**Exercise 1:** Problem Statement on Design patterns Come up creatively with six different use cases to demonstrate your understanding of the following software design patterns by coding the same.

1. Two use cases to demonstrate two behavioural design pattern.

2. Two use cases to demonstrate two creational design pattern.

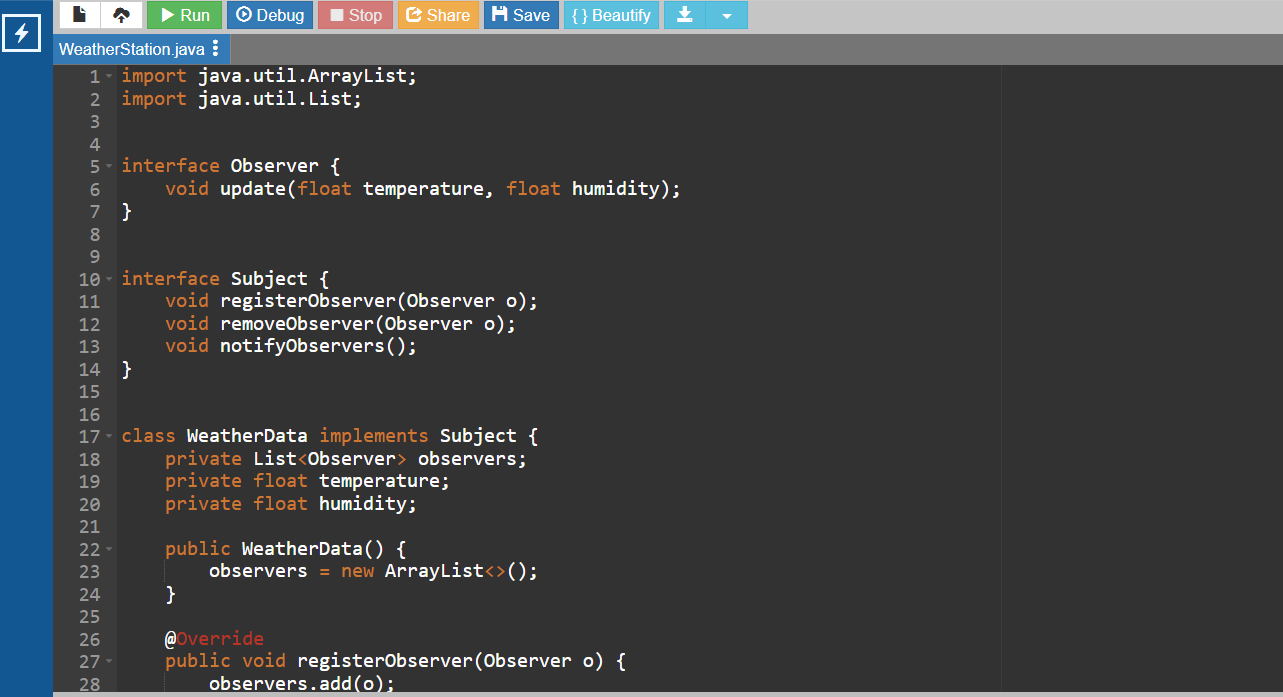
3. Two use cases to demonstrate two structural design pattern.

