Design Patterns

SE 3109

Basics of OOP

Design Pattern

- Represent the best practices used by experienced object-oriented software developers.
- Design patterns are solutions to general problems that software developers faced during software development
- It describes the problem, the solution, when to apply the solution, and its consequences
- Also gives implementation hints and examples.

Design Pattern

- Problem
  - Suppose in a project may contains many classes only a single instance (or object) should be created for a class and that single object can be used by all other classes.
- Solution
  - Singleton design pattern is the best solution of above specific problem.
  - Every design pattern has some specification or set of rules for solving the problems
- design patterns are programming language independent strategies for solving the common object-oriented design problems

Design Pattern

- Design pattern represents an idea, not a particular implementation
- It makes code more flexible, reusable and maintainable
- Gang of Four (GOF)
  - In 1994, four authors Erich Gamma, Richard Helm, Ralph Johnson and John Vlissides published a book titled Design Patterns - Elements of Reusable Object-Oriented Software which initiated the concept of Design Pattern in Software development

Benefit

- Common platform for developers
- Best Practices

Design pattern vs algorithm

- A design pattern is a general guideline for how to go about writing and organizing a piece of code.

- An algorithm is a specific set of steps that can be used to solve a problem.

Right time to use design pattern

- During the analysis and requirement phase of SDLC(Software Development Life Cycle)
- Design patterns ease the analysis and requirement phase of SDLC by providing information

Type of pattern

- Creational design patterns
  - These design patterns are all about class instantiation.
  - This pattern can be further divided into class-creation patterns and object-creational patterns.
- Structural design patterns
  - These design patterns are all about Class and Object composition.
  - Structural object-patterns define ways to compose objects to obtain new functionality.
- Behavioral design patterns
  - These design patterns are all about Class's objects communication

Lets have a journey with the following problem

- There are few soldiers in a war & the soldiers fighting mode change according to a command by a commander. The modes like aggressive mode, defensive mode etc. Now solve it.

Strategy Pattern

- A Strategy Pattern says that "**defines a family of functionality, encapsulate each one, and make them interchangeable**".
- It provides a substitute to subclassing.
- It defines each behavior within its own class, eliminating the need for conditional statements.
- It makes it easier to extend and incorporate new behavior without changing the application.
- Enables an algorithm’s behavior to be selected at runtime
- Usage:
  - When the multiple classes differ only in their behaviors.e.g. Servlet API.
  - It is used when you need different variations of an algorithm.

Solution

- The Strategy pattern suggests that you take a class that does something specific in a lot of different ways and extract all of these algorithms into separate classes called strategies.

Advantages

- A family of algorithms can be defined as a class hierarchy and can be used interchangeably to alter application behavior without changing its architecture.
- By encapsulating the algorithm separately, new algorithms complying with the same interface can be easily introduced.
- The application can switch strategies at run-time.
- Strategy enables the clients to choose the required algorithm, without using a “switch” statement or a series of “if-else” statements.
- Data structures used for implementing the algorithm are completely encapsulated in Strategy classes. Therefore, the implementation of an algorithm can be changed without affecting the Context class.