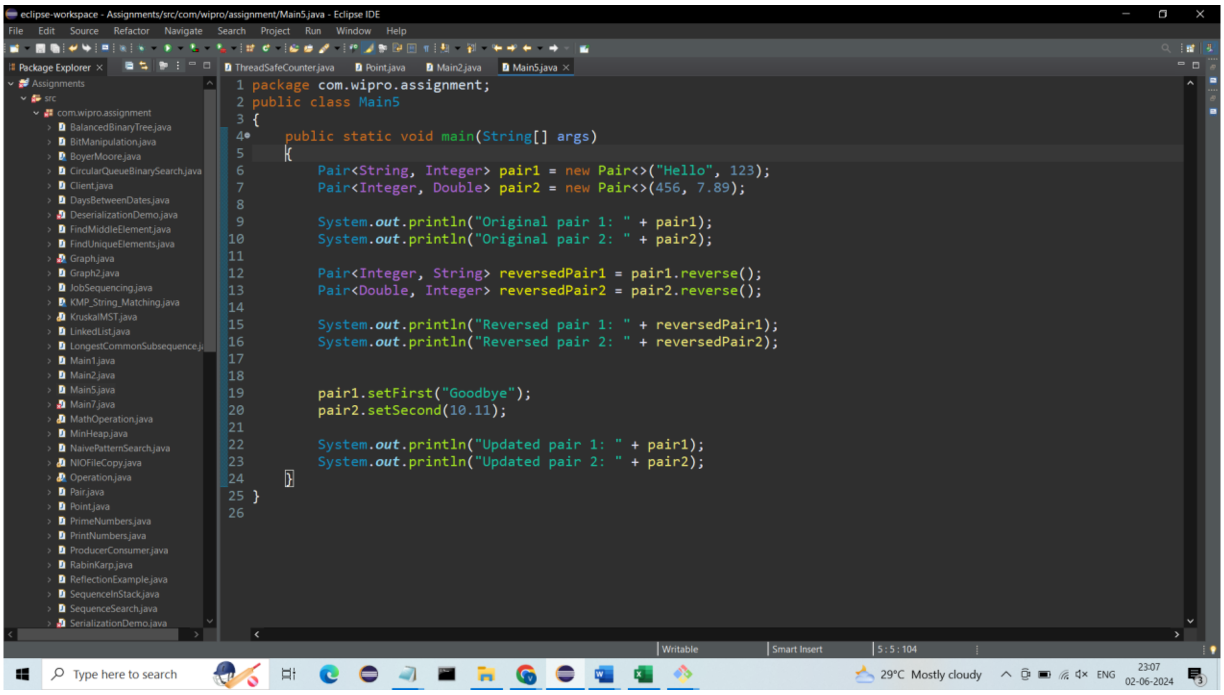
## Task 1: Generics and Type Safety

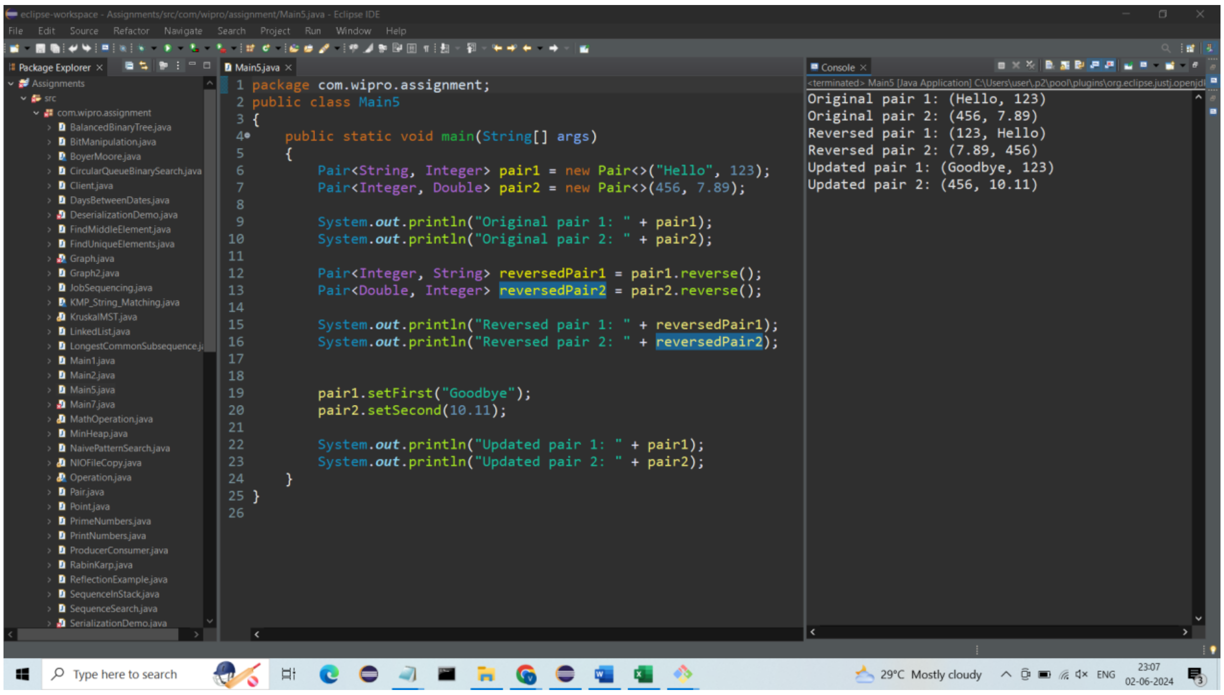
Create a generic Pair class that holds two objects of different types, and write a method to return a reversed version of the pair.