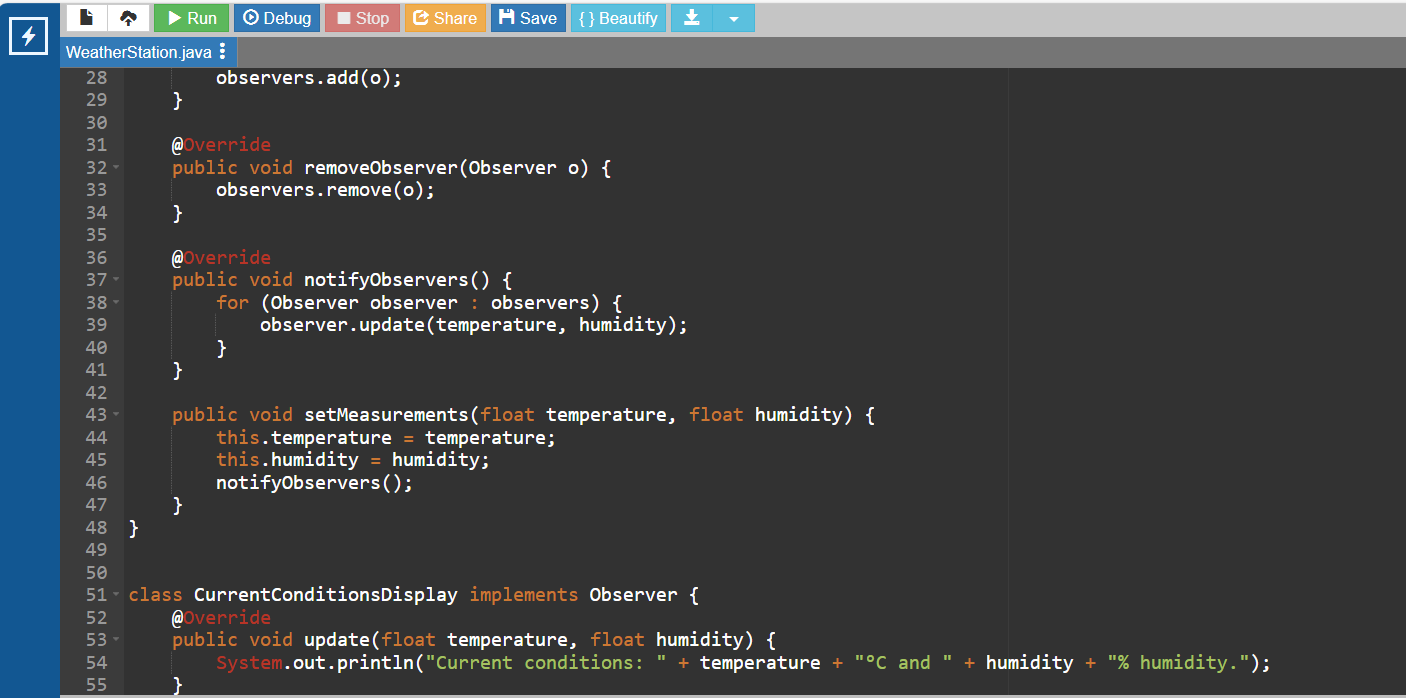
**1. Two use cases to demonstrate two behavioural design pattern.**

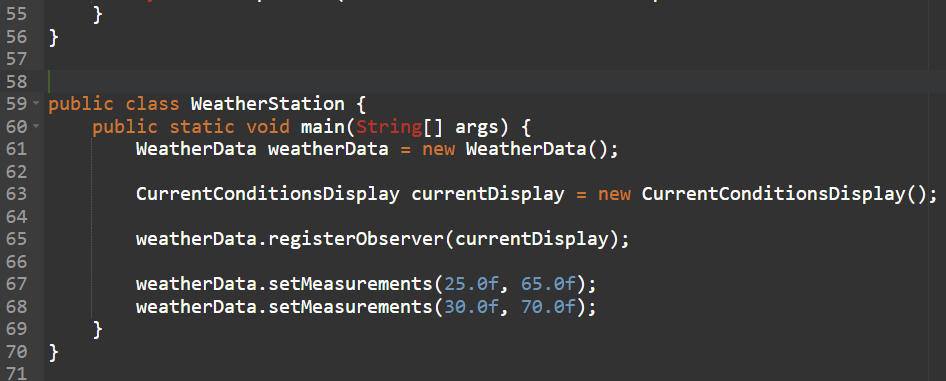
**1. Observer Pattern**

Use Case: A weather station that notifies various display elements (like temperature and humidity displays) whenever there's a change in weather data.

