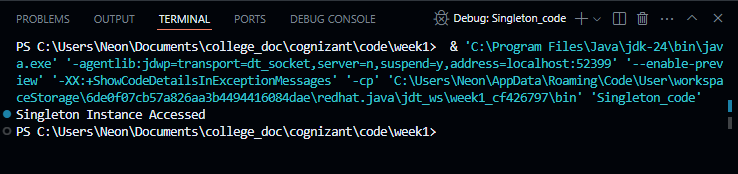
Week 1: Design Principles and Patterns - Hands-On

# Exercise 1: Implementing the Singleton Pattern

## Code:

public class Singleton\_code {  
 private static Singleton\_code instance;  
  
 private Singleton\_code() {  
 // private constructor  
 }  
  
 public static Singleton\_code getInstance() {  
 if (instance == null) {  
 instance = new Singleton\_code();  
 }  
 return instance;  
 }  
  
 public void showMessage() {  
 System.out.println("Singleton Instance Accessed");  
 }  
  
 public static void main(String[] args) {  
 Singleton\_code obj = Singleton\_code.getInstance();  
 obj.showMessage();  
 }  
}

## Output Screenshot:

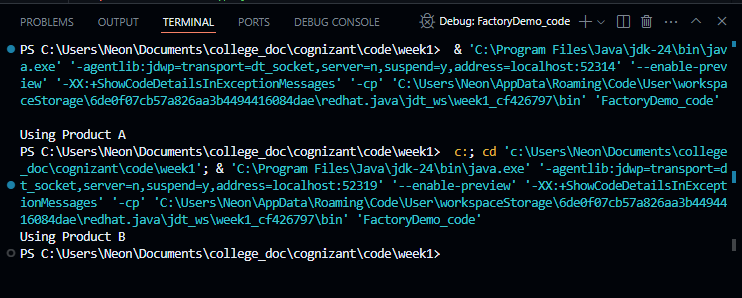


# Exercise 2: Implementing the Factory Method Pattern

## Code:

interface Product {  
 void use();  
}  
  
class ConcreteProductA implements Product {  
 @Override  
 public void use() {  
 System.out.println("Using Product A");  
 }  
}  
  
class ConcreteProductB implements Product {  
 @Override  
 public void use() {  
 System.out.println("Using Product B");  
 }  
}  
  
class ProductFactory {  
 public static Product createProduct(String type) {  
 if (type.equalsIgnoreCase("A")) {  
 return new ConcreteProductA();  
 } else if (type.equalsIgnoreCase("B")) {  
 return new ConcreteProductB();  
 }  
 return null;  
 }  
}  
  
public class FactoryDemo\_code {  
 public static void main(String[] args) {  
 Product product = ProductFactory.createProduct("B");  
 product.use();  
 }  
}

## Output Screenshot:

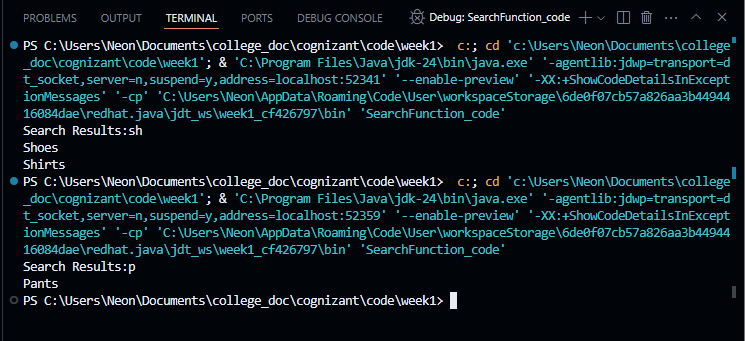


# Exercise 2: E-commerce Platform Search Function

## Code:

import java.util.\*;  
  
public class SearchFunction\_code {  
 public static List<String> search(List<String> products, String keyword) {  
 List<String> result = new ArrayList<>();  
 for (String product : products) {  
 if (product.toLowerCase().contains(keyword.toLowerCase())) {  
 result.add(product);  
 }  
 }  
 return result;  
 }  
  
 public static void main(String[] args) {  
 List<String> products = Arrays.asList("Shoes", "Shirts", "Pants", "Socks", "Hat");  
  
 String keyword = "p";  
 List<String> found = search(products, keyword);  
  
 System.out.println("Search Results:" + keyword);  
 for (String product : found) {  
 System.out.println(product);  
 }  
 }  
}

## Output Screenshot:



# Exercise 7: Financial Forecasting

## Code:

public class FinancialForecast\_code {  
 public static double forecastRevenue(double currentRevenue, double growthRate, int years) {  
 return currentRevenue \* Math.pow((1 + growthRate), years);  
 }  
  
 public static void main(String[] args) {  
 double currentRevenue = 100000; // 1 Lakh  
 double growthRate = 0.06; // 8%  
 int years = 4;  
  
 System.out.println("current revenue:" + currentRevenue);  
 System.out.println("growth rate:" + growthRate);  
 System.out.println("years:" + years);  
  
 double forecastedRevenue = forecastRevenue(currentRevenue, growthRate, years);  
 System.out.printf("Forecasted Revenue in %d years: %.2f\n", years, forecastedRevenue);  
 }  
}

## Output Screenshot:

