# UGANDA CHRISTIAN UNIVERSITY

A Centre of Excellence in the Heart of Africa

| | | |
|---|---|---|
| **Name** | : | KAUTA MARVIN |
| **Registration Number** | : | S20B23/204 |
| **Faculty/School** | : | SCIENCE AND TECHNOLOGY |
| **Course** | : | BACHOLERS OF SCIENCE IN COMPUTER SCIENCE |
| **Lecturer** | : | ***MR LUBAMBO SIMON*** |

Design patterns are reusable solutions to commonly occurring software design problems. They provide a standardized way to solve a particular problem, and can improve the quality, maintainability, and scalability of software.

Design patterns can be divided into three categories:

1. Creational
2. Structural
3. behavioral.

Creational patterns focus on object creation mechanisms, while structural patterns deal with object composition, and behavioral patterns define how objects interact and communicate with one another.

Some common design patterns include the

1. Factory pattern
2. Singleton pattern
3. Observer pattern
4. Command pattern.

Each pattern is designed to solve a particular problem or address a specific aspect of software design.

**Other design patterns include.**

1. Factory pattern: The Factory pattern provides a way to create objects without specifying the exact class of object that will be created. It defines an interface or abstract class for creating objects, but allows subclasses to decide which class to instantiate.
2. Abstract factory pattern: The Abstract Factory pattern provides an interface for creating families of related objects. It allows the client to create objects without knowing the specific classes of objects it is creating.
3. Singleton pattern: The Singleton pattern ensures that a class has only one instance and provides a global point of access to it. It is commonly used for resources that are expensive to create or when only one instance is needed for coordination.
4. Observer pattern: The Observer pattern defines a one-to-many relationship between objects, so that when one object changes state, all its dependents are notified and updated automatically. It is commonly used for event handling and UI updates.
5. Strategy pattern: The Strategy pattern defines a family of algorithms, encapsulates each one, and makes them interchangeable. It allows the algorithm to vary independently from the client that uses it.
6. Decorator pattern: The Decorator pattern attaches additional responsibilities to an object dynamically, providing a flexible alternative to subclassing for extending functionality. It allows the behavior of an object to be modified without affecting the behavior of other objects of the same class.
7. Adapter pattern: The Adapter pattern converts the interface of a class into another interface that clients expect, allowing classes to work together that otherwise couldn't. It is commonly used when integrating existing code or libraries.
8. Template method pattern: The Template Method pattern defines the skeleton of an algorithm in a superclass, but allows subclasses to override specific steps of the algorithm without changing its structure. It is commonly used to define a high-level algorithm while allowing customization of its details by subclasses.
9. Command pattern: The Command pattern encapsulates a request as an object, allowing clients to parameterize different requests, queue or log requests, and support undoable operations. It is commonly used in UI design and to implement undo-redo functionality.