MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)
Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

Estd. 2000

IQAC

L-1

LECTURE HANDOUTS

AI&DS

II/III

Course Name with Code : **OBJECT ORIENTED PROGRAMMING-19ADC06**

Course Faculty : **T.Divya**

Unit : **I- Introduction to OOP and Java Fundamentals** **Date of Lecture:**

Topic of Lecture: Object Oriented Programming ,Abstraction ,objects and classes

Introduction : (Maximum 5 sentences) :

- Object-oriented programming aims to implement real-world entities like inheritance, hiding, polymorphism etc in programming.
- The main aim of OOP is to bind together the data and the functions that operate on them so that no other part of the code can access this data except that function

Prerequisite knowledge for Complete understanding and learning of Topic:
(Max. Four important topics)

- Basic terminologies of Program
- C program concepts

Detailed content of the Lecture:

Object Oriented Programming – Abstraction:

- Data Abstraction is the property by virtue of which only the essential details are displayed to the user
- The trivial or the non-essentials units are not displayed to the user.
- Ex: A car is viewed as a car rather than its individual components
- Data Abstraction may also be defined as the process of identifying only the required characteristics of an object ignoring the irrelevant details.
- The properties and behaviors of an object differentiate it from other objects of similar type and also help in classifying/grouping the objects

Object:

- It is a basic unit of Object Oriented Programming and represents the real life entities.
- A typical Java program creates many objects, which as you know, interact by invoking methods. An object consists of:
- **State** : It is represented by attributes of an object. It also reflects the properties of an object.
- **Behavior** : It is represented by methods of an object. It also reflects the response of an object with other objects.
- **Identity** : It gives a unique name to an object and enables one object to interact with other objects.
- In C++, an object is created from a class. We have already created the class named MyClass, so now we can use this to create objects
- To create an object of MyClass, specify the class name, followed by the object name
- To access the class attributes (myNum and myString), use the dot syntax (.) on the object

//Java Program to illustrate how to define a class and fields

//Defining a Student class.

```
class Student{
 //defining fields
 int id;//field or data member or instance variable
 String name;
 //creating main method inside the Student class
 public static void main(String args[]){
  //Creating an object or instance
  Student s1=new Student();//creating an object of Student
  //Printing values of the object
  System.out.println(s1.id);//accessing member through reference variable
  System.out.println(s1.name);
 }
}
```

Class:

- A class is a user defined blueprint or prototype from which objects are created.
- It represents the set of properties or methods that are common to all objects of one type
- **Body:** The class body surrounded by braces, { }.
- **To create a class, use the class keyword**

Example

```
Create a class called "MyClass"
class MyClass {       // The class
  public:             // Access specifier
    int myNum;        // Attribute (int variable)
    string myString;  // Attribute (string variable)
};
```

Abstraction:

- Data abstraction refers to providing only essential information about the data to the outside world, hiding the background details or implementation
- Abstraction using Classes: We can implement Abstraction in C++ using classes.
- Class helps us to group data members and member functions using available access specifiers

Benefits of Data Abstraction

Data abstraction provides two important advantages −

- Class internals are protected from inadvertent user-level errors, which might corrupt the state of the object.
- The class implementation may evolve over time in response to changing requirements or bug reports without requiring change in user-level code.

Video Content / Details of website for further learning (if any):

https://medium.com/@viktor.kukurba/object-oriented-programming-in-javascript-1-abstraction-c47307c469d1

https://www.youtube.com/watch?v=57SIpmA3vcg

Important Books/Journals for further learning including the page nos.:

David J., Object-oriented programming, University of KwaZulu-Natal,Version 5, 2006Page No:11-39

Course Faculty

Verified by HOD

MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)
Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

Estd. 2000

IQAC

L-2

LECTURE HANDOUTS

AI&DS

II/III

Course Name with Code : **OBJECT ORIENTED PROGRAMMING-19ADC06**

Course Faculty : **T.Divya**

Unit : **I- Introduction to OOP and Java Fundamentals** **Date of Lecture:**

Topic of Lecture: Encapsulation, Inheritance ,Polymorphism

Introduction : (Maximum 5 sentences) :

- Encapsulation is defined as wrapping up of data and information under a single unit.
- The capability of a class to derive properties and characteristics from another class is called Inheritance
- The word polymorphism means having many forms. In simple words, we can define polymorphism as the ability of a message to be displayed in more than one form.

Prerequisite knowledge for Complete understanding and learning of Topic:
(Max. Four important topics)

- Object oriented programming
- Class and objects

Detailed content of the Lecture:

Encapsulation:

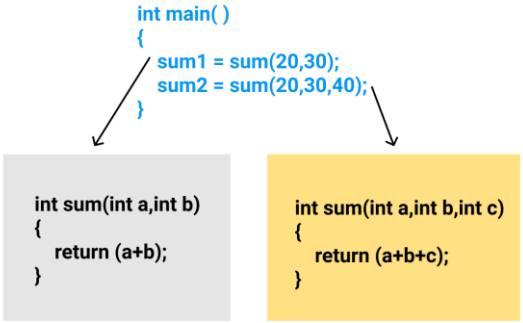
- Encapsulation is defined as wrapping up of data and information under a single unit.
- In Object-Oriented Programming, Encapsulation is defined as binding together the data and the functions that manipulate them.
- Encapsulation also leads to data abstraction or hiding.
- As using encapsulation also hides the data.
- Encapsulation is a process of combining data and function into a single unit like capsule.
- This is to avoid the access of private data members from outside the class.
- To achieve encapsulation, we make all data members of class private and create public functions, using them we can get the values from these data members or set the value to these data members.

Inheritance

- The capability of a class to derive properties and characteristics from another class is called Inheritance. Inheritance is one of the most important features of Object-Oriented Programming.
- Sub Class: The class that inherits properties from another class is called Sub class or Derived Class.
- Super Class: The class whose properties are inherited by sub class is called Base Class or Super class.
- Reusability: Inheritance supports the concept of "reusability", i.e. when we want to create a new class and there is already a class that includes some of the code that we want, we can derive our new class from the existing class. By doing this, we are reusing the fields and methods of the existing class.
- Example: Dog, Cat, Cow can be Derived Class of Animal Base Class.

Polymorphism:

- The word polymorphism means having many forms. In simple words, we can define polymorphism as the ability of a message to be displayed in more than one form.
- A person at the same time can have different characteristic. Like a man at the same time is a father, a husband, an employee. So the same person posses different behavior in different situations. This is called polymorphism.
- An operation may exhibit different behaviors in different instances. The behavior depends upon the types of data used in the operation.
- Operator Overloading: The process of making an operator to exhibit different behaviors in different instances is known as operator overloading.
- Function Overloading: Function overloading is using a single function name to perform different. Polymorphism is extensively used in implementing inheritance.



Video Content / Details of website for further learning (if any):
https://www.fusion-reactor.com/blog/technical-blogs/object-oriented-programming-what-is-inheritance-polymorphism-abstraction-encapsulation/
https://www.youtube.com/watch?v=QzX7REqciPY

Important Books/Journals for further learning including the page nos.:
Herbert Schildt, " Java : The Complete Reference" , Mc Graw Hill Publication, 2007, Page No: 157-162

Course Faculty

Verified by HOD

MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)

Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

L-3

LECTURE HANDOUTS

AI&DS

II/III

Course Name with Code : **OBJECT ORIENTED PROGRAMMING-19ADC06**

Course Faculty : **T.Divya**

Unit : **I- Introduction to OOP and Java Fundamentals** **Date of Lecture:**

Topic of Lecture: OOP in Java , Characteristics of Java ,The Java Environment

Introduction : (Maximum 5 sentences) :

- In Java, everything is an Object.
- Java can be easily extended since it is based on the Object model.
- Java syntax is based on C++
- Java has removed many complicated and rarely-used features, for example, explicit pointers, operator overloading, etc.
- There is no need to remove unreferenced objects because there is an Automatic Garbage Collection in Java

Prerequisite knowledge for Complete understanding and learning of Topic:
(Max. Four important topics)

- Objects
- Classes
- OOPs concepts

Detailed content of the Lecture:

Characteristics of Java:

Java has the following characteristics:

- Object oriented - Java provides the basic object technology of C++ with some enhancements and some deletions.
- Architecture neutral - Java source code is compiled into architecture-independent object code.
- The object code is interpreted by a Java Virtual Machine (JVM) on the target architecture.
- Portable - Java implements additional portability standards. For example, ints are always 32-bit, 2's-complemented integers.
- User interfaces are built through an abstract window system that is readily implemented in Solaris and other operating environments.
- Distributed - Java contains extensive TCP/IP networking facilities. Library routines support protocols such as HyperText Transfer Protocol (HTTP) and file transfer protocol (FTP).
- Robust - Both the Java compiler and the Java interpreter provide extensive error checking.
- Java manages all dynamic memory, checks array bounds, and other exceptions.
- Secure - Features of C and C++ that often result in illegal memory accesses are not in the Java language.
- The interpreter also applies several tests to the compiled code to check for illegal code.
- After these tests, the compiled code causes no operand stack over- or underflows, performs no

illegal data conversions, performs only legal object field accesses, and all opcode parameter types are verified as legal.

- High performance - Compilation of programs to an architecture independent machine-like language, results in a small efficient interpreter of Java programs. The Java environment also compiles the Java bytecode into native machine code at runtime.
- Multithreaded - Multithreading is built into the Java language.
- It can improve interactive performance by allowing operations, such as loading an image, to be performed while continuing to process user actions.
- Dynamic - Java does not link invoked modules until runtime.
- Simple - Java is similar to C++, but with most of the more complex features of C and C++ removed.

  Java does not provide:
  - Programmer-controlled dynamic memory
  - Pointer arithmetic
  - struct
  - typedefs
  - #define

The Java Environment:

The Java Runtime Environment, or JRE, is a software layer that runs on top of a computer's operating system software and provides the class libraries and other resources that a specific Java program needs to run.The JRE is one of three interrelated components for developing and running Java programs. The other two components are as follows:

- The Java Development Kit, or JDK, is a set of tools for developing Java applications. Developers choose JDKs by Java version and by package or edition—Java Enterprise Edition (Java EE), Java Special Edition (Java SE), or Java Mobile Edition (Java ME). Every JDK always includes a compatible JRE, because running a Java program is part of the process of developing a Java program.
- The Java Virtual Machine, or JVM, executes live Java applications. Every JRE includes a default JRE, but developers are free to choose another that meets the specific resource needs of their applications.

Video Content / Details of website for further learning (if any):
https://developer.ibm.com/languages/java/tutorials/j-introtojava1/
https://intellipaat.com/blog/tutorial/java-tutorial/java-introduction-installation/

Important Books/Journals for further learning including the page nos.:
Herbert Schildt,Java, The Complete Reference ,8th Edition, McGraw Hill Education, 2011,Page No:15 - 32

Course Faculty

Verified by HOD

MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)

Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

Estd. 2000

IQAC

L-4

LECTURE HANDOUTS

AI&DS

II/III

Course Name with Code : **OBJECT ORIENTED PROGRAMMING-19ADC06**

Course Faculty : **T.Divya**

Unit : **I- Introduction to OOP and Java Fundamentals** **Date of Lecture:**

Topic of Lecture: Java Source File, Structure ,Compilation

Introduction : (Maximum 5 sentences) :

- Many implementations of the Java platform rely on hierarchical file systems to manage source and class files, although The Java Language Specification does not require this
- Java programs need to be compiled to bytecode.
- When the bytecode is run, it needs to be converted to machine code.

Prerequisite knowledge for Complete understanding and learning of Topic:
(Max. Four important topics)

- OOPs concepts
- JAVA environment

Detailed content of the Lecture:

Java Source File:

- Many implementations of the Java platform rely on hierarchical file systems to manage source and class files, although The Java Language Specification does not require this. The strategy is as follows.
- Put the source code for a class, interface, enumeration, or annotation type in a text file whose name is the simple name of the type and whose extension is .java. For example:

```
//in the Rectangle.java file
package graphics;
public class Rectangle {
   ...
}
```

- Then, put the source file in a directory whose name reflects the name of the package to which the type belongs:

.....\graphics\Rectangle.java

- The qualified name of the package member and the path name to the file are parallel, assuming the Microsoft Windows file name separator backslash (for UNIX, use the forward slash).

class name – graphics.Rectangle
pathname to file – graphics\Rectangle.java

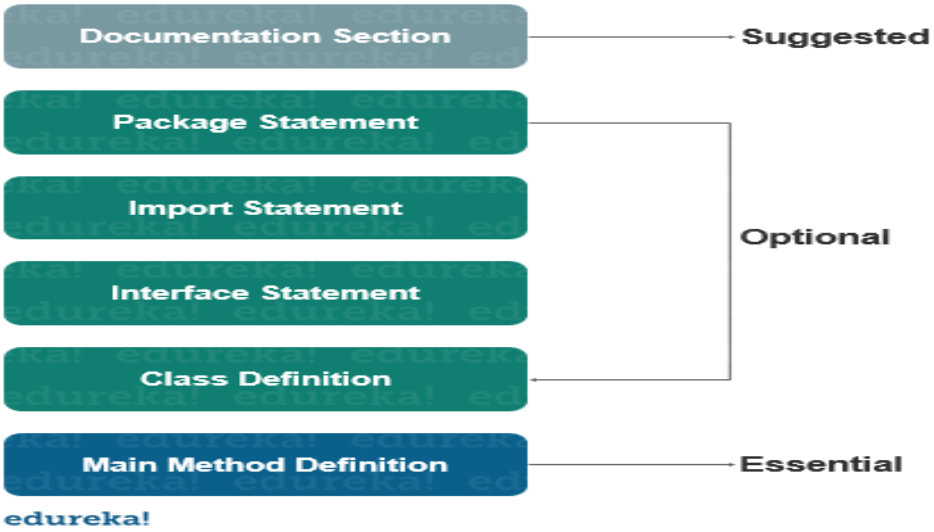
Structure:

- Documentation Section
- Package Statement
- Import Statements
- Interface Statement
- Class Definition
- Main Method Class

- Main Method Definition

There are three types of comments that Java supports:

- Single line Comment
- Multi-line Comment
- Documentation Comment



Compilation:

- In Java, programs are not compiled into executable files; they are compiled into bytecode (as discussed earlier), which the JVM (Java Virtual Machine) then executes at runtime.
- Java source code is compiled into bytecode when we use the javac compiler.
- The bytecode gets saved on the disk with the file extension .class.
- When the program is to be run, the bytecode is converted, using the just-in-time (JIT) compiler.
- The result is machine code which is then fed to the memory and is executed.

Java code needs to be compiled twice in order to be executed:

1. Java programs need to be compiled to bytecode.
2. When the bytecode is run, it needs to be converted to machine code.

- The Java classes/bytecode are compiled to machine code and loaded into memory by the JVM when needed the first time.
- This is different from other languages like C/C++ where programs are to be compiled to machine code and linked to create an executable file before it can be executed.

Video Content / Details of website for further learning (if any):
https://www.youtube.com/watch?v=do2zwKh9Q8E
https://www.youtube.com/watch?v=9KOmK1ulObA

Important Books/Journals for further learning including the page nos.:
Herbert Schildt,Java, The Complete Reference ,8th Edition, McGraw Hill Education, 2011,Page No:21 -26

Course Faculty

Verified by HOD

MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)
Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

Estd. 2000

IQAC

L -5

LECTURE HANDOUTS

AI&DS

II/III

Course Name with Code : **OBJECT ORIENTED PROGRAMMING-19ADC06**

Course Faculty : **T.Divya**

Unit : **I- Introduction to OOP and Java Fundamentals** **Date of Lecture:**

Topic of Lecture: Fundamental Programming, Structures in Java

Introduction : (Maximum 5 sentences) :

- Many implementations of the Java platform rely on hierarchical file systems to manage source and class files, although The Java Language Specification does not require this
- Java programs need to be compiled to bytecode.
- When the bytecode is run, it needs to be converted to machine code.

Prerequisite knowledge for Complete understanding and learning of Topic:
(Max. Four important topics)

- Java Source File Structures
- Compilation

Detailed content of the Lecture:

Fundamental Programming:

- Java is case sensitive
- 'public' is access modifier
- 'class' reminds that everything in a Java program lives inside a class
- Think 'class' as a container for the program logic that defines the behavior of an application
- class 'name' must begin with a letter, and after that they can have any combination of letters and digits with unlimited length
- HelloWorld.java Source code compilation using javac produces the platform independent byte code HelloWorld.class
- When you use java Hello World to run the program, the Java interpreter always starts execution with the code in the main method in the class
- Thus, you must have a main methods in the source file for your class in order for your code to execute
- The main method in Java is always static
- Unlike C/C++, the main method does not return an "exit code" to the operating system
- If the main method exits normally, the Java program has the exit code 0, indicating successful completion
- To terminate the program with different exit code, use the System.exit method
- Here System.out object calls the println method, object.method(parameters)

Structures in Java:

- Every Java program consists of a collection of classes--nothing else.
- A class is a template for creating a particular form of object.
- Each object created by the template contains the same members, each of which is either a field

or a method.

- A java program may contain many classes of which only one class will have a main method.
- Class will contain data members and methods that operate on the data members of the class.
- To write a Java program, we first need to define classes and then put them together.

```
Package list;
public Class <class name>
{
    Data members;
    Constructor functions;
    User-defined methods;

    public static void main (String arguments[])
    {
        Block of statements;
    }
}
```

More Information regarding Java Class:

- Java is an object-oriented language, which means that it has constructs to represent objects from the real world.
- Each Java program has at least one class that knows how to do certain things or how to represent some type of object.
- Classes in Java may have methods (or functions) and fields (or attributes or properties).

Video Content / Details of website for further learning (if any):
https://www.edx.org/course/introduction-to-java-programming-fundamental-data
https://www.youtube.com/watch?v=210liKT3_n8

Important Books/Journals for further learning including the page nos.:
Herbert Schildt,Java, The Complete Reference ,8th Edition, McGraw Hill Education, 2011,Page No:15 -32

Course Faculty

Verified by HOD

MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)
Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

Estd. 2000

IQAC

L-6

LECTURE HANDOUTS

AI&DS

II/III

Course Name with Code : **OBJECT ORIENTED PROGRAMMING-19ADC06**

Course Faculty : **T.Divya**

Unit : **I- Introduction to OOP and Java Fundamentals** **Date of Lecture:**

Topic of Lecture: Defining classes in Java, Constructors, Methods ,Access Specifiers

Introduction : (Maximum 5 sentences) :

- Classes and Objects are basic concepts of Object Oriented Programming which revolve around the real life entities.
- A class is a user defined blueprint or prototype from which objects are created.
- It represents the set of properties or methods that are common to all objects of one type.
- Constructor is a block of code that initializes the newly created object but it's not a method as it doesn't have a return type
- Constructor has same name as the class and looks like this in a java code

Prerequisite knowledge for Complete understanding and learning of Topic:
(Max. Four important topics)

- OOPs concepts
- JAVA environment

Detailed content of the Lecture:

Defining classes in Java:

- A class can be defined as a template/blueprint that describes the behavior/state that the object of its type support

Create a Class

To create a class, use the keyword `class`:

MyClass.java

Create a class named "MyClass" with a variable x:

```
public class MyClass {

  int x = 5;

}
```

Classes in Java:

- A class is a blueprint from which individual objects are created.

```
public class Dog {
   String breed;
   int age;
   String color;
```

```
  void barking() {
  }
  void hungry() {
  }
  void sleeping() {
  }
}
```

A class can contain any of the following variable types.

