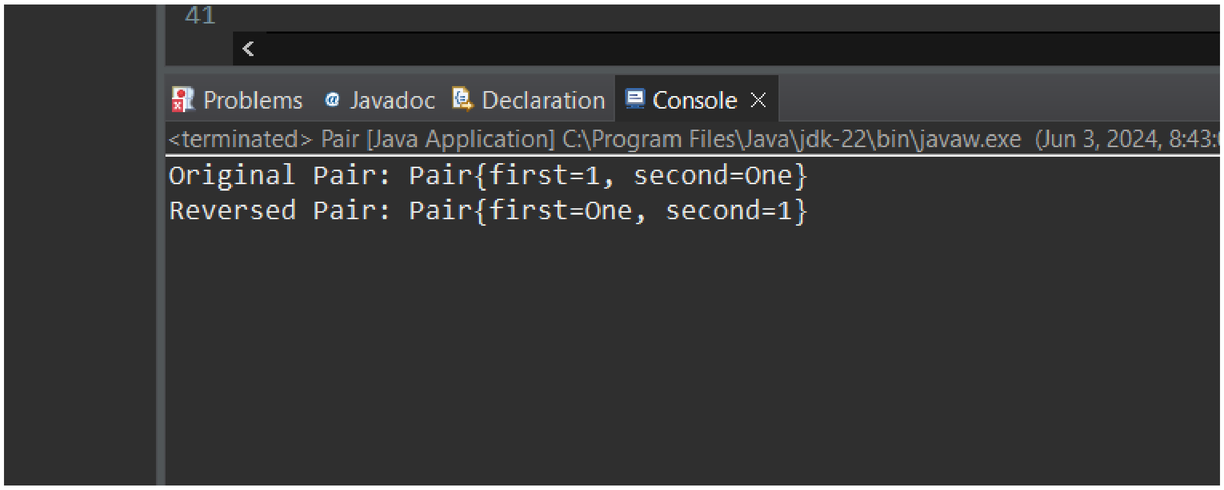
Task 1: Generics and Type Safety

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

Ans:

```
package Practice3;


public class Pair<T, U> {

 private T first;

 private U second;


 public Pair(T first, U second) {

     this.first = first;

     this.second = second;

 }


 public T getFirst() {

     return first;

 }


 public U getSecond() {

     return second;

 }


 public Pair<U, T> reverse() {

     return new Pair<>(second, first);

 }
 @Override
 public String toString() {
    return "Pair{" +
          "first=" + first +
          ", second=" + second +
          '}';
 }

 public static void main(String[] args) {
    Pair<Integer, String> originalPair = new Pair<>(1, "One");
    System.out.println("Original Pair: " + originalPair);

    Pair<String, Integer> reversedPair = originalPair.reverse();
    System.out.println("Reversed Pair: " + reversedPair);
 }
}
```

Output:



```
<terminated> Pair [Java Application] C:\Program Files\Java\jdk-22\bin\javaw.exe  (Jun 3, 2024, 8:43:
Original Pair: Pair{first=1, second=One}
Reversed Pair: Pair{first=One, second=1}
```

