Week - 1. Design principles & Patterns

i) Implementing the Singleton Pattern

What is Singleton Pattern *?*

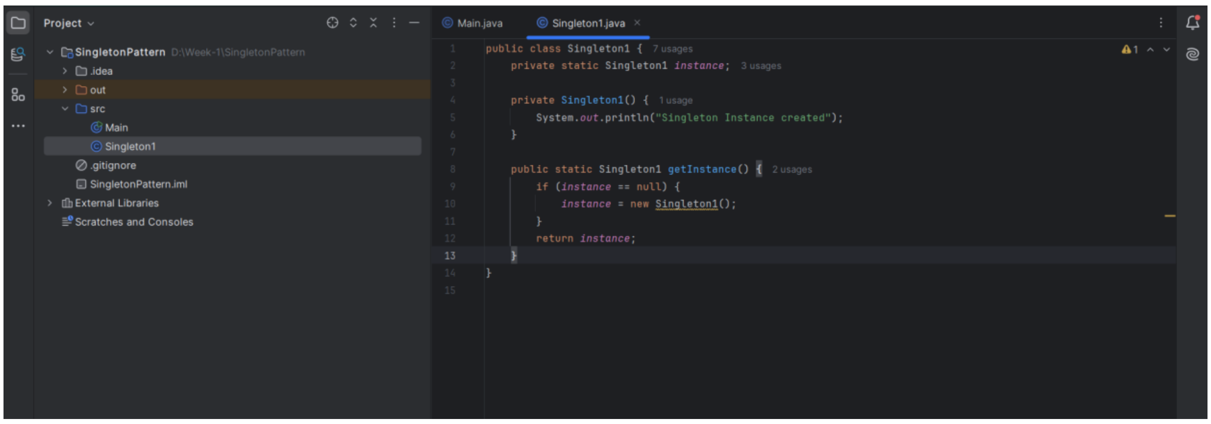
Ans *:*

The singleton pattern ensures that a *class* has "only 1 instance" throughout the application and providing a global access point to the instance

Disadvantage:

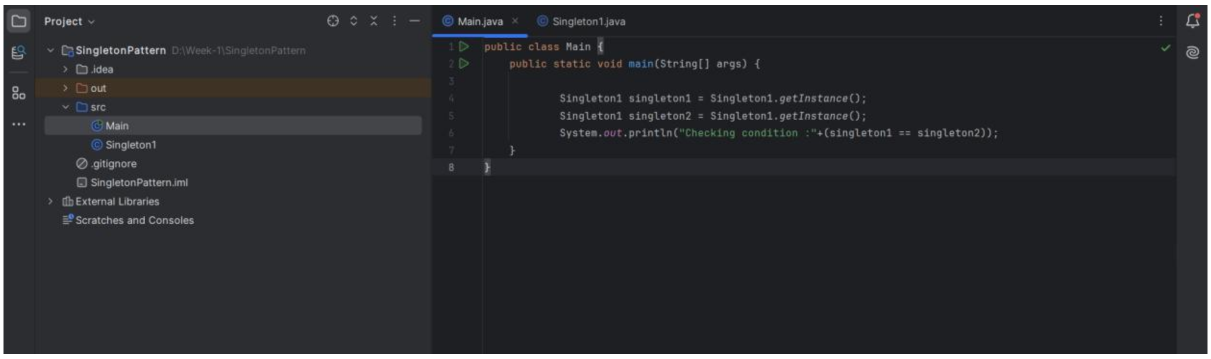
Not Thread safe but we can counter that to using the different approaches

Singleton Class



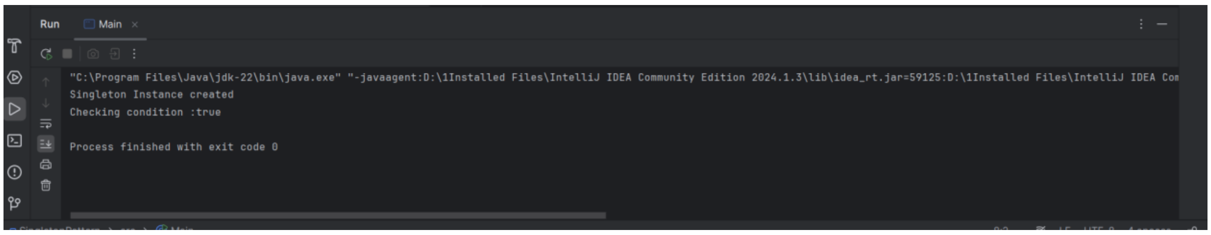
```java
public class Singleton1 {  7 usages
    private static Singleton1 instance;  3 usages

    private Singleton1() {  1 usage
        System.out.println("Singleton Instance created");
    }

    public static Singleton1 getInstance() {  2 usages
        if (instance == null) {
            instance = new Singleton1();
        }
        return instance;
    }
}
```

