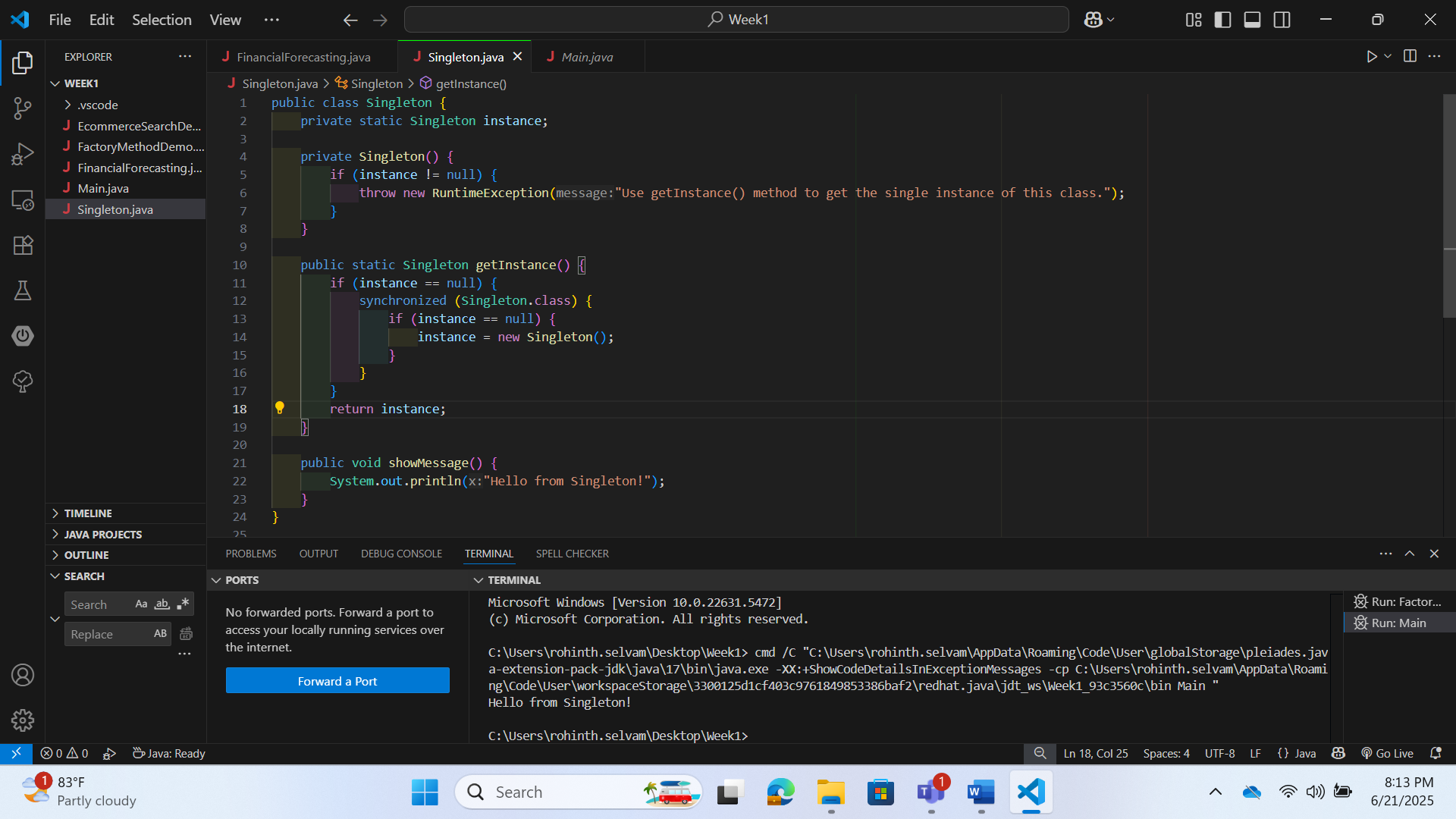
# Week1\_DesignPatternsandPrinciples\_HandsOn

## Exercise 1: Implement a Singleton Pattern

Code:

public class Singleton {  
 private static Singleton instance;  
 private Singleton() {  
 if (instance != null) {  
 throw new RuntimeException("Use getInstance() method to get the single instance of this class.");  
 }  
 }  
 public static Singleton getInstance() {  
 if (instance == null) {  
 synchronized (Singleton.class) {  
 if (instance == null) {  
 instance = new Singleton();  
 }  
 }  
 }  
 return instance;  
 }  
 public void showMessage() {  
 System.out.println("Hello from Singleton!");  
 }  
}  
  
public class Main {  
 public static void main(String[] args) {  
 Singleton obj = Singleton.getInstance();  
 obj.showMessage();  
 }  
}

**OUTPUT:  
**

------------------------------------------------------------

## Exercise 2: Implement a Factory Method Pattern

Code:

interface Product {  
 void use();   
}  
  
class ConcreteProductA implements Product {  
 public void use() {  
 System.out.println("Product A is being used.");  
 }  
}  
  
class ConcreteProductB implements Product {  
 public void use() {  
 System.out.println("Product B is being used.");  
 }  
}  
  
abstract class Creator {  
 public abstract Product createProduct();  
}  
  
class ConcreteCreatorA extends Creator {  
 public Product createProduct() {  
 return new ConcreteProductA();   
 }  
}  
  
class ConcreteCreatorB extends Creator {  
 public Product createProduct() {  
 return new ConcreteProductB();   
 }  
}  
  
public class FactoryMethodDemo {  
 public static void main(String[] args) {  
 Creator creatorA = new ConcreteCreatorA();  
 Product productA = creatorA.createProduct();  
 productA.use();   
  
 Creator creatorB = new ConcreteCreatorB();  
 Product productB = creatorB.createProduct();  
 productB.use();   
 }  
}

**OUTPUT:  
A screenshot of a computer screen

AI-generated content may be incorrect.**