## Method: -


eclipse-workspace - Assignments/src/com/wipro/assignment/Main5.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer
ThreadSafeCounter.java
Point.java
Main2.java
Main5.java
Writable
Smart Insert
5 : 5 : 104

```java
package com.wipro.assignment;
public class Main5
{
    public static void main(String[] args)
    {
        Pair<String, Integer> pair1 = new Pair<>("Hello", 123);
        Pair<Integer, Double> pair2 = new Pair<>(456, 7.89);

        System.out.println("Original pair 1: " + pair1);
        System.out.println("Original pair 2: " + pair2);

        Pair<Integer, String> reversedPair1 = pair1.reverse();
        Pair<Double, Integer> reversedPair2 = pair2.reverse();

        System.out.println("Reversed pair 1: " + reversedPair1);
        System.out.println("Reversed pair 2: " + reversedPair2);


        pair1.setFirst("Goodbye");
        pair2.setSecond(10.11);

        System.out.println("Updated pair 1: " + pair1);
        System.out.println("Updated pair 2: " + pair2);
    }
}
```

## Output: -



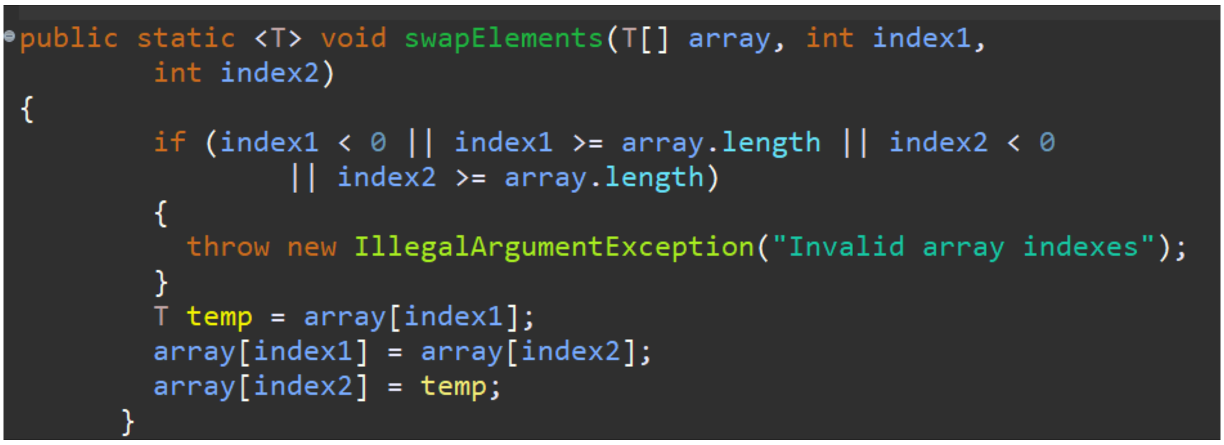
```java
package com.wipro.assignment;
public class Main5
{
    public static void main(String[] args)
    {
        Pair<String, Integer> pair1 = new Pair<>("Hello", 123);
        Pair<Integer, Double> pair2 = new Pair<>(456, 7.89);

        System.out.println("Original pair 1: " + pair1);
        System.out.println("Original pair 2: " + pair2);

        Pair<Integer, String> reversedPair1 = pair1.reverse();
        Pair<Double, Integer> reversedPair2 = pair2.reverse();

        System.out.println("Reversed pair 1: " + reversedPair1);
        System.out.println("Reversed pair 2: " + reversedPair2);


        pair1.setFirst("Goodbye");
        pair2.setSecond(10.11);

        System.out.println("Updated pair 1: " + pair1);
        System.out.println("Updated pair 2: " + pair2);
    }
}
```

```
<terminated> Main5 [Java Application] C:\Users\user\.p2\pool\plugins\org.eclipse.justj.openjdk
Original pair 1: (Hello, 123)
Original pair 2: (456, 7.89)
Reversed pair 1: (123, Hello)
Reversed pair 2: (7.89, 456)
Updated pair 1: (Goodbye, 123)
Updated pair 2: (456, 10.11)
```