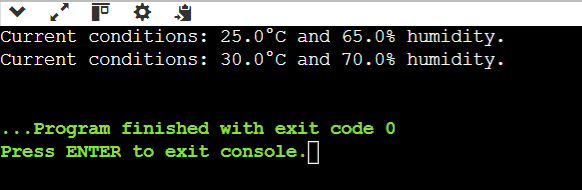
**CODE:**

****

****

****

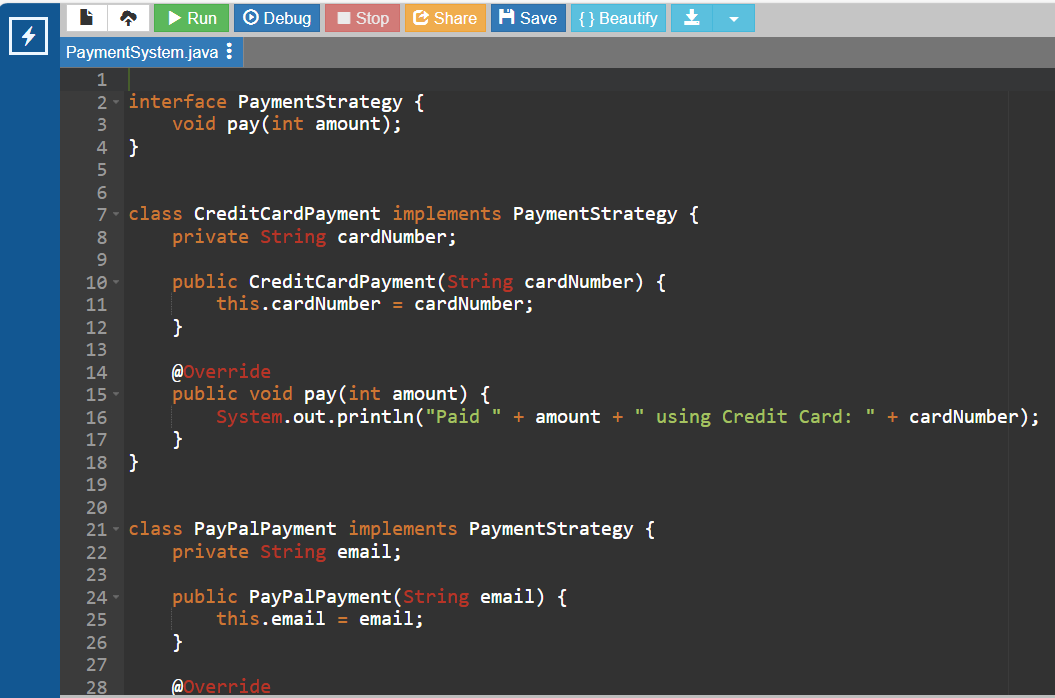
**OUTPUT:**

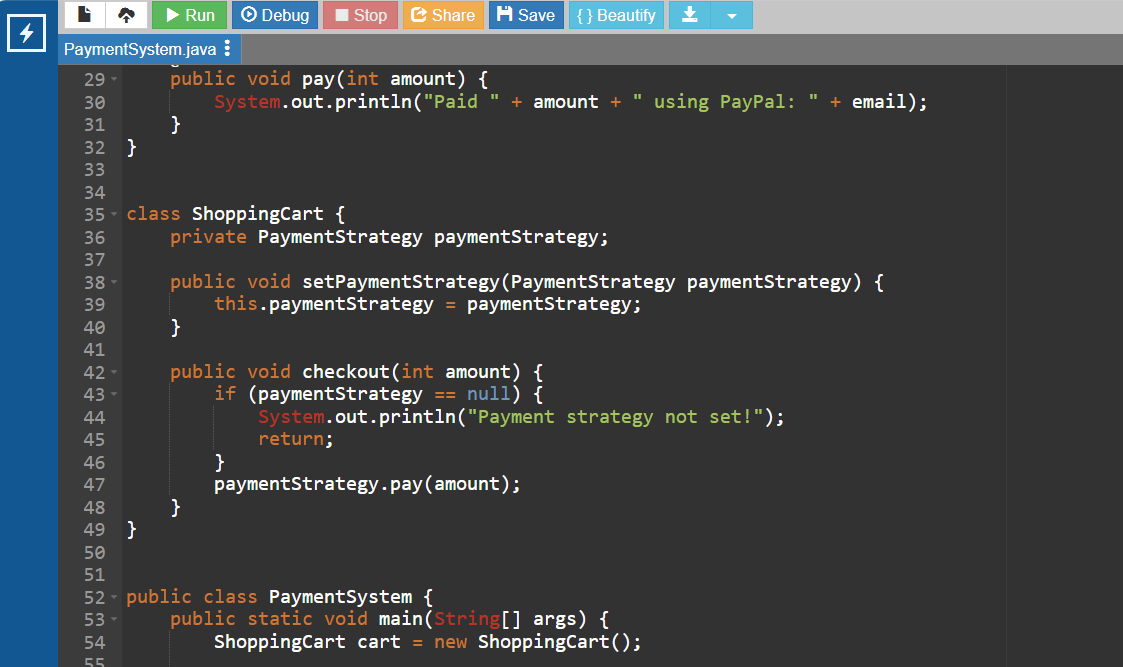
****

**2. Strategy Pattern**

Use Case: A payment system that can use different payment strategies (like Credit Card, PayPal) based on user preference.