Main Class



```java
public class Main {
    public static void main(String[] args) {

            Singleton1 singleton1 = Singleton1.getInstance();
            Singleton1 singleton2 = Singleton1.getInstance();
            System.out.println("Checking condition :"+(singleton1 == singleton2));
    }
}
```

Output:



```
"C:\Program Files\Java\jdk-22\bin\java.exe" "-javaagent:D:\1Installed Files\IntelliJ IDEA Community Edition 2024.1.3\lib\idea_rt.jar=59125:D:\1Installed Files\IntelliJ IDEA Com
Singleton Instance created
Checking condition :true

Process finished with exit code 0
```

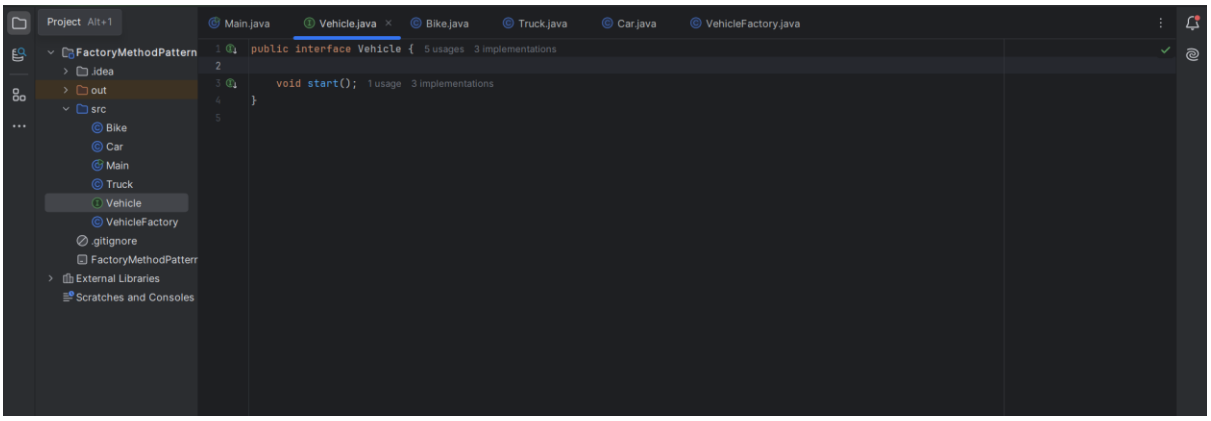
ii) Implementing the Factory Method Pattern

What is Factory Method?

It helps us create objects without exposing the object creation logic to the client. *The* client just asks the Factory, and the factory returns the correct object.

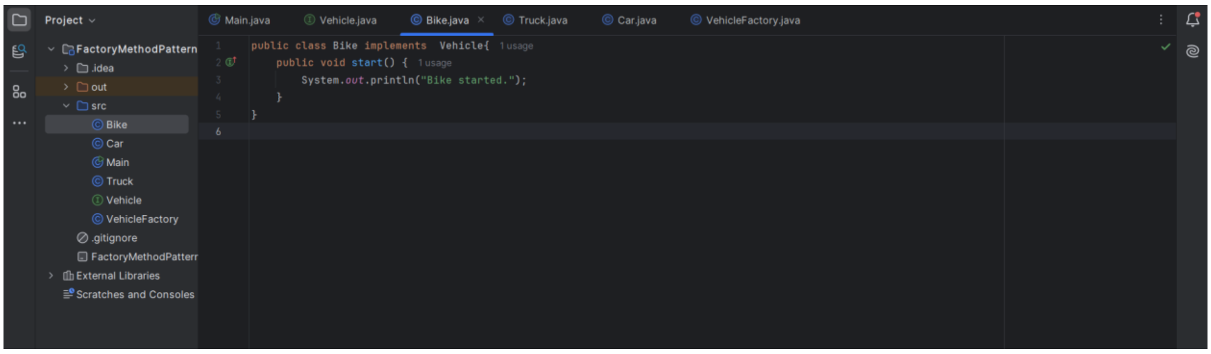
Example for Factory Method is Vehicle

Vehicle Interface:



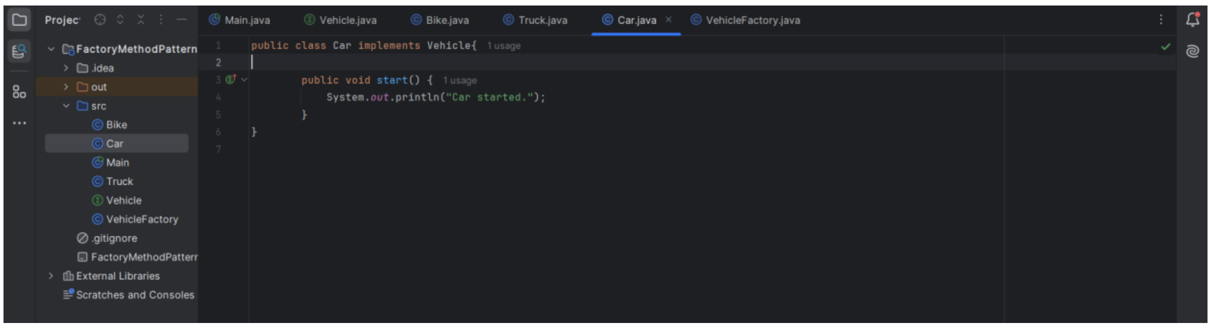
```java
public interface Vehicle {

    void start();
}
```

