Exercise 1: Implementing the Singleton Pattern

public class Singleton {

private static Singleton instance;

private Singleton() {

}

public static Singleton getInstance() {

if (instance == null) {

synchronized (Singleton.class) {

if (instance == null) {

instance = new Singleton();

}

}

}

return instance;

}

public void showMessage() {

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

}

public static void main(String[] args) {

Singleton obj = Singleton.getInstance();

obj.showMessage();

}

}

Exercise 2: Implementing the Factory Method Pattern

interface Product {

void use();

}

class ConcreteProductA implements Product {

public void use() {

System.out.println("Using Product A");

}

}

class ConcreteProductB implements Product {

public void use() {

System.out.println("Using Product B");

}

}

abstract class Creator {

public abstract Product factoryMethod();

public void someOperation() {

Product product = factoryMethod();

product.use();

}

}

class ConcreteCreatorA extends Creator {

public Product factoryMethod() {

return new ConcreteProductA();

}

}

class ConcreteCreatorB extends Creator {

public Product factoryMethod() {

return new ConcreteProductB();

}

}

public class FactoryMethodPattern {

public static void main(String[] args) {

Creator creatorA = new ConcreteCreatorA();

creatorA.someOperation();

Creator creatorB = new ConcreteCreatorB();

creatorB.someOperation();

}

}