



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



Must Know Concepts (MKC)

CSE

2021-2022

| Subject | | 19CSC06/OBJECT ORIENTED PROGRAMMING | | |
|--|---|-------------------------------------|---|-------|
| S. No. | Term | Notation (Symbol) | Concept/Definition/Meaning/Units/Equation/Expression | Units |
| UNIT –I-INTRODUCTION TO OOP AND JAVA FUNDAMENTALS | | | | |
| 1. | Principles of Oops | | <ol style="list-style-type: none">1. Objects2. Classes3. Data abstraction and encapsulation4. Inheritance5. Polymorphism6. Dynamic Binding7. Message passing | |
| 2. | Objects | | Objects are the basic run time entities in an object-oriented system. They may represent a person, a place, a bank account, a table of data or any item that the program has to handle. | |
| 3. | Classes | | Objects contain data, and code to manipulate that data. The entire set of data and code of an object can be made a user-defined data type called class. Objects are variables of the type class | |
| 4. | Data Abstraction and Encapsulation | | *The wrapping up of data and function into a single unit (called class) is known as encapsulation. * Abstraction refers to the act of representing essential features without including the background details or explanation | |
| 5. | Inheritance | | <ol style="list-style-type: none">1. Inheritance is the process by which objects of one class acquired the properties of objects of another classes. It supports the concept of hierarchical classification.2. Provides the idea of reusability | |
| 6. | Polymorphism | | <ul style="list-style-type: none">• Polymorphism is another important OOP concept.• Polymorphism, a Greek term, means the ability to take more than one form.• An operation may exhibit different behaviors in different instances. | |
| 7. | Dynamic Binding | | Dynamic binding means that the code associated with a given procedure call is not known until the time of the call at run time. | |
| 8. | Applications of Oop | | <ol style="list-style-type: none">1) Real-time systems.2) Simulation & modeling3) Object Oriented Databases4) Artificial Intelligence (AI) & Expert Systems.5) Neural networks & Parallel Programming.6) Decision Support Systems. | |

| | | | | |
|-----|--|--|--|--|
| 9. | C++ | | C++ is an Object Oriented Program Language C++ supports inheritance, polymorphism, Virtual functions, classes & object concepts | |
| 10. | Tokens | | The smallest individual units in a program are known as tokens. C++ has the following tokens 1. Keywords 2. Identifiers 3. Constants 4. Strings 5. Operators | |
| 11. | Datatypes | | Datatypes specify the size and type of values that can be stored. The variety of data types available allow the programmer to select the type appropriate to needs of the application. | |
| 12. | Types of Datatypes | | 1. Built-in-type 2. User-defined type. 3. Derived type | |
| 13. | Variable | | Variable is an identifier which holds data or another one variable is an identifier whose value can be changed at the execution time of program. Variable is an identifier which can be used to identify input data in a program | |
| 14. | Operators | | <ul style="list-style-type: none"> • An Operator is a symbol that specifies an Operator to be Performed on the Operands. • Some Operators require two Operands called “Binary Operators”. • Some Operators require only one Operand called “Unary Operators”. | |
| 15. | Types of operators | | 1) Arithmetic Operators(+, -, *, /, %,) 2) Relational Operators (<, >, <=, >=, !=, ==) 3) Logical Operators (&&, , !) 4) Assignment Operator(=) 5) Bitwise Operators (&, !, , ^) 6) Other Operators (, sizeof, &&*, .&->) | |
| 16. | Functions in C++ | | A function is a group of statements that together perform a task. A function declaration tells the compiler about a function's name, return type, and parameters. | |
| 17. | Advantage of Function | | Advantage of Function ➤ Code Re-usability ➤ Develop an application in module format. ➤ Easily to debug the program. ➤ Code optimization: No need to write lot of code. | |
| 18. | Object-Oriented Programming (OOP) | | It is a programming language model organized around objects rather than actions and data. An object-oriented program can be characterized as data controlling access to the code. | |
| 19. | Method overloading | | When a method in a class having the same method name with different arguments is said to be method overloading. | |
| 20. | Method overriding | | When a method in a class having the same method name with same arguments is said to be method overriding | |
| 21. | Constructor | | Constructor is an operation that creates an object and/or initializes its state. | |
| 22. | Destructor | | Destructor is an operation that frees the state of an object and/or destroys the object itself. In Java, there is no concept of destructors. Its taken care by the JVM. | |
| 23. | Java Virtual Machine | | JVM is an abstract computing machine like any other real computing machine which first converts .java file into .class file by using Compiler (.class is nothing but byte code file.) and | |