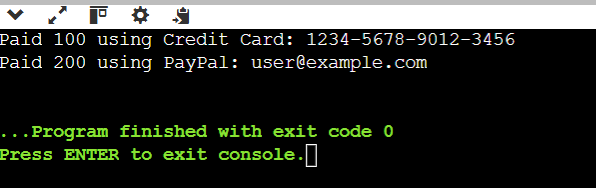
**CODE:**







**OUTPUT:**

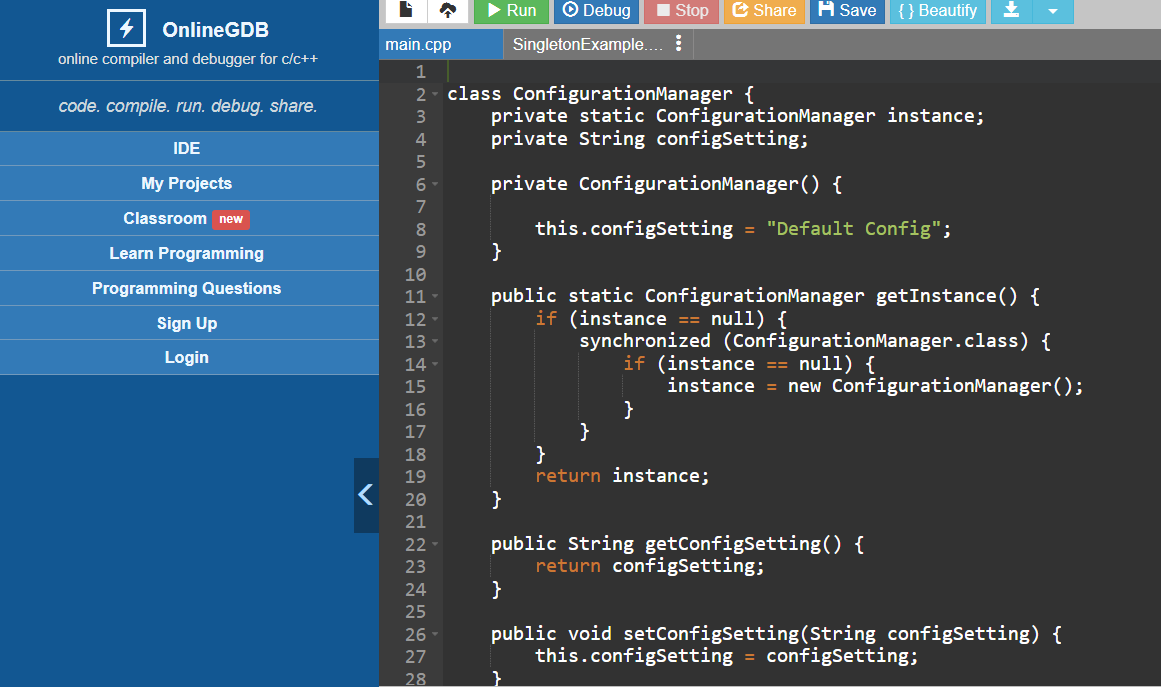


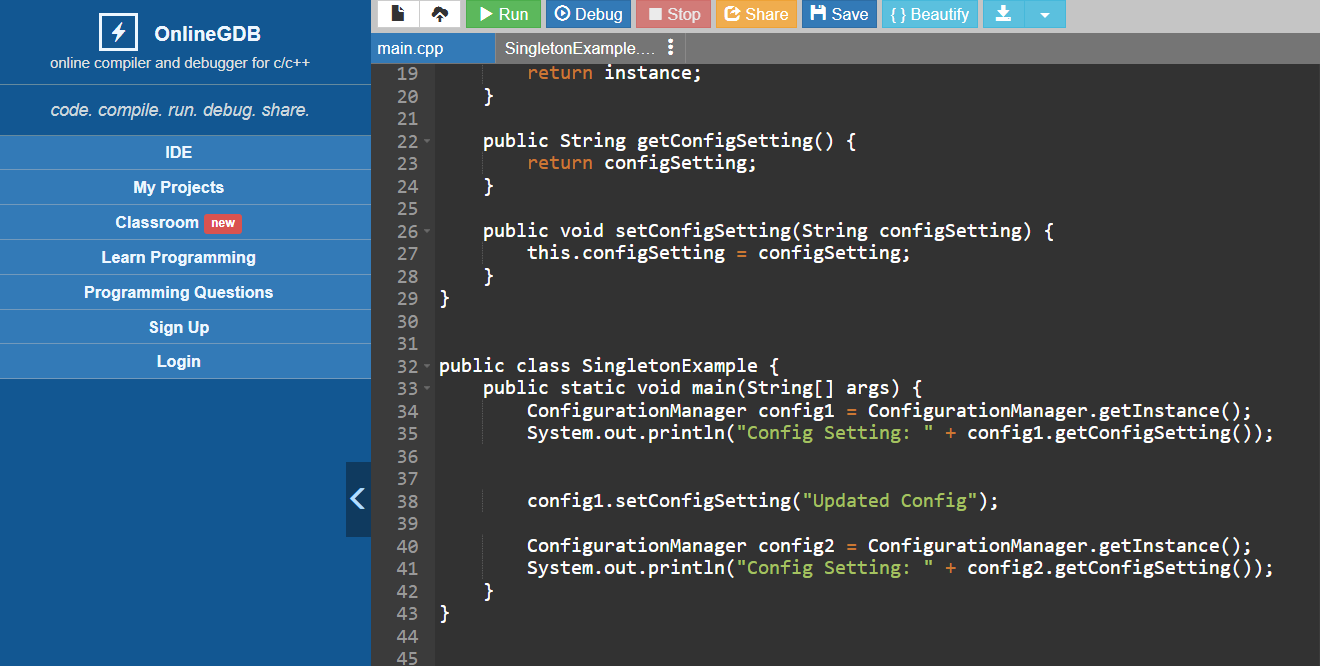
**2. Two use cases to demonstrate two creational design pattern.**

