**Exercise 1: Implementing the Singleton Pattern**

public class Singleton {  
 private static Singleton *instance*;  
 private Singleton() {  
 System.*out*.println("Singleton instance has created");  
 }  
 public static synchronized Singleton getInstance() {  
 if (*instance* == null) {  
 *instance* = new Singleton();  
 }  
 return *instance*;  
 }  
 public void showMessage() {  
 System.*out*.println("Hello!");  
 }  
 public static void main(String[] args) {  
 Singleton obj1 = Singleton.*getInstance*();  
 obj1.showMessage();  
 Singleton obj2 = Singleton.*getInstance*();  
 System.*out*.println("Are both instances same? " + (obj1 == obj2));  
 }  
}

**#OUTPUT:**

Singleton instance has created

Hello!

Are both instances same? true

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

interface Notification {  
 void notifyUser();  
}  
class SMSNotification implements Notification {  
 public void notifyUser() {  
 System.*out*.println("Sending an SMS notification");  
 }  
}  
class EmailNotification implements Notification {  
 public void notifyUser() {  
 System.*out*.println("Sending an Email notification");  
 }  
}  
class PushNotification implements Notification {  
 public void notifyUser() {  
 System.*out*.println("Sending a Push notification");  
 }  
}  
class NotificationFactory {  
 public Notification createNotification(String type) {  
 if (type == null || type.isEmpty())  
 return null;  
 if (type.equalsIgnoreCase("SMS"))  
 return new SMSNotification();  
 if (type.equalsIgnoreCase("EMAIL"))  
 return new EmailNotification();  
 if (type.equalsIgnoreCase("PUSH"))  
 return new PushNotification();  
 return null;  
 }  
}  
public class FactoryMethodPatternDemo {  
 public static void main(String[] args) {  
 NotificationFactory factory = new NotificationFactory();  
  
 Notification notif1 = factory.createNotification("EMAIL");  
 notif1.notifyUser();  
  
 Notification notif2 = factory.createNotification("PUSH");  
 notif2.notifyUser();  
 }  
}

**#OUTPUT:**

Sending an Email notification

Sending a Push notification

**Exercise 2: E-commerce Platform Search Function**

package ECom;  
import java.util.Arrays;  
public class ECommerceSearch {  
 public static int linearSearch(String[] products, String target) {  
 for (int i = 0; i < products.length; i++) {  
 if (products[i].equalsIgnoreCase(target)) {  
 return i;  
 }  
 }  
 return -1;  
 }  
 public static int binarySearch(String[] products, String target) {  
 int left = 0;  
 int right = products.length - 1;  
  
 while (left <= right) {  
 int mid = left + (right - left) / 2;  
 int compare = products[mid].compareToIgnoreCase(target);  
  
 if (compare == 0) {  
 return mid;  
 } else if (compare < 0) {  
 left = mid + 1;  
 } else {  
 right = mid - 1;  
 }  
 }  
 return -1;  
 }  
 public static void main(String[] args) {  
 String[] productList = {"Soap", "Toothpaste", "Shampoo", "Milk", "Bread", "Butter"};  
 String searchTerm = "Milk";  
 int linearIndex = *linearSearch*(productList, searchTerm);  
 if (linearIndex != -1) {  
 System.*out*.println("Linear Search: '" + searchTerm + "' found at index " + linearIndex);  
 } else {  
 System.*out*.println("Linear Search: '" + searchTerm + "' not found");  
 }  
 System.*out*.println("Before Sorted Product List: "+Arrays.*toString*(productList));  
 Arrays.*sort*(productList);  
 int binaryIndex = *binarySearch*(productList, searchTerm);  
 if (binaryIndex != -1) {  
 System.*out*.println("Binary Search: '" + searchTerm + "' found at index " + binaryIndex + " in sorted list");  
 } else {  
 System.*out*.println("Binary Search: '" + searchTerm + "' not found in sorted list");  
 }  
 System.*out*.println("Sorted Product List: " + Arrays.*toString*(productList));  
 }  
}

**#OUTPUT:**

EX 1-

Linear Search: 'Milk' found at index 3

Before Sorted Product List: [Soap, Toothpaste, Shampoo, Milk, Bread, Butter]

Binary Search: 'Milk' found at index 2 in sorted list

Sorted Product List: [Bread, Butter, Milk, Shampoo, Soap, Toothpaste]

EX 2-

Linear Search: 'bresh' not found

Before Sorted Product List: [Soap, Toothpaste, Shampoo, Milk, Bread, Butter]

Binary Search: 'bresh' not found in sorted list

Sorted Product List: [Bread, Butter, Milk, Shampoo, Soap, Toothpaste]

EX 3-

Linear Search: 'Butter' found at index 5

Before Sorted Product List: [Soap, Toothpaste, Shampoo, Milk, Bread, Butter]

Binary Search: 'Butter' found at index 1 in sorted list

Sorted Product List: [Bread, Butter, Milk, Shampoo, Soap, Toothpaste]

**Exercise 7: Financial Forecasting**

import java.util.Scanner;  
public class FinancialForecast {  
 public static void main(String[] args) {  
 Scanner sc = new Scanner(System.*in*);  
 System.*out*.print("Enter initial savings (₹): ");  
 double initialSavings = sc.nextDouble();  
 System.*out*.print("Enter expected monthly income (₹): ");  
 double monthlyIncome = sc.nextDouble();  
 System.*out*.print("Enter expected monthly expenses (₹): ");  
 double monthlyExpenses = sc.nextDouble();  
 System.*out*.print("Enter number of months to forecast: ");  
 int months = sc.nextInt();  
 double currentSavings = initialSavings;  
 System.*out*.println("\nFinancial Forecast:");  
 System.*out*.printf("Month\tIncome\tExpenses\tNet Savings\n");  
 for (int i = 1; i <= months; i++) {  
 double net = monthlyIncome - monthlyExpenses;  
 currentSavings += net;  
 System.*out*.printf("%d\t₹%.2f\t₹%.2f\t\t₹%.2f\n", i, monthlyIncome, monthlyExpenses, currentSavings);  
 }  
 System.*out*.printf("\nFinal Forecasted Savings after %d months: ₹%.2f\n", months, currentSavings);  
 }  
}

**#OUTPUT:**

Enter initial savings (₹): 5000

Enter expected monthly income (₹): 20000

Enter expected monthly expenses (₹): 15000

Enter number of months to forecast: 3

Financial Forecast:

Month Income Expenses Net Savings

1 ₹20000.00 ₹15000.00 ₹10000.00

2 ₹20000.00 ₹15000.00 ₹15000.00

3 ₹20000.00 ₹15000.00 ₹20000.00

Final Forecasted Savings after 3 months: ₹20000.00

**THANK YOU**