| | | | | |
|--|--|--|---|--|
| | | | Interpreter reads byte code. | |
| 24. | difference between this () and super () | | This () can be used to invoke a constructor of the same class whereas super() can be used to invoke a super class constructor | |
| 25. | Package | | A package is a collection of classes and interfaces that provides a high-level layer of access protection and name space management | |
| UNIT –II-INHERITANCE AND INTERFACES | | | | |
| 26. | super class and subclass | | Super class is a class from which another class inherits. Subclass is a class that inherits from one or more classes. | |
| 27. | Difference B/W superclass and subclass | | A super class is a class that is inherited whereas sub class is a class that does the inheriting. | |
| 28. | Interface | | Interface is an outside view of a class or object which emphasizes its abstraction while hiding its structure and secrets of its behavior . | |
| 29. | Inheritance | | Inheritance is the process of creating new classes from the existing classes. The new classes are called derived classes. The existing classes are called base classes. | |
| 30. | FINAL KEYWORD | | Final keyword can be used along with variables, methods and classes. 1) final variable 2) final method 3) final class | |
| 31. | Object Cloning | | 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. | |
| 32. | Advantage of Object cloning | | <ul style="list-style-type: none"> You don't need to write lengthy and repetitive codes. Just use an abstract class with a 4- or 5-line long clone() method. Clone() is the fastest way to copy array. | |
| 33. | Disadvantage of Object cloning | | <ul style="list-style-type: none"> Object.clone() is protected, so we have to provide our own clone() and indirectly call Object.clone() from it. Object.clone() does not invoke any constructor so we don't have any control over object construction | |
| 34. | Inner class | | Inner class means one class which is a member of another class. There are basically four types of inner classes in java. 1) Nested Inner class 2) Method Local inner classes 3) Anonymous inner classes 4) Static nested class | |
| 35. | Nested Inner class | | It can access any private instance variable of outer class. Like any other instance variable, we can have access modifier private, protected, public and default modifier. | |
| 36. | Method Local inner classes | | Inner class can be declared within a method of an outer class. In the following example, Inner is an inner class in outerMethod(). | |
| 37. | Anonymous inner classes | | Anonymous inner classes are declared without any name at all. They are created in two ways. a) As subclass of specified type b) As implementer of the specified interface | |
| 38. | Static nested classes | | Static nested classes are not technically an inner class. They are like a static member of outer class. | |

| | | | |
|-----|--|--|---|
| 39. | Strings in Java | | In java, string is basically an object that represents sequence of char values. |
| 40. | ways to create String object | | 1. By string literal 2. By new keyword |
| 41. | Advantages of interface in java | | <ul style="list-style-type: none"> • Without bothering about the implementation part, we can achieve the security of implementation • In java, multiple inheritance is not allowed, however you can use interface to make use of it as you can implement more than one interface. |
| 42. | Object Class | | There is one special class, Object, defined by Java. All other classes are subclasses of Object. That is, Object is a superclass of all other classes. This means that a reference variable of type Object can refer to an object of any other class. |
| 43. | “super” KEYWORD Usage of super keyword | | <ol style="list-style-type: none"> 1. super() invokes the constructor of the parent class. 2. super.variable_name refers to the variable in the parent class. 3. super.method_name refers to the method of the parent class. |
| 44. | Java Array List class | | Java Array List class uses a dynamic array for storing the elements. It inherits Abstract List class and implements List interface |
| 45. | Nested Interface in Java | | An interface can have another interface i.e. known as nested interface. <pre>interface printable{ void print(); interface MessagePrintable{ void msg(); } }</pre> |
| 46. | Multiple inheritance in Java by interface | | If a class implements multiple interfaces, or an interface extends multiple interfaces i.e. known as multiple inheritance. |
| 47. | Sub Class/Child Class | | Subclass is a class which inherits the other class. It is also called a derived class, extended class, or child class. |
| 48. | Super Class/Parent Class | | Superclass is the class from where a subclass inherits the features. It is also called a base class or a parent class |
| 49. | Reusability | | As the name specifies, reusability is a mechanism which facilitates you to reuse the fields and methods of the existing class when you create a new class. You can use the same fields and methods already defined in previous class. |
| 50. | Class | | A class is a group of objects which have common properties. It is a template or blueprint from which objects are created. |