Bike Class:



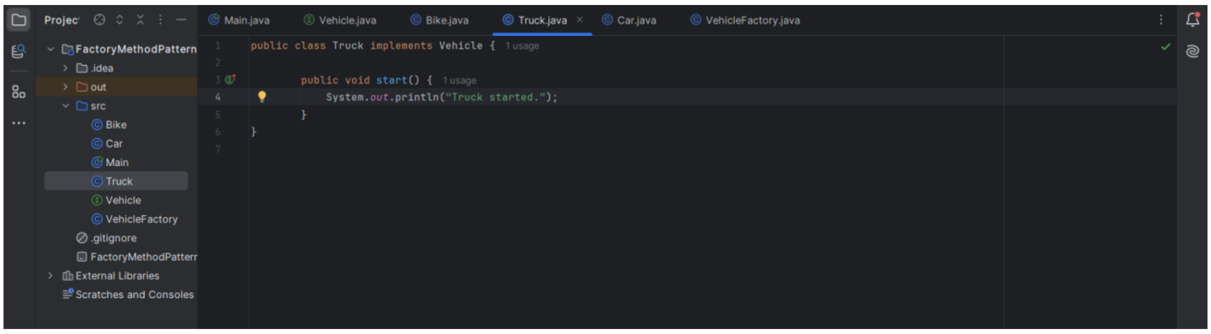
```java
public class Bike implements  Vehicle{
    public void start() {
        System.out.println("Bike started.");
    }
}
```

Car Class:



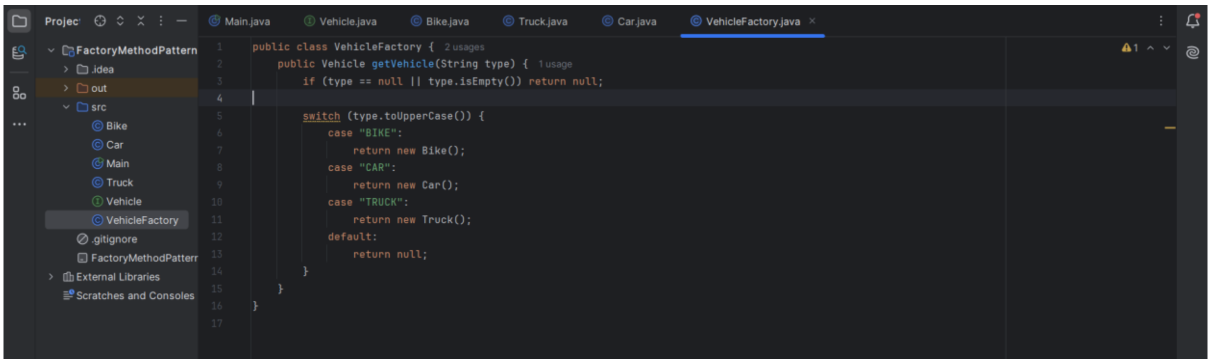
```java
public class Car implements Vehicle{

        public void start() {
            System.out.println("Car started.");
        }
}
```

Truck Class:



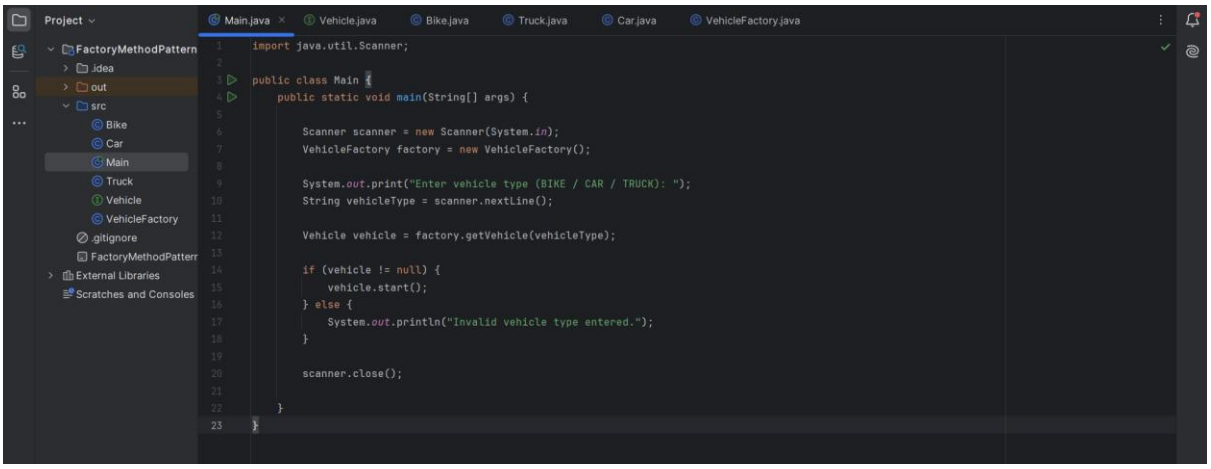
```java
public class Truck implements Vehicle {

    public void start() {
        System.out.println("Truck started.");
    }
}
```