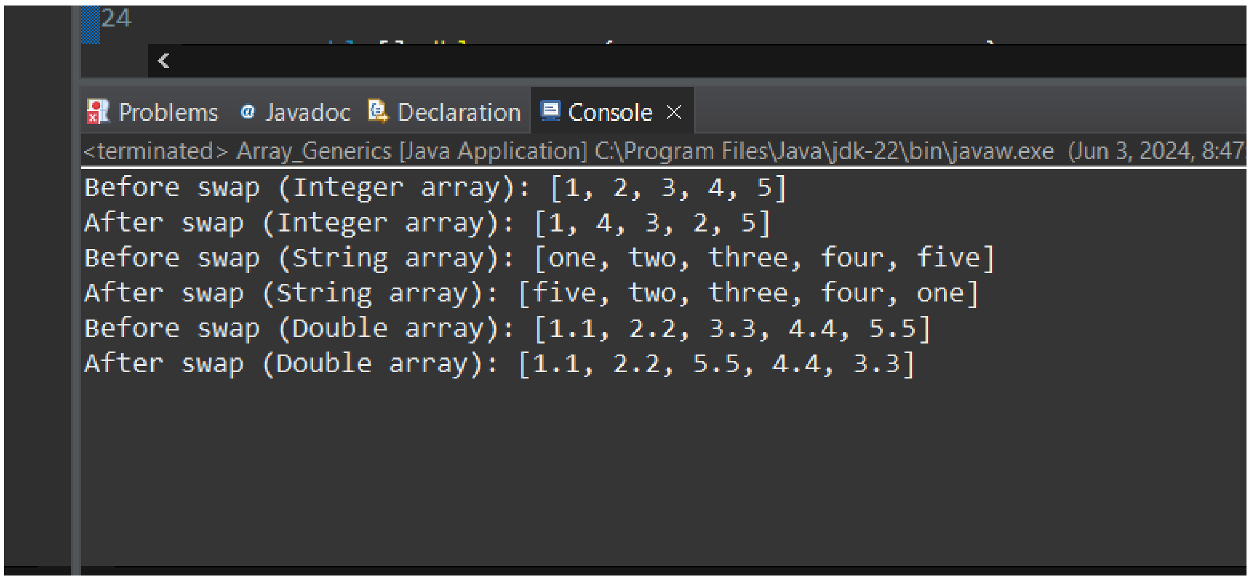
Task 2: Generic Classes and Methods

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.

Ans:

```
package Practice3;

import java.util.Arrays;

public class Array_Generics {

  public static <T> void swap(T[] array, int index1, int index2) {
    T temp = array[index1];
    array[index1] = array[index2];
    array[index2] = temp;
  }

  public static void main(String[] args) {

    Integer[] intArray = {1, 2, 3, 4, 5};
    System.out.println("Before swap (Integer array): " + Arrays.toString(intArray));
    swap(intArray, 1, 3);
    System.out.println("After swap (Integer array): " + Arrays.toString(intArray));

    String[] strArray = {"one", "two", "three", "four", "five"};
    System.out.println("Before swap (String array): " + Arrays.toString(strArray));
    swap(strArray, 0, 4);
```

```
        System.out.println("After swap (String array): " + Arrays.toString(strArray));


        Double[] dblArray = {1.1, 2.2, 3.3, 4.4, 5.5};
        System.out.println("Before swap (Double array): " + Arrays.toString(dblArray));
        swap(dblArray, 2, 4);
        System.out.println("After swap (Double array): " + Arrays.toString(dblArray));
    }
}
```



```
<terminated> Array_Generics [Java Application] C:\Program Files\Java\jdk-22\bin\javaw.exe (Jun 3, 2024, 8:47
Before swap (Integer array): [1, 2, 3, 4, 5]
After swap (Integer array): [1, 4, 3, 2, 5]
Before swap (String array): [one, two, three, four, five]
After swap (String array): [five, two, three, four, one]
Before swap (Double array): [1.1, 2.2, 3.3, 4.4, 5.5]
After swap (Double array): [1.1, 2.2, 5.5, 4.4, 3.3]
```

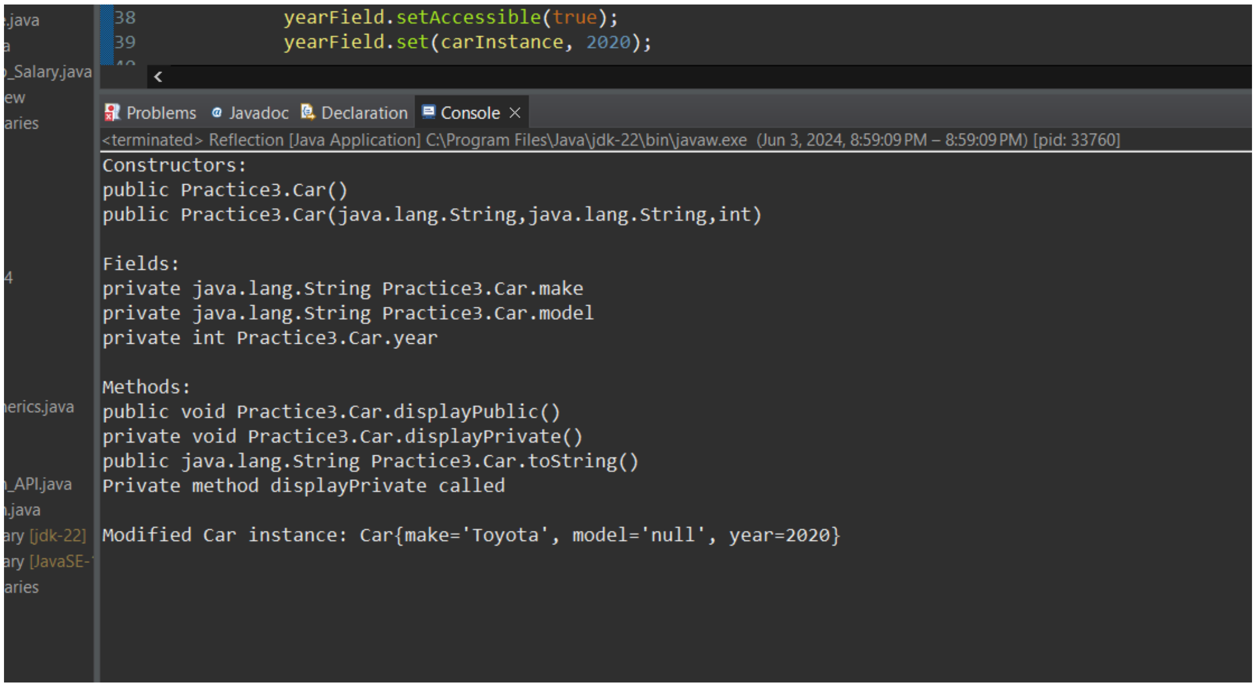
Task 3: Reflection API

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

```
package Practice3;
public class Car {
    private String make;
    private String model;
    private int year;

    public Car() {
        // Default constructor
    }

    public Car(String make, String model, int year) {
        this.make = make;
        this.model = model;
        this.year = year;
    }

    private void displayPrivate() {
        System.out.println("Private method displayPrivate called");
    }

    public void displayPublic() {
        System.out.println("Public method displayPublic called");
    }

    @Override
    public String toString() {
        return "Car{make='" + make + "', model='" + model + "', year=" + year + '}';
    }
}
```

```java
package Practice3;


import java.lang.reflect.Constructor;
import java.lang.reflect.Field;
import java.lang.reflect.Method;


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


            Class<?> carClass = Car.class;


            Constructor<?>[] constructors = carClass.getDeclaredConstructors();
            System.out.println("Constructors:");
            for (Constructor<?> constructor : constructors) {
                System.out.println(constructor);
            }


            Field[] fields = carClass.getDeclaredFields();
            System.out.println("\nFields:");
            for (Field field : fields) {
                System.out.println(field);
            }




            Method[] methods = carClass.getDeclaredMethods();
            System.out.println("\nMethods:");
            for (Method method : methods) {
                System.out.println(method);
            }

            Object carInstance = carClass.getDeclaredConstructor().newInstance();
            Field makeField = carClass.getDeclaredField("make");
            makeField.setAccessible(true);
            makeField.set(carInstance, "Toyota");

            Field yearField = carClass.getDeclaredField("year");
            yearField.setAccessible(true);
            yearField.set(carInstance, 2020);



            Method privateMethod = carClass.getDeclaredMethod("displayPrivate");
            privateMethod.setAccessible(true);
            privateMethod.invoke(carInstance);

            System.out.println("\nModified Car instance: " + carInstance);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

OutPut:



```
            yearField.setAccessible(true);
            yearField.set(carInstance, 2020);
```

```
<terminated> Reflection [Java Application] C:\Program Files\Java\jdk-22\bin\javaw.exe (Jun 3, 2024, 8:59:09 PM – 8:59:09 PM) [pid: 33760]
Constructors:
public Practice3.Car()
public Practice3.Car(java.lang.String,java.lang.String,int)

Fields:
private java.lang.String Practice3.Car.make
private java.lang.String Practice3.Car.model
private int Practice3.Car.year

Methods:
public void Practice3.Car.displayPublic()
private void Practice3.Car.displayPrivate()
public java.lang.String Practice3.Car.toString()
Private method displayPrivate called

Modified Car instance: Car{make='Toyota', model='null', year=2020}
```

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..