UNIT –III-EXCEPTION HANDLING AND I/O

| | | | |
|-----|---------------------------|--|--|
| 51. | Exception handling | | The ios class provides operations common to both input and output. It contains a pointer to a buffer object. It has constants and member functions that are useful in handling formatted I/O operations. |
| 52. | istream | | Input stream-A file input stream is an input stream for reading data from a File or from a FileDescriptor . |
| 53. | ostream | | output stream -Creates a file output stream to write to the file represented by the specified File object. |
| 54. | iostream | | The InputStream is used to read data from a source and the OutputStream is used for writing data to a destination. Here is a hierarchy of classes to deal with Input and Output streams. |
| 55. | ios::in | | open for reading |

| | | | | |
|---|-------------------------------|--|---|--|
| 56. | ios::ate | | seek to the end of file at opening time | |
| 57. | ios::binary | | opens a binary file | |
| 58. | ios::nocreate | | open fails if file does not exist | |
| 59. | Sequential access | | This type of file is to be accessed sequentially that is to access a particular data all the preceding data items have to be read and discarded. | |
| 60. | Random access | | This type of file allows access to the specific data directly without accessing its preceding data items | |
| 61. | Synchronous exception | | The exceptions, which occur during the program execution, due to some fault in input data, within the program, is known as Synchronous exception. | |
| 62. | Asynchronous exception | | The exceptions caused by events or a fault unrelated to the program and beyond the control of the program is known as asynchronous exception. | |
| 63. | try | | <ul style="list-style-type: none"> • This keyword defines a boundary within which an exception can occur. • A block of code in which an exception may occur must be prefixed by this keyword. | |
| 64. | Throw | | <ul style="list-style-type: none"> • Throw is used to raise an exception when an error is generated in the computation. • It initializes a temporary object to be used in throw. | |
| 65. | Catch | | <ul style="list-style-type: none"> • This keyword represents exception handler. It must be compulsorily used immediately after the statements marked by try keyword. • It can also occur immediately after catch keyword. | |
| 66. | Hit the exception | | Detect the problem causing exception | |
| 67. | Throw the exception | | Inform that an error has occurred | |
| 68. | Catch the exception | | Receive the error information | |
| 69. | Handle the exceptions | | Take corrective actions | |
| 70. | terminate() | | It is invoked when an exception is raised and the handler is not found. | |
| 71. | set_terminate() | | Allows the user to install a function that defines the program's actions to be taken to terminate the program when a handler for the exception cannot be found | |
| 72. | unexpected() | | This function is called when a function throws an exception not listed in its exception specification | |
| 73. | set_unexpected() | | It allows the user to install a function that defines the program's actions to be taken when a function throws an exception not listed in its exception specification | |
| 74. | Fault avoidance | | It deals with the prevention of fault occurrence by construction. | |
| 75. | Fault tolerance | | This deals with the method of providing services complying with the specification in spite of false occurring by redundancy. | |
| UNIT IV-MULTITHREADING AND GENERIC PROGRAMMING | | | | |
| 76. | Java | | <ul style="list-style-type: none"> • Java is an object-oriented programming language with its runtime environment. • It is a combination of features of C and C++ with some essential additional concepts | |