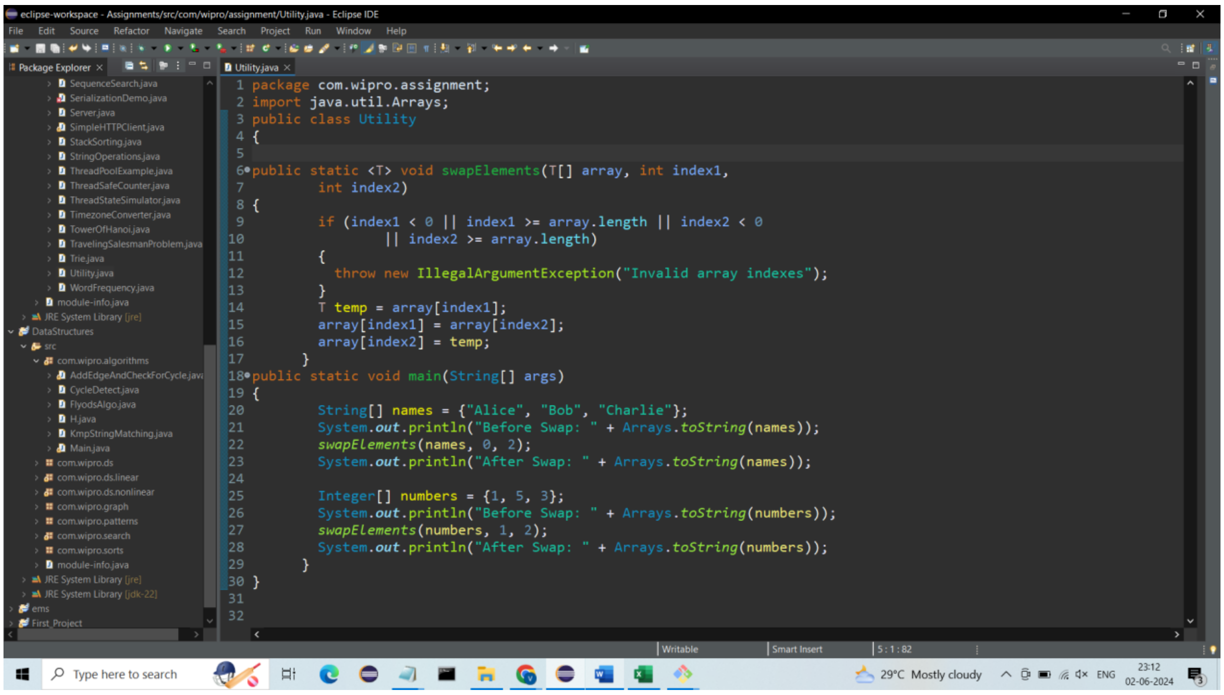
## Task 2: Generic Classes and Methods

Implement a generic method that swaps the positions of two elements in an array, regardless of their type, and demonstrate its usage with different object types.

**Method: -**

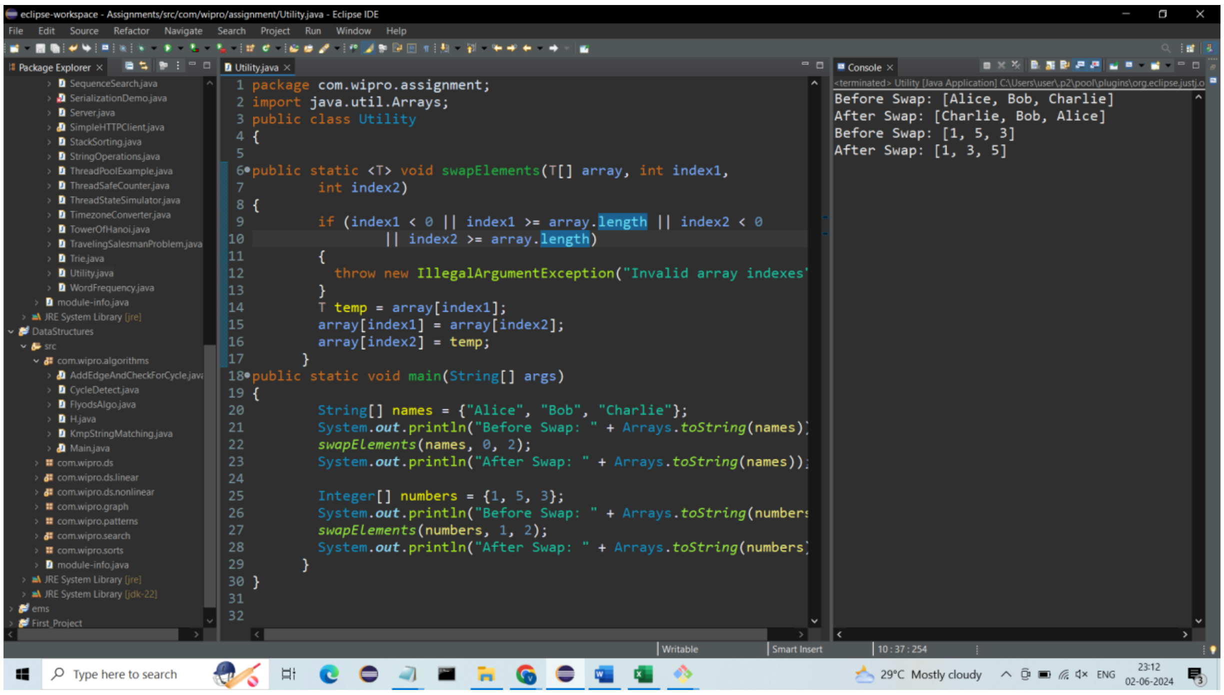


```java
public static <T> void swapElements(T[] array, int index1,
        int index2)
{
        if (index1 < 0 || index1 >= array.length || index2 < 0
                || index2 >= array.length)
        {
          throw new IllegalArgumentException("Invalid array indexes");
        }
        T temp = array[index1];
        array[index1] = array[index2];
        array[index2] = temp;
    }
```



```java
package com.wipro.assignment;
import java.util.Arrays;
public class Utility
{

public static <T> void swapElements(T[] array, int index1,
        int index2)
{
        if (index1 < 0 || index1 >= array.length || index2 < 0
                || index2 >= array.length)
        {
          throw new IllegalArgumentException("Invalid array indexes");
        }
        T temp = array[index1];
        array[index1] = array[index2];
        array[index2] = temp;
      }
public static void main(String[] args)
{
        String[] names = {"Alice", "Bob", "Charlie"};
        System.out.println("Before Swap: " + Arrays.toString(names));
        swapElements(names, 0, 2);
        System.out.println("After Swap: " + Arrays.toString(names));

        Integer[] numbers = {1, 5, 3};
        System.out.println("Before Swap: " + Arrays.toString(numbers));
        swapElements(numbers, 1, 2);
        System.out.println("After Swap: " + Arrays.toString(numbers));
      }
}
```