**1. Singleton Pattern**

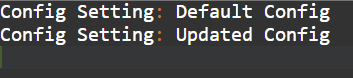
Use Case: A configuration manager that ensures there is only one instance throughout the application.

**CODE:**

****

****

**OUTPUT:**

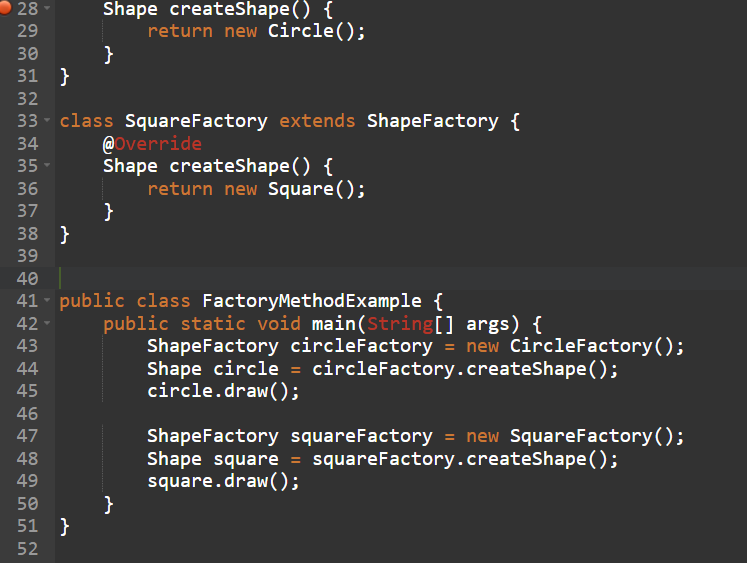


**2. Factory Method Pattern**

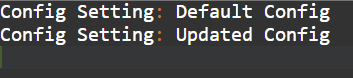
Use Case: A shape factory that creates different shapes based on input, allowing for flexibility and extensibility.