VehicleFactory Class:



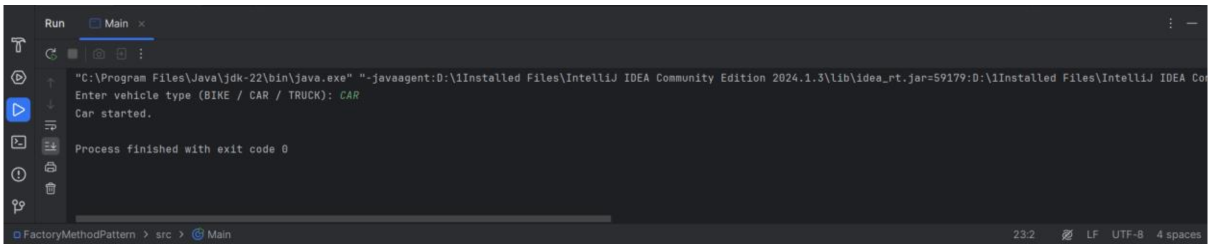
```java
public class VehicleFactory {
    public Vehicle getVehicle(String type) {
        if (type == null || type.isEmpty()) return null;

        switch (type.toUpperCase()) {
            case "BIKE":
                return new Bike();
            case "CAR":
                return new Car();
            case "TRUCK":
                return new Truck();
            default:
                return null;
        }
    }
}
```

Main Class:



```java
import java.util.Scanner;

public class Main {
    public static void main(String[] args) {

        Scanner scanner = new Scanner(System.in);
        VehicleFactory factory = new VehicleFactory();

        System.out.print("Enter vehicle type (BIKE / CAR / TRUCK): ");
        String vehicleType = scanner.nextLine();

        Vehicle vehicle = factory.getVehicle(vehicleType);

        if (vehicle != null) {
            vehicle.start();
        } else {
            System.out.println("Invalid vehicle type entered.");
        }

        scanner.close();

    }
}
```

Output:

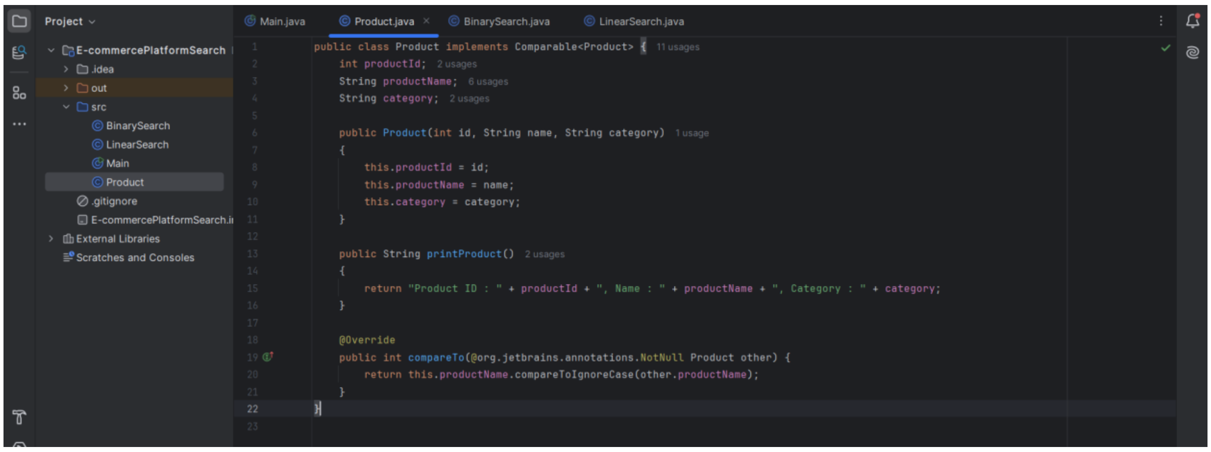


```
"C:\Program Files\Java\jdk-22\bin\java.exe" "-javaagent:D:\1Installed Files\IntelliJ IDEA Community Edition 2024.1.3\lib\idea_rt.jar=59179:D:\1Installed Files\IntelliJ IDEA Co
Enter vehicle type (BIKE / CAR / TRUCK): CAR
Car started.

Process finished with exit code 0
```

