WEEK – I HANDS ON EXERCISES

DESIGN PATTERNS AND PRINCIPLES

1. Singleton Pattern

```
class Logger {
   private static Logger instance;

   private Logger() {
      System.out.println("Logger initialized.");
   }

   public static Logger getInstance() {
      if (instance == null) {
         instance = new Logger();
}

      return instance;
   }

   public void log(String message) {
      System.out.println("Log: " + message);
   }
}

public class Main {
   public static void main(String[] args) {
      Logger logger1 = Logger.getInstance();
      Logger logger2 = Logger.getInstance();

      logger1.log("First message");
      logger2.log("Second message");

      if (logger1 == logger2) {
         System.out.println("Both logger instances are the same.
Singleton works!");
      } else {
```

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

Output:

```
Logger initialized.
Log: First message
Log: Second message
Both logger instances are the same. Singleton works!
```

2. Implementing the Factory Method Pattern

```
interface Document {
    void open();
}

class WordDocument implements Document {
    public void open() {
        System.out.println("Opening a Word document...");
    }
}

class PdfDocument implements Document {
    public void open() {
        System.out.println("Opening a PDF document...");
    }
}

class ExcelDocument implements Document {
    public void open() {
        System.out.println("Opening an Excel document...");
    }
}
```

```
abstract class DocumentFactory {
   public abstract Docum
ent createDocument();
}

class WordDocumentFactory extends DocumentFactory {
   public Document createDocument() {
      return new WordDocument();
   }
}

class PdfDocumentFactory extends DocumentFactory {
   public Document createDocument() {
      return new PdfDocument();
   }
}

class ExcelDocumentFactory extends DocumentFactory {
   public Document createDocument() {
      return new ExcelDocument();
   }
}

public class Main {
   public static void main(String[] args) {
      DocumentFactory wordFactory = new WordDocumentFactory();
      Document wordDoc = wordFactory.createDocument();
      wordDoc.open();

      DocumentFactory pdfFactory = new PdfDocumentFactory();
      Document pdfDoc = pdfFactory.createDocument();
      pdfDoc.open();

      DocumentFactory excelFactory = new ExcelDocumentFactory();
      Document excelDoc = excelFactory.createDocument();
      excelDoc.open();
   }
}
```

Output:

```
Opening a Word document...
Opening a PDF document...
Opening an Excel document...
```

DATA STRUCTURES AND ALGORITHMS

1. E-commerce Platform Search Function

```
import java.util.Arrays;

class Product {
    int productId;
    String productName;
    String category;

    Product(int id, String name, String category) {
        this.productId = id;
        this.productName = name;
        this.category = category;
    }

    public String toString() {
        return "ID: " + productId + ", Name: " + productName + ", Category:
" + category;
    }
}

public class Main {


    public static Product linearSearch(Product[] products, String name) {
        for (Product p : products) {
```

```
            if (p.productName.equalsIgnoreCase(name)) {
                return p;
            }
        }
        return null;
    }


    public static Product binarySearch(Product[] products, String name) {
        int left = 0;
        int right = products.length - 1;

        while (left <= right) {
            int mid = (left + right) / 2;
            int result =
name.compareToIgnoreCase(products[mid].productName);

            if (result == 0) {
                return products[mid];
            } else if (result < 0) {
                right = mid - 1;
            } else {
                left = mid + 1;
            }
        }

        return null;
    }

    public static void main(String[] args) {
        Product[] products = {
            new Product(101, "iPhone", "Electronics"),
            new Product(102, "T-shirt", "Clothing"),
            new Product(103, "Laptop", "Electronics"),
            new Product(104, "Blender", "Home Appliance"),
            new Product(105, "Shoes", "Footwear")
        };
```

```
        Arrays.sort(products, (a, b) ->
a.productName.compareToIgnoreCase(b.productName));

        String searchQuery = "Laptop";

        System.out.println("Searching for product: " + searchQuery);


        Product result1 = linearSearch(products, searchQuery);
        System.out.println("\nLinear Search Result:");
        if (result1 != null)
            System.out.println(result1);
        else
            System.out.println("Product not found!");

        Product result2 = binarySearch(products, searchQuery);
        System.out.println("\nBinary Search Result:");
        if (result2 != null)
            System.out.println(result2);
        else
            System.out.println("Product not found!");

        System.out.println("\nTime Complexity Comparison:");
        System.out.println("Linear Search: O(n)");
        System.out.println("Binary Search: O(log n) (only works on sorted
data)");

        System.out.println("\nBinary search is preferred for performance
when data is sorted.");
    }
}
```

Output:

```
Searching for product: Laptop

Linear Search Result:
ID: 103, Name: Laptop, Category: Electronics

Binary Search Result:
ID: 103, Name: Laptop, Category: Electronics

Time Complexity Comparison:
Linear Search: O(n)
Binary Search: O(log n) (only works on sorted data)

Binary search is preferred for performance when data is sorted.
```

2. **Financial Forecasting**

```
public class Main {

  public static double futureValue(double presentValue, double growthRate, int years) {
    if (years == 0) return presentValue;
    return futureValue(presentValue, growthRate, years - 1) * (1 + growthRate);
  }

  public static void main(String[] args) {
    double present = 1000.0;
    double rate = 0.10;
    int years = 5;

    double result = futureValue(present, rate, years);
    System.out.println("Future Value after %d years: %.2f\n", years, result);
  }
```

```
}
```

Output:

```
Future Value after 5 years: 1610.51
```