**CODE:**





**OUTPUT:**

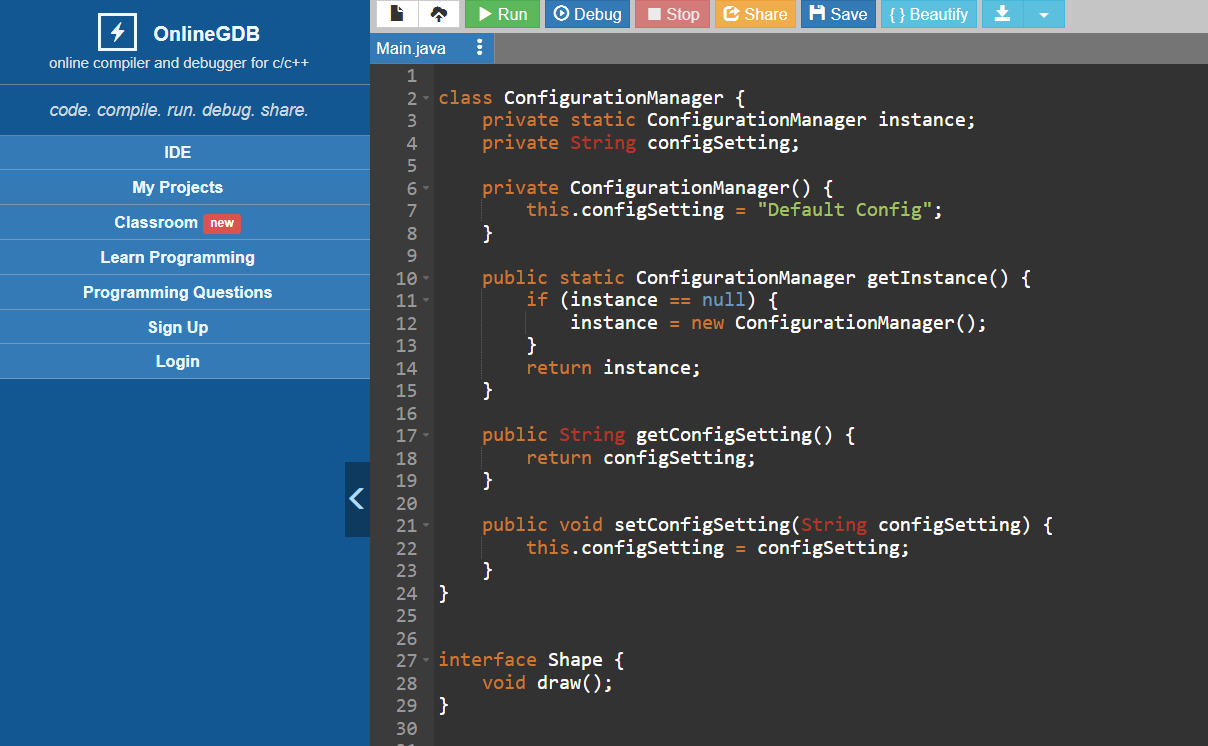
**:**

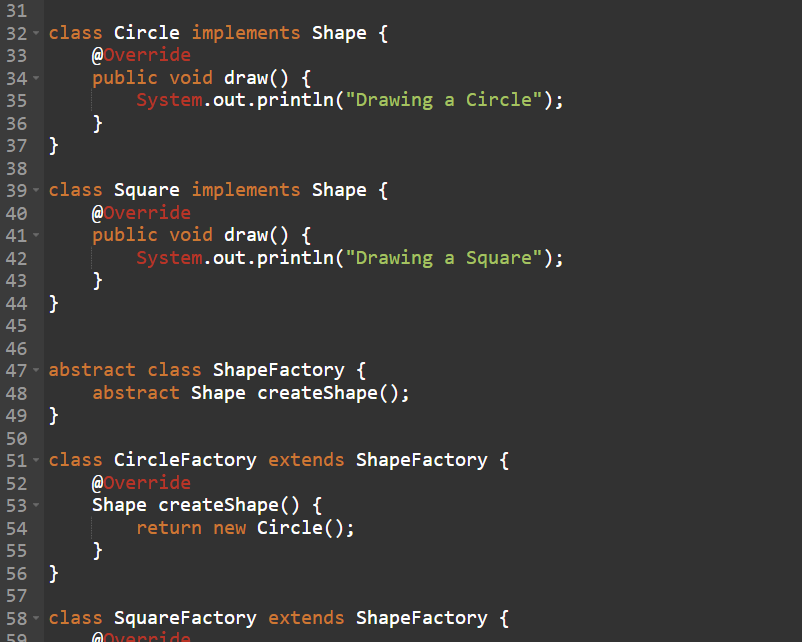
**3. Two use cases to demonstrate two structural design pattern.**