- **Local variables** − Variables defined inside methods, constructors or blocks are called local variables. The variable will be declared and initialized within the method and the variable will be destroyed when the method has completed.
- **Instance variables** − Instance variables are variables within a class but outside any method. These variables are initialized when the class is instantiated. Instance variables can be accessed from inside any method, constructor or blocks of that particular class.
- **Class variables** − Class variables are variables declared within a class, outside any method, with the static keyword.

A class can have any number of methods to access the value of various kinds of methods. In the above example, barking(), hungry() and sleeping() are methods.

- A constructor in Java is a **special method** that is used to initialize objects.
- The constructor is called when an object of a class is created.
- It can be used to set initial values for object attributes:
- In Java, a constructor is a block of codes similar to the method.
- It is called when an instance of the class is created.
- At the time of calling constructor, memory for the object is allocated in the memory.
- It is a special type of method which is used to initialize the object.
- Every time an object is created using the new() keyword, at least one constructor is called.
- A constructor is used to initialize the state of an object.
- A constructor must not have a return type.

Rules for creating Java constructor:

There are two rules defined for the constructor.

- Constructor name must be the same as its class name
- A Constructor must have no explicit return type
- A Java constructor cannot be abstract, static, final, and synchronized

Types of Java constructors

There are two types of constructors in Java:

1. Default constructor (no-arg constructor)
2. Parameterized constructor

Java Default Constructor

A constructor is called "Default Constructor" when it doesn't have any parameter.

Syntax of default constructor:

<class_name>(){}

They are:

- By constructor
- By assigning the values of one object into another
- By clone() method of Object class

Methods:

- A Java method is a collection of statements that are grouped together to perform an operation.
- When you call the System.out.println() method, for example, the system actually executes several statements in order to display a message on the console.

Creating Method

Considering the following example to explain the syntax of a method −

Syntax

```
public static int methodName(int a, int b) {
  // body
}
```

Here,

- **public static** − modifier
- **int** − return type
- **methodName** − name of the method
- **a, b** − formal parameters
- **int a, int b** − list of parameters

The syntax shown above includes −

- **modifier** − It defines the access type of the method and it is optional to use.
- **returnType** − Method may return a value.
- **nameOfMethod** − This is the method name. The method signature consists of the method name and the parameter list.
- **Parameter List** − The list of parameters, it is the type, order, and number of parameters of a method. These are optional, method may contain zero parameters.
- **method body** − The method body defines what the method does with the statements.

Access Specifiers:

- Access modifiers (or access specifiers) are keywords in object-oriented languages that set the accessibility of classes, methods, and other members.
- public - members are accessible from outside the class
- private - members cannot be accessed (or viewed) from outside the class
- protected - members cannot be accessed from outside the class, however, they can be accessed in inherited classes.

Example

```
class MyClass {
  public:    // Public access specifier
    int x;   // Public attribute
  private:   // Private access specifier
    int y;   // Private attribute
};
int main() {
  MyClass myObj;
  myObj.x = 25;  // Allowed (public)
  myObj.y = 50;  // Not allowed (private)
  return 0;}
```

Video Content / Details of website for further learning (if any):

https://www.youtube.com/watch?v=4xKihjI6HJ0

https://www.guru99.com/java-oops-class-objects.html

Important Books/Journals for further learning including the page nos.:

Herbert Schildt,Java, The Complete Reference ,8th Edition, McGraw Hill Education, 2011,Page No:105 - 122

Course Faculty

Verified by HOD

MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)
Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

Estd. 2000

IQAC

L-7

LECTURE HANDOUTS

AI&DS

II/III

Course Name with Code : **OBJECT ORIENTED PROGRAMMING-19ADC06**

Course Faculty : **T.Divya**

Unit : **I- Introduction to OOP and Java Fundamentals** **Date of Lecture:**

Topic of Lecture: Static members –Comments, Data Types

Introduction : (Maximum 5 sentences) :

- Static methods can access only static fields, methods.
- To access static methods there is no need to instantiate the class
- Program comments are explanatory statements that you can include in the C++ code.
- These comments help anyone reading the source code.

Prerequisite knowledge for Complete understanding and learning of Topic:
(Max. Four important topics)

- Defining classes in Java
- Constructors
- Methods

Video Content / Details of website for further learning (if any):

Static Methods

- You can create a static method by using the keyword static.
- Static methods can access only static fields, methods. To access static methods there is no need to instantiate the class

Example

```
public class MyClass {
  public static void sample(){
    System.out.println("Hello");
  }
  public static void main(String args[]){
    MyClass.sample();
  }
}
```

Output

```
Hello
```

Static Fields –

- You can create a static field by using the keyword static.
- The static fields have the same value in all the instances of the class.
- These are created and initialized when the class is loaded for the first time.
- Just like static methods you can access static fields using the class name (without instantiation).

Static Blocks :

- These are a block of codes with a static keyword.
- In general, these are used to initialize the static members.
- JVM executes static blocks before the main method at the time of class loading.

Comments :

- C++ supports single-line and multi-line comments.
- All characters available inside any comment are ignored by C++ compiler.

C++ comments start with /* and end with */. For example −

```
/* This is a comment */

/* C++ comments can also
   * span multiple lines
*/
```

A comment can also start with //, extending to the end of the line.

Data Types:

- All variables use data-type during declaration to restrict the type of data to be stored.
- Therefore, we can say that data types are used to tell the variables the type of data it can store.

Primitive Data Types:

These data types are built-in or predefined data types and can be used directly by the user to declare variables. example: int, char , float, bool etc. Primitive data types available in C++ are:

- Integer
- Character
- Boolean
- Floating Point
- Double Floating Point
- Valueless or Void
- Wide Character

Derived Data Types: The data-types that are derived from the primitive or built-in datatypes are referred to as Derived Data Types. These can be of four types namely:

- Function
- Array
- Pointer
- Reference

Abstract or User-Defined Data Types:

These data types are defined by user itself. Like, defining a class in C++ or a structure. C++ provides the following user-defined datatypes:

- Class , Structure, Union , Enumeration, Typedef defined DataType

Video Content / Details of website for further learning (if any):

https://www.youtube.com/watch?v=0dSDJJzWqQI
https://classes.mst.edu/compsci1570/staticmembers.htm

Important Books/Journals for further learning including the page nos.:

Herbert Schildt,Java, The Complete Reference ,8th Edition, McGraw Hill Education, 2011,Page No:33-40,141

Course Faculty

Verified by HOD

MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)
Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

Estd. 2000

IQAC

L-8

LECTURE HANDOUTS

AI&DS

II/III

Course Name with Code : **OBJECT ORIENTED PROGRAMMING-19ADC06**

Course Faculty : **T.Divya**

Unit : **I- Introduction to OOP and Java Fundamentals** **Date of Lecture:**

Topic of Lecture: Variables, Operators, Control Flow

Introduction : (Maximum 5 sentences) :

- A variable is a name which is associated with a value that can be changed.
- An operator is a symbol that tells the compiler to perform specific mathematical or logical manipulations.
- Control flow or flow of control is the order in which instructions, statements and function calls being executed or evaluated when a program is running

Prerequisite knowledge for Complete understanding and learning of Topic:
(Max. Four important topics)

- Class
- Object
- Programming structure in JAVA

Video Content / Details of website for further learning (if any):

Variables :

- A variable is a name which is associated with a value that can be changed.
- For example when I write int num=20; here variable name is num which is associated with value 20, int is a data type that represents that this variable can hold integer values.

Syntax of declaring a variable in C++

```
data_type variable1_name = value1, variable2_name = value2;
```

For example:

```
int num1=20, num2=100;
```

We can also write it like this:

```
int num1,num2;
num1=20;
num2=100
```

Operators :

- An operator is a symbol that tells the compiler to perform specific mathematical or logical manipulations. C++ is rich in built-in operators and provide the following types of operators −
- Arithmetic Operators: A + B, A – B, A * B, B % A , A++, A--
- Relational Operators: (A == B) , (A != B) , (A > B), (A < B) , (A >= B), (A <= B)
- Logical Operators: (A && B) , (A || B), !(A && B)
- Bitwise Operators: (A & B) will give 12 which is 0000 1100, (A | B) will give 61 which is 0011 1101
- Assignment Operators: C = A + B, C += A, C -= A
- This chapter will examine the arithmetic, relational, logical, bitwise, assignment and other operators one by one.
- **Arithmetic Operator**s (+, -, *, /, %, post-increment, pre-increment, post-decrement, pre-

decrement)

- **Relational Operators** (==, !=, >, <, >= & <=) Logical Operators (&&, || and !)
- **Bitwise Operators** (&, |, ^, ~, >> and <<)
- **Assignment Operator**s (=, +=, -=, *=, etc)
- **Other Operators** (conditional, comma, sizeof, address, redirection)

Control flow

- Control flow or flow of control is the order in which instructions, statements and function calls being executed or evaluated when a program is running.
- The control flow statements are also called as Flow Control Statements.

IF:

```
if (booleanExpression)
{
   expressions1;
}
[else
{
   expressions2;
}]
```

DO…WHILE:

```
do
{
   expressions;
} while(booleanExpression);
```

FOR:

```
for (initCommand; boolExpression; loopExpression)
{
   expressions;
}
```

SWITCH:

```
switch(integerExpression)
{
  case val1:
        expressions1;
        break;
  case val2:
        expressions2;
        break;
  [default:
        expressionsN;
  ]
}
```

Video Content / Details of website for further learning (if any):

https://sites.google.com/view/sirallan-programmingtutorials/c-basics/using-c-operators
https://cosmolearning.org/video-lectures/c-arithmetic-operators-for-beginners/

Important Books/Journals for further learning including the page nos.:

Herbert Schildt,Java, The Complete Reference ,8th Edition, McGraw Hill Education, 2011,Page No:41- 103

Course Faculty

Verified by HOD

MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)

Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

Estd. 2000

IQAC

LECTURE HANDOUTS

L-9

AI&DS

II/III

Course Name with Code : **OBJECT ORIENTED PROGRAMMING-19ADC06**

Course Faculty : **T.Divya**

Unit : **I- Introduction to OOP and Java Fundamentals** **Date of Lecture:**

Topic of Lecture: Arrays , Packages , JavaDoc comments

Introduction : (Maximum 5 sentences) :

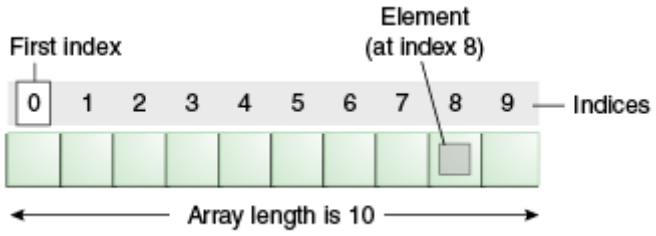
- A variable is a name which is associated with a value that can be changed.
- An operator is a symbol that tells the compiler to perform specific mathematical or logical manipulations.
- Control flow or flow of control is the order in which instructions, statements and function calls being executed or evaluated when a program is running

Prerequisite knowledge for Complete understanding and learning of Topic:
(Max. Four important topics)

- OOPs concepts
- JAVA source file structure

Detailed content of the Lecture:

- An array is a collection of items stored at contiguous memory locations.
- The idea is to store multiple items of the same type together.
- This makes it easier to calculate the position of each element by simply adding an offset to a base value, i.e., the memory location of the first element of the array (generally denoted by the name of the array).
- The base value is index 0 and the difference between the two indexes is the offset.



Types of Array in java

There are two types of array.

- Single Dimensional Array
- Two Dimensional Array
- Multidimensional Array

Single Dimensional Array in Java

Syntax to Declare an Array in Java

dataType[] arr; (or)
dataType []arr; (or)
dataType arr[];

Instantiation of an Array in Java

arrayRefVar=**new** datatype[size];

Example :

```
class Testarray{
public static void main(String args[]){
int a[]=new int[5];//declaration and instantiation
a[0]=10;//initialization
a[1]=20;
a[2]=70;
a[3]=40;
a[4]=50;
//traversing array
for(int i=0;i<a.length;i++)//length is the property of array
System.out.println(a[i]);
}}
```

COMMENTS

- Comments can be used to explain C++ code, and to make it more readable. It can also be used to prevent execution when testing alternative code. Comments can be singled-lined or multi-lined.
- Single-line comments start with two forward slashes (//).

Video Content / Details of website for further learning (if any):
https://www.javatpoint.com/array-in-java

Important Books/Journals for further learning including the page nos.:
Herbert Schildt,Java, The Complete Reference ,8th Edition, McGraw Hill Education, 2011,Page No:48 - 56,183 - 190

Course Faculty

Verified by HOD

MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)
Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

Estd. 2000

IQAC

L-10

LECTURE HANDOUTS

AI&DS

II/III

Course Name with Code : **OBJECT ORIENTED PROGRAMMING – 19ADC06**

Course Faculty : **T.Divya**

Unit : **II –Inheritance and Interfaces** **Date of Lecture:**

Topic of Lecture: Inheritance ,Super classes,Sub classes

Introduction : (Maximum 5 sentences) :

- Inheritance is basically the process of basing a class on another class i.e. to build a class on a existing class.
- A super class is the class from which many subclasses can be created.
- A subclass is a class derived from the super class.
- It inherits the properties of the super class and also contains attributes of its own.

Prerequisite knowledge for Complete understanding and learning of Topic:
(Max. Four important topics)

- Data
- Variable
- Data Hiding
- Super Class
- Sub Class

Detailed content of the Lecture:

Inheritance:

- Inheritance is basically the process of basing a class on another class i.e to build a class on an existing class.
- The new class contains all the features and functionalities of the old class in addition to its own.
- The class which is newly created is known as the subclass or child class and the original class is the parent class or the super class.

Types of Inheritance:

Single Inheritance :

- In single inheritance, subclasses inherit the features of one super class. In image below, the class A serves as a base class for the derived class B.

Multilevel Inheritance :

- In Multilevel Inheritance, a derived class will be inheriting a base class and as well as the derived class also act as the base class to other class. In below image, the class A serves as a base class for the derived class B, which in turn serves as a base class for the derived class C. In Java, a class cannot directly access the grandparent's members.

Hierarchical Inheritance :

- In Hierarchical Inheritance, one class serves as a superclass (base class) for more than one sub class.In below image, the class A serves as a base class for the derived class B,C and D.

Multiple Inheritance (Through Interfaces) :

- In Multiple inheritance ,one class can have more than one superclass and inherit features from all parent classes. Please note that Java does **not** support multiple inheritance with classes.
- In java, we can achieve multiple inheritance only through Interfaces. In image below, Class C is derived from interface A and B.

Super classes

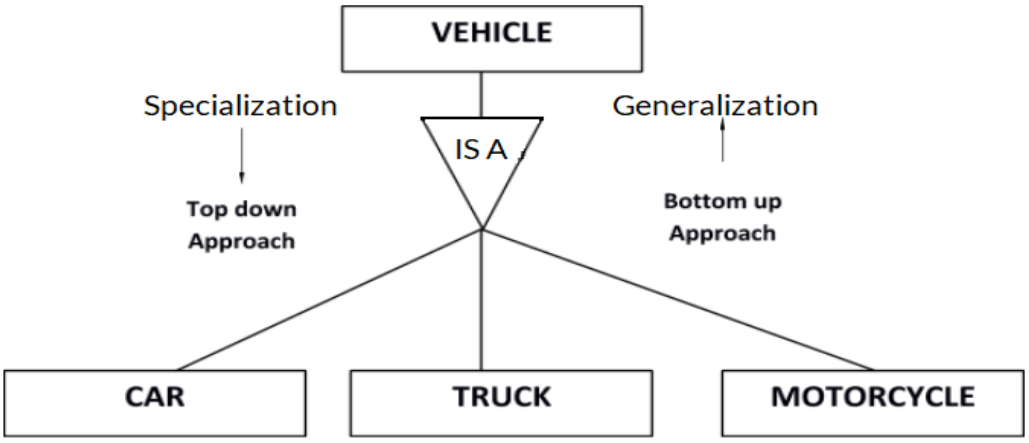
- A superclass is the class from which many subclasses can be created.
- The subclasses inherit the characteristics of a superclass.
- The superclass is also known as the parent class or base class.
- In the above example, Vehicle is the Superclass and its subclasses are Car, Truck and Motorcycle.

Subclasses

- A subclass is a class derived from the superclass.
- It inherits the properties of the superclass and also contains attributes of its own.

An example is:

- Car, Truck and Motorcycle are all subclasses of the superclass Vehicle.
- They all inherit common attributes from vehicle such as speed, colour etc.
- while they have different attributes also i.e Number of wheels in Car is 4 while in Motorcycle is 2.



Video Content / Details of website for further learning (if any):

https://www.youtube.com/watch?v=MHZHwbUZYJM
https://www.youtube.com/watch?v=0PPKccntohM

Important Books/Journals for further learning including the page nos.:

Herbert Schildt, " Java : The Complete Reference" , Mc Graw Hill Publication, 2007, Page No: 205-212

Course Faculty

Verified by HOD

MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)

Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

Estd. 2000

IQAC

L-11

LECTURE HANDOUTS

AI&DS

II/III

Course Name with Code : **OBJECT ORIENTED PROGRAMMING – 19ADC06**

Course Faculty : **T.Divya**

Unit : **II –Inheritance and Interfaces** **Date of Lecture:**

Topic of Lecture: Protected members

Introduction : (Maximum 5 sentences) :

- Protected member variable or function is very similar to a private member but it provided one additional benefit that they can be accessed in child classes which are called derived classes.

Prerequisite knowledge for Complete understanding and learning of Topic:
(Max. Four important topics)

- Data
- Variable
- Data Hiding
- Class
- Members

Detailed content of the Lecture:

Access modifiers:

1) private
2) protected
3) public.

They are used based on their properties

Protected members:

Java protected keyword

A Java protected keyword is an access modifier. It can be assigned to variables, methods, constructors and inner classes.

