Design Patterns

Dr.K.Ramar
Professor / CSE
Dean R& D
MEC (Autonomous)

References

- Web page:
  http://hillside.net/patterns/DPBook/DPBook.html
- Source code:
  http://hillside.net/patterns/DPBook/Source.html
- Comments:
  gang-of-4-patterns@cs.uiuc.edu

Organization of Book

Book is divided into two parts:

- Chapters 1 & 2
  - describe what design patterns are
  - describe how design patterns help design object-oriented software
  - include a design case study that demonstrates how design patterns apply in practice
- Chapters 3, 4, and 5 are a catalog of design patterns

Appendices

- Appendix A: Glossary of terminology
- Appendix B: Explains notation used in diagrams throughout the book.
- Appendix C: Contains source code for the foundation classes in the code samples

Goal of the Book

"To capture design experience in a form that people can use effectively."

Contributions of Book

1. Shows the role that patterns can play in architecting complex systems.
2. Provides a pragmatic reference to a set of well-engineered patterns that can be applied by other developers.

What is a design pattern?

Or what attributes describe a design pattern?

Responses on board.

In what ways are design patterns useful in design and development?

Responses on board.

What are the four essential elements of a design pattern?

Responses on board.

Design Template

Purpose: “Lends uniform structure to the information, making design patterns easier to learn, compare and use.”

How do the four essential elements related to the design pattern template?

Design Pattern Template

- **Pattern Name and Classification**: Succinct, captures essences of pattern, reflects scheme
- **Intent**: Short statement that answers the following:
  - What does it do?
  - What is its rationale and intent
  - What particular design issue or problem does it address?

More design pattern template

- **Also Know As**: aliases
- **Motivation**: Scenario which illustrates the design problem and how class and object structures in the pattern solve the problem.
- **Applicability**:
  - When design pattern can be applied
  - Examples of poor design the pattern can address
  - How to recognize design patterns

Still more design pattern template

- **Structure**: Graphical representation of classes in the design pattern using OMT (Object Modeling Technique). Also utilizes interaction diagrams.
- **Participants**: Classes and/or objects participating in the design pattern and their responsibilities.
- **Collaborations**: How participants carry out their responsibilities.

Even more design pattern template

- **Consequences**:
  - How a design pattern supports its objectives
  - Trade-offs, results of using design pattern
  - Aspects of system which can vary independently
- **Implementation**:
  - pitfalls, hints or techniques
  - language-specific issues
- **Sample Code**: Code fragments in C++ or SmallTalk.

Finally, last part of the design pattern template

- **Known Uses**: At least two examples of the pattern found in real systems.
- **Related Patterns**:
  - What patterns are closely related?
  - Important differences between closely related patterns.
  - Which patterns is this design pattern used with?

Organization of Catalog

- Design patterns vary in granularity and level of abstraction.
- Families of related patterns help to programmer to learn patterns faster and can direct efforts to find new patterns.
- 23 design patterns total

Two classifications of a design pattern

- **Purpose**: What a design pattern does
- **Scope**: Specifies whether a design pattern applies primarily to classes or to objects

Design Pattern Purpose

- **Creational** (5): Concerns object creation
- **Structural** (7): Deals with composition of classes or objects
- **Behavioral** (11): Characterizes ways in which classes or objects interact and distribute responsibility.

Design Pattern Scope

- **Class Patterns** (4)
  - Deal with relationships between classes and their subclasses
  - Relationships established through inheritance, so they are fixed at compile time (static)
- **Object patterns** (20)
  - Deal with object relationships
  - Relationships can be changed at runtime (dynamic)

Six types of design patterns

1. **Creational class patterns** defer some part of object creation to subclasses
2. **Creational object patterns** defer some part of object creation to another object
3. **Structural class patterns** use inheritance to compose classes
4. **Structural object patterns** describe ways to assemble objects
5. **Behavioral class patterns** use inheritance to describe algorithms and flow of control
6. **Behavioral object patterns** describe how a group of objects cooperate to perform a task that no single object can carry out alone

Additional ways to organize design patterns

- Patterns which are used together
- Some patterns are alternatives for one another
- Some patterns result in similar design although they have different intents
- Patterns which reference one another (see Figure 1.1, p. 12)

MVC (quick review)

- **MVC**
  - **Model**: application object
  - **View**: screen presentation
  - **Controller**: defines the way the user interface reacts to user input

Draw diagram on board

MVC Design Pattern Examples

- **Observer Pattern**:
  - decoupling of model from view
  - changes to one object can affect multiple views, without requiring object to know details of view
- **Composite Pattern**:
  - nesting of views
  - class hierarchy in which some subclasses define primitive objects and other classes define composite objects that assemble primitives into more complex objects
- **Strategy Pattern**:
  - view-controller relationship allows controller to replaced statically or dynamically

What don’t these design patterns apply to?

- Concurrency
- Distributed programming
- Real-time programming
- Domain-specific patterns
- User interface design
- Device drivers
- Object-oriented DB