OO Design Patterns

Creational, Structural, and Behavioral Patterns

Presented by: T.Tharshana.

Department of Computing and Information Systems

Design Patterns

- Design patterns are typical solutions to common problems in software design.
- Design patterns are divided into 3 Categories:

  - ✓ Creational Patterns.
  - ✓ Structural Patterns.
  - ✓ Behavioral Patterns.

Creational Patterns

Creational patterns provide various object creation mechanisms, which increase flexibility and reuse of existing code.

- Focus on object creation.

Key Points:

- Ensures a class has only one instance.

- Provides a global point of access to it.

Example:

- Singleton Pattern
- Factory Method Pattern
- Abstract Factory Pattern
- Builder
- Prototype

Singleton Pattern

The singleton Pattern Ensures a class has only one instance and provides a global point of access to it.

Purpose:

- Control access to a single instance of a class.
- Prevent multiple instances of the class from being created, which could lead to inconsistent behavior or resource contention.

Implementation Details:

- Private Constructor: Prevents direct instantiation of the class from outside the class.
- Static Instance: Holds the single instance of the class.
- Public Static Method: Provides a way to access the instance, often called getInstance().

Structural Patterns

Structural patterns explain how to assemble objects and classes into larger structures while keeping these structures flexible and efficient.

- Deal with object composition.
- Example: Adapter Pattern

Key Points:

- Allows objects with incompatible interfaces to collaborate.

- Converts the interface of a class into another interface clients expect.

Example:

- Bridge patterns
- Composite patterns
- Facade patterns
- Proxy

Bridge Pattern

The Bridge Pattern is a structural design pattern that separates an object's abstraction from its implementation so that the two can vary independently. This pattern is useful when both the abstractions and their implementations need to be extended using inheritance

Purpose:

- To decouple an abstraction from its implementation.
- To allow the abstraction and implementation to evolve independently.

Behavioral Patterns

Behavioral patterns are concerned with algorithms and the assignment of responsibilities between objects

- Focus on object communication.
- Example: Observer Pattern

Key Points:

- Defines a one-to-many dependency between objects.

- When one object changes state, all its dependents are notified and updated automatically.

Example:

- Interpreter
- Mediator
- Iterator
- Observer

Conclusion

- Design patterns are essential for building robust, reusable, and maintainable software.

Thank you