1. You are developing a game that features various characters, including warriors and mages. Each character type has unique abilities and characteristics. Implement the Factory Method design pattern to create instances of these characters. Define an interface Character with methods for getting the character's name and abilities. Create concrete classes for Warrior and Mage, both implementing the Character interface with their own unique abilities and names. Then, create a CharacterFactory class with factory methods to create characters of each type. Write a Java program to demonstrate the use of the Factory Method pattern by creating instances of warriors and mages and displaying their names and abilities.

   Character should have methods getName() and getAbilities()

   Observer design pattern: https://refactoring.guru/design-patterns/observer

2. You are developing a weather monitoring application that needs to notify multiple displays whenever the weather conditions change. Implement the Observer design pattern to achieve this functionality. Define a Subject interface with methods for registering, removing, and notifying observers. Create a concrete WeatherStation class that implements the Subject interface and simulates weather updates. WeatherDisplay and PhoneApp are the observers which displays the weather whenever the weather station updates the weather. Write a Java program to demonstrate the use of the Observer pattern by registering the WeatherDisplay and PhoneApp as observers and simulating weather updates.
3. You are developing a messaging application, and you want to implement the Observer design pattern to notify users when new messages arrive. Implement a basic version of the Observer pattern. Define MessageService class with methods for registering, removing, and notifying observers while sending messages. Define an Observer interface with a method for receiving and displaying new messages. Create a concrete class User that implements the Observer interface and displays received messages. Write a Java program to demonstrate the use of the Observer pattern by registering a User as an observer and simulating the arrival of new messages.

   Decorator design pattern: https://www.geeksforgeeks.org/decorator-design-pattern-in-java-with-example/
4. You are designing a flower bouquet arrangement application and want to implement the Decorator design pattern to create different types of bouquets with various features. Define a base interface Bouquet with a method getDescription to get the description of the bouquet. Create a concrete class BasicBouquet that implements the Bouquet interface, representing a simple bouquet of flowers. Implement the Decorator pattern to create two decorators: RibbonDecorator and FragranceDecorator. These decorators should add additional features to the bouquet. The additional feature each decorator adds should get reflected in the getDescription method of the decorator.