2.Algorithms_Data Structures

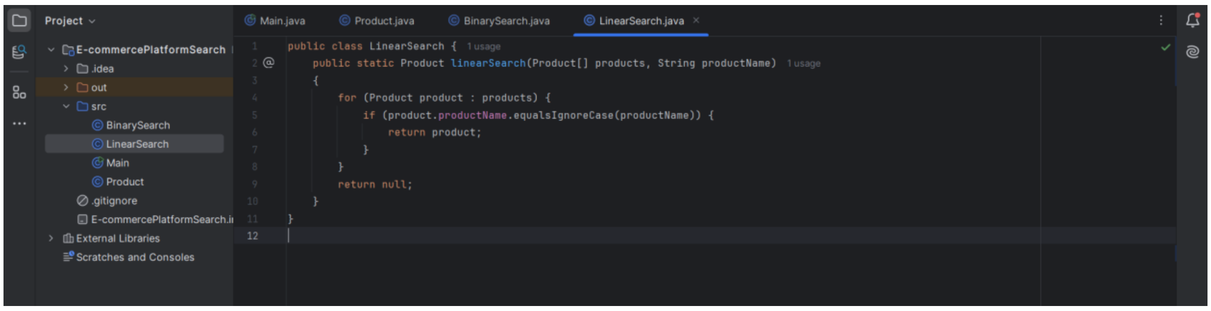
i) E-commerce Platform Search Function

Product Class:



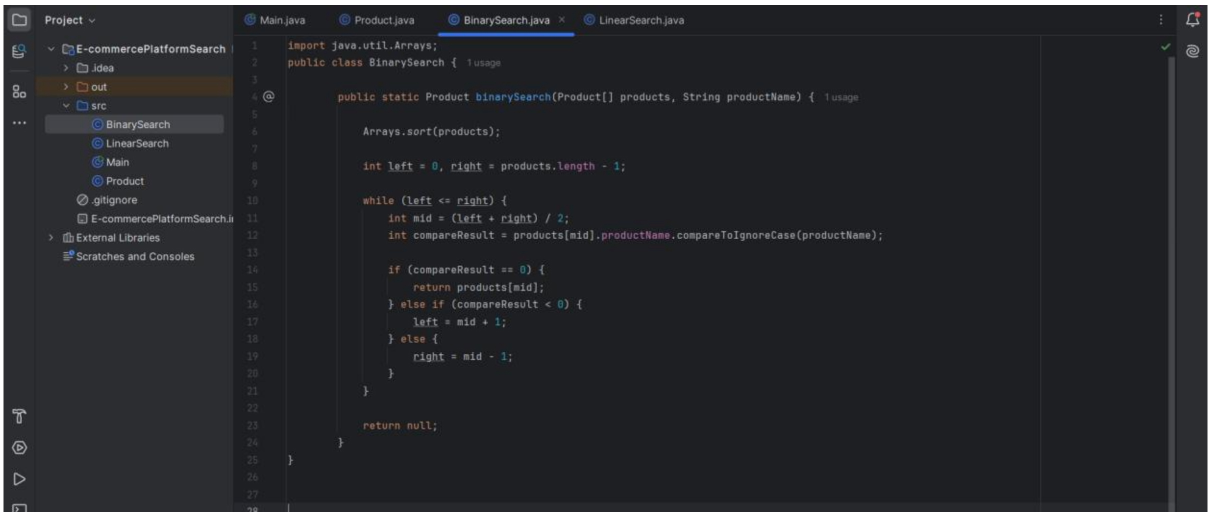
```java
public class Product implements Comparable<Product> {
    int productId;
    String productName;
    String category;

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

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

    @Override
    public int compareTo(@org.jetbrains.annotations.NotNull Product other) {
        return this.productName.compareToIgnoreCase(other.productName);
    }
}
```

LinearSearch Class:



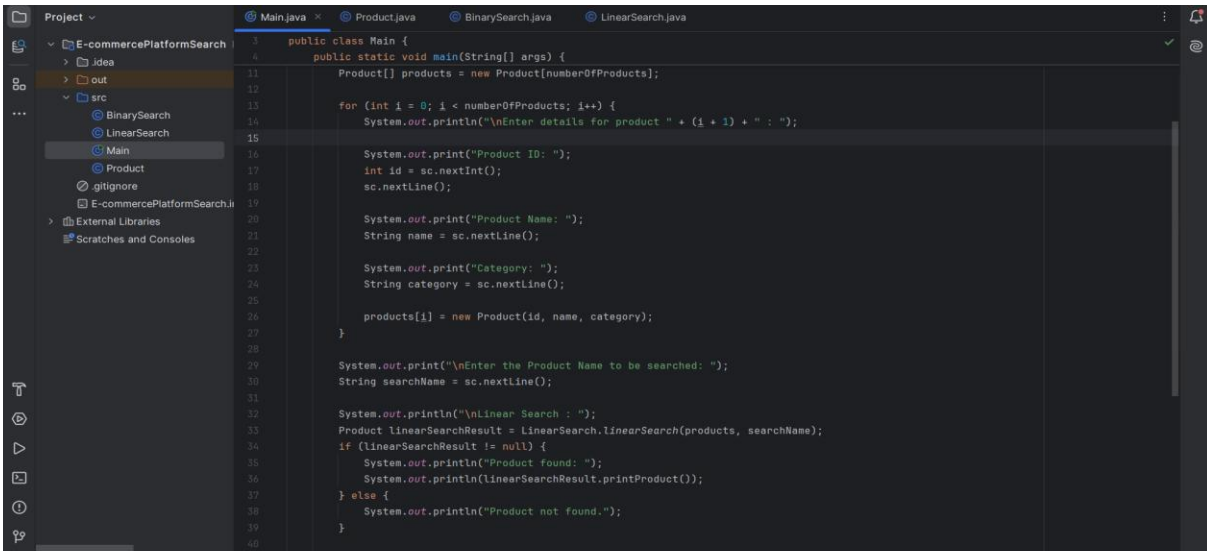
```java
public class LinearSearch {
    public static Product linearSearch(Product[] products, String productName)
    {
        for (Product product : products) {
            if (product.productName.equalsIgnoreCase(productName)) {
                return product;
            }
        }
        return null;
    }
}
```

BinarySearch Class:

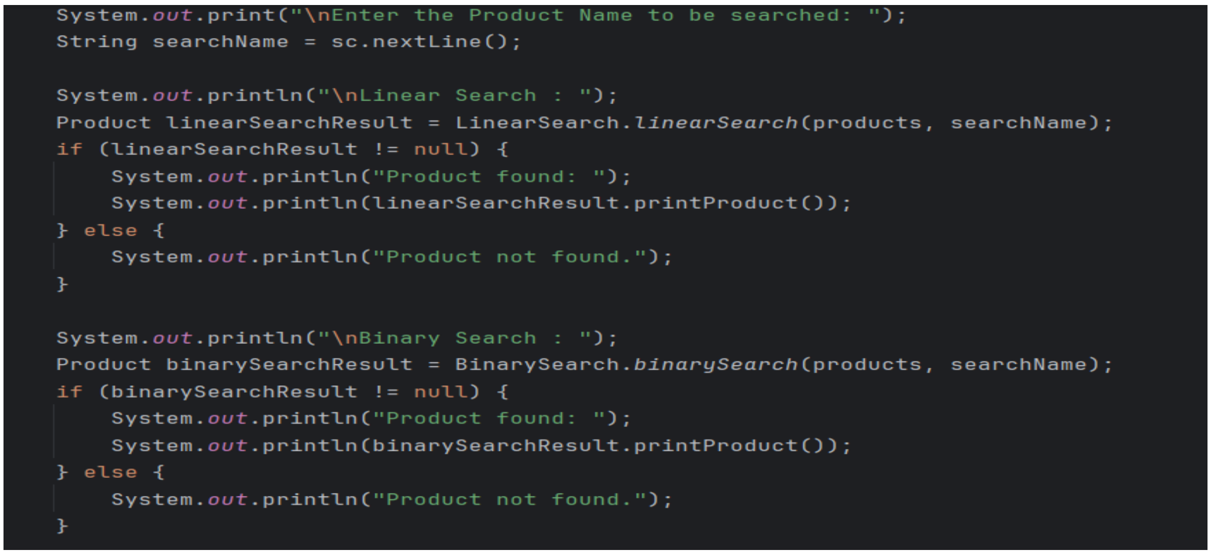


```java
import java.util.Arrays;
public class BinarySearch {

        public static Product binarySearch(Product[] products, String productName) {

            Arrays.sort(products);

            int left = 0, right = products.length - 1;

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

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

            return null;
        }
}
```

Main Class:

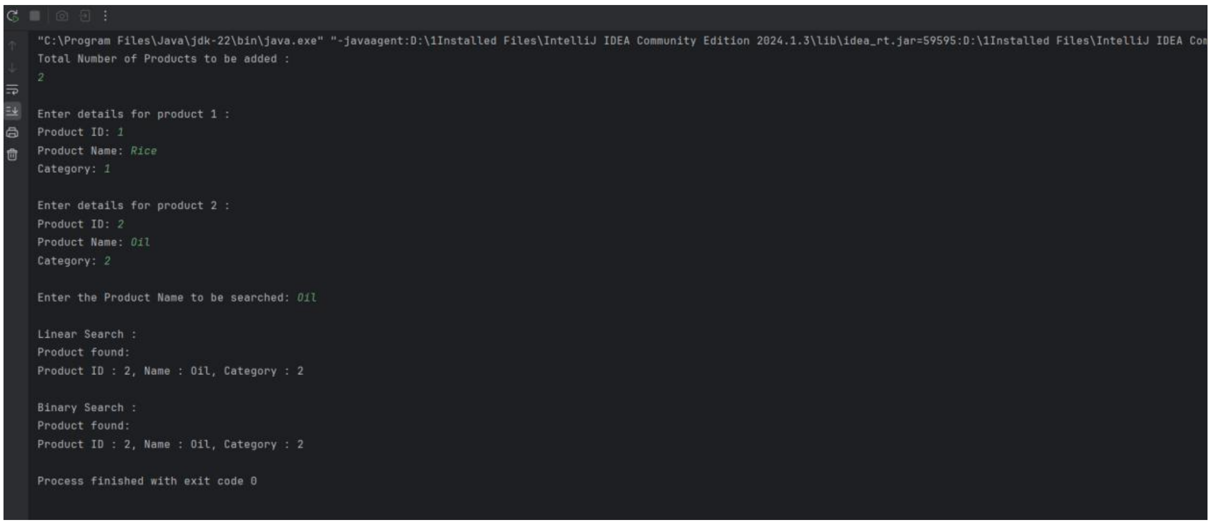


```java
public class Main {
    public static void main(String[] args) {
        Product[] products = new Product[numberOfProducts];

        for (int i = 0; i < numberOfProducts; i++) {
            System.out.println("\nEnter details for product " + (i + 1) + " : ");

            System.out.print("Product ID: ");
            int id = sc.nextInt();
            sc.nextLine();

            System.out.print("Product Name: ");
            String name = sc.nextLine();

            System.out.print("Category: ");
            String category = sc.nextLine();

            products[i] = new Product(id, name, category);
        }

        System.out.print("\nEnter the Product Name to be searched: ");
        String searchName = sc.nextLine();

        System.out.println("\nLinear Search : ");
        Product linearSearchResult = LinearSearch.linearSearch(products, searchName);
        if (linearSearchResult != null) {
            System.out.println("Product found: ");
            System.out.println(linearSearchResult.printProduct());
        } else {
            System.out.println("Product not found.");
        }
```



```java
System.out.print("\nEnter the Product Name to be searched: ");
String searchName = sc.nextLine();

System.out.println("\nLinear Search : ");
Product linearSearchResult = LinearSearch.linearSearch(products, searchName);
if (linearSearchResult != null) {
    System.out.println("Product found: ");
    System.out.println(linearSearchResult.printProduct());
} else {
    System.out.println("Product not found.");
}

System.out.println("\nBinary Search : ");
Product binarySearchResult = BinarySearch.binarySearch(products, searchName);
if (binarySearchResult != null) {
    System.out.println("Product found: ");
    System.out.println(binarySearchResult.printProduct());
} else {
    System.out.println("Product not found.");
}
```

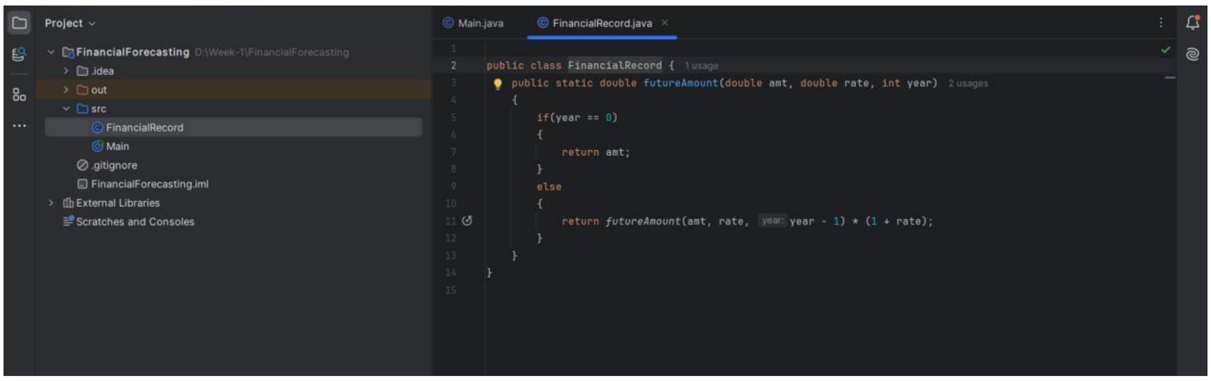
Output:



```
"C:\Program Files\Java\jdk-22\bin\java.exe" "-javaagent:D:\1Installed Files\IntelliJ IDEA Community Edition 2024.1.3\lib\idea_rt.jar=59595:D:\1Installed Files\IntelliJ IDEA Com
Total Number of Products to be added :
2

Enter details for product 1 :
Product ID: 1
Product Name: Rice
Category: 1

Enter details for product 2 :
Product ID: 2
Product Name: Oil
Category: 2

Enter the Product Name to be searched: Oil

Linear Search :
Product found:
Product ID : 2, Name : Oil, Category : 2

Binary Search :
Product found:
Product ID : 2, Name : Oil, Category : 2

Process finished with exit code 0
```

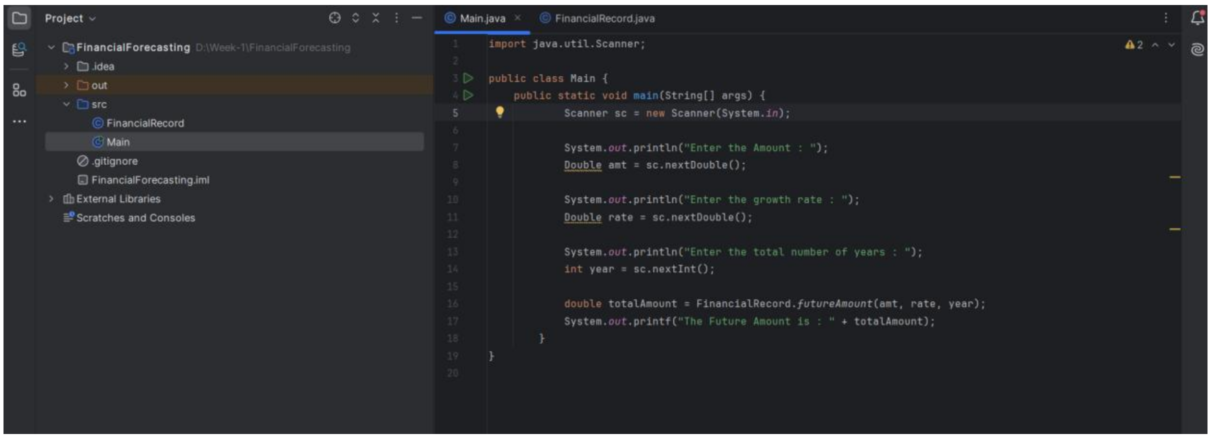
ii) Financial Forecasting

FinancialRecord Class:



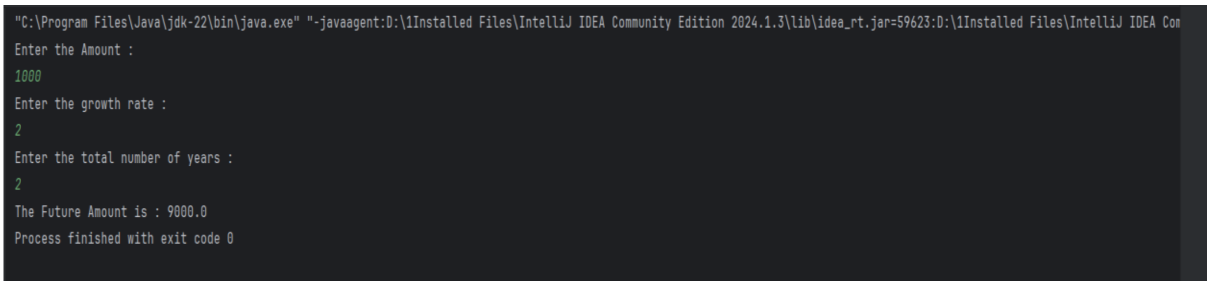
```java

public class FinancialRecord {
    public static double futureAmount(double amt, double rate, int year)
    {
        if(year == 0)
        {
            return amt;
        }
        else
        {
            return futureAmount(amt, rate, year - 1) * (1 + rate);
        }
    }
}
```

Main Class:



```java
import java.util.Scanner;

public class Main {
    public static void main(String[] args) {
            Scanner sc = new Scanner(System.in);

            System.out.println("Enter the Amount : ");
            Double amt = sc.nextDouble();

            System.out.println("Enter the growth rate : ");
            Double rate = sc.nextDouble();

            System.out.println("Enter the total number of years : ");
            int year = sc.nextInt();

            double totalAmount = FinancialRecord.futureAmount(amt, rate, year);
            System.out.printf("The Future Amount is : " + totalAmount);
        }
}
```

Output:



```
"C:\Program Files\Java\jdk-22\bin\java.exe" "-javaagent:D:\1Installed Files\IntelliJ IDEA Community Edition 2024.1.3\lib\idea_rt.jar=59623:D:\1Installed Files\IntelliJ IDEA Con
Enter the Amount :
1000
Enter the growth rate :
2
Enter the total number of years :
2
The Future Amount is : 9000.0
Process finished with exit code 0
```