| | | | | |
|-----|------------------------------------|--|--|--|
| 77. | Types of Java Application | | 1.Web Application 2. Standalone Application 3.Enterprise Application 4.Mobile Application | |
| 78. | BASIC JAVA CONCEPTS | | Java supports the following fundamental concepts – <ul style="list-style-type: none"> • Object • Class • Inheritance • Polymorphism • Abstraction • Encapsulation | |
| 79. | Object Creation | | Creating an object from a class: <ul style="list-style-type: none"> • Declaration: A variable declaration with a variable name with an object type. • Instantiation: The 'new' keyword is used to create the object. • Initialization: The 'new' keyword is followed by a call to a constructor. • This call initializes the new object. | |
| 80. | CLASSES | | <ul style="list-style-type: none"> • A class is a group of objects that has common properties. It is a template or blueprint from which objects are created. • Class keyword is used to declare a class. • Class does not store any space | |
| 81. | ABSTRACTIO N | | <ul style="list-style-type: none"> • Abstraction is a process where you show only “relevant” data and “hide” unnecessary details of an object from the user. • We can achieve “abstraction” in Java using two ways. <ol style="list-style-type: none"> 1. Abstract class 2. Abstract methods | |
| 82. | Advantages of Encapsulation | | <ul style="list-style-type: none"> • Using getter and setter method, the field of the class can be made read-only or write-only. • It improves the flexibility & maintainability of the code. | |
| 83. | Use of inheritance in java | | For Method Overriding (so runtime polymorphism can be achieved). <ul style="list-style-type: none"> • For Code Reusability. • Consistency in using an interface | |
| 84. | Types of inheritance | | <ol style="list-style-type: none"> 1. Single level 2. Multilevel inheritance 3. Hierarchical | |
| 85. | Abstract method | | <ul style="list-style-type: none"> • A method that is declared as abstract and does not have implementation is known as abstract method. • To create an abstract method, just write the method declaration without the body and use the keyword “abstract” | |
| 86. | POLYMORPHI SM | | <ul style="list-style-type: none"> • When one task is performed by different ways i.e. known as polymorphism. • For example: to converse the customer differently, to draw something e.g. shape | |
| 87. | Compile Time Polymorphism | | <ul style="list-style-type: none"> • Method overloading is nothing but compile time polymorphism in Java. • It is checked by the compiler at compile time. • Compile time polymorphism is also known as static polymorphism. | |
| 88. | Runtime polymorphism | | Compiler cannot determine the method at compile time. | |

| | | | | |
|---|------------------------------------|--|--|--|
| | | | <ul style="list-style-type: none"> • Method overriding is the perfect example of runtime polymorphism. • It is also called as dynamic polymorphism | |
| 89. | Rules for method overriding | | <ul style="list-style-type: none"> • Method must be written in child class, not in same class. • Method name and argument must be same. • A final method cannot be overridden. • Method should be inherited means IS-A relationship. | |
| 90. | four access levels | | <ul style="list-style-type: none"> • Visible to the package, the default. No modifiers are needed. • Visible to the class only (private). • Visible to the world (public). • Visible to the package and all subclasses (protected). | |
| 91. | STATIC MEMBERS | | static keyword with variables, methods, blocks and nested class. The static keyword belongs to the class than instance of the class. | |
| 92. | Types of constructors | | There are two types of constructors: 1. default constructor (no-arg constructor) 2. parameterized constructor | |
| 93. | Constructor Overloading | | Constructor overloading is a technique in which a class can have any number of constructors that differ in parameter lists. Constructor overloading means declaring multiple constructors with different parameter in the similar class. | |
| 94. | Features of Java | | Object oriented * Platform independent * Simple *Secure *Portable | |
| 95. | Default constructor | | The default constructor can refer to a constructor that is automatically generated by the compiler in the absence of any programmer-defined constructors | |
| 96. | parameterized constructor | | A constructor is called Parameterized Constructor when it accepts a specific number of parameters. To initialize data members of a class with distinct values. | |
| 97. | hierarchical inheritance. | | <ul style="list-style-type: none"> • when a class has more than one child classes (sub classes) or in other words more than one child classes have the same parent class • This type of inheritance is known as hierarchical inheritance. | |
| 98. | Multiple inheritance | | Multiple inheritance is a feature of some object-oriented computer programming languages in which an object or class can inherit characteristics and features from more than one parent object or parent class. | |
| 99. | Access Modifiers | | The access modifiers in Java specifies the accessibility or scope of a field, method, constructor, or class and change the access level of fields, constructors, methods, and class by applying the access modifier on it. | |
| 100. | Protected access modifier | | The protected access modifier is accessible within package and outside the package but through inheritance only. | |
| UNIT –V-EVENT DRIVEN PROGRAMMING | | | | |
| 101. | Overriding member functions | | If base class and derived class have member functions with same name and arguments. | |
| 102. | Virtual base class | | When two or more objects are derived from a common base class, we can prevent multiple copies of the base class being present in | |

