Definitions

Design pattern: is a general repeatable solution to a commonly occurring problem in software design

- It is not a finished design that can be transformed directly into code
- It is a description or template for how to solve a problem that can be used in many different situations
- Design patterns can speed up the development process by providing tested, proven development paradigms
- Allow developers to communicate using well-known, well-understood names for software interactions

Creational Design Patterns

They are patterns that deal with object creation mechanisms, trying to create objects in a manner suitable to the solution

Examples:

Name	Definition
Abstract Factory	Creates an instance of several families of classes

Builder	Separates object construction from its representation
Factory Method	Creates an instance of several derived classes
Object Pool	Avoid expensive acquisition and release of resources by recycling objects that are no longer in use
Prototype	A fully initialized instance to be copied or clone
Singleton	A class of which only a single instance can exist

Structural Design Patterns

Patterns that ease the design by identifying a simple way to realize relationships between entities

They explain how to assemble objects and classes into larger structures while keeping these structures flexible and efficient

Examples:

Name	Definition
Adapter	Match interfaces of different classes
Bridge	Separates an object’s interface from its implementation
Composite	A tree structure of simple and composite objects
Decorator	Add responsibilities to objects dynamically
Facade	A single class that represents an entire subsystem

Behavioral Design Patterns

Patterns that identify common communication patterns between objects and realize these patterns. By doing so, these patterns increase flexibility in carrying out this communication

Example:

Name	Definition
Chain of Responsibility	A way of passing a request between a chain of objects

Command	Encapsulate a command request as an object
Interpreter	A way to include language elements in a program
Iterator	Sequentially access the elements of a collection
Mediator	Defines simplified communication between classes
Observer	A way of notifying change to a number of classes