What Are Design Patterns And How To Use Them

Juan Sebastian Rodriguez Trujillo
Cali, Colombia
jsrodriguez.user@gmail.com

***Abstract*— The following article explores the three main categories of design patterns: creational, structural, and behavioral. Examples and code snippets are provided in each category to illustrate the implementation of each of these design patterns.**

I. INTRODUCTION

This article is an introduction to creational, structural, and behavioral design patterns to understand their fundamental concepts. Understanding and applying these principles allows developers to increase code quality, optimize development processes, and efficiently approach software design. The exercises proposed in this article can be consulted in the following link: DesignPatternsProject.

TABLE I

Design Patterns	
Creational design patterns provide various object creation mechanisms, which increase flexibility and reuse of existing code [1]	• Factory Method • Prototype • Singleton
Structural design patterns explain how to assemble objects and classes into larger structures, while keeping these structures flexible and efficient [2]	• Adapter • Facade • Bridge
Behavioral design patterns are concerned with algorithms and the assignment of responsibilities between objects [3]	• Mediator • Observer • Strategy

II. FACTORY METHOD

Factory Method is a creational design pattern that provides an interface for creating objects in a superclass, but allows subclasses to alter the type of objects that will be created. [1] **Code:** Factory Method

III. PROTOTYPE

Prototype is a creational design pattern that lets you copy existing objects without making your code dependent on their classes. [1] **Code:** Prototype

IV. SINGLETON

Singleton is a creational design pattern that lets you ensure that a class has only one instance, while providing a global access point to this instance. [1] **Code:** Singleton

V. ADAPTER

Adapter is a structural design pattern that allows objects with incompatible interfaces to collaborate. [2] **Code:** Adapter

VI. FACADE

Facade is a structural design pattern that provides a simplified interface to a library, a framework, or any other complex set of classes. [2] **Code:** Facade

VII. BRIDGE

Bridge is a structural design pattern that lets you split a large class or a set of closely related classes into two separate hierarchies (abstraction and implementation) which can be developed independently of each other. [2] **Code:** Bridge

VIII. MEDIATOR

Mediator is a behavioral design pattern that lets you reduce chaotic dependencies between objects. The pattern restricts direct communications between the objects and forces them to collaborate only via a mediator object. [3] **Code:** Mediator

IX. OBSERVER

Observer is a behavioral design pattern that lets you define a subscription mechanism to notify multiple objects about any events that happen to the object they're observing. [3] **Code:** Observer

X. STRATEGY

Strategy is a behavioral design pattern that lets you define a family of algorithms, put each of them into a separate class, and make their objects interchangeable. [3] **Code:** Strategy

XI. CONCLUSIONS

Design patterns are important tools that help developers solve specific problems. Each of these tools has its own strengths and weaknesses, making it crucial to assess the context of the problem in order to choose the most suitable design pattern for implementation.

REFERENCES

[1] **Refactoring Guru.** https://refactoring.guru/design-patterns/creational-patterns. July, 2023.

[2] **Refactoring Guru.** https://refactoring.guru/design-patterns/structural-patterns. July, 2023.

[3] **Refactoring Guru.** https://refactoring.guru/design-patterns/behavioral-patterns. July, 2023.