| | | | | |
|------|---|--|--|--|
| | | | an object derived from those objects by declaring the base class as virtual when it is being inherited. Such a base class is known as virtual base class. | |
| 103. | Abstract Class | | An abstract class is a class that cannot be instantiated and is usually implemented as a class that has one or more pure virtual (abstract) functions. | |
| 104. | This Pointer | | C++ uses a unique keyword called this to represent an object that invokes a member function. this is a pointer that points to the object for which <i>this</i> function was called. | |
| 105. | JAVA | | <ul style="list-style-type: none"> • Java is an object-oriented programming language with its runtime environment. • Java code that runs on one platform does not need to be recompiled to run on another platform • Write Once, Run Anywhere(WORA) | |
| 106. | JVM | | Converts Java bytecode into machine language and executes it. | |
| 107. | Methods | | Collection of statements that are grouped together to perform an operation | |
| 108. | Java Beans | | This is a set of reusable software components that can be easily used to create new and advanced applications | |
| 109. | J2EE | | Java 2 Enterprise Edition is a platform-independent environment that is a set of different protocols and APIs and is used by various organizations to transfer data between each other. | |
| 110. | JSP | | In Java, JSP (Java Server Pages) is used to create dynamic web pages, such as in PHP and ASP. | |
| 111. | JDBC Rowset | | For sending the tabular format between remote components to distributed application JDBC RowSet is used as it provides direct access to the Database | |
| 112. | Messages | | Objects communicate with one another by sending messages. A message is a method call from a message-sending object to a message-receiving object. | |
| 113. | Encapsulation | | Binding (or wrapping) code and data together into a single unit is known as encapsulation. For example: capsule, it is wrapped with different medicines. | |
| 114. | Abstract Class | | A class that is declared as abstract is known as abstract class. It needs to be extended and its method implemented | |
| 115. | Subclasses | | A subclass is a class derived from the superclass. It inherits the properties of the superclass and also contains attributes of its own. | |
| 116. | Java Array List | | The Array List class is a resizable array, which can be found in the java.util package | |
| 117. | Arrays in C++ | | An array is a collection of elements of the same type placed in contiguous memory locations that can be individually referenced by using an index to a unique identifier | |
| 118. | C++ Strings | | A string variable contains a collection of characters surrounded by double quotes | |
| 119. | Multithreading in Java | | Multithreading in java is a process of executing multiple threads simultaneously. A thread is a lightweight sub-process, the smallest unit of processing. | |
| 120. | Dynamic binding | | Dynamic binding also called dynamic dispatch is the process of linking procedure call to a specific sequence of code (method) at run-time. | |
| 121. | Operators that cannot be overloaded in C++ | | <p>Class member access operator (, , .*)</p> <ul style="list-style-type: none"> • Scope resolution operator (::) • Size operator (sizeof) • Conditional operator (?:) | |

| | | | |
|------|---------------------------------------|--|---|
| 122. | Copy constructor | | A copy constructor is used to declare and initialize an object from another object. Copy constructor takes a reference to an object of the same class as itself as an argument. |
| 123. | Data hiding | | The insulation of data from direct access by the program |
| 124. | Function overloading | | Function Overloading is defined as the process of having two or more function with the same name, but different in parameters. |
| 125. | Characteristics of constructor | | <ul style="list-style-type: none"> • They should be declared in the public section. • They are invoked directly when an object is created. |