Points to remember

- The protected access modifier is accessible within the package. However, it can also accessible outside the package but through inheritance only.
- We can't assign protected to outer class and interface.
- If you make any constructor protected, you cannot create the instance of that class from outside the package.
- If you are overriding any method, overridden method (i.e., declared in the subclass) must not be more restrictive.
- According to the previous point, if you assign protected to any method or variable, that method

or variable can be overridden to sub-class using public or protected access modifier only.

Examples of protected keyword

Example 1

```
//save by A.java
package com.java;

public class A {

 protected String msg="Try to access the protected variable outside the package";

}
//save by ProtectedExample1.java
package com.javatpoint;
import com.java.A;

public class ProtectedExample1 {
public static void main(String[] args) {
   A a=new A();
  System.out.println(a.msg);

}
}
```

Output:

```
Exception in thread "main" java.lang.Error: Unresolved compilation problem:
        The field A.msg is not visible
```

Video Content / Details of website for further learning (if any):
https://www.javatpoint.com/protected-keyword-in-java

Important Books/Journals for further learning including the page nos.:
Herbert Schildt, " Java : The Complete Reference" , Mc Graw Hill Publication, 2007, Page No: 205-212

Course Faculty

Verified by HOD

MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)

Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

Estd. 2000

L-12

LECTURE HANDOUTS

AI&DS

II/III

Course Name with Code : **OBJECT ORIENTED PROGRAMMING – 19ADC06**

Course Faculty : **T.Divya**

Unit : **II –Inheritance and Interfaces** **Date of Lecture:**

Topic of Lecture: Constructors in sub classes, the Object class

Introduction : (Maximum 5 sentences) :

- A **subclass** inherits all the members (fields, methods, and **nested classes**) from its super class.
- **Constructors** are not members, so they are not inherited by **subclasses**, but the **constructor** of the super class can be invoked from the **subclass**.
- Benefit that they can be accessed in child classes which are called derived classes.

Prerequisite knowledge for Complete understanding and learning of Topic:
(Max. Four important topics)

- Data
- Variable
- Data Hiding
- Class
- Members

Video Content / Details of website for further learning (if any):

A Java constructor is special method that is called when an object is instantiated. In other words, when you use the new keyword. The purpose of a Java constructor is to initializes the newly created object before it is used.

Here is a simple example that creates an object, which results in the class constructor being called:

```
MyClass myClassObj = new MyClass();
```

This example results in a new MyClass object being created, and the no-arg constructor of MyClass to be called. You will learn what the no-arg constructor is later.

A Java class constructor initializes instances (objects) of that class. Typically, the constructor initializes the fields of the object that need initialization. Java constructors can also take parameters, so fields can be initialized in the object at creation time.

Defining a Constructor in Java

Here is a simple Java constructor declaration example. The example shows a very simple Java class with a single constructor.

```
public class MyClass {

    public MyClass() {

    }
}
```

The constructor is this part:

```
    public MyClass() {

    }
```

The first part of a Java constructor declaration is an access modifier. The access modifier have the same meanings as for methods and fields. They determine what classes can access (call) the constructor.

Calling a Constructor

You call a constructor when you create a new instance of the class containing the constructor. Here is a Java constructor call example:

```
MyClass myClassVar = new MyClass();
```

This example invokes (calls) the no-argument constructor for MyClass as defined earlier in this text.

In case you want to pass parameters to the constructor, you include the parameters between the parentheses after the class name, like this:

```
MyClass myClassVar = new MyClass(1975);
```

Calling Constructors in Superclasses

In Java a class can extend another class. When a class extends another class it is also said to "inherit" from the class it extends. The class that extends is called the subclass, and the class being extended is called the superclass. Inheritance is covered in more detail in my tutorial about **inheritance in Java**.

Video Content / Details of website for further learning (if any):
http://tutorials.jenkov.com/java/constructors.html

Important Books/Journals for further learning including the page nos.:
Herbert Schildt, " Java : The Complete Reference" , Mc Graw Hill Publication, 2007, Page No: 105-112

Course Faculty

Verified by HOD

MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)

Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

Estd. 2000

L-13

LECTURE HANDOUTS

AI&DS

II/III

Course Name with Code : **OBJECT ORIENTED PROGRAMMING – 19ADC06**

Course Faculty : **T.Divya**

Unit : **II –Inheritance and Interfaces** **Date of Lecture:**

Topic of Lecture: Abstract classes and methods

Introduction : (Maximum 5 sentences) :

- Abstract class cannot be instantiated, but pointers and references of Abstract class type can be created.
- Abstract class can have normal functions and variables along with a pure virtual function.
- Abstract classes are mainly used for Up casting, so that its derived classes can use its interface.

Prerequisite knowledge for Complete understanding and learning of Topic:
(Max. Four important topics)

- Data
- Variable
- Data Hiding
- Class

Detailed content of the Lecture:

Abstract class in Java

A class which is declared with the abstract keyword is known as an abstract class in Java. It can have abstract and non-abstract methods (method with the body).

Abstraction in Java

Abstraction is a process of hiding the implementation details and showing only functionality to the user.

Another way, it shows only essential things to the user and hides the internal details, for example, sending SMS where you type the text and send the message. You don't know the internal processing about the message delivery.

- An abstract class must be declared with an abstract keyword.
- It can have abstract and non-abstract methods.
- It cannot be instantiated.
- It can have constructors and static methods also.
- It can have final methods which will force the subclass not to change the body of the method.

Example of abstract class

```
abstract class A{}
```

Abstract Method in Java

A method which is declared as abstract and does not have implementation is known as an abstract method.

Example of abstract method

```
abstract void printStatus();//no method body and abstract
abstract class Bike{
  abstract void run();
}
class Honda4 extends Bike{
void run(){System.out.println("running safely");}
public static void main(String args[]){
 Bike obj = new Honda4();
 obj.run();
}
}
```

Output:
running safely

Video Content / Details of website for further learning (if any):
https://www.javatpoint.com/abstract-class-in-java

Important Books/Journals for further learning including the page nos.:
Herbert Schildt, " Java : The Complete Reference" , Mc Graw Hill Publication, 2007, Page No: 177-180

Course Faculty

Verified by HOD

MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)

Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

Estd. 2000

L-14

LECTURE HANDOUTS

AI&DS

II/III

Course Name with Code : **OBJECT ORIENTED PROGRAMMING – 19ADC06**

Course Faculty : **T.Divya**

Unit : **II –Inheritance and Interfaces** **Date of Lecture:**

Topic of Lecture: Final methods and classes

Introduction : (Maximum 5 sentences) :

- The final specifier can provide the compiler with more opportunities for de virtualization by helping it identify more cases where virtual calls can be resolved at compile time.

Prerequisite knowledge for Complete understanding and learning of Topic:
(Max. Four important topics)

- Data
- Variable
- Data Hiding
- Super Class
- Sub Class

Detailed content of the Lecture:

A final method cannot be overridden. Which means even though a sub class can call the final method of parent class without any issues but it cannot override it.

Example:

```
class XYZ{
  final void demo(){
    System.out.println("XYZ Class Method");
  }
}

class ABC extends XYZ{
  void demo(){
    System.out.println("ABC Class Method");
  }

  public static void main(String args[]){
    ABC obj= new ABC();
    obj.demo();
  }
```

} final class

We cannot extend a final class. Consider the below example:

```
final class XYZ{
}
```

```
class ABC extends XYZ{
  void demo(){
    System.out.println("My Method");
  }
  public static void main(String args[]){
    ABC obj= new ABC();
    obj.demo();
  }
}
```

Output:

The type ABC cannot subclass the final class XYZ

Points to Remember:

1) A constructor cannot be declared as final.
2) Local final variable must be initializing during declaration.
3) All variables declared in an interface are by default final.
4) We cannot change the value of a final variable.
5) A final method cannot be overridden.
6) A final class not be inherited.
7) If method parameters are declared final then the value of these parameters cannot be changed.
8) It is a good practice to name final variable in all CAPS.
9) final, finally and finalize are three different terms. finally is used in exception handling and finalize is a method that is called by JVM during garbage collection.

Video Content / Details of website for further learning (if any):

https://beginnersbook.com/2014/07/final-keyword-java-final-variable-method-class/

Important Books/Journals for further learning including the page nos.:

Herbert Schildt, " Java : The Complete Reference" , Mc Graw Hill Publication, 2007, Page No: 180-182

Course Faculty

Verified by HOD

MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)
Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

Estd. 2000

IQAC

L-15

LECTURE HANDOUTS

AI&DS

II/III

Course Name with Code : **OBJECT ORIENTED PROGRAMMING – 19ADC06**

Course Faculty : **T.Divya**

Unit : **II –Inheritance and Interfaces** **Date of Lecture:**

Topic of Lecture: Interfaces ,defining an interface, implementing interface

Introduction : (Maximum 5 sentences) :

- An interface is a description of what member functions must a class, which inherits this interface, implement.
- Interfaces are used to implement abstraction
- A class that implements an interface must implement all the methods declared in the interface.

Prerequisite knowledge for Complete understanding and learning of Topic:
(Max. Four important topics)

- Data
- Variable
- Data Hiding
- Super Class
- Sub Class

Video Content / Details of website for further learning (if any):

- An **interface in Java** is a blueprint of a class. It has static constants and abstract methods.
- The interface in Java is *a mechanism to achieve abstraction*
- There can be only abstract methods in the Java interface, not method body. It is used to achieve abstraction and multiple inheritance in Java .
- It cannot be instantiated just like the abstract class.
- Since Java 8, we can have **default and static methods** in an interface.
- Since Java 9, we can have **private methods** in an interface.

Why use Java interface?

There are mainly three reasons to use interface. They are given below.

- It is used to achieve abstraction.
- By interface, we can support the functionality of multiple inheritance.
- It can be used to achieve loose coupling.

How to declare an interface?

An interface is declared by using the interface keyword. It provides total abstraction; means all the methods in an interface are declared with the empty body, and all the fields are public, static and final by default. A class that implements an interface must implement all the methods declared in the interface.

Syntax:

```
interface <interface_name>{

   // declare constant fields
   // declare methods that abstract
   // by default.
}
```

Example :

```
interface printable{
void print();
}
class A6 implements printable{
public void print(){System.out.println("Hello");}

public static void main(String args[]){
A6 obj = new A6();
obj.print();
 }
}
```

Video Content / Details of website for further learning (if any):
https://www.javatpoint.com/interface-in-java

Important Books/Journals for further learning including the page nos.:
Herbert Schildt, " Java : The Complete Reference" , Mc Graw Hill Publication, 2007, Page No: 192-198

Course Faculty

Verified by HOD

MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)

Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

Estd. 2000

IQAC

L-16

LECTURE HANDOUTS

AI&DS

II/III

Course Name with Code : **OBJECT ORIENTED PROGRAMMING – 19ADC06**

Course Faculty : **T.Divya**

Unit : **II –Inheritance and Interfaces** **Date of Lecture:**

Topic of Lecture: Differences between classes and interfaces and extending interfaces

Introduction : (Maximum 5 sentences) :

- A class describes the attributes and behaviors of an object.
- An interface contains behaviors that a class implements.
- A class may contain abstract methods, concrete methods.
- An interface contains only abstract methods.

Prerequisite knowledge for Complete understanding and learning of Topic:
(Max. Four important topics)

- Data
- Variable
- Data Hiding
- Super Class
- Sub Class

Video Content / Details of website for further learning (if any):

Differences between classes and interfaces and extending interfaces:

- A class is a user-defined blueprint or prototype from which objects are created.
- It represents the set of properties or methods that are common to all objects of one type.
- In general, class declarations can include these components, in order:

Modifiers : A class can be public or has default access

Class name: The name should begin with a initial letter (capitalized by convention).

Superclass:

The name of the class's parent (superclass), if any, preceded by the keyword extends.

A class can only extend (subclass) one parent.

Interfaces:

- A comma-separated list of interfaces implemented by the class, if any, preceded by the keyword implements.
- A class can implement more than one interface.

Body: The class body surrounded by braces, { }.

Interface:

- Interfaces specify what a class must do and not how. It is the blueprint of the class.
- An Interface is about capabilities like a Player may be an interface and any class implementing Player must be able to (or must implement) move().
- So it specifies a set of methods that the class has to implement.
- If a class implements an interface and does not provide method bodies for all functions specified in the interface, then class must be declared abstract.
- A Java library example is, Comparator Interface.
- If a class implements this interface, then it can be used to sort a collection.

Syntax :

```
interface <interface_name> {
   // declare constant fields
   // declare methods that abstract
   // by default.
}
```

CLASS	INTERFACE
The keyword used to create a class is “class”	The keyword used to create an interface is “interface”
A class can be instantiated i.e, objects of a class can be created.	An Inteface cannot be instantiated i.e, objects cannot be created.
Classes does not support multiple inheritance.	Inteface supports multiple inheritance.
It can be inherit another class.	It cannot inherit a class.
It can be inherited by another class using the keyword ‘extends’.	It can be inherited by a class by using the keyword ‘implements’ and it can be inherited by an interface using the keyword ‘extends’.
It can contain constructors.	It cannot contain constructors.
It cannot contain abstract methods.	It contains abstract methods only.
Variables and methods in a class can be declared using any access specifier(public, private, default, protected)	All variables and methods in a interface are declared as public.
Variables in a class can be static, final or neither.	All variables are static and final.

Video Content / Details of website for further learning (if any):
https://www.youtube.com/watch?v=J-4-UpywYOU
https://intellipaat.com/blog/tutorial/java-tutorial/abstract-class-interface/

Important Books/Journals for further learning including the page nos.:
Herbert Schildt, “ Java : The Complete Reference” , Mc Graw Hill Publication, 2007, Page No: 105-110

Course Faculty

Verified by HOD

MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)

Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

Estd. 2000

IQAC

L-17

LECTURE HANDOUTS

AI&DS

II/III

Course Name with Code : **OBJECT ORIENTED PROGRAMMING – 19ADC06**

Course Faculty : **T.Divya**

Unit : **II –Inheritance and Interfaces** **Date of Lecture:**

Topic of Lecture: Object cloning ,Inner classes

Introduction : (Maximum 5 sentences) :

- If a copy constructor is not defined in a class, the compiler itself defines one.
- If the class has pointer variables and has some dynamic memory allocations, then it is a must to have a copy constructor.
- A nested class is a class that is declared in another class.
- The nested class is also a member variable of the enclosing class and has the same access rights as the other members.

Prerequisite knowledge for Complete understanding and learning of Topic:
(Max. Four important topics)

- Data
- Variable
- Class
- Object

Video Content / Details of website for further learning (if any):

Object Cloning in Java

- The object cloning is a way to create exact copy of an object. The clone() method of Object class is used to clone an object.
- The java.lang.Cloneable interface must be implemented by the class whose object clone we want to create. If we don't implement Cloneable interface, clone() method generates CloneNotSupportedException.

The clone() method is defined in the Object class. Syntax of the clone() method is as follows:

protected Object clone() throws CloneNotSupportedException

- The clone() method saves the extra processing task for creating the exact copy of an object. If we perform it by using the new keyword, it will take a lot of processing time to be performed that is why we use object cloning.
  - You don't need to write lengthy and repetitive codes. Just use an abstract class with a 4- or 5-line long clone() method.
  - It is the easiest and most efficient way for copying objects, especially if we are applying it to an already developed or an old project. Just define a parent class, implement Cloneable in it, provide the definition of the clone() method and the task will be done.
  - Clone() is the fastest way to copy array.

- Example:

```
class Student18 implements Cloneable{
int rollno;
String name;
Student18(int rollno,String name){
this.rollno=rollno;
this.name=name;
}
public Object clone()throws CloneNotSupportedException{
return super.clone();
}
public static void main(String args[]){
try{
Student18 s1=new Student18(101,"amit");
Student18 s2=(Student18)s1.clone();
System.out.println(s1.rollno+" "+s1.name);
System.out.println(s2.rollno+" "+s2.name);
}catch(CloneNotSupportedException c){}
}
}
```

Java Inner Classes (Nested Classes)

- Java inner class or nested class is a class that is declared inside the class or interface.
- We use inner classes to logically group classes and interfaces in one place to be more readable and maintainable.
- Additionally, it can access all the members of the outer class, including private data members and methods.

Syntax of Inner class

```
class Java_Outer_class{
 //code
 class Java_Inner_class{
  //code
 }
}
```

Video Content / Details of website for further learning (if any):
https://www.javatpoint.com/java-inner-class
https://www.javatpoint.com/object-cloning

Important Books/Journals for further learning including the page nos.:
Herbert Schildt, " Java : The Complete Reference" , Mc Graw Hill Publication, 2007, Page No: 145-148

Course Faculty

Verified by HOD

MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)

Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

Estd. 2000

IQAC

L-18

LECTURE HANDOUTS

AI&DS

II/III

Course Name with Code : **OBJECT ORIENTED PROGRAMMING – 19ADC06**

Course Faculty : **T.Divya**

Unit : **II –Inheritance and Interfaces** **Date of Lecture:**

Topic of Lecture: Array Lists , Strings

Introduction : (Maximum 5 sentences) :

- C++ provides a data structure, the array, which stores a fixed-size sequential collection of elements of the same type.
- An array is used to store a collection of data, but it is often more useful to think of an array as a collection of variables of the same type

Prerequisite knowledge for Complete understanding and learning of Topic:
(Max. Four important topics)

- Data
- Variable
- Class
- Object

Video Content / Details of website for further learning (if any):

Array Lists :

- ArrayList is a part of **collection framework** and is present in java.util package. It provides us with dynamic arrays in Java. Though, it may be slower than standard arrays but can be helpful in programs where lots of manipulation in the array is needed. This class is found in **java.util** package.


```
// Java program to demonstrate the
// working of ArrayList in Java
 import java.io.*;
import java.util.*;

class ArrayListExample {
    public static void main(String[] args)
    {
        // Size of the
        // ArrayList
        int n = 5;
```

```
        // Declaring the ArrayList with
        // initial size n
        ArrayList<Integer> arrli
            = new ArrayList<Integer>(n);

        // Appending new elements at
        // the end of the list
        for (int i = 1; i <= n; i++)
            arrli.add(i);

        // Printing elements
        System.out.println(arrli);

        // Remove element at index 3
        arrli.remove(3);

        // Displaying the ArrayList
        // after deletion
        System.out.println(arrli);

        // Printing elements one by one
        for (int i = 0; i < arrli.size(); i++)
            System.out.print(arrli.get(i) + " ");
    }
}
```

Java String

In Java, string is basically an object that represents sequence of char values. An array of characters works same as Java string. For example:

```
char[] ch={'j','a','v','a','t','p','o','i','n','t'};
String s=new String(ch);
```

is same as:

```
String s="javatpoint";
```

Java String class provides a lot of methods to perform operations on strings such as compare(), concat(), equals(), split(), length(), replace(), compareTo(), intern(), substring() etc.

