```java
//Structural pattern

//Bride pattern

// Abstraction

   interface Shape {

    void draw();

}

// Refined Abstraction

   class Circle implements Shape {

    private Color color;

    Circle (Color color) {

        this.color = color;

    }

    public void draw() {

        System.out.print("Drawing Circle in  ");
        color.fill();
    }

}

// Implementor
     interface Color {
    void fill();

    }
// Concrete Implementor

    class Red implements Color {

        public void fill() {
            System.out.println("Red");
        }

        }
```

```java
//Structural pattern

//Bride pattern

public class Bride_pattern {

    public static void main(String[] args) {
```

```
        Shape circle = new Circle(new Red()

        );
        circle.draw();
        // Output: Drawing Circle in Red

    }

}
```

```
// Behavioral pattern

//Adapter pattern

interface MediaPlayer {

    void play (String audioType, String fileName);

}

class AudioPlayer implements MediaPlayer {

    public void play (String audioType, String fileName) {

        if (audioType.equalsIgnoreCase("mp3")) {

            System.out.println("Playing mp3 file: " + fileName);

        }
        else if (audioType.equalsIgnoreCase("vlc")) {

            System.out.println("Cannot play vlc file: " + fileName);
        }

    }

}

class MediaAdapter extends AudioPlayer {


    public void play(String audioType, String fileName) {

        if (audioType.equalsIgnoreCase("vlc")) {

        super.play("mp3", fileName);
```

```
    }
  }
}
```

```
// Behavioral pattern
//Adapter pattern
public class Adapter_pattern {
    public static void main(String[] args) {
        MediaPlayer mediaPlayer = new MediaAdapter();
        mediaPlayer.play("vlc", "song1.vlc");
    }
}
```

```
//creational pattern
//Singleton pattern
public class Singleton {
    private static Singleton instance;
    private String value;
    private Singleton (String value) {
        this.value = value;
    }
    public static Singleton getInstance(String value) {
        if (instance == null) {
            instance = new Singleton(value);
        }
            return instance;
```

```
    }

    public String getValue() {

        return value;

    }

    public void setValue(String value) {

        this.value = value;

    }


    public static void main(String[] args) {

        Singleton singleton1 = Singleton.getInstance("First Instance");

        System.out.println(singleton1.getValue());
        // Output: First Instance

        Singleton singleton2 = Singleton.getInstance("Second Instance");

        System.out.println(singleton2.getValue());
        // Output: First Instance

        singleton2.setValue("Updated Instance");

        System.out.println(singleton1.getValue());
        // Output: Updated Instance
        }

    }
```