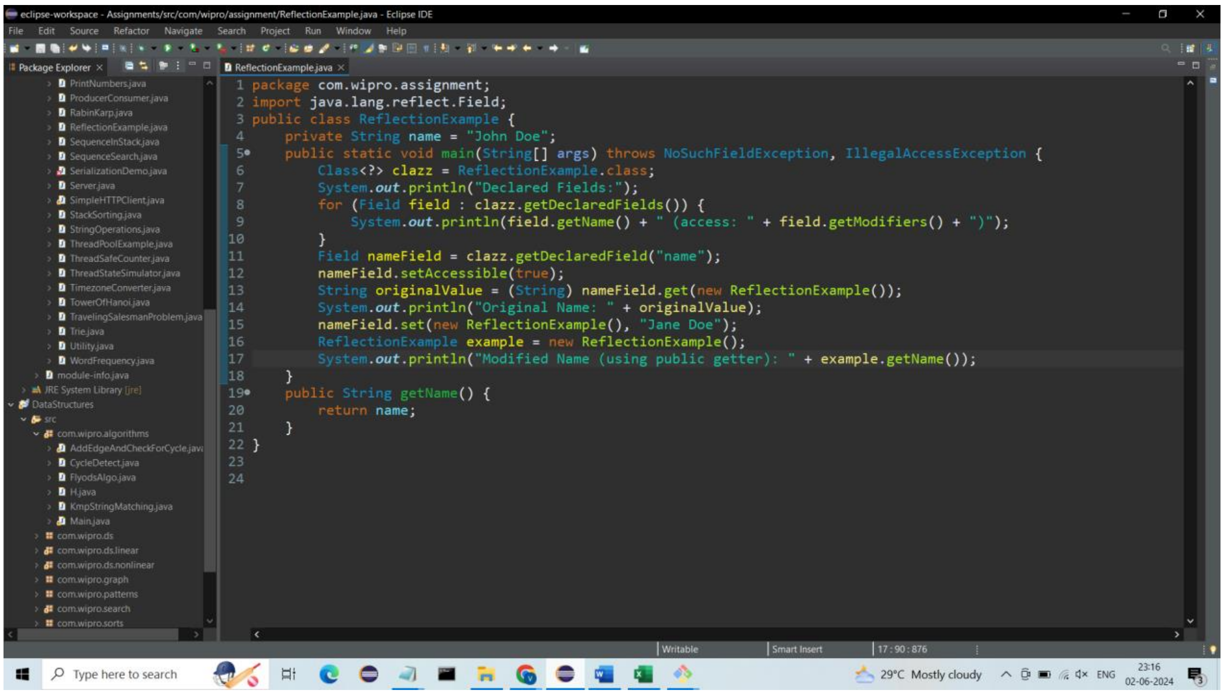
**Output: -**



```
Before Swap: [Alice, Bob, Charlie]
After Swap: [Charlie, Bob, Alice]
Before Swap: [1, 5, 3]
After Swap: [1, 3, 5]
```

