**Exercise 1: Implementing the Singleton Pattern**

public class Singleton {

private static Singleton instance;

private Singleton() {

System.out.println("Singleton instance created.");

}

public static Singleton getInstance() {

if (instance == null) {

instance = new Singleton();

}

return instance;

}

public void showMessage() {

System.out.println("Hello from Singleton!");

}

public static void main(String[] args) {

Singleton s1 = Singleton.getInstance();

Singleton s2 = Singleton.getInstance();

s1.showMessage();

if (s1 == s2) {

System.out.println("Both instances are the same (Singleton confirmed).");

} else {

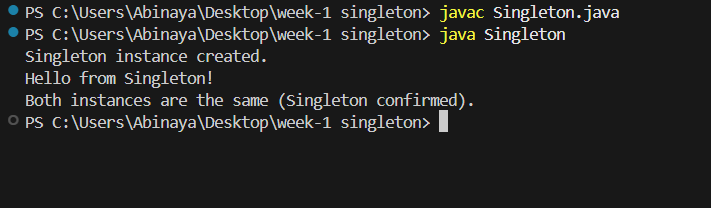
System.out.println("Different instances (Singleton failed).");

}

}

}

**Output:**



**Exercise 2: Implementing the Factory Method Pattern**

interface Animal {

void speak();

}

class Dog implements Animal {

public void speak() {

System.out.println("Dog says: Woof Woof!");

}

}

class Cat implements Animal {

public void speak() {

System.out.println("Cat says: Meow Meow!");

}

}

abstract class AnimalFactory {

public abstract Animal createAnimal();

}

class DogFactory extends AnimalFactory {

public Animal createAnimal() {

return new Dog();

}

}

class CatFactory extends AnimalFactory {

public Animal createAnimal() {

return new Cat();

}

}

public class FactoryPatternDemo {

public static void main(String[] args) {

AnimalFactory dogFactory = new DogFactory();

Animal dog = dogFactory.createAnimal();

dog.speak();

AnimalFactory catFactory = new CatFactory();

Animal cat = catFactory.createAnimal();

cat.speak();

}

}

**OUTPUT:**