Video Content / Details of website for further learning (if any):

https://examples.javacodegeeks.com/arraylist-java-example/
https://www.javatpoint.com/java-string

Important Books/Journals for further learning including the page nos.:

Herbert Schildt, " Java : The Complete Reference" , Mc Graw Hill Publication, 2007, Page No: 145-148

Course Faculty

Verified by HOD

MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)
Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

Estd. 2000

IQAC

L-19

LECTURE HANDOUTS

AI&DS

II/III

Course Name with Code : **Object Oriented Programming - 19ADC06**

Course Teacher : **T.Divya**

Unit : **III - Exception Handling and I/O** **Date of Lecture:**

Topic of Lecture: Exceptions - exception hierarchy

Introduction : (Maximum 5 sentences)

- An exception is an unwanted or unexpected event, which occurs during the execution of a program i.e at run time, that disrupts the normal flow of the program's instructions.
- Exception indicates conditions that a reasonable application might try to catch.

Prerequisite knowledge for Complete understanding and learning of Topic:
(Max. Four important topics)

- OOP concept
- Java Fundamental

Detailed content of the Lecture:

An exception is an unwanted or unexpected event, which occurs during the execution of a program i.e at run time, that disrupts the normal flow of the program's instructions.

try : The "try" keyword is used to specify a block where we should place an exception code. It means we can't use try block alone. The try block must be followed by either catch or finally.

catch : The "catch" block is used to handle the exception. It must be preceded by try block which means we can't use catch block alone. It can be followed by finally block later.

finally : The "finally" block is used to execute the necessary code of the program. It is executed whether an exception is handled or not.

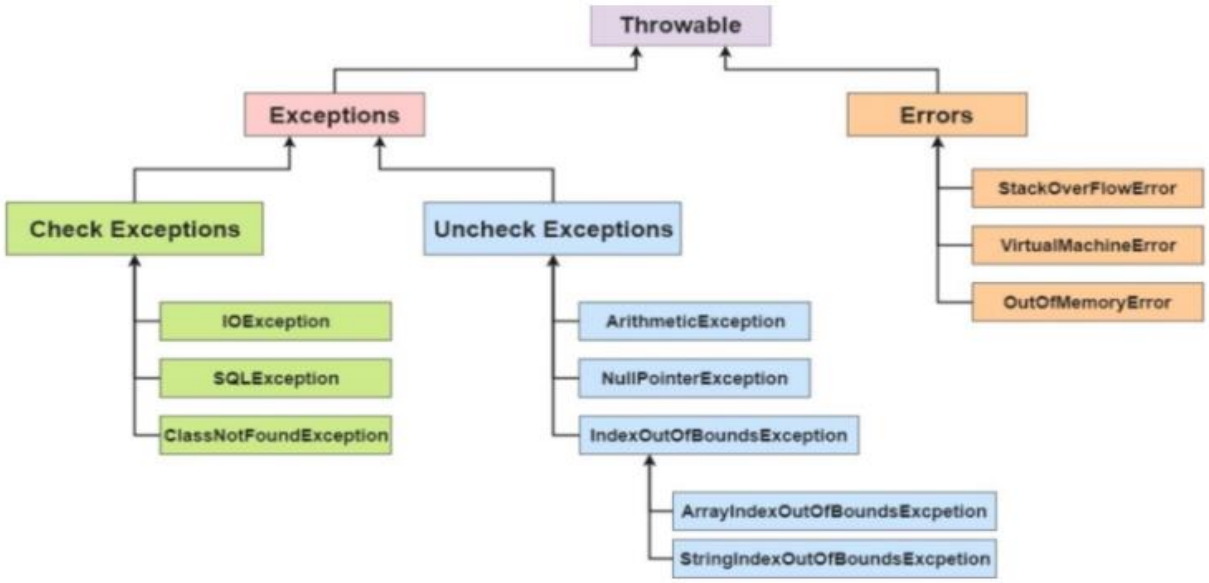
throw : The "throw" keyword is used to throw an exception.

throws : The "throws" keyword is used to declare exceptions. It specifies that there may occur an exception in the method. It doesn't throw an exception. It is always used with method signature.

```
// Example program
public class JavaExceptionExample{
public static void main(String args[]){
        try{
          //code that may raise exception
          int data=100/0;
        }catch(ArithmeticException e){System.out.println(e);}
        //rest code of the program
        System.out.println("rest of the code...");
       }
      }
```

Exception Hierarchy

- All exception and errors types are sub classes of class **Throwable**, which is base class of hierarchy.One branch is headed by **Exception**.
- This class is used for exceptional conditions that user programs should catch. **Error** are used by the Java run-time system(JVM) to indicate errors having to do with the run-time environment itself(JRE).
- StackOverflowError is an example of such an error.



Video Content / Details of website for further learning (if any):
https://www.geeksforgeeks.org/exceptions-in-java/

Important Books/Journals for further learning including the page nos.:

Book: Herbert Schildt, " Java : The Complete Reference" , Mc Graw Hill Publication, 2007, Page No: 205-212

Course Teacher

Verified by HOD

MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)
Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

Estd. 2000

IQAC

L-20

LECTURE HANDOUTS

AI&DS

II/III

Course Name with Code : **Object Oriented Programming – 19ADC06**

Course Teacher : **T.Divya**

Unit : **III – Exception Handling and I/O** **Date of Lecture:**

Topic of Lecture: Throwing and catching exceptions

Introduction : (Maximum 5 sentences)

- An exception is an "unwanted or unexpected event", which occurs during the execution of the program i.e, at run-time, that disrupts the normal flow of the program's instructions.
- When an exception occurs, execution of the program gets terminated.

Prerequisite knowledge for Complete understanding and learning of Topic:
(Max. Four important topics)

- Error
- Exception Hierarchy

Detailed content of the Lecture:

- An exception can occur due to several reasons like Network connection problem, Bad input provided by user, Opening a non-existing file in your program etc

Blocks & Keywords used for exception handling

1.try: The try block contains set of statements where an exception can occur.

```
try
{
   // statement(s) that might cause exception
}
```

2.**catch** : Catch block is used to handle the uncertain condition of try block. A try block is always followed by a catch block, which handles the exception that occurs in associated try block.

```
catch
{
  // statement(s) that handle an exception
  // examples, closing a connection, closing
  // file, exiting the process after writing
  // details to a log file.
}
```

3.**throw**: Throw keyword is used to transfer control from try block to catch block.

4.**throws**: Throws keyword is used for exception handling without try & catch block. It specifies the exceptions that a method can throw to the caller and does not handle itself.

5.**finally**: It is executed after catch block. We basically use it to put some common code when there are multiple catch blocks.

```
// Java program to demonstrate working of throws
class ThrowsExecp {

    // This method throws an exception
    // to be handled
    // by caller or caller
    // of caller and so on.
    static void fun() throws IllegalAccessException
    {
        System.out.println("Inside fun(). ");
        throw new IllegalAccessException("demo");
    }

    // This is a caller function
    public static void main(String args[])
    {
        try {
            fun();
        }
        catch (IllegalAccessException e) {
            System.out.println("caught in main.");
        }
    }
}
```

Video Content / Details of website for further learning (if any):
https://www.geeksforgeeks.org/try-catch-throw-and-throws-in-java/

Important Books/Journals for further learning including the page nos.:

Book: Herbert Schildt, " Java : The Complete Reference" , Mc Graw Hill Publication, 2007, Page No: 205-220.

Course Teacher

Verified by HOD

MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)
Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

Estd. 2000

IQAC

L-21

LECTURE HANDOUTS

AI&DS

II/III

Course Name with Code : **Object Oriented Programming – 19ADC06**

Course Teacher : **T.Divya**

Unit :**III - Exception Handling and I/O Date of Lecture:**

Topic of Lecture: Built-in exceptions

Introduction : (Maximum 5 sentences)

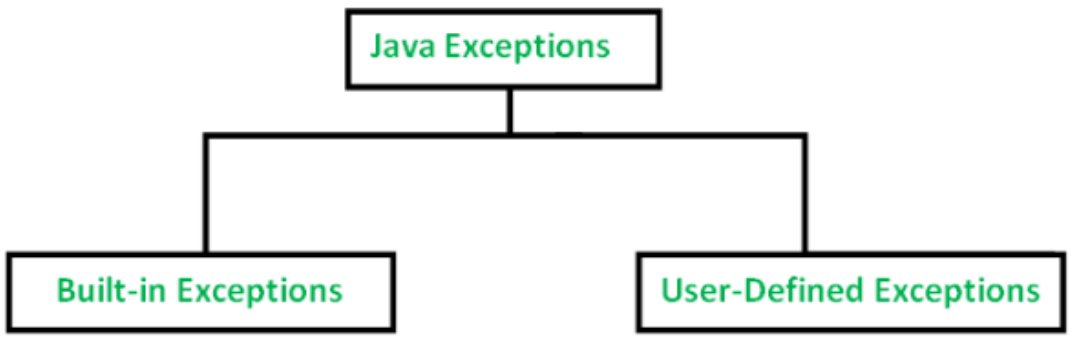
- An exception is an "unwanted or unexpected event", which occurs during the execution of the program i.e, at run-time, that disrupts the normal flow of the program's instructions.
- Built-in exceptions are the exceptions which are available in Java libraries. These exceptions are suitable to explain certain error situations.

Prerequisite knowledge for Complete understanding and learning of Topic:
(Max. Four important topics)

- Exception
- Error

Detailed content of the Lecture:

- Built-in exceptions are the exceptions which are available in Java libraries. These exceptions are suitable to explain certain error situations.




```
// Java program to demonstrate ArithmeticException
class ArithmeticException_Demo
{
   public static void main(String args[])
   {
      try {
         int a = 30, b = 0;
         int c = a/b;  // cannot divide by zero
         System.out.println ("Result = " + c);
      }
```

```
        catch(ArithmeticException e) {
           System.out.println ("Can't divide a number by 0");
        }
    }
}
```

Output:
Can't divide a number by 0

Built in Exception:

Exception	Meaning
ArithmeticException	Arithmetic error, such as divide-by-zero.
ArrayIndexOutOfBoundsException	Array index is out-of-bounds.
ArrayStoreException	Assignment to an array element of an incompatible type.
ClassCastException	Invalid cast.
EnumConstantNotPresentException	An attempt is made to use an undefined enumeration value.
IllegalArgumentException	Illegal argument used to invoke a method.
IllegalMonitorStateException	Illegal monitor operation, such as waiting on an unlocked thread.
IllegalStateException	Environment or application is in incorrect state.
IllegalThreadStateException	Requested operation not compatible with current thread state.
IndexOutOfBoundsException	Some type of index is out-of-bounds.
NegativeArraySizeException	Array created with a negative size.
NullPointerException	Invalid use of a null reference.
NumberFormatException	Invalid conversion of a string to a numeric format.
SecurityException	Attempt to violate security.
StringIndexOutOfBounds	Attempt to index outside the bounds of a string.
TypeNotPresentException	Type not found.
UnsupportedOperationException	An unsupported operation was encountered.

Video Content / Details of website for further learning (if any):
https://www.geeksforgeeks.org/types-of-exception-in-java-with-examples/

Important Books/Journals for further learning including the page nos.:

Book: Herbert Schildt, " Java : The Complete Reference" , Mc Graw Hill Publication, 2007, Page No: 205-220.

Course Teacher

Verified by HOD

MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)
Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

Estd. 2000

IQAC

L-22

LECTURE HANDOUTS

AI&DS

II/III

Course Name with Code : **Object Oriented Programming – 19ADC06**

Course Teacher : **T.Divya**

Unit :**III – Exception Handling and I/O** **Date of Lecture:**

Topic of Lecture: Creating own exceptions

Introduction : (Maximum 5 sentences)

- An exception is an "unwanted or unexpected event", which occurs during the execution of the program i.e, at run-time, that disrupts the normal flow of the program's instructions.
- Creating our own Exception is known as custom exception or user-defined exception.

Prerequisite knowledge for Complete understanding and learning of Topic:
(Max. Four important topics)

- Exception
- Error
- Built in Exception

Detailed content of the Lecture:

- Creating our own Exception is known as custom exception or user-defined exception.
- Following are few of the reasons to use custom exceptions:
  - To catch and provide specific treatment to a subset of existing Java exceptions.
  - Business logic exceptions: These are the exceptions related to business logic and workflow. It is useful for the application users or the developers to understand the exact problem.

// A Class that represents use-defined expception

```
class MyException extends Exception
{
    public MyException(String s)
    {
        // Call constructor of parent Exception
        super(s);
    }
}
```

// A Class that uses above MyException

```
public class setText
{
    // Driver Program
    public static void main(String args[])
    {
        try
```

```
        {
            // Throw an object of user defined exception
            throw new MyException();
        }
        catch (MyException ex)
        {
            System.out.println("Caught");
            System.out.println(ex.getMessage());
        }
    }
}
```

Video Content / Details of website for further learning (if any):
https://www.geeksforgeeks.org/g-fact-32-user-defined-custom-exception-in-java/

Important Books/Journals for further learning including the page nos.:

Book: Herbert Schildt, " Java : The Complete Reference" , Mc Graw Hill Publication, 2007, Page No: 205-220.

Course Teacher

Verified by HOD

MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)
Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

Estd. 2000

IQAC

L-23

LECTURE HANDOUTS

AI&DS

II/III

Course Name with Code : **Object Oriented Programming – 19ADC06**

Course Teacher : **T.Divya**

Unit :**III – Exception Handling and I/O** **Date of Lecture:**

Topic of Lecture: Stack Trace Elements

Introduction : (Maximum 5 sentences)

- In Java, stack trace is nothing but location of the exceptions. In other words, we can say that stack trace hunt (trace) for the next line where exception may raise.
- The stack frames represent the movement of an application during the execution of the program. It traces the locations where exception raised.

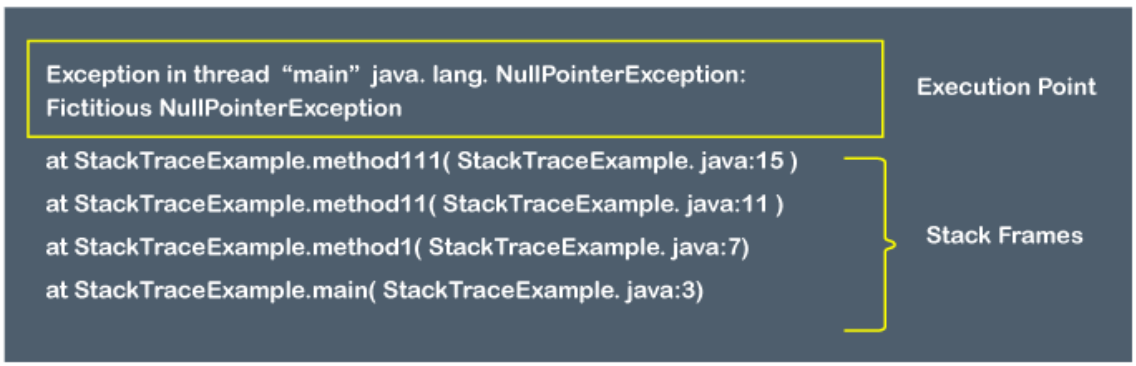
Prerequisite knowledge for Complete understanding and learning of Topic:
(Max. Four important topics)

- Exception
- Error
- Built in Exception

Detailed content of the Lecture:

- In Java, the stack trace is an array of **stack frames**. It is also known as **stack backtrace** (or backtrace). The stack frames represent the movement of an application during the execution of the program. It traces the locations where exception raised.
- It collects the information of all the methods that are invoked by a program. When we do not care about the unhandled exceptions and the program throws the exceptions then Java stack trace prints the stack trace on the console by default.
- The JVM automatically produces the stack trace when an exception is thrown. In stack trace, each element represents a method invocation.

Java Stack Trace



- After introducing Java 1.5, the stack trace is encapsulated into an array of a Java class called **StackTraceElement**. The array returned by the **getStackTrace()** method of the Throwable class.
- Each element is represented by a single stack frame. All stack frames denote the method invocation except for the first frame (at the top). The first frame denotes the execution point on which JVM generates the stack trace.
- The StackTraceElement class provides a constructor that parses four parameters as an argument and creates a stack trace element that denotes the specified execution point.

public StackTraceElement(String declaringClass, String methodName, String fileName, **int** lineNumber

Parameters:

- **declaringClass:** the qualified name of the class that contains the execution point.
- **methodName:** It represents the method name that contains the execution point.
- **fileName:** It represents the file name that contains the execution point.
- **lineNumber:** It represents the line number of the source of the execution point.

It throws the **NullPointerException** if the parameters **declaringClass** and **methodName** are null.

Syntax of the stack trace. :

Exception in thread "main" java.lang.NullPointerException: Fictitious NullPointerException
at ClassName.methodName1(ClassName.java:lineNumber)
at ClassName.methodName2(ClassName.java:lineNumber)
at ClassName.methodName3(ClassName.java:lineNumber)
at ClassName.main(ClassName.java:lineNumber)

Video Content / Details of website for further learning (if any):
https://www.javatpoint.com/java-stack-trace

Important Books/Journals for further learning including the page nos.:

Book: Herbert Schildt, " Java : The Complete Reference" , Mc Graw Hill Publication, 2007, Page No: 205-220.

Course Teacher

Verified by HOD

MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)
Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

Estd. 2000

IQAC

L-24

LECTURE HANDOUTS

AI&DS

II/III

Course Name with Code : **Object Oriented Programming – 19ADC06**

Course Teacher : **T.Divya**

Unit :**III – Exception Handling and I/O** **Date of Lecture:**

Topic of Lecture: Input / Output Basics – Streams

Introduction : (Maximum 5 sentences)

- **Java I/O** (Input and Output) is used *to* process the input *and* produce the output.
- Java uses the concept of a stream to make I/O operation fast. The java.io package contains all the classes required for input and output operations.

Prerequisite knowledge for Complete understanding and learning of Topic:
(Max. Four important topics)

- Java Fundamentals
- Input
- Output

Detailed content of the Lecture:

• In Java, 3 streams are created for us automatically. All these streams are attached with the console.

- **Standard Input** − This is used to feed the data to user's program and usually a keyboard is used as standard input stream and represented as **System.in**.
- **Standard Output** − This is used to output the data produced by the user's program and usually a computer screen is used for standard output stream and represented as **System.out**.
- **Standard Error** − This is used to output the error data produced by the user's program and usually a computer screen is used for standard error stream and represented as **System.err**.

