

**JOMO KENYATTA UNIVERSITY OF AGRICULTURE AND TECHNOLOGY.**

**M.Sc. SOFTWARE ENGINEERING**

**ICS 3105: - OBJECT ORIENTED SOFTWARE ENGINEERING.**

**ASSIGNMENT 2**

**Dr Eunice Njeri.**

**ALEX KEMBOI SCT 313-0530/2023 OBJECT ORIENTED SOFTWARE ENGINEERING.**

A Research proposal submitted to Jomo Kenyatta University of Agriculture and Technology, School of Computing. Partial fulfillment of the requirement for the award of degree of Masters of Science in Software Engineering.

2025

**Design Patterns**

Design patterns are reusable solutions to common software design problems. They help make code more **maintainable, scalable, and flexible**.

**Types of Design Patterns**

Design patterns are categorized into **three main types**:

1. **Creational Patterns** – Handle object creation.
2. **Structural Patterns** – Organize and structure code.
3. **Behavioral Patterns** – Manage communication between objects.

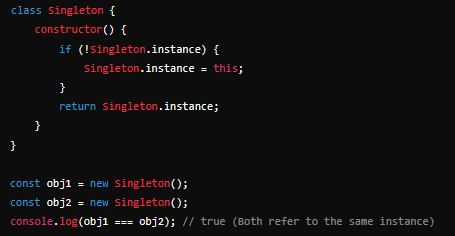
**1. Creational Patterns**

Creational patterns focus on the best ways to create objects.

**1.1 Singleton Pattern**

Ensures that only one instance of a class is created.

**Example (Singleton in JavaScript)**



## ****1.2 Factory Pattern****

Encapsulates object creation logic in a separate function.

### ****Example (Factory Pattern in JavaScript)****



## ****1.3 Builder Pattern****

Separates object construction from representation, allowing step-by-step object creation.

### ****Example (Builder Pattern in JavaScript)****



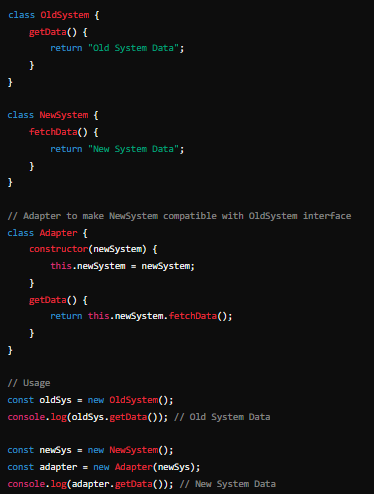
# ****2. Structural Patterns****

Structural patterns deal with organizing classes and objects to form larger structures.

## ****2.1 Adapter Pattern****

Converts one interface into another expected by the client.

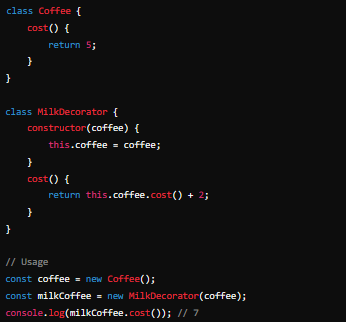
### ****Example (Adapter Pattern in JavaScript)****



## ****2.2 Decorator Pattern****

Dynamically adds behavior to objects without modifying their structure.

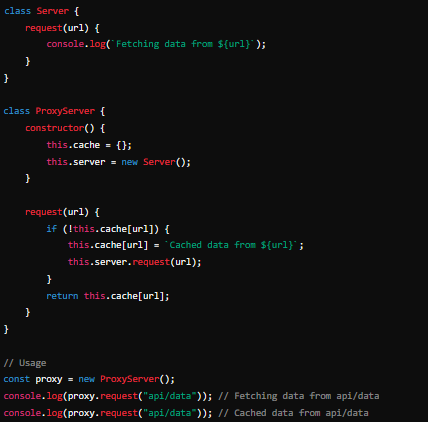
### ****Example (Decorator Pattern in JavaScript)****



## ****2.3 Proxy Pattern****

Controls access to an object, adding security or performance optimizations.

### ****Example (Proxy Pattern in JavaScript)****



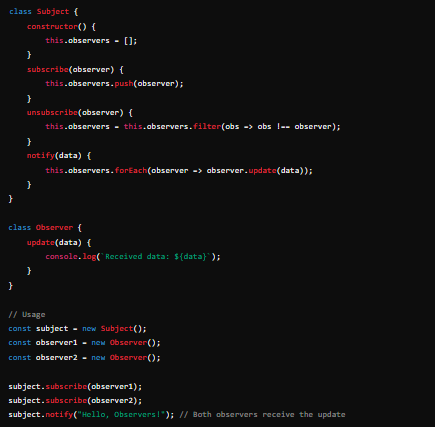
# ****3. Behavioral Patterns****

Behavioral patterns manage communication between objects.

## ****3.1 Observer Pattern****

Allows multiple objects to react to changes in another object.

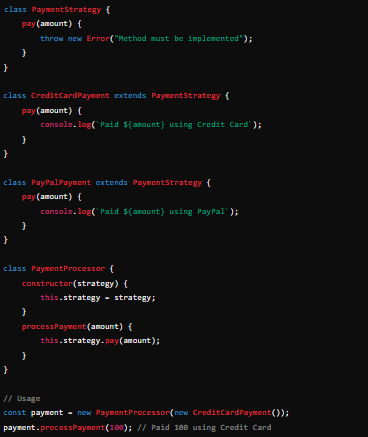
### ****Example (Observer Pattern in JavaScript)****



## ****3.2 Strategy Pattern****

Defines a family of algorithms and makes them interchangeable.

### ****Example (Strategy Pattern in JavaScript)****



**GIT HUB LINK**

[**https://github.com/alexkemboi/Design-Patterns.git**](https://github.com/alexkemboi/Design-Patterns.git)