DIGITAL-NURTURE-4.0-JavaFSE

WEEK 1

Design principles & Patterns

Exercise 1: Implementing the Singleton Pattern

```
public class Singleton {
   private static volatile 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!");
   }
}
```

DIGITAL-NURTURE-4.0-JavaFSE

TEST CLASS

```
public class Main {
    public static void main(String[] args) {
        Singleton singleton1 = Singleton.getInstance();
        Singleton singleton2 = Singleton.getInstance();
        singleton1.showMessage();
        if (singleton1 == singleton2) {
            System.out.println("Both references point to the same instance.");
        } else {
            System.out.println("Different instances exist!");
        }
    }
}
```

Exercise 2: Implementing the Factory Method Pattern

SHAPE.java

```
public interface Shape {
    void draw();
}
```

Circle.java

```
public class Circle implements Shape {
    @Override
    public void draw() {
        System.out.println("Drawing a Circle");
    }
}
```

Rectangle.java

```
public class Rectangle implements Shape {
```

```
    @Override
    public void draw() {
        System.out.println("Drawing a Rectangle");
    }
}
```

ShapeFactory.java

```
public class ShapeFactory {
    public Shape createShape(String type) {
        if (type == null) {
            return null;
        }
        if (type.equalsIgnoreCase("CIRCLE")) {
            return new Circle();
        } else if (type.equalsIgnoreCase("RECTANGLE")) {
            return new Rectangle();
        }
        return null;
    }
}
```

Main.java

```
public class Main {
    public static void main(String[] args) {
        ShapeFactory factory = new ShapeFactory();
        Shape shape1 = factory.createShape("CIRCLE");
        shape1.draw();
        Shape shape2 = factory.createShape("RECTANGLE");
        shape2.draw();
    }
}
```

DIGITAL-NURTURE-4.0-JavaFSE

Algorithms Data Structures

Exercise 2: E-commerce Platform Search Function

Product.java

```
public class Product {
    private int id;
    private String name;
    private String category;
    private double price;
    private String description;
    public Product(int id, String name, String category, double price, String description) {
        this.id = id;
        this.name = name;
        this.category = category;
        this.price = price;
        this.description = description;
    }

    public boolean matchesKeyword(String keyword) {
        String lowerKeyword = keyword.toLowerCase();
        return name.toLowerCase().contains(lowerKeyword) ||
                category.toLowerCase().contains(lowerKeyword) ||
                description.toLowerCase().contains(lowerKeyword);
    }

    @Override
    public String toString() {
        return String.format("Product[id=%d, name=%s, category=%s, price=%.2f]", id, name, category,
price);
    }
}
```

DIGITAL-NURTURE-4.0-JavaFSE

ProductCatalog.java

```
import java.util.ArrayList;
import java.util.List;
public class ProductCatalog {
    private List<Product> products;
    public ProductCatalog() {
        products = new ArrayList<>();
    }
    public void addProduct(Product product) {
        products.add(product);
    }
    public List<Product> search(String keyword) {
        List<Product> result = new ArrayList<>();
        for (Product product : products) {
            if (product.matchesKeyword(keyword)) {
                result.add(product);
            }
        }
        return result;
    }
}
```

Main.java

```
import java.util.List;
public class Main {
    public static void main(String[] args) {
        ProductCatalog catalog = new ProductCatalog();

        catalog.addProduct(new Product(1, "Laptop", "Electronics", 999.99, "15-inch high-performance laptop"));
        catalog.addProduct(new Product(2, "Headphones", "Audio", 199.99, "Noise-cancelling headphones"));
```

```
        catalog.addProduct(new Product(3, "Coffee Mug", "Kitchen", 12.99, "Ceramic mug with
handle"));

        catalog.addProduct(new Product(4, "Monitor", "Electronics", 249.99, "24-inch 1080p monitor"));

        String keyword = "Electronics";

        List<Product> results = catalog.search(keyword);

        System.out.println("Search results for keyword: " + keyword);

        for (Product product : results) {

            System.out.println(product);

        }

    }

}
```