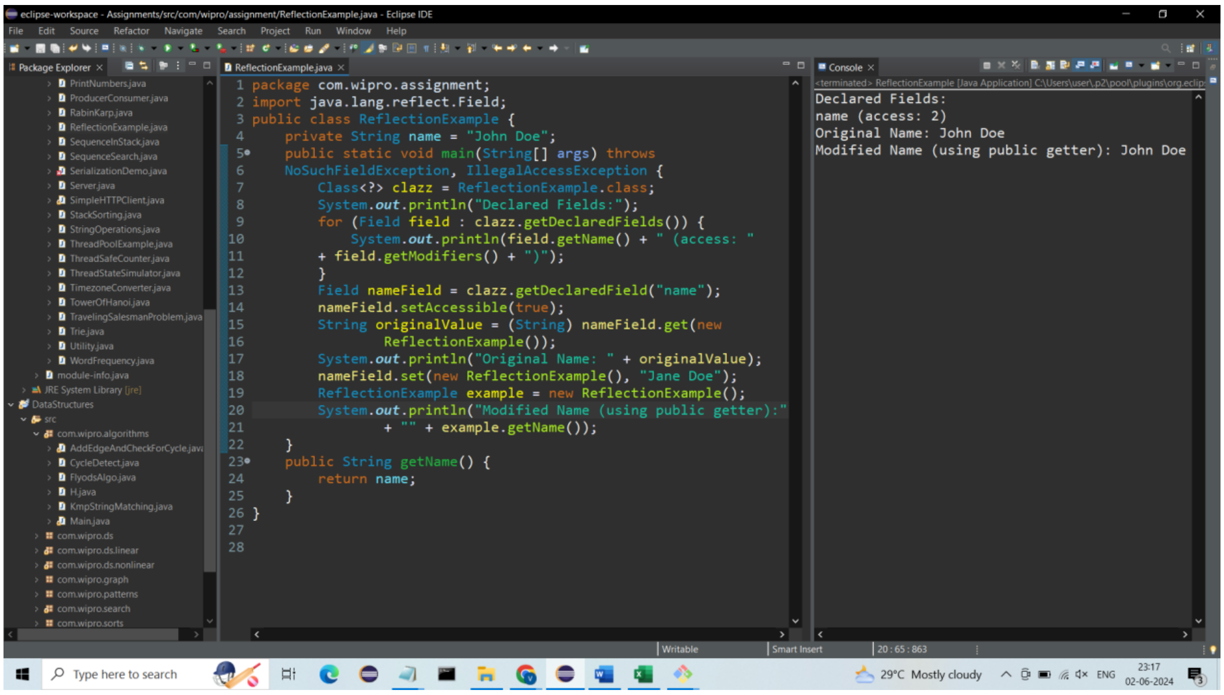
## Task 3: Reflection API

Use reflection to inspect a class's methods, fields, and constructors, and modify the access level of a private field, setting its value during runtime



```java
package com.wipro.assignment;
import java.lang.reflect.Field;
public class ReflectionExample {
    private String name = "John Doe";
    public static void main(String[] args) throws NoSuchFieldException, IllegalAccessException {
        Class<?> clazz = ReflectionExample.class;
        System.out.println("Declared Fields:");
        for (Field field : clazz.getDeclaredFields()) {
            System.out.println(field.getName() + " (access: " + field.getModifiers() + ")");
        }
        Field nameField = clazz.getDeclaredField("name");
        nameField.setAccessible(true);
        String originalValue = (String) nameField.get(new ReflectionExample());
        System.out.println("Original Name: " + originalValue);
        nameField.set(new ReflectionExample(), "Jane Doe");
        ReflectionExample example = new ReflectionExample();
        System.out.println("Modified Name (using public getter): " + example.getName());
    }
    public String getName() {
        return name;
    }
}
```

## Output: -



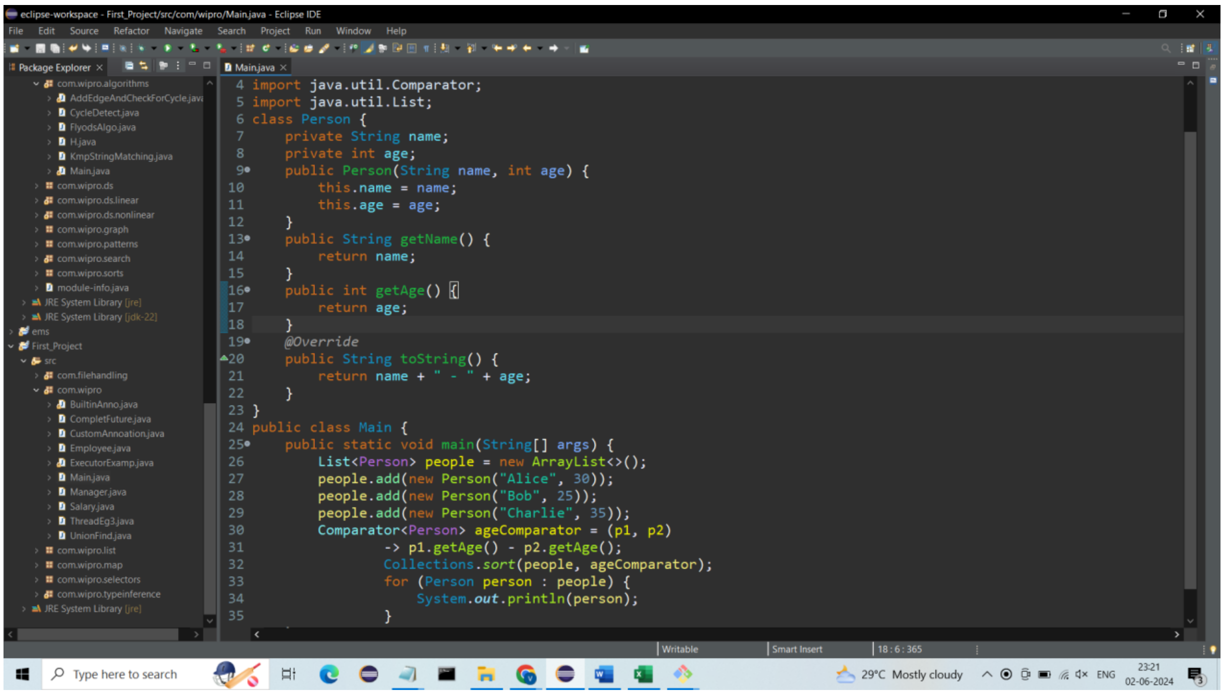
```java
package com.wipro.assignment;
import java.lang.reflect.Field;
public class ReflectionExample {
    private String name = "John Doe";
    public static void main(String[] args) throws
    NoSuchFieldException, IllegalAccessException {
        Class<?> clazz = ReflectionExample.class;
        System.out.println("Declared Fields:");
        for (Field field : clazz.getDeclaredFields()) {
            System.out.println(field.getName() + " (access: "
        + field.getModifiers() + ")");
        }
        Field nameField = clazz.getDeclaredField("name");
        nameField.setAccessible(true);
        String originalValue = (String) nameField.get(new
                ReflectionExample());
        System.out.println("Original Name: " + originalValue);
        nameField.set(new ReflectionExample(), "Jane Doe");
        ReflectionExample example = new ReflectionExample();
        System.out.println("Modified Name (using public getter):"
                + "" + example.getName());
    }
    public String getName() {
        return name;
    }
}
```

```
Declared Fields:
name (access: 2)
Original Name: John Doe
Modified Name (using public getter): John Doe
```