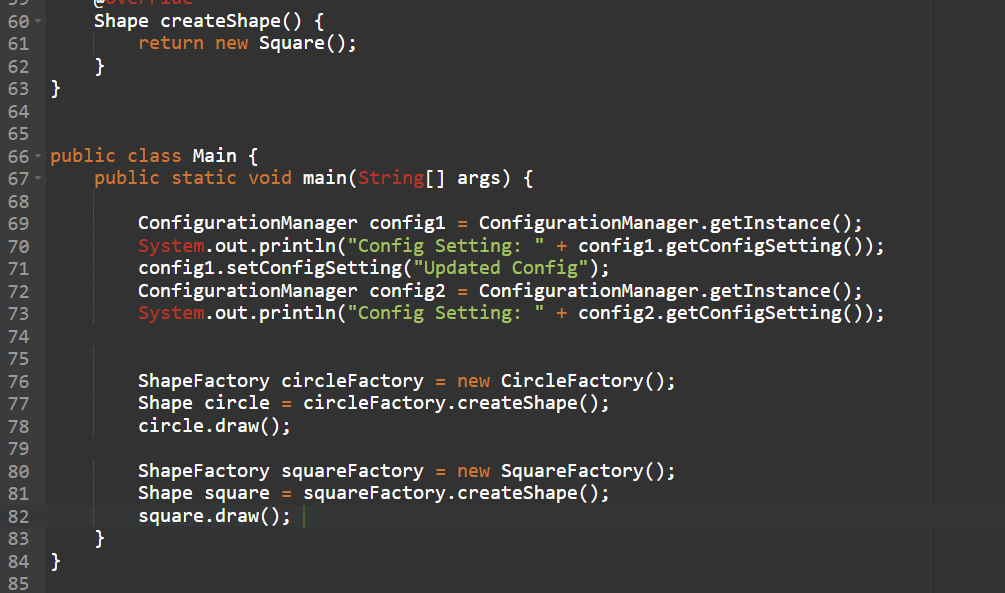
**Use Case 1: Adapter Pattern**

The Adapter Pattern allows incompatible interfaces to work together.

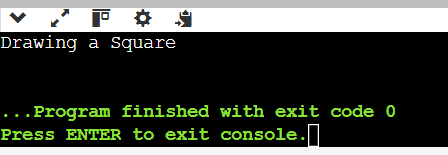
**CODE:**







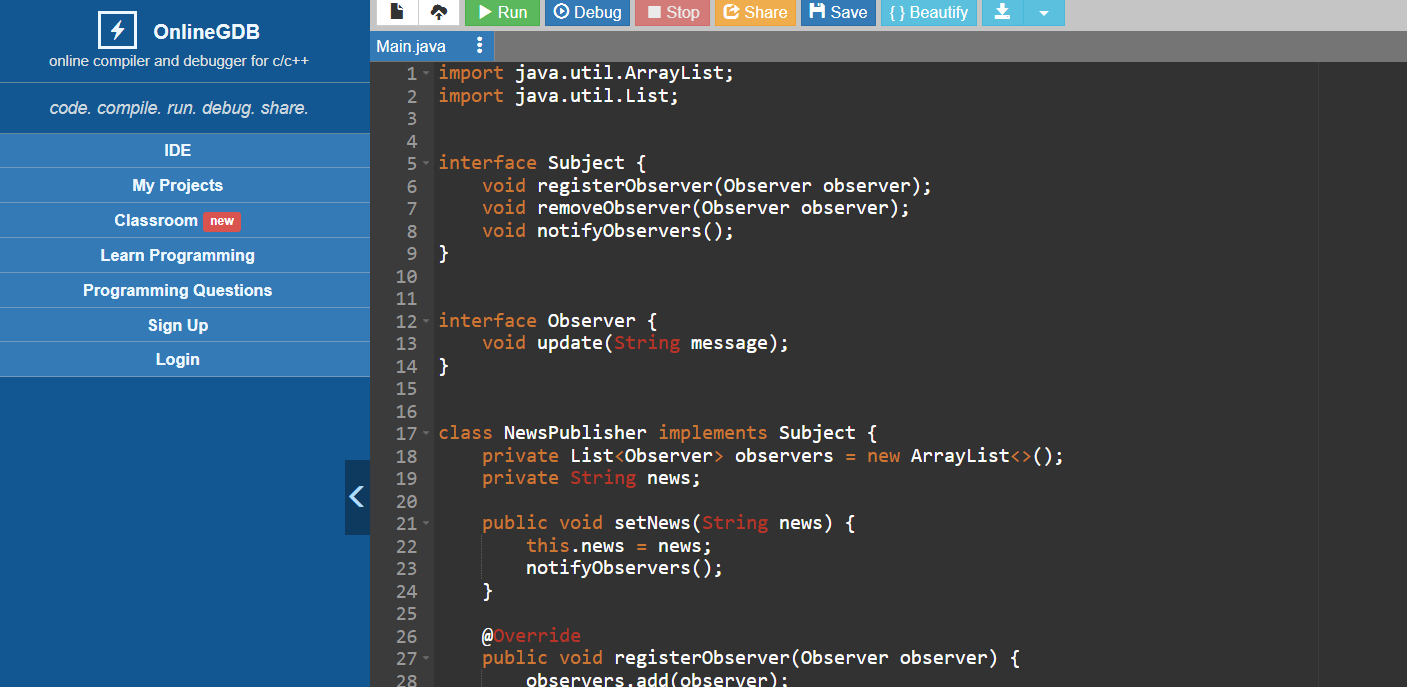
**OUTPUT:**

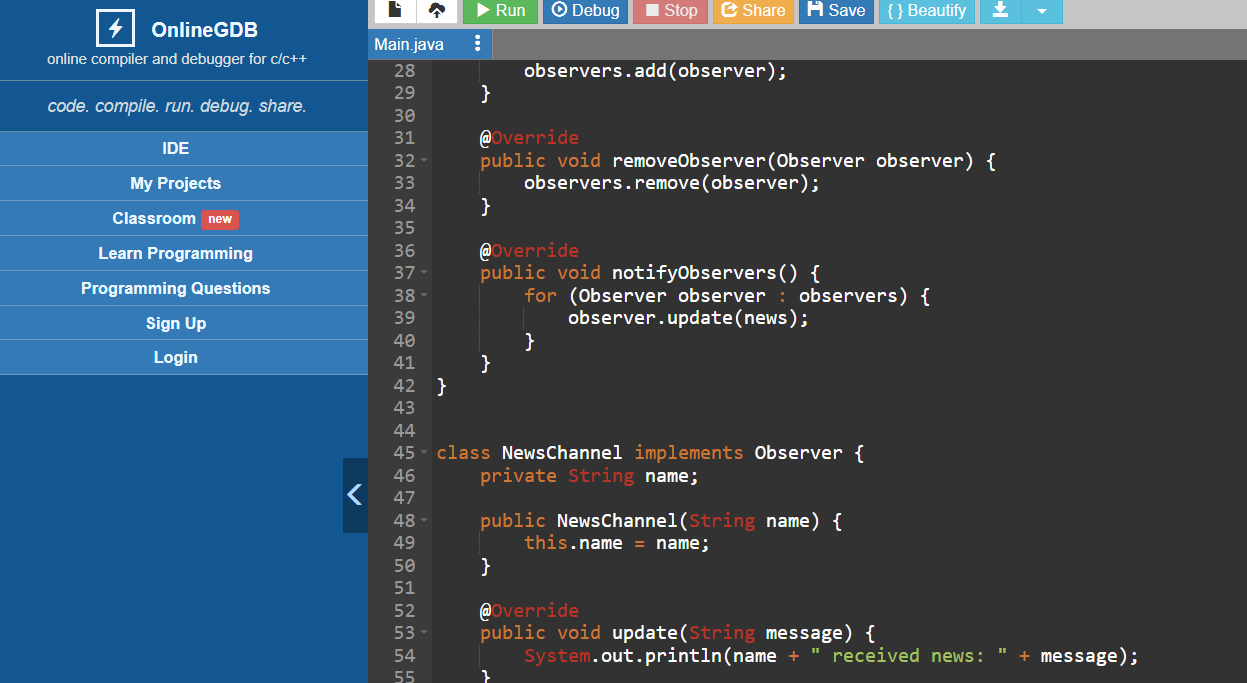


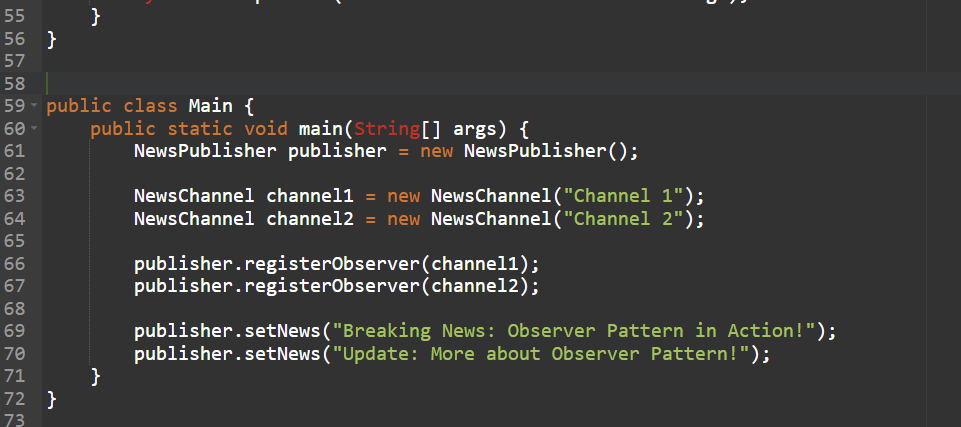
**Use Case 2: Decorator Pattern**

The Decorator Pattern allows behavior to be added to individual objects, either statically or dynamically, without affecting the behavior of other objects.

**CODE:**







**OUTPUT:**