GATE QUESTIONS

| | | | |
|------|------------------------------------|--|--|
| 126. | Overriding | | Many object-oriented programming languages permit a class or object to replace the implementation of an aspect—typically a behavior—that it has inherited. This process is usually called overriding. |
| 127. | This pointer | | C++ uses the unique keyword called this to represent an object that invokes a member function |
| 128. | pure virtual function | | A virtual function that is declared in a base class but not defined there. The responsibility for defining the function falls on the derived classes, each of which generally provides a different definitions. |
| 129. | virtual function | | It is a function qualified by the virtual keyword. When a virtual function is called via a pointer, the class of the object pointed to determines which function definition will be used. |
| 130. | virtual base class | | A base class that has been qualified as virtual in the inheritance definition. |
| 131. | Code re-use | | One of the earliest motivations for using inheritance was to allow a new class to re-use code which already existed in another class. This practice is usually called implementation inheritance. |
| 132. | Extension | | Another reason to use inheritance is to provide additional data or behavior features. This practice is sometimes called extension or subclassing. |
| 133. | Overriding | | Many object-oriented programming languages permit a class or object to replace the implementation of an aspect—typically a behavior—that it has inherited. This process is usually called overriding. |
| 134. | Specialization | | One common reason to use inheritance is to create specializations of existing classes or objects. This is often called subtyping when applied to classes. |
| 135. | destructor | | It is used to destroy the objects that have been created by a constructor, when they no longer required. |
| 136. | const member function | | If a member function does not alter any data in the class, then we declare it as const member function. The keyword const is appended to the function prototype. |
| 137. | friend function | | The functions that are declared with the keyword friend are known as friend functions. A function can be declared as a friend in any number of classes, it has full access rights to the private members of the class. |
| 138. | nesting of member functions | | A member function can be called by using its name inside another member function of the same class, is known as member function |
| 139. | advantages of new operator | | It automatically computes the size of the data object. So there is no need to use sizeof operator |
| 140. | types of OOP | | Object-Based Programming Languages, Object-Oriented Programming Languages |

| | | | | |
|---|---------------------------------|--|--|--|
| 141. | Break | | Break statement takes the control to the outside of loop | |
| 142. | Continue | | Continue statement takes the control to the beginning of loop | |
| 143. | Implicit Type Conversion | | When an expression consists of data items of different types, the compiler performs type conversions automatically. This is referred as Implicit Type Conversion. | |
| 144. | ADT | | The classes which are using the concept of data abstraction is known as abstract data types (ADT). | |
| 145. | Pointer | | pointer is a data type that holds the address of a location in memory. | |
| 146. | Null Pointer | | Null Pointer = is a pointer that does not point to any data object. In C++, the null pointer can be represented by the constant 0. | |
| 147. | Usage of ios class | | The ios class provides operations common to both input and output. It contains a pointer to a buffer object. It has constants and member functions that are useful in handling formatted I/O operations. | |
| 148. | N-version programming | | N-programmers develop N algorithms for the same problem with out interacting with each other | |
| 149. | Recovery block | | This structure represents the dynamic redundancy approach to s/w fault tolerance. | |
| 150. | daemon thread | | These are the threads which can run without user intervention. The JVM can exit when there are daemon thread by killing them abruptly. | |
| Faculty Team Prepared | | | | |
| 1) K.Sudha, ASP/CSE 2) M.Buveneswari, AP/CSE 3) K.Papithasri, AP/CSE 4) M.Ramya, AP/CSE 5) S.Pragadeeswaran, AP/CSE | | | Signature:1) 2) 3) 4) 5) | |

Subject Expert

HoD/CSE