## Task 4: Lambda Expressions

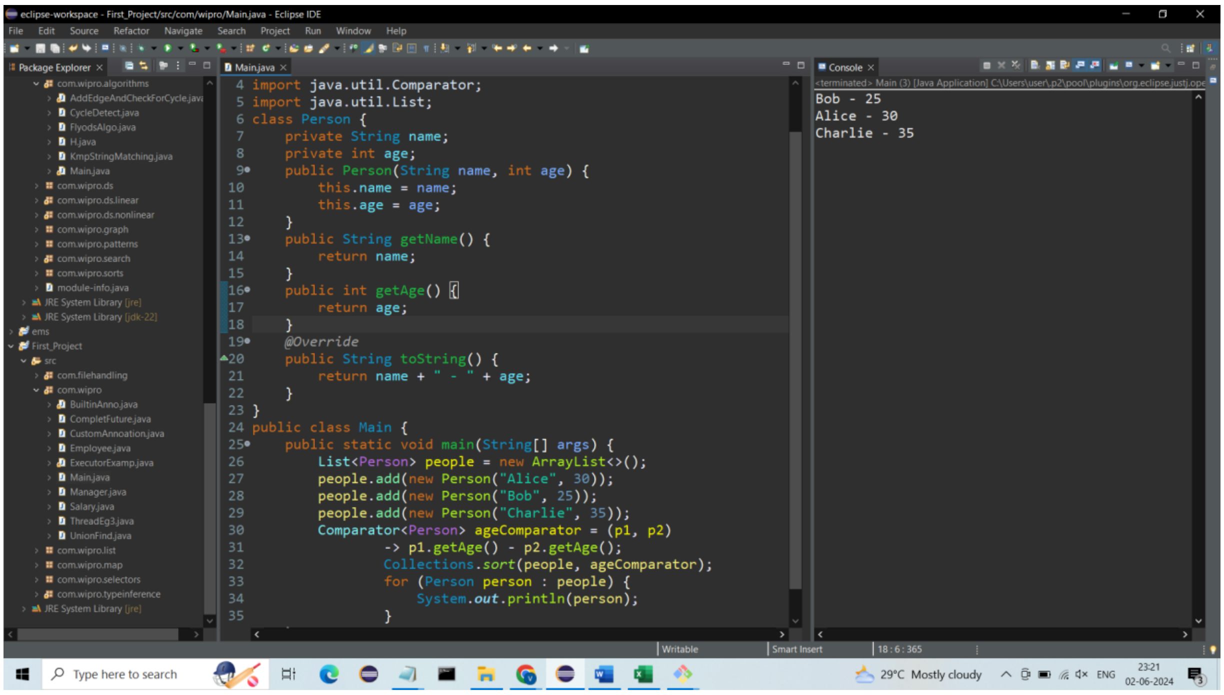
Implement a Comparator for a Person class using a lambda expression, and sort a list of Person objects by their age..

**Solution: -**



```java
import java.util.Comparator;
import java.util.List;
class Person {
    private String name;
    private int age;
    public Person(String name, int age) {
        this.name = name;
        this.age = age;
    }
    public String getName() {
        return name;
    }
    public int getAge() {
        return age;
    }
    @Override
    public String toString() {
        return name + " - " + age;
    }
}
public class Main {
    public static void main(String[] args) {
        List<Person> people = new ArrayList<>();
        people.add(new Person("Alice", 30));
        people.add(new Person("Bob", 25));
        people.add(new Person("Charlie", 35));
        Comparator<Person> ageComparator = (p1, p2)
                -> p1.getAge() - p2.getAge();
                Collections.sort(people, ageComparator);
                for (Person person : people) {
                    System.out.println(person);
                }
```

**Output: -**



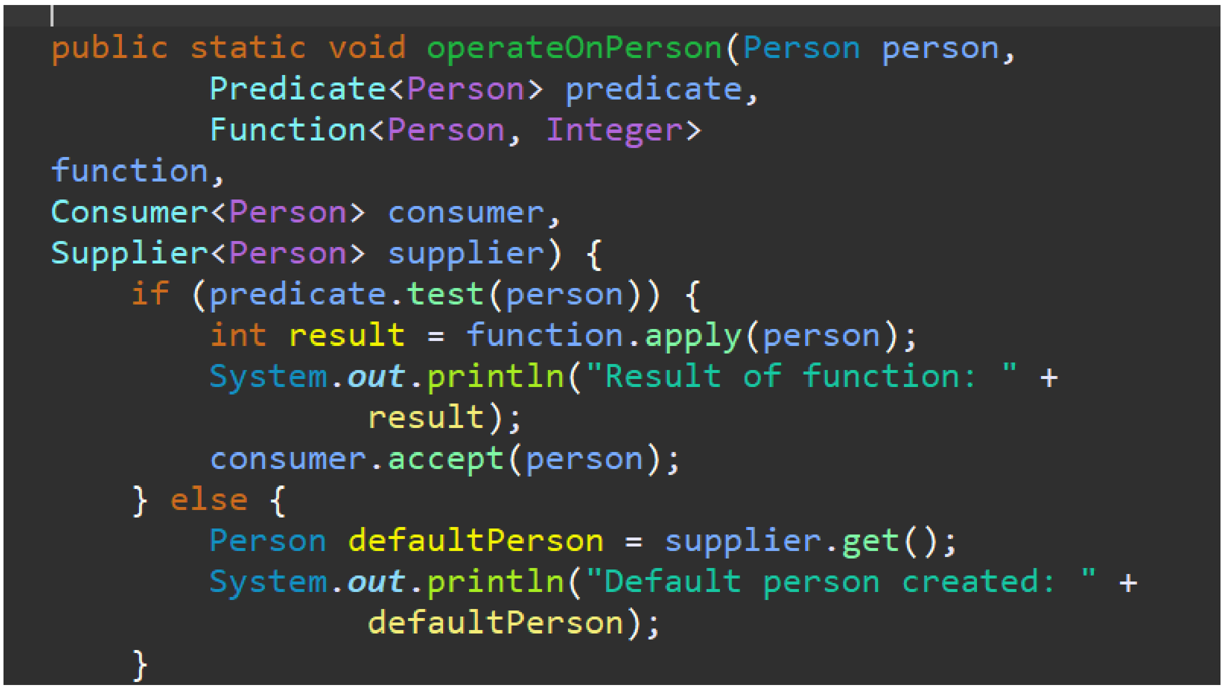
```java
import java.util.Comparator;
import java.util.List;
class Person {
    private String name;
    private int age;
    public Person(String name, int age) {
        this.name = name;
        this.age = age;
    }
    public String getName() {
        return name;
    }
    public int getAge() {
        return age;
    }
    @Override
    public String toString() {
        return name + " - " + age;
    }
}
public class Main {
    public static void main(String[] args) {
        List<Person> people = new ArrayList<>();
        people.add(new Person("Alice", 30));
        people.add(new Person("Bob", 25));
        people.add(new Person("Charlie", 35));
        Comparator<Person> ageComparator = (p1, p2)
                -> p1.getAge() - p2.getAge();
                Collections.sort(people, ageComparator);
                for (Person person : people) {
                    System.out.println(person);
                }
```

```
Bob - 25
Alice - 30
Charlie - 35
```

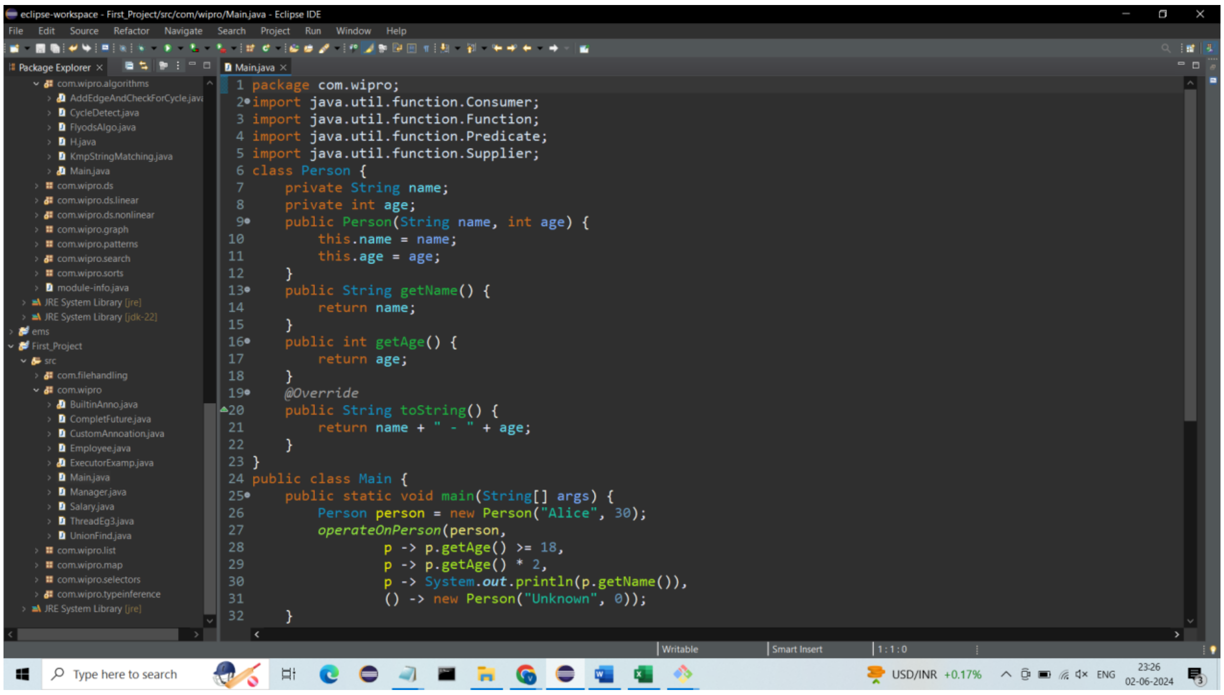
## Task 5: Functional Interfaces

Create a method that accepts functions as parameters using Predicate, Function, Consumer, and Supplier interfaces to operate on a Person object.