Code to print output and an error **message to the console:**

```
System.out.println("simple message");
System.err.println("error message");
```

Code to get input **from console:**

```
int i=System.in.read();//returns ASCII code of 1st character
System.out.println((char)i);//will print the character
```

OutputStream

Java application uses an output stream to write data to a destination; it may be a file, an array,

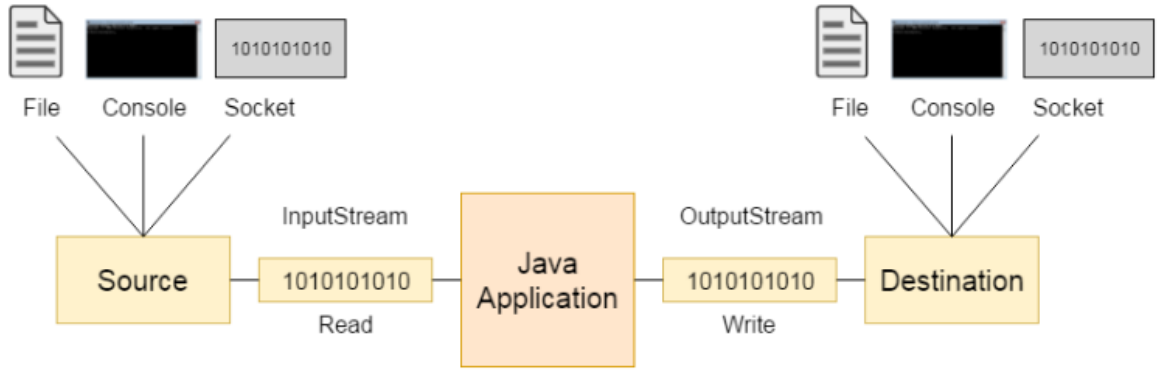
peripheral device or socket.

Methods of OutputStream:

Method	Description
1) public void write(int)throws IOException	is used to write a byte to the current output stream.
2) public void write(byte[])throws IOException	is used to write an array of byte to the current output stream.
3) public void flush()throws IOException	flushes the current output stream.
4) public void close()throws IOException	is used to close the current output stream.

InputStream

Java application uses an input stream to read data from a source; it may be a file, an array, peripheral device or socket.



Methods of InputStream:

Method	Description
1) public abstract int read()throws IOException	reads the next byte of data from the input stream. It returns -1 at the end of the file.
2) public int available()throws IOException	returns an estimate of the number of bytes that can be read from the current input stream.
3) public void close()throws IOException	is used to close the current input stream.

Video Content / Details of website for further learning (if any):
https://www.javatpoint.com/java-io

Important Books/Journals for further learning including the page nos.:

Book: Herbert Schildt, " Java : The Complete Reference" , Mc Graw Hill Publication, 2007, Page No: 285-294

Course Teacher

Verified by HOD

MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)
Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

Estd. 2000

IQAC

L-25

LECTURE HANDOUTS

AI&DS

II/III

Course Name with Code : **Object Oriented Programming – 19ADC06**

Course Teacher : **T.Divya**

Unit :**III – Exception Handling and I/O** **Date of Lecture:**

Topic of Lecture: Byte streams and Character streams

Introduction : (Maximum 5 sentences)

- A stream can be defined as a sequence of data.
- The java.io package contains nearly every class you might ever need to perform input and output (I/O) in Java. All these streams represent an input source and an output destination.
- The stream in the java.io package supports many data such as primitives, object, localized characters, etc.

Prerequisite knowledge for Complete understanding and learning of Topic:
(Max. Four important topics)

- Java Fundamentals
- Input
- Output

Detailed content of the Lecture:

- A stream can be defined as a sequence of data. There are two kinds of Streams −
- **InPutStream** − The InputStream is used to read data from a source.
- **OutPutStream** − The OutputStream is used for writing data to a destination.
- Java provides strong but flexible support for I/O related to files and networks but this tutorial covers very basic functionality related to streams and I/O.

Byte Streams :

- Java byte streams are used to perform input and output of 8-bit bytes. Though there are many classes related to byte streams but the most frequently used classes are, **FileInputStream** and **FileOutputStream**.

// Example Program
import java.io.*;

```
public class CopyFile {
  public static void main(String args[]) throws IOException {
    FileInputStream in = null;
    FileOutputStream out = null;
    try {
      in = new FileInputStream("input.txt");
      out = new FileOutputStream("output.txt");
     int c;
      while ((c = in.read()) != -1) {
        out.write(c);
      }
    }finally {
      if (in != null) {
```

```
        in.close();
      }
      if (out != null) {
        out.close();
      }
    }
  }
```

}

input.txt with the following content

This is test for copy file.

Character Streams:

- Java **Byte** streams are used to perform input and output of 8-bit bytes, whereas Java **Character** streams are used to perform input and output for 16-bit unicode. Though there are many classes related to character streams but the most frequently used classes are, **FileReader** and **FileWriter**.
- Though internally FileReader uses FileInputStream and FileWriter uses FileOutputStream but here the major difference is that FileReader reads two bytes at a time and FileWriter writes two bytes at a time.

```
import java.io.*;
public class CopyFile {
  public static void main(String args[]) throws IOException {
    FileReader in = null;
    FileWriter out = null;
    try {
      in = new FileReader("input.txt");
      out = new FileWriter("output.txt");
     int c;
      while ((c = in.read()) != -1) {
        out.write(c);
      }
    }finally {
      if (in != null) {
        in.close();
      }
      if (out != null) {
        out.close();
      }
    }
  }
}
```

Video Content / Details of website for further learning (if any):

https://www.youtube.com/watch?v=kD_HqZP8MLY

Important Books/Journals for further learning including the page nos.:

Book: Herbert Schildt, " Java : The Complete Reference" , Mc Graw Hill Publication, 2007, Page No: 285-294

Course Teacher

Verified by HOD

MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)
Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

L-26

LECTURE HANDOUTS

AI&DS

II/III

Course Name with Code : **Object Oriented Programming – 19ADC06**

Course Teacher : **T.Divya**

Unit :**III – Exception Handling and I/O** **Date of Lecture:**

Topic of Lecture: Reading and Writing Console

Introduction : (Maximum 5 sentences)

- A console is generally connected with Java processes which are started using the command-line tool.
- If the Java process has started automatically, (for example a background task), the console may not be available for input and output purposes.

Prerequisite knowledge for Complete understanding and learning of Topic:
(Max. Four important topics)

- Java Fundamentals
- Input / Output
- Stream

Detailed content of the Lecture:

- A console is generally connected with Java processes which are started using the command-line tool. If the Java process has started automatically, (for example a background task), the console may not be available for input and output purposes.

1. Java read input from console

By default, to read from system console, we can use the Console class. This class provides methods to access the character-based console, if any, associated with the current Java process. To get access to Console, call the method **System.console()**.
Console gives three ways to read the input:

- String readLine() – reads a single line of text from the console.
- char[ ] readPassword() – reads a password or encrypted text from the console with echoing disabled
- Reader reader() – retrieves the Reader object associated with this console. This reader is supposed to be used by sophisticated applications.

Java program to read console input with readLine()

```
Console console = System.console();

if(console == null) {
   System.out.println("Console is not available to current JVM process");
```

```
    return;
}

String userName = console.readLine("Enter the username: ");
System.out.println("Entered username: " + userName);
```
Program output

Console
Enter the username: lokesh
Entered username: lokesh

Java print output to console

The easiest way to write the output data to console is System.out.println() statements. Still, we can use printf() methods to write formatted text to console.

Java program to write to console with System.out.println

System.out.println() method example
```
System.out.println("Hello, world!");
```
Program output

Console
Hello, world!

Java program to write to console with printf()

The printf(String format, Object... args) method takes an output string and multiple parameters which are substituted in the given string to produce the formatted output content. This formatted output is written in the console.

Console printf() method example
```
String name = "Lokesh";
int age = 38;

console.printf("My name is %s and my age is %d", name, age);
```
Program output

Console
My name is Lokesh and my age is 38

Video Content / Details of website for further learning (if any):
https://howtodoinjava.com/java-examples/console-input-output/

Important Books/Journals for further learning including the page nos.:

Book: Herbert Schildt, " Java : The Complete Reference" , Mc Graw Hill Publication, 2007, Page No: 288-293

Course Teacher

Verified by HOD

MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)
Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

L-27

LECTURE HANDOUTS

AI&DS

II/III

Course Name with Code : **Object Oriented Programming – 19ADC06**

Course Teacher : **T.Divya**

Unit :**III – Exception Handling and I/O** **Date of Lecture:**

Topic of Lecture: Reading and Writing Files

Introduction : (Maximum 5 sentences)

- File handling is an important part of any application.
- Java has several methods for creating, reading, updating, and deleting files.
- The File class from the java.io package, allows us to work with files.
- To use the File class, create an object of the class, and specify the filename or directory name

Prerequisite knowledge for Complete understanding and learning of Topic:
(Max. Four important topics)

- Java Fundamentals
- Input / Output
- Stream

Detailed content of the Lecture:

Write To a File :

- In the following example, we use the FileWriter class together with its write() method to write some text to the file we created in the example above. Note that when you are done writing to the file, you should close it with the close() method

```
import java.io.FileWriter;   // Import the FileWriter class
import java.io.IOException;  // Import the IOException class to handle errors

public class WriteToFile {
  public static void main(String[] args) {
    try {
      FileWriter myWriter = new FileWriter("filename.txt");
      myWriter.write("Files in Java might be tricky, but it is fun enough!");
      myWriter.close();
      System.out.println("Successfully wrote to the file.");
    } catch (IOException e) {
      System.out.println("An error occurred.");
      e.printStackTrace();
    }
  }
}
```

The output will be:

```
Successfully wrote to the file.
```

Read a File :

In the previous chapter, you learned how to create and write to a file.

In the following example, we use the Scanner class to read the contents of the text file we created in the previous chapter:

Example

```
import java.io.File;  // Import the File class
import java.io.FileNotFoundException;  // Import this class to handle errors
import java.util.Scanner; // Import the Scanner class to read text files

public class ReadFile {
  public static void main(String[] args) {
    try {
      File myObj = new File("filename.txt");
      Scanner myReader = new Scanner(myObj);
      while (myReader.hasNextLine()) {
        String data = myReader.nextLine();
        System.out.println(data);
      }
      myReader.close();
    } catch (FileNotFoundException e) {
      System.out.println("An error occurred.");
      e.printStackTrace();
    }
  }
}
```

The output will be:

Files in Java might be tricky, but it is fun enough!

Video Content / Details of website for further learning (if any):

https://www.youtube.com/watch?v=SslMi6ptwH8

Important Books/Journals for further learning including the page nos.:

Book: Herbert Schildt, " Java : The Complete Reference" , Mc Graw Hill Publication, 2007, Page No: 293-296

Course Teacher

Verified by HOD

MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)

Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

Estd. 2000

LECTURE HANDOUTS

L-28

AI&DS

II/III

Course Name with Code : **Object Oriented Programming-19ADC06**

Course Faculty : **T.Divya**

Unit : **IV - Multithreading and Generic Programming**

Date of Lecture:

Topic of Lecture: Differences between multi-threading and multitasking

Introduction : (Maximum 5 sentences) :

- Multithreading is different from multitasking in a sense that multitasking allows multiple tasks at the same time, whereas, the Multithreading allows multiple threads of a single task (program, process) to be processed by CPU at the same time.
- Multitasking is when a single CPU performs several tasks (program, process, task, threads) at the same time. To perform multitasking, the CPU switches among these tasks very frequently so that user can interact with each program simultaneously.

Prerequisite knowledge for Complete understanding and learning of Topic:
(Max. Four important topics)

- Input / Output Basics
- Reading and Writing Console
- Reading and Writing Files

Detailed content of the Lecture:

In programming, there are two main ways to improve the throughput of a program:

i) by using multi-threading

ii) by using multitasking

Both these methods take advantage of parallelism to efficiently utilize the power of CPU and improve the throughput of program.

Difference between multithreading and multi-tasking

1. The basic difference between multitasking and multithreading is that in multitasking, the system allows executing multiple programs and tasks at the same time, whereas, in multithreading, the system executes multiple threads of the same or different processes at the same time.

2. Multi-threading is more granular than multi-tasking. In multi-tasking, CPU switches between multiple programs to complete their execution in real time, while in multithreading CPU switches between multiple threads of the same program. Switching between multiple processes has more context switching cost than switching between multiple threads of the same program.

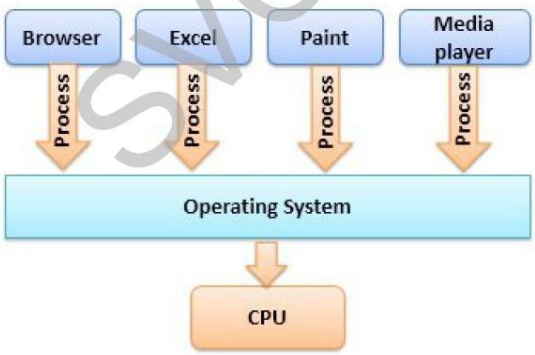
3. Processes are heavyweight as compared to threads. They require their own address space, which means multi-tasking is heavy compared to multithreading.

4. Multitasking allocates separate memory and resources for each process/program whereas, in multithreading threads belonging to the same process shares the same memory and resources as that of the process.

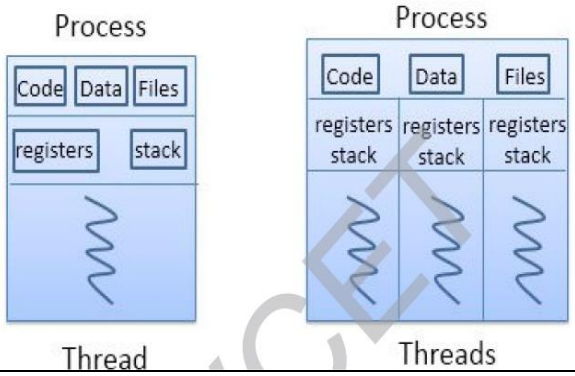
Comparison between multithreading and multi-tasking

Parameter	Multi Tasking	Multi Threading
Basic	Multitasking lets CPU to execute multiple tasks at the same time.	Multithreading lets CPU to execute multiple threads of a process simultaneously
Switching	In multitasking, CPU switches between programs frequently.	In multithreading, CPU switches between the threads frequently.
Memory and Resource	In multitasking, system has to allocate separate memory and resources to each program that CPU is executing	In multithreading, system has to allocate memory to a process, multiple threads of that process shares the same memory and resources allocated to the process.

Multitasking - In a multitasking operating system, several users can share the system simultaneously. CPU rapidly switches among the tasks, so a little time is needed to switch from one user to the next user. This puts an impression on a user that entire computer system is dedicated to him.



Multithreading- A thread is a basic execution unit which has its own program counter, set of the register and stack. But it shares the code, data, and file of the process to which it belongs. A process can have multiple threads simultaneously, and the CPU switches among these threads so frequently making an impression on the user that all threads are running simultaneously.



Video Content / Details of website for further learning (if any):
https://www.youtube.com/watch?v=L95658yXRgI
https://www.youtube.com/watch?v=Xj1uYKa8rIw

Important Books/Journals for further learning including the page nos.:
Herbert Schildt, "Java The complete reference", 8th EditionMcGraw Hill Education2011,Page No:227-258

Course Faculty

Verified by HOD

MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)
Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

Estd. 2000

IQAC

LECTURE HANDOUTS | **L-29**

AI&DS | **II/III**

Course Name with Code : **Object Oriented Programming-19ADC06**

Course Faculty : **T.Divya**

Unit : **IV-Multithreading and Generic Programming**

Date of Lecture:

Topic of Lecture: Thread life cycle, creating threads

Introduction : (Maximum 5 sentences) :

- A thread in Java at any point of time exists in any one of the following states

Prerequisite knowledge for Complete understanding and learning of Topic:
(Max. Four important topics)

- Multitasking
- Multithreading

Detailed content of the Lecture:

A thread lies only in one of the shown states at any instant:

1) New
2) Runnable
3) Blocked
4) Waiting
5) Timed Waiting
6) Terminated

The following figure represents various states of a thread at any instant of time:

- **New Thread:**
  - When a new thread is created, it is in the new state.
  - The thread has not yet started to run when thread is in this state.
  - When a thread lies in the new state, it's code is yet to be run and hasn't started to execute.
- **Runnable State:**
  - A thread that is ready to run is moved to runnable state.
  - In this state, a thread might actually be running or it might be ready run at any instant of time.
- It is the responsibility of the thread scheduler to give the thread, time to run.
- Blocked/Waiting state:
  - When a thread is temporarily inactive, then it's in one of the following states:
  - Blocked

- Waiting
    - For example, when a thread is waiting for I/O to complete, it lies in the blocked state. It's the responsibility of the thread scheduler to reactivate and schedule a blocked/waiting thread.
    - A thread in this state cannot continue its execution any further until it is moved to runnable state. Any thread in these states do not consume any CPU cycle.
- A thread is in the blocked state when it tries to access a protected section of code that is
- currently locked by some other thread. When the protected section is unlocked, the schedule picks one of the threads which is blocked for that section and moves it to the runnable state. A thread is in the waiting state when it waits for another thread on a condition. When this condition is fulfilled, the scheduler is notified
- Timed Waiting:
    - A thread lies in timed waiting state when it calls a method with a time out parameter.
    - A thread lies in this state until the timeout is completed or until a notification is received.
    - For example, when a thread calls sleep or a conditional wait, it is moved to timed waiting state.
- Terminated State:
    - A thread terminates because of either of the following reasons:
- Because it exits normally. This happens when the code of thread has entirely executed by the program.
- Because there occurred some unusual erroneous event, like segmentation fault
- or an unhandled exception.
    - A thread that lies in terminated state does no longer consumes any cycles of CPU.

There are two ways to create a thread:

1) By extending Thread class. 2) By implementing Runnable interface.

Java Thread Benefits

1. Java Threads are lightweight compared to processes as they take less time and resource to create a thread.
2. Threads share their parent process data and code
3. Context switching between threads is usually less expensive than between processes.
4. Thread intercommunication is relatively easy than process communication.

Thread class:

Thread class provide constructors and methods to create and perform operations on a thread. Thread class extends Object class and implements Runnable interface.

Video Content / Details of website for further learning (if any):

https://www.youtube.com/watch?v=TCd8QIS-2KI
https://www.youtube.com/watch?v=b5sj13Z7aho

Important Books/Journals for further learning including the page nos.:

Herbert Schildt, "Java The complete reference", 8th EditionMcGraw Hill Education2011.Page No: 227-258

Course Faculty

Verified by HOD

MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)

Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

LECTURE HANDOUTS

L-30

AI&DS

II/III

Course Name with Code : **Object Oriented Programming-19ADC06**

Course Faculty : **T.Divya**

Unit : **IV-Multithreading and Generic Programming**

Date of Lecture:

Topic of Lecture: Synchronizing threads

Introduction : (Maximum 5 sentences) :

- Threading is a facility to allow multiple tasks to run concurrently within a single process. Threads are independent, concurrent execution through a program, and each thread has its own stack.
- Synchronization in java is the capability to control the access of multiple threads toany shared resource.

Prerequisite knowledge for Complete understanding and learning of Topic:
(Max. Four important topics)

- Multithreading
- Thread Life Cycle

Detailed content of the Lecture:

Naming Thread

- The Thread class provides methods to change and get the name of a thread.
- By default,each thread has a name i.e. thread-0, thread-1 and so on.
- But we can change the name of thethread by using setName() method.
- The syntax of setName() and getName() methods aregiven below:
- public String getName(): is used to return the name of a thread.
- public void setName(String name): is used to change the name of a thread.

Extending Thread

- The first way to create a thread is to create a new class that extends Thread, and then to create an instance of that class.
- The extending class must override the run( ) method, which is the entry point for the new thread. It must also call start( ) to begin execution of the new thread.

Synchronization

- Synchronization in java is the capability to control the access of multiple threads to any shared resource.
  - Java Synchronization is better option where we want to allow only one thread to access the shared resource.
  - When two or more threads need access to a shared resource, they need some way to ensure that the resource will be used by only one thread at a time.

- The process by which this is achieved is called synchronization.

- Java provides unique, language level support for it.
- Key to synchronization is the concept of the monitor (also called a semaphore).
- A monitor is an object that is used as a mutually exclusive lock, or mutex.
- Only one thread can own a monitor at a given time.
- When a thread acquires a lock, it is said to have entered the monitor.
- All other threads attempting to enter the locked monitor will be suspended until the first thread exits the monitor.
- These other threads are said to be waiting for the monitor.
- A thread that owns a monitorcan reenter the same monitor if it so desires.

- Approaches:
- Using synchronized Method
- Using synchronized Statement

Video Content / Details of website for further learning (if any):
https://www.youtube.com/watch?v=TCd8QIS-2KI
https://www.youtube.com/watch?v=b5sj13Z7aho
https://www.youtube.com/watch?v=IIgHG_YHXPE

Important Books/Journals for further learning including the page nos.:
Herbert Schildt, "Java The complete reference", 8th EditionMcGraw Hill Education2011.Page No: 227-258

Course Faculty

Verified by HOD

MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)
Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

Estd. 2000

IQAC

LECTURE HANDOUTS | **L-31**

AI&DS | **II/III**

Course Name with Code : **Object Oriented Programming-19ADC06**

Course Faculty : **T.Divya**

Unit : **IV-Multithreading and Generic Programming**

Date of Lecture:

Topic of Lecture: Inter-thread communication

Introduction : (Maximum 5 sentences) :

- Inter-process communication (IPC) is a mechanism that allows the exchange of data between Processes.
- By providing a user with a set of programming interfaces, IPC helps a programmer organize the activities among different processes.

Prerequisite knowledge for Complete understanding and learning of Topic:
(Max. Four important topics)

- Thread Life Cycle
- Multi threading

Detailed content of the Lecture:

- IPC enables data communication by allowing processes to use segments, semaphores, and other methods to share memory and information. IPC facilitates efficient message transferbetween processes.

The idea of IPC is based on Task Control Architecture (TCA). It is a flexibletechnique that can send and receive variable length arrays, data structures, and lists.

It is implemented by following methods of Object class:

• wait() • notify() • notifyAll()

All these methods belong to object class as final so that all classes have them.They must be used within a synchronized block only.

1) wait() method

Causes current thread to release the lock and wait until either another thread invokes the notify() method or the notifyAll() method for this object, or a specified amount of time has elapsed. The current thread must own this object's monitor, so it must be called from the synchronized method only otherwise it will throw exception.

2) notify() method

Wakes up a single thread that is waiting on this object's monitor. If any threads are waiting on this object, one of them is chosen to be awakened. The choice is arbitrary and occurs at the discretion of the implementation. Syntax:

public final void notify()

3) notifyAll() method

Wakes up all threads that are waiting on this object's monitor. Syntax:

```
public final void notifyAll()
// Java program to demonstrate inter-thread communication (wait(), join() and notify()) in Java
import java.util.Scanner;
public class Thread_Example
{     public static void main(String[] args) throws InterruptedException
{     final Producer_Consumer pc = new Producer_Consumer ();
Thread t1 = new Thread(new Runnable()
{      public void run()
{      try
{     pc.producer();    }
catch(InterruptedException e)
{    e.printStackTrace();
} } });
Thread t2 = new Thread(new Runnable()
{  public void run()
{   try
{    pc.consumer();  }
catch(InterruptedException e)
{ e.printStackTrace();
}   }   });
t1.start();
t2.start();
t1.join();
t2.join();     }
public static class Producer_Consumer
{   public void producer()throws InterruptedException
{   synchronized(this)
{   System.out.println("producer thread running");
wait();
System.out.println("Resumed");      } }
public void consumer()throws InterruptedException
{ Thread.sleep(1000);
Scanner ip = new Scanner(System.in);
synchronized(this)
{ System.out.println("Waiting for return key.");
ip.nextLine();
System.out.println("Return key pressed");
notify();
Thread.sleep(1000);
} } } }
```

Video Content / Details of website for further learning (if any):
https://www.youtube.com/watch?v=TCd8QIS-2KI
https://www.youtube.com/watch?v=b5sj13Z7aho

Important Books/Journals for further learning including the page nos.:
Herbert Schildt, "Java The complete reference", 8th EditionMcGraw Hill Education2011.Page No: 227-258

Course Faculty

Verified by HOD

MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)
Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

Estd. 2000

IQAC

LECTURE HANDOUTS | **L-32**

AI&DS | **II/III**

Course Name with Code : **Object Oriented Programming-19ADC06**

Course Faculty : **T.Divya**

Unit : **IV-Multithreading and Generic Programming**

Date of Lecture:

Topic of Lecture: Daemon threads & Thread groups

Introduction : (Maximum 5 sentences) :

- Daemon thread in java is a service provider thread that provides services to the user thread.

Prerequisite knowledge for Complete understanding and learning of Topic:
(Max. Four important topics)

- Synchronize Threads
- Inter – Thread Communication

Detailed content of the Lecture:

- Daemon thread is a low priority thread that runs in background to perform tasks such as garbage collection.
- Its life depend on the mercy of user threads i.e. when all the user threadsdies, JVM terminates this thread automatically.
- There are many java daemon threads running automatically e.g. gc, finalize etc.
- It provides services to user threads for background supporting tasks. It has no role in life than to serve user threads.
- Its life depends on user threads.
- It is a low priority thread.

Properties:

• They cannot prevent the JVM from exiting when all the user threads finish their execution.

• JVM terminates itself when all user threads finish their execution

• If JVM finds running daemon thread, it terminates the thread and after that shutdown itself.

JVM does not care whether Daemon thread is running or not.

• It is an utmost low priority thread.

Methods for Java Daemon thread by Thread class

The java.lang.Thread class provides two methods for java daemon thread.

Method	Description
public void setDaemon(boolean status)	used to mark the current thread as daemon thread or user thread.
public boolean isDaemon()	used to check that current is daemon.

Daemon vs User Threads

Priority:

- When the only remaining threads in a process are daemon threads, the interpreter exits.
- This makes sense because when only daemon threads remain, there is no other thread for which a daemon thread can provide a service.

Usage: Daemon thread is to provide services to user thread for background supporting task.

Thread Group in Java

- Java provides a convenient way to group multiple threads in a single object.
- In such way, we can suspend, resume or interrupt group of threads by a single method call. ThreadGroup creates a group of threads.
- It offers a convenient way to manage groups of threads as a unit.
- This is particularly valuable in situation in which you want to suspend and resume a number of related threads.
- The thread group form a tree in which every thread group except the initial thread group has a parent
- A thread is allowed to access information about its own thread group but not to access information about its thread group's parent thread group or any other thread group.

CONSTRUCTORS OF THREADGROUP CLASS

There are only two constructors of ThreadGroup class.

Constructor	Description
Thread Group (String name)	creates a thread group with given name.
Thread Group (ThreadGroup parent, String name)	creates a thread group with given parent group

Video Content / Details of website for further learning (if any):

https://www.youtube.com/watch?v=6TqUlAfhpVM

https://www.youtube.com/watch?v=GZbf6FXWmps

Important Books/Journals for further learning including the page nos.:

Herbert Schildt,"Java The complete reference", 8th EditionMcGraw Hill Education2011.Page No: 227-258

Course Faculty

Verified by HOD

MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)

Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

Estd. 2000

LECTURE HANDOUTS | **L-33**

AI&DS | **II/III**

Course Name with Code : **Object Oriented Programming-19ADC06**

Course Faculty : **T.Divya**

Unit : **IV-Multithreading and Generic Programming**

Date of Lecture:

Topic of Lecture: Generic Programming

Introduction : (Maximum 5 sentences) :

- Generic programming is a programming style in which algorithms are written at the most abstract possible level independent of the form of the data on which these algorithms will be carried out

Prerequisite knowledge for Complete understanding and learning of Topic:
(Max. Four important topics)

- Object Oriented Programming
- Functions and Methods

Detailed content of the Lecture:

The roots of generic programming

- David Musser and Alexander Stepanov, in the early 1970s
- the term 'generic programming' is coined in 1989
- the generic programming approach was pioneered by ML in 1973 (?)
- the generic programming approach was pioneered by ADA in 1983 (?)
- Different terms (& implementation) → similar concept

– generics

- Ada, Eiffel, Java, C#, VisualBasic.NET

– parametric polymorphism

- ML, Scala, Haskell

– templates

- C++

Generic Programming

- functions (methods) or types (classes) that differ only in the set of types on which theyoperate
- generic programming is a way to make a language more expressive, while still maintaining

full static type-safety

- reduce duplication of code

- algorithms are written in terms of generic types
- types are passed as parameters later when needed
- generic function
- performs the same operation on different data types

Generic type

– store values and perform operation on different data types

Java

– generics

C++

– templates

– (concepts)

generics add a way to specify concrete types to general purpose classes and methods that operated on Object before

● Java Specification Request 14, Add Generic Types To The Java Programming Language

● Java Enhancements in JDK 5 (originally numbered 1.5) (2005)

● "Generics

This long-awaited enhancement to the type system allows a type or method to operate on objects of various types while providing compile-time type safety.

- It adds compile-time type safety tothe Collections Framework and eliminates the drudgery of casting″

Video Content / Details of website for further learning (if any):
https://www.youtube.com/watch?v=XMvznsY02Mk

Important Books/Journals for further learning including the page nos.:
Herbert Schildt, "Java The complete reference", 8th EditionMcGraw Hill Education2011,Page No:325

Course Faculty

Verified by HOD

MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)

Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

Estd. 2000

LECTURE HANDOUTS | **L-34**

AI&DS | **II/III**

Course Name with Code : **Object Oriented Programming-19ADC06**

Course Faculty : **T.Divya**

Unit : **IV-Multithreading and Generic Programming**

Date of Lecture:

Topic of Lecture: Generic Class

Introduction : (Maximum 5 sentences) :

Generic classes enable programmers to specify, with a single method declaration, a set of related methods, or with a single class declaration, a set of related types, respectively.

Prerequisite knowledge for Complete understanding and learning of Topic:
(Max. Four important topics)

- Classes and Objects
- Functions and Methods

Detailed content of the Lecture:

Generic classes

- A generic class declaration looks like a non-generic class declaration, except that the class name is followed by a type parameter section.
- As with generic methods, the type parameter section of a generic class can have one or more type parameters separated by commas.
- These classes are known as parameterized classes or parameterized types because they accept one or more parameters.

Example

Following example illustrates how we can define a generic class −

```
public class Box<T> {
   private T t;

   public void add(T t) {
      this.t = t;
   }

   public T get() {
      return t;
   }

   public static void main(String[] args) {
```

```
    Box<Integer> integerBox = new Box<Integer>();
    Box<String> stringBox = new Box<String>();

    integerBox.add(new Integer(10));
    stringBox.add(new String("Hello World"));

    System.out.printf("Integer Value :%d\n\n", integerBox.get());
    System.out.printf("String Value :%s\n", stringBox.get());
  }
}
```

This will produce the following result −

Output

```
Integer Value :10
String Value :Hello World
```

Video Content / Details of website for further learning (if any):

https://www.tutorialspoint.com/java/java_generics.htm

Important Books/Journals for further learning including the page nos.:

Herbert Schildt, "Java The complete reference", 8th EditionMcGraw Hill Education2011,Page No:325

Course Faculty

Verified by HOD

MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)

Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

Estd. 2000

LECTURE HANDOUTS | **L-35**

AI&DS | **II/III**

Course Name with Code : **Object Oriented Programming-19ADC06**

Course Faculty : **T.Divya**

Unit : **IV-Multithreading and Generic Programming**

Date of Lecture:

Topic of Lecture: Generic methods

Introduction : (Maximum 5 sentences) :

- A Class that hold elements of various type
- A Generic method (with generics) is a method with a type parameter.

Prerequisite knowledge for Complete understanding and learning of Topic:
(Max. Four important topics)

- Generic Programming
- Generic methods

Detailed content of the Lecture:

Generic Methods:

- A generic class declaration looks like a non-generic class declaration, except that the class name is followed by a type parameter section.
- As with generic methods, the type parameter section of a generic class can have one or more type parameters separated by commas.
- These classes are known as parameterized classes or
- Parameterized types because they accept one or more parameters.

EXAMPLE:

```
public class Box<T>
{
private T t;
public void add(T t)
{
this.t = t;
}
public T get() {
return t;        }
public static void main(String[] args) {
Box<Integer> integerBox = new Box<Integer>();
Box<String> stringBox = new Box<String>();
```

integerBox.add(new Integer(10));

stringBox.add(new String("Hello World"));

System.out.printf("Integer Value :%d\n\n", integerBox.get());

System.out.printf("String Value :%s\n", stringBox.get());

} }

Output

Integer Value :10

String Value :Hello World

Generic methods

- You can write a single generic method declaration that can be called with arguments of different

  types.
- Based on the types of the arguments passed to the generic method, the compiler handles
- Each method call appropriately. Following are the rules to define Generic Methods:
- All generic method declarations have a type parameter section delimited by angle brackets < and > that precedes the method's return type < E > inthenextexample.
- Each type parameter section contains one or more type parameters separated by commas.
- A type parameter, also known as a type variable, is an identifier that specifies a generic type name.
- The type parameters can be used to declare the return type and act as placeholders for the types of the arguments passed to the generic method, which are known as actual type arguments.
- A generic method's body is declared like that of any other method.
- Note that type parameters can represent only reference types, not primitive types like int, double and char
- Following example illustrates how we can print array of different type using a single Generic

Method.

Video Content / Details of website for further learning (if any):
https://www.youtube.com/watch?v=22AUtQnTZkY

Important Books/Journals for further learning including the page nos.:
Herbert Schildt, “Java The complete reference”, 8th EditionMcGraw Hill Education2011,Page No:325

Course Faculty

Verified by HOD

MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)

Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

Estd. 2000

LECTURE HANDOUTS | **L-36**

AI&DS | **II/III**

Course Name with Code : **Object Oriented Programming-19ADC06**

Course Faculty : **T.Divya**

Unit : **IV-Multithreading and Generic Programming**

Date of Lecture:

Topic of Lecture : Bounded Types , Restrictions and Limitations

Introduction : (Maximum 5 sentences) :

- In the earlier posts, we have seen that while creating objects to generic classes we can pass any derived type as type parameters.

Prerequisite knowledge for Complete understanding and learning of Topic:
(Max. Four important topics)

- Generic classes
- Generic methods

Detailed content of the Lecture:

- Many times it will be useful to limit the types that can be passed to type parameters.
- For that purpose, bounded types or bounded type parameters are introduced in generics.
- Using bounded types, you can make the objects of generic class to have data of specific derived types.
- For example, If you want a generic class that works only with numbers (like int, double, float, long …..) then declare type parameter of that class as a bounded type to java.lang.Number class.
- Then while creating objects to that class you have to pass only Number types or it's subclass types as type parameters.
- Here is the syntax for declaring Bounded type parameters.

<T extends SuperClass>

This specifies that 'T' can only be replaced by 'SuperClass' or it's sub classes. Remember that extends clause is an inclusive bound. That means bound includes 'SuperClass' also.

Here is an example which demonstrates the bounded type parameters.

```
class GenericClass<T extends Number>    //Declaring Number class as upper bound of T
{
   T t;
    public GenericClass(T t)
   {
      this.t = t;
```

```
    }
    public T getT()
    {
       return t;
    }
}
```

- In this example, T has been declared as bounded type to Number class.
- So while creating objects to this class, you have to pass either Number type or it's subclass types (Integer, Double, Float, Byte…) as a type parameter. It wouldn't allow other than these types to pass as a type parameter. If you try to pass, compiler will throw compile time error.

We discuss a number of restrictions that you need to considerwhen working with Java generics.

Type Parameters Cannot Be Instantiated with Primitive Types

- Thus, there is no Pair<double>, only Pair<Double>.
- The reason is, of course, type erasure.
- After erasure, the Pair class has fields of type Object, and you can't use them to store double values.
- This is an annoyance, to be sure, but it is consistent with the separate status of primitive types in the Java language.
- It is not a fatal flaw—there are only eight primitive types, and you can always handle them with separate classes and methods when wrapper types are not an acceptable substitute.

Runtime Type Inquiry Only Works with Raw Types

Objects in the virtual machine always have a specific nongeneric type. Therefore, all type inquiries yield only the raw type.

You Cannot Throw or Catch Instances of a Generic Class

You can neither throw nor catch objects of a generic class. In fact, it is not even legal for a generic class to extend Throwable.

Video Content / Details of website for further learning (if any):
https://www.youtube.com/watch?v=KJJG-zBqSwE

Important Books/Journals for further learning including the page nos.:
Herbert Schildt , "Java The complete reference", 8th EditionMcGraw Hill Education2011,Page No:327

Course Faculty

Verified by HOD

MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)
Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

Estd. 2000

IQAC

L -37

LECTURE HANDOUTS

AI&DS

II/III

Course Name with Code : **Object Oriented Programming-19ADC06**

Course Faculty : **T.Divya**

Unit : **V-Event Driven Programming** **Date of Lecture:**

Topic of Lecture: Graphics programming - Frame – Components

Introduction : (Maximum 5 sentences) :

- It is difficult to display an image of any size on the computer screen.
- This method is simplified by using Computer graphics.
- Graphics on the computer are produced by using various algorithms and techniques.
- This tutorial describes how a rich visual experience is provided to the user by explaining how all these processed by the computer.

Prerequisite knowledge for Complete understanding and learning of Topic:
(Max. Four important topics)

- Real-World Entities
- Objects
- Classes
- Data

Detailed content of the Lecture:

- It is difficult to display an image of any size on the computer screen.
- This method is simplified by using Computer graphics.
- Graphics on the computer are produced by using various algorithms and techniques.
- This tutorial describes how a rich visual experience is provided to the user by explaining how all these processed by the computer.

INTRODUCTION OF COMPUTER GRAPHICS

- Computer Graphics involves technology to access.
- The Process transforms and presents information in a visual form.
- The role of computer graphics insensible.
- In today life, computer graphics has now become a common element in user interfaces, T.V. commercial motion pictures.
- Computer Graphics is the creation of pictures with the help of a computer.
- The end product of the computer graphics is a picture it may be a business graph, drawing, and engineering.
- In computer graphics, two or three-dimensional pictures can be created that are used for research.
- Many hardware devices algorithm has been developing for improving the speed of picture generation with the passes of time.
- It includes the creation storage of models and image of objects.
- These models for various fields like engineering, mathematical and so on.
- Today computer graphics is entirely different from the earlier one.
- It is not possible. It is an interactive user can control the structure of an object of various input devices.

WHY COMPUTER GRAPHICS USED:

- Suppose a shoe manufacturing company want to show the sale of shoes for five years.

- For this vast amount of information is to store. So a lot of time and memory will be needed.
- This method will be tough to understand by a common man. In this situation graphics is a better alternative.
- Graphics tools are charts and graphs. Using graphs, data can be represented in pictorial form.
- A picture can be understood easily just with a single look.
- Interactive computer graphics work using the concept of two-way communication between computer users.
- The computer will receive signals from the input device, and the picture is modified accordingly.
- Picture will be changed quickly when we apply command.

FRAMES

- The Leap Motion API presents motion tracking data to your application as a series of snapshots called frames.
- Each frame of tracking data contains the measured positions and other information about each entity detected in that snapshot.
- This article discusses the details of getting Frame objects from the Leap Motion controller.
- Each Frame object contains an instantaneous snapshot of the scene recorded by the Leap Motion controller. Hands, fingers, and tools are the basic physical entities tracked by the Leap Motion system.
- Get a Frame object containing tracking data from a connected Controller object. You can get a frame whenever your application is ready to process it using the frame() method of the Controller class:

```
if( controller.isConnected()) //controller is a Controller object

{

   Leap::Frame frame = controller.frame(); //The latest frame

   Leap::Frame previous = controller.frame(1); //The previous frame

}
```

Video Content / Details of website for further learning (if any):

https://medium.com/@viktor.kukurba/object-oriented-programming-in-javascript-1-abstraction-c47307c469d1

https://www.youtube.com/watch?v=57SIpmA3vcg

Important Books/Journals for further learning including the page nos.:

Timothy Budd, Understanding Object-oriented programming with Java, Pearson Education, 2000,Page No.566-568.

Course Faculty

Verified by HOD

MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)
Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

Estd. 2000

IQAC

L -38

LECTURE HANDOUTS

AI&DS

II / III

Course Name with Code : **Object Oriented Programming-19ADC06**

Course Faculty : **T.Divya**

Unit : **V-Event Driven Programming** **Date of Lecture:**

Topic of Lecture: Working with 2D shapes - Using color, fonts, and images

Introduction : (Maximum 5 sentences) :

- It is difficult to display an image of any size on the computer screen.
- This method is simplified by using Computer graphics.
- Graphics on the computer are produced by using various algorithms and techniques.
- This tutorial describes how a rich visual experience is provided to the user by explaining how all these processed by the computer.

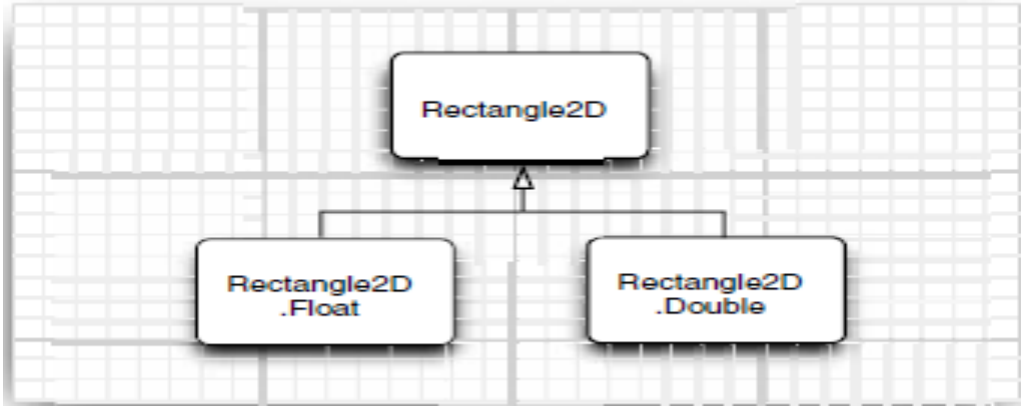
Prerequisite knowledge for Complete understanding and learning of Topic:
(Max. Four important topics)

- Real-World Entities
- Objects
- Classes
- Data

Detailed content of the Lecture:

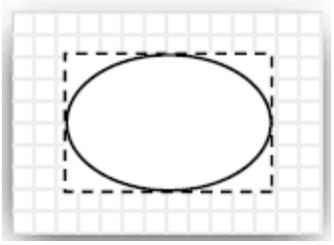
- A shape with only two dimensions (such as width and height) and no thickness.
- Squares, Circles, Triangles etc are two dimensional objects.
- These shapes mostly contain mathematical figures such are a line, square, triangle, rectangle and hexagon.

2D rectangle classes

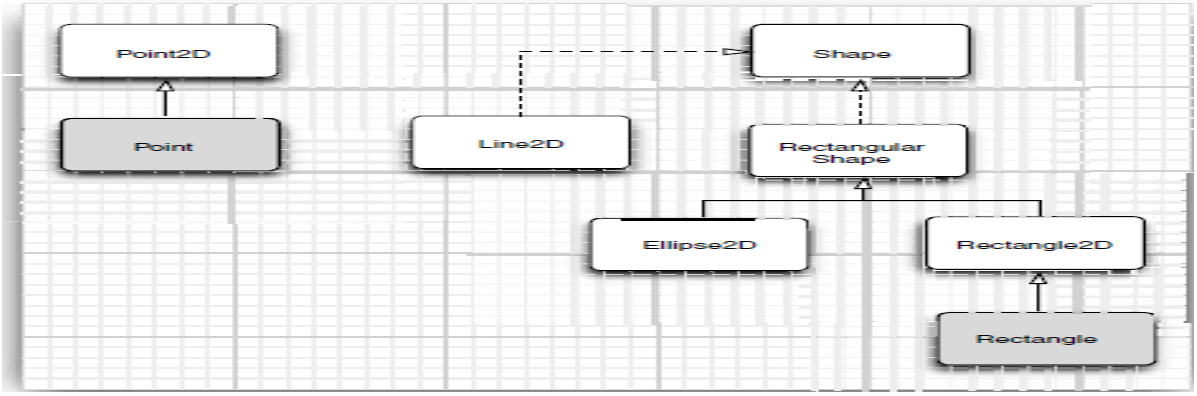


- The classes Rectangle2D and Ellipse2D both inherit from the common superclass Rectangular Shape. Admittedly, ellipses are not rectangular

The bounding rectangle of an ellipse



Relationships between the shape classes



However, the Double and Float subclasses are omitted. Legacy classes are marked with a gray fill. Rectangle2D and Ellipse 2D objects are simple to construct. You need to specify

- The *x*- and *y*-coordinates of the top-left corner; and
- The width and height.

For ellipses, these refer to the bounding rectangle. For example, Ellipse2D e = new Ellipse2D.Double(150, 200, 100, 50); constructs an ellipse that is bounded by a rectangle with the top -left corner at (150, 200),width 100, and height 50.

java.awt.geom.RectangularShape

doublegetCenterX()
doublegetCenterY()
doublegetMinX()
doublegetMinY()
doublegetMaxX()
doublegetMaxY()

Video Content / Details of website for further learning (if any):

https://medium.com/@viktor.kukurba/object-oriented-programming-in-javascript-1-abstraction-c47307c469d1

https://www.youtube.com/watch?v=57SIpmA3vcg

Important Books/Journals for further learning including the page nos.:

Timothy Budd, Understanding Object-oriented programming with Java, Pearson Education, 2000,Page.No-570-572

Course Faculty

Verified by HOD

MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)
Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

Estd. 2000

IQAC

L -39

LECTURE HANDOUTS

AI&DS

II/III

Course Name with Code : **Object Oriented Programming-19ADC06**

Course Faculty : **T.Divya**

Unit : **V-Event Driven Programming** **Date of Lecture:**

Topic of Lecture: Basics of event handling - event handlers

Introduction : (Maximum 5 sentences) :

- Change in the state of an object is known as event i.e. event describes the change in state of source.
- Events are generated as result of user interaction with the graphical user interface components.
- For example, clicking on a button, moving the mouse, entering a character through keyboard,selecting an item from list, scrolling the page are the activities that causes an event to happen

Prerequisite knowledge for Complete understanding and learning of Topic:
(Max. Four important topics)

- Real-World Entities
- Objects
- Classes
- Data

Detailed content of the Lecture:

WHAT IS AN EVENT

- Change in the state of an object is known as event i.e. event describes the change in state of source.
- Events are generated as result of user interaction with the graphical user interface components.
- For example, clicking on a button, moving the mouse, entering a character through keyboard,selecting an item from list, scrolling the page are the activities that causes an event to happen.

Types of Event

The events can be broadly classified into two categories:

- Foreground Events - Those events which require the direct interaction of user.
- They are generated as consequences of a person interacting with the graphical components in Graphical User Interface.
- For example, clicking on a button, moving the mouse, entering a character through keyboard,selecting an item from list, scrolling the page etc.
- Operating system interrupts, hardware or software failure, timer expires, an operation completion are the example of background event

Event Handling:

- Event Handling is the mechanism that controls the event and decides what should happen if an event occurs. This mechanism have the code which is known as event handler that is executed when an even
- t occurs. Java Uses the Delegation Event Model to handle the events.
- This model defines the standard mechanism to generate and handle the events.Let's have a brief introduction to this model.

The Delegation Event Model has the following key participants namely:

- Source - The source is an object on which event occurs. Source is responsible for providing information of the occurred event to it's handler.
- Java provide as with classes for source object.
- Listener - It is also known as event handler.Listener is responsible for generating response to an event. .
- The benefit of this approach is that the user interface logic is completely separated from the logic that generates the event.
- The user interface element is able to delegate the processing of an event to the separate piece of code.
- In this model ,Listener needs to be registered with the source object so that the listener can receive the event notification.
- This is an efficient way of handling the event because the event notifications are sent only to those listener that want to receive them.

Steps involved in event handling

- The User clicks the button and the event is generated.
- Now the object of concerned event class is created automatically and information about the source and the event get populated with in same object.
- Event object is forwarded to the method of registered listener class.
- the method is now get executed and returns.

Points to remember about listener

- In order to design a listener class we have to develop some listener interfaces.These Listener interfaces forecast some public abstract callback methods which must be implemented by the listener class.
- If you do not implement the any if the predefined interfaces then your class can not act as a listener class for a source object.

Callback Methods

- These are the methods that are provided by API provider and are defined by the application programmer and invoked by the application developer.
- Here the callback methods represents an event method. In response to an event java jre will fire callback method. All such callback methods are provided in listener interfaces.

Video Content / Details of website for further learning (if any):

https://medium.com/@viktor.kukurba/object-oriented-programming-in-javascript-1-abstraction-c47307c469d1

https://www.youtube.com/watch?v=57SIpmA3vcg

Important Books/Journals for further learning including the page nos.:

Timothy Budd, Understanding Object-oriented programming with Java, Pearson Education, 2000,Page.No-574-577

Course Faculty

Verified by HOD

MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)
Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

Estd. 2000

IQAC

L -40

LECTURE HANDOUTS

AI&DS

II/III

Course Name with Code : **Object Oriented Programming-19ADC06**

Course Faculty : **T.Divya**

Unit : **V-Event Driven Programming** **Date of Lecture:**

Topic of Lecture: Adapter classes - actions - mouse events

Introduction : (Maximum 5 sentences) :

- Adapter classes provide the default implementation of listener interfaces.
- If you inherit the adapter class, you will not be forced to provide the implementation of all the methods of listener interfaces. So it saves code.

Prerequisite knowledge for Complete understanding and learning of Topic:
(Max. Four important topics)

- Adapter classes
- Actions
- Mouse events

Detailed content of the Lecture:

- Adapter classes provide the default implementation of listener interfaces.
- If you inherit the adapter class, you will not be forced to provide the implementation of all the methods of listener interfaces. So it saves code.
- The adapter classes are found in **java.awt.event**, **java.awt.dnd** and **javax.swing.event** packages.
- The Adapter classes with their corresponding listener interfaces are given below.

JAVA.AWT.EVENT ADAPTER CLASSES

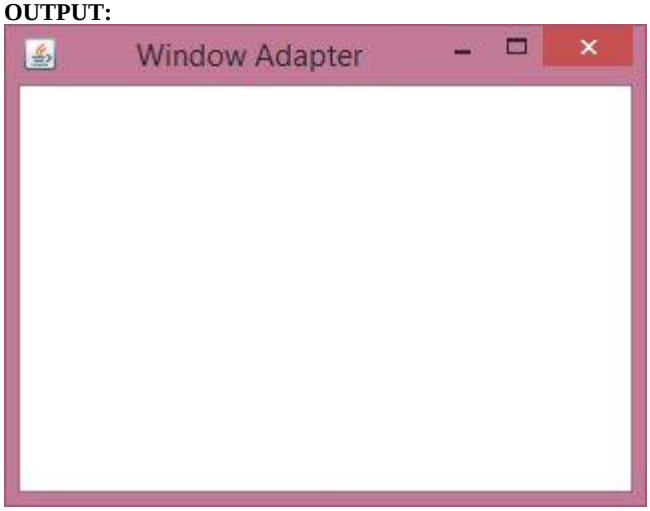
Adapter class	Listener interface
WindowAdapter	WindowListener
KeyAdapter	KeyListener
MouseAdapter	MouseListener
MouseMotionAdapter	MouseMotionListener
FocusAdapter	FocusListener
ComponentAdapter	ComponentListener
ContainerAdapter	ContainerListener
HierarchyBoundsAdapter	HierarchyBoundsListener

EXAMPLE :

```
import java.awt.*;
import java.awt.event.*;
public class AdapterExample{
   Frame f;
   AdapterExample(){
```

```
        f=new Frame("Window Adapter");
        f.addWindowListener(new WindowAdapter(){
          public void windowClosing(WindowEvent e) {
            f.dispose();  }}
        f.setSize(400,400);
        f.setLayout(null);
        f.setVisible(true);
      }
    public static void main(String[] args) {
      new AdapterExample();
    }
    }
```

OUTPUT:



MOUSE EVENTS

- The mouse_event function synthesizes mouse motion and button clicks.

PARAMETERS

Type: DWORD

- Controls various aspects of mouse motion and button clicking. This parameter can be certain combinations of the following values

SYNTAX:

C++Copy

```
void mouse_event(
  DWORD     dwFlags,
  DWORD     dx,
  DWORD     dy,
  DWORD     dwData,
  ULONG_PTR dwExtraInfo
);
```

Video Content / Details of website for further learning (if any):

https://medium.com/@viktor.kukurba/object-oriented-programming-in-javascript-1-abstraction-c47307c469d1

https://www.youtube.com/watch?v=57SIpmA3vcg

Important Books/Journals for further learning including the page nos.:

Timothy Budd, Understanding Object-oriented programming with Java, Pearson Education, 2000, Page.No-577-579

Course Faculty

Verified by HOD

MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)
Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

Estd. 2000

IQAC

L -41

LECTURE HANDOUTS

AI&DS

II/III

Course Name with Code : **Object Oriented Programming-19ADC06**

Course Faculty : **T.Divya**

Unit : **V-Event Driven Programming** **Date of Lecture:**

Topic of Lecture: AWT event hierarchy

Introduction : (Maximum 5 sentences) :

- Change in the state of an object is known as event i.e. event describes the change in state of source.
- Events are generated as result of user interaction with the graphical user interface components.
- For example, clicking on a button, moving the mouse, entering a character through keyboard,selecting an item from list, scrolling the page are the activities that causes an event to happen.

Prerequisite knowledge for Complete understanding and learning of Topic:
(Max. Four important topics)

- AWT event hierarchy
- Actions
- Event
- Mouse events

Detailed content of the Lecture:

TYPES OF EVENT

The events can be broadly classified into two categories:

- Foreground Events - Those events which require the direct interaction of user.They are generated as consequences of a person interacting with the graphical components in Graphical User Interface. For example, clicking on a button, moving the mouse, entering a character through keyboard,selecting an item from list, scrolling the page etc.
- Background Events - Those events that require the interaction of end user are known as background events. Operating system interrupts, hardware or software failure, timer expires, an operation completion are the example of background events.

WHAT IS EVENT HANDLING

Event Handling is the mechanism that controls the event and decides what should happen if an event occurs. This mechanism have the code which is known as event handler that is executed when an event occurs. The Delegation Event Model has the following key participants namely:

- Source - The source is an object on which event occurs. Source is responsible for providing information of the occurred event to it's handler. Java provide as with classes for source object.
- Listener - It is also known as event handler.Listener is responsible for generating response to an event. From java implementation point of view the listener is also an object. Listener waits until it receives an event. Once the event is received , the listener process the event an then returns.

The benefit of this approach is that the user interface logic is completely separated from the logic that generates the event.

STEPS INVOLVED IN EVENT HANDLING

- The User clicks the button and the event is generated.
- Now the object of concerned event class is created automatically and information about the source and the event get populated with in same object.
- Event object is forwarded to the method of registered listener class.
- the method is now get executed and returns.

POINTS TO REMEMBER ABOUT LISTENER

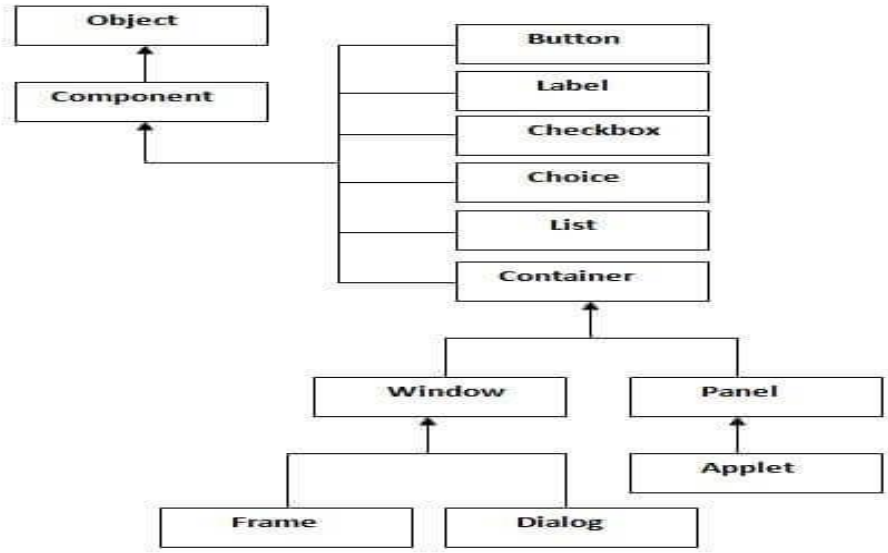
- In order to design a listener class we have to develop some listener interfaces.These Listener interfaces forecast some public abstract callback methods which must be implemented by the listener class.
- If you do not implement the any if the predefined interfaces then your class can not act as a listener class for a source object.

CALLBACK METHODS

- These are the methods that are provided by API provider and are defined by the application programmer and invoked by the application developer. Here the callback methods represents an event method. In response to an event java jre will fire callback method. All such callback methods are provided in listener interfaces.
- If a component wants some listener will listen to it's events the the source must register itself to the listener.

JAVA AWT HIERARCHY

The hierarchy of Java AWT classes are given below.



Video Content / Details of website for further learning (if any):

https://medium.com/@viktor.kukurba/object-oriented-programming-in-javascript-1-abstraction-c47307c469d1

https://www.youtube.com/watch?v=57SIpmA3vcg

Important Books/Journals for further learning including the page nos.:

Timothy Budd, Understanding Object-oriented programming with Java, Pearson Education, 2000, Page.No-579-581

Course Faculty

Verified by HOD

MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)
Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

Estd. 2000

IQAC

L -42

LECTURE HANDOUTS

AI&DS

II/III

Course Name with Code : **Object Oriented Programming-19ADC06**

Course Faculty : **T.Divya**

Unit : **V-Event Driven Programming** **Date of Lecture:**

Topic of Lecture: Introduction to Swing – layout management

Introduction : (Maximum 5 sentences) :

- Layout refers to the arrangement of components within the container.
- In another way, it could be said that layout is placing the components at a particular position within the container.
- The task of laying out the controls is done automatically by the Layout Manager.

Prerequisite knowledge for Complete understanding and learning of Topic:
(Max. Four important topics)

- Swing
- Layout management
- Event
- Mouse events

Detailed content of the Lecture:

LAYOUT MANAGERS

- The LayoutManagers are used to arrange components in a particular manner.
- LayoutManager is an interface that is implemented by all the classes of layout managers.
- There are following classes that represents the layout managers:
  1. java.awt.BorderLayout
  2. java.awt.FlowLayout
  3. java.awt.GridLayout
  4. java.awt.CardLayout
  5. java.awt.GridBagLayout
  6. javax.swing.BoxLayout
  7. javax.swing.GroupLayout
  8. javax.swing.ScrollPaneLayout
  9. javax.swing.SpringLayout etc.

JAVA BORDERLAYOUT

The BorderLayout is used to arrange the components in five regions: north, south, east, west and center. Each region (area) may contain one component only. It is the default layout of frame or window. The BorderLayout provides five constants for each region:

1. **public static final int NORTH**
2. **public static final int SOUTH**
3. **public static final int EAST**
4. **public static final int WEST**
5. **public static final int CENTER**

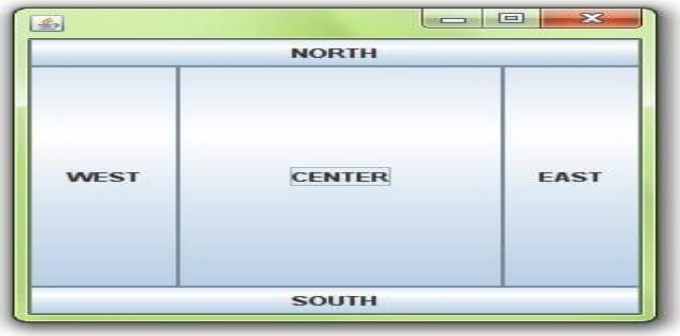
6.

CONSTRUCTORS OF BORDERLAYOUT CLASS:

- **BorderLayout():** creates a border layout but with no gaps between the components.
- **JBorderLayout(int hgap, int vgap):** creates a border layout with the given horizontal and vertical gaps between the components.

EXAMPLE:

```
import java.awt.*;
import javax.swing.*;
public class Border {
JFrame f;
Border(){
   f=new JFrame();
   JButton b1=new JButton("NORTH");;
   JButton b2=new JButton("SOUTH");;
   JButton b3=new JButton("EAST");;
   JButton b4=new JButton("WEST");;
   JButton b5=new JButton("CENTER");;
   f.add(b1,BorderLayout.NORTH);
   f.add(b2,BorderLayout.SOUTH);
   f.add(b3,BorderLayout.EAST);
   f.add(b4,BorderLayout.WEST);
   f.add(b5,BorderLayout.CENTER);
   f.setSize(300,300);
   f.setVisible(true);
}
public static void main(String[] args) {
   new Border();
}
}
```



Video Content / Details of website for further learning (if any):

https://medium.com/@viktor.kukurba/object-oriented-programming-in-javascript-1-abstraction-c47307c469d1

https://www.youtube.com/watch?v=57SIpmA3vcg

Important Books/Journals for further learning including the page nos.:

Timothy Budd, Understanding Object-oriented programming with Java, Pearson Education, 2000, Page.No-581-583

Course Faculty

Verified by HOD

MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)
Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

Estd. 2000

IQAC

L-43

LECTURE HANDOUTS

AI&DS

II/III

Course Name with Code : **Object Oriented Programming-19ADC06**

Course Faculty : **T.Divya**

Unit : **V-Event Driven Programming** **Date of Lecture:**

Topic of Lecture: Swing Components – Text Fields , Text Areas

Introduction : (Maximum 5 sentences) :

- Layout refers to the arrangement of components within the container.
- In another way, it could be said that layout is placing the components at a particular position within the container.
- The task of laying out the controls is done automatically by the Layout Manager.

Prerequisite knowledge for Complete understanding and learning of Topic:
(Max. Four important topics)

- Swing
- Swing Components
- Text Fields
- Text Areas

Detailed content of the Lecture:

- The Text Area and TextField classes display selectable text and, optionally, allow the user to edit the text. You can subclass Text Area and Text Field to perform such tasks as checking for errors in the input. As with any Component, you can specify the background and foreground colors and font used by TextAreas and TextFields. You can't, however, change their basic appearance.
- Both TextArea and TextField are subclasses of TextComponent[api]. From TextComponent they inherit methods that allow them to set and get the current selection, enable and disable editing, get the currently selected text (or all the text), and set the text.
- Below is an applet that displays first a TextField and then a TextArea. The TextField is editable; the TextArea isn't. When the user presses Return in the TextField, its contents are copied to the TextArea and then selected in the TextField.
- Here's the program. Here's just its code that creates, initializes, and handles events in the TextArea and TextField:

```
//Where instance variables are defined:
TextField textField;
TextArea textArea;

public void init() {
    textField = new TextField(20);
    textArea = new TextArea(5, 20);
    textArea.setEditable(false);

    ...//Add the two components to the panel.
}
```

```
public boolean action(Event evt, Object arg) {
   String text = textField.getText();
   textArea.appendText(text + "\n");
   textField.selectAll();
   return true;
}
```

- The TextComponent superclass of TextArea and TextField supplies the getText(), setText(), setEditable(), and selectAll() methods used in the above code example. It also supplies the following useful methods: getSelectedText(), isEditable(), getSelectionStart(), and getSelectionEnd().
- It also provides a select() method that lets you select text between beginning and end positions that you specify.
- The TextField class has four constructors: TextField(), TextField(int), TextField(String), and TextField(String, int).
- The integer argument specifies the number of columns in the text field. The String argument specifies the text initially displayed in the text field.
- The TextField class also supplies the following handy methods:

int getColumns()
Returns the number the columns in the text field.

setEchoChar()
Sets the echo character, which is useful for password fields.

char getEchoChar() boolean echoCharIsSet()
These methods let you ask about the echo character.

- The TextArea class supplies the `appendText()` method used in the code example above.
- It also supplies these methods:

int getRows(), int getColumns()
Return the number of rows or columns in the text area.

void insertText(String, int)
Inserts the specified text at the specified position.

void replaceText(String, int, int)
Replaces text from the indicated start position (the first integer) to the indicated end position.

Video Content / Details of website for further learning (if any):
https://medium.com/@viktor.kukurba/object-oriented-programming-in-javascript-1-abstraction-c47307c469d1
https://www.youtube.com/watch?v=57SIpmA3vcg

Important Books/Journals for further learning including the page nos.:
Timothy Budd, Understanding Object-oriented programming with Java, Pearson Education, 2000, Page.No-583-586

Course Faculty

Verified by HOD

MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)
Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

Estd. 2000

IQAC

L-44

LECTURE HANDOUTS

AI&DS

II/III

Course Name with Code : **Object Oriented Programming-19ADC06**

Course Faculty : **T.Divya**

Unit : **V-Event Driven Programming** **Date of Lecture:**

Topic of Lecture: Buttons- Check Boxes – Radio Buttons – Lists

Introduction : (Maximum 5 sentences) :

- A radio button is a control that appears as a dot surrounded by a round box.
- In reality, a radio button is accompanied by one or more other radio buttons that appear and behave as a group.

Prerequisite knowledge for Complete understanding and learning of Topic:
(Max. Four important topics)

- Buttons
- Check Boxes
- Radio Buttons
- Lists

Detailed content of the Lecture:

- A radio button is a control that appears as a dot surrounded by a round box.
- In reality, a radio button is accompanied by one or more other radio buttons that appear and behave as a group

HOW TO USE BUTTONS, CHECK BOXES, AND RADIO BUTTONS

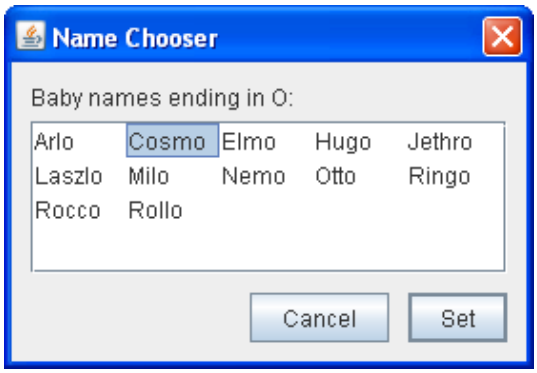
- To create a button, you can instantiate one of the many classes that descend from the AbstractButton class.
- The following table shows the Swing-defined AbstractButton subclasses that you might want to use:

Class	Summary	Where Described
`JButton`	A common button.	How to Use the Common Button API and How to Use JButton Features
`JCheckBox`	A check box button.	How to Use Check Boxes
`JRadioButton`	One of a group of radio buttons.	How to Use Radio Buttons
`JMenuItem`	An item in a menu.	How to Use Menus
`JCheckBoxMenuItem`	A menu item that has a check box.	How to Use Menus and How to Use Check Boxes
`JRadioButtonMenuItem`	A menu item that has a radio button.	How to Use Menus and How to Use Radio Buttons
`JToggleButton`	Implements toggle functionality inherited by `JCheckBox` and `JRadioButton`. Can be instantiated or subclassed to create two-state buttons.	Used in some examples

HOW TO USE JBUTTON FEATURES

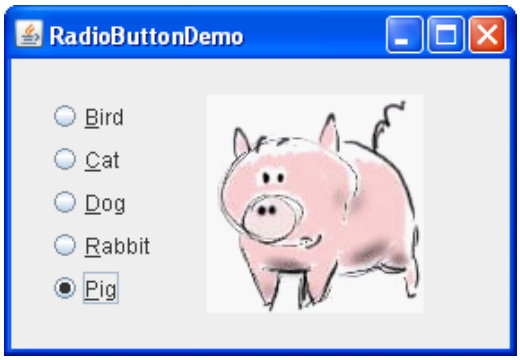
- Ordinary buttons — JButton objects — have just a bit more functionality than the AbstractButton class provides: You can make a JButton be the default button.
- At most one button in a top-level container can be the default button.

- The default button typically has a highlighted appearance and acts clicked whenever the top-level container has the keyboard focus and the user presses the Return or Enter key.
- Here is a picture of a dialog, implemented in the ListDialog example, in which the Set button is the default button:



HOW TO USE RADIO BUTTONS

- Radio buttons are groups of buttons in which, by convention, only one button at a time can be selected.
- The Swing release supports radio buttons with the JRadioButton and ButtonGroup classes.
- To put a radio button in a menu, use the JRadioButtonMenuItem class.
- Other ways of displaying one-of-many choices are combo boxes and lists. Radio buttons look similar to check boxes, but, by convention, check boxes place no limits on how many items can be selected at a time.
- Because JRadioButton inherits from AbstractButton, Swing radio buttons have all the usual button characteristics, as discussed earlier in this section.
- Here is a picture of an application that uses five radio buttons to let you choose which kind of pet is displayed:



LIST

- Lists are sequence containers that allow constant time insert and erase operations anywhere within the sequence, and iteration in both directions.
- List containers are implemented as doubly-linked lists; Doubly linked lists can store each of the elements they contain in different and unrelated storage locations.
- The ordering is kept internally by the association to each element of a link to the element preceding it and a link to the element following it.

Video Content / Details of website for further learning (if any):
https://medium.com/@viktor.kukurba/object-oriented-programming-in-javascript-1-abstraction-c47307c469d1
https://www.youtube.com/watch?v=57SIpmA3vcg

Important Books/Journals for further learning including the page nos.:
Timothy Budd, Understanding Object-oriented programming with Java, Pearson Education, 2000, Page.No-586-589

Course Faculty

Verified by HOD

MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)
Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

Estd. 2000

IQAC

L -45

LECTURE HANDOUTS

AI&DS

II/III

Course Name with Code : **Object Oriented Programming-19ADC06**

Course Faculty : **T.Divya**

Unit : **V-Event Driven Programming** **Date of Lecture:**

Topic of Lecture: Choices- Scrollbars – Windows –Menus – Dialog Boxes

Introduction : (Maximum 5 sentences) :

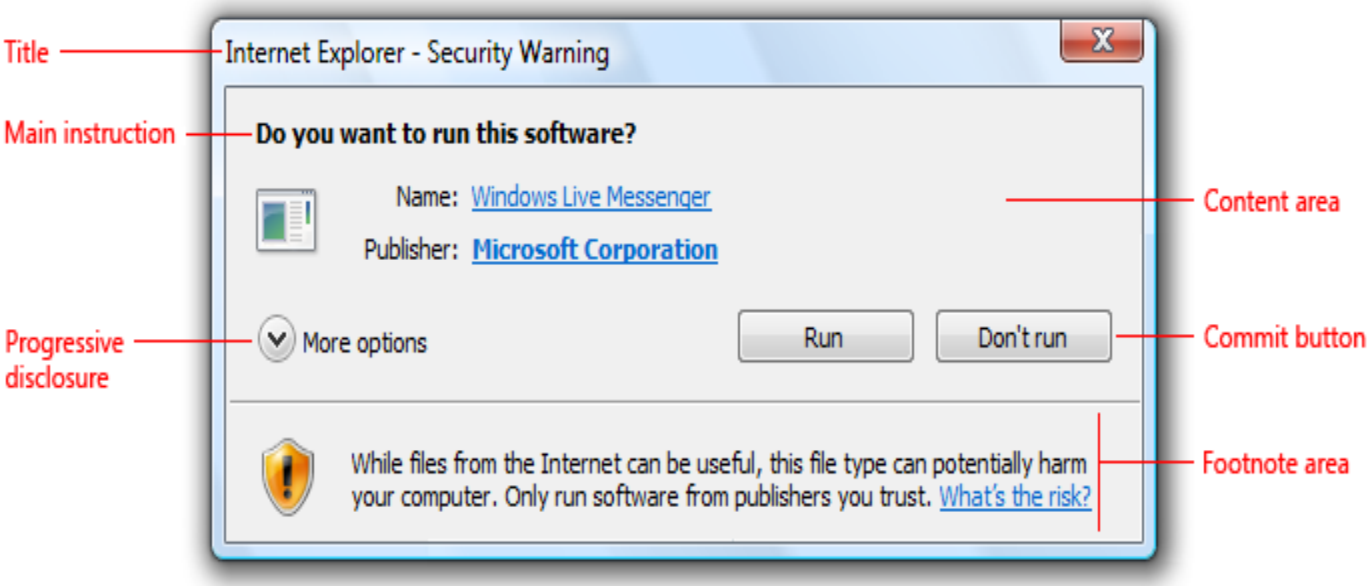
-

Prerequisite knowledge for Complete understanding and learning of Topic:
(Max. Four important topics)

- Choices- Scrollbars
- Windows
- Menus
- Dialog Boxes

Detailed content of the Lecture:

- A dialog box is a secondary window that allows users to perform a command, asks users a question, or provides users with information or progress feedback.



- A typical dialog box.
- Dialog boxes consist of a title bar (to identify the command, feature, or program where a dialog box came from), an optional main instruction (to explain the user's objective with the dialog box), various controls in the content area (to present options), and commit buttons (to indicate how the user wants to commit to the task).

WINDOWS

- Windows are the main "canvases" or UI surfaces of your desktop app, including the main windows itself and pop-ups, dialogs, and wizards.
- Follow these guidelines when deciding which surface to use and how best to use them.

Topic	Description
Window Management	This article covers default placement of windows when initially displayed on the screen, their stacking order relative to other windows (Z order), their initial size, and how their display affects input focus.
Window Frames	Most programs should use standard window frames. Immersive applications can have a full screen mode that hides the window frame. Consider using glass strategically for a simpler, lighter, more cohesive look.
Dialog Boxes	A dialog box is a secondary window that allows users to perform a command, asks users a question, or provides users with information or progress feedback.
Common Dialogs	The Microsoft Windows common dialogs consist of the Open File, Save File, Open Folder, Find and Replace, Print, Page Setup, Font, and Color dialog boxes.
Wizards	Despite that wonderful, whimsical name, wizards are not really a special form of user interface, and they have only a particular range of utility.

SCROLLBARS - BUILT IN OR SEPAR

- The first thing I want to discuss before we start is what type of scrollbar to use.
- We have two choices - “built-in” scrollbars which would be part of the TextView’s non-client window area, or separate scrollbar controls.
- There isn’t much difference between the two types. With built-in scrollbars, the WM_xSCROLL messages are sent to the same window to which the scrollbars belong.
- With separate scrollbar controls, the scrollbar messages get sent to the control’s parent window.
- However it would be very simple to forward the scrollbar messages from a parent window to the real TextView window, so this wouldn’t be any kind of set-back.

Video Content / Details of website for further learning (if any):
https://medium.com/@viktor.kukurba/object-oriented-programming-in-javascript-1-abstraction-c47307c469d1
https://www.youtube.com/watch?v=57SIpmA3vcg

Important Books/Journals for further learning including the page nos.:
Timothy Budd, Understanding Object-oriented programming with Java, Pearson Education, 2000Page.No-589-592

Course Faculty

Verified by HOD