**Method: -**

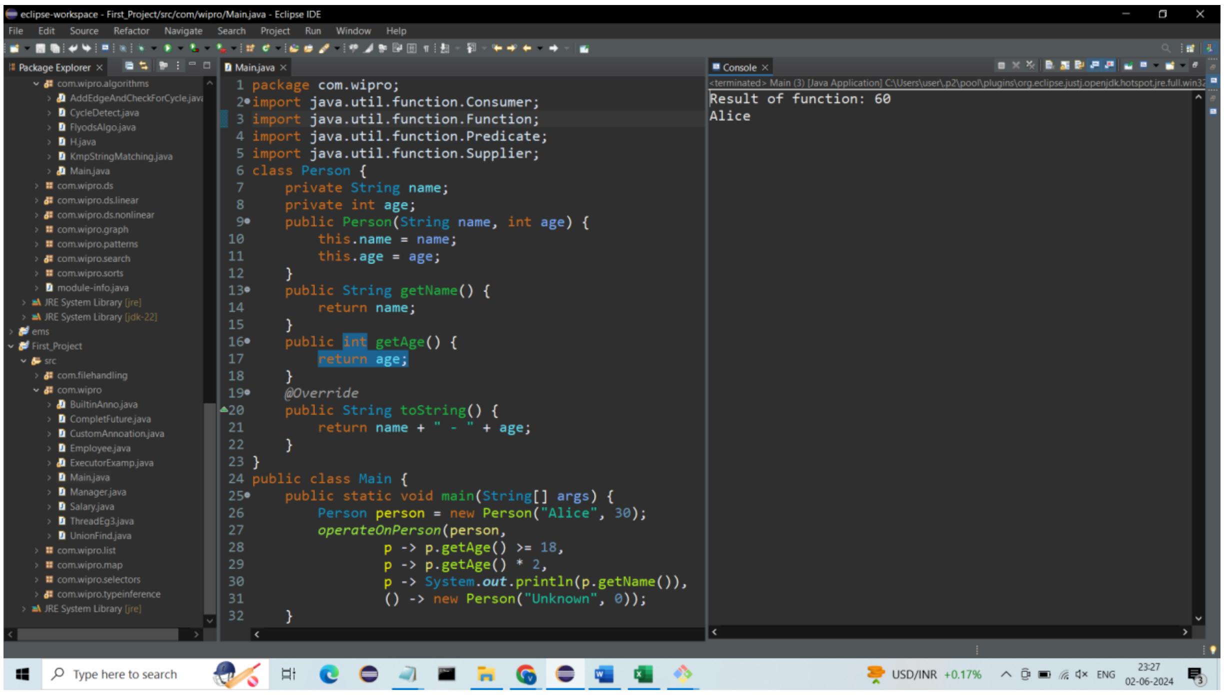


```java
public static void operateOnPerson(Person person,
        Predicate<Person> predicate,
        Function<Person, Integer>
function,
Consumer<Person> consumer,
Supplier<Person> supplier) {
    if (predicate.test(person)) {
        int result = function.apply(person);
        System.out.println("Result of function: " +
            result);
        consumer.accept(person);
    } else {
        Person defaultPerson = supplier.get();
        System.out.println("Default person created: " +
            defaultPerson);
    }
```


eclipse-workspace - First_Project/src/com/wipro/Main.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer
Main.java
Writable
Smart Insert
1 : 1 : 0
Type here to search
USD/INR +0.17%
ENG
23:26
02-06-2024

```java
package com.wipro;
import java.util.function.Consumer;
import java.util.function.Function;
import java.util.function.Predicate;
import java.util.function.Supplier;
class Person {
    private String name;
    private int age;
    public Person(String name, int age) {
        this.name = name;
        this.age = age;
    }
    public String getName() {
        return name;
    }
    public int getAge() {
        return age;
    }
    @Override
    public String toString() {
        return name + " - " + age;
    }
}
public class Main {
    public static void main(String[] args) {
        Person person = new Person("Alice", 30);
        operateOnPerson(person,
                p -> p.getAge() >= 18,
                p -> p.getAge() * 2,
                p -> System.out.println(p.getName()),
                () -> new Person("Unknown", 0));
    }
```

## Output: -



```java
package com.wipro;
import java.util.function.Consumer;
import java.util.function.Function;
import java.util.function.Predicate;
import java.util.function.Supplier;
class Person {
    private String name;
    private int age;
    public Person(String name, int age) {
        this.name = name;
        this.age = age;
    }
    public String getName() {
        return name;
    }
    public int getAge() {
        return age;
    }
    @Override
    public String toString() {
        return name + " - " + age;
    }
}
public class Main {
    public static void main(String[] args) {
        Person person = new Person("Alice", 30);
        operateOnPerson(person,
                p -> p.getAge() >= 18,
                p -> p.getAge() * 2,
                p -> System.out.println(p.getName()),
                () -> new Person("Unknown", 0));
    }
```

```
<terminated> Main (3) [Java Application] C:\Users\user\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32
Result of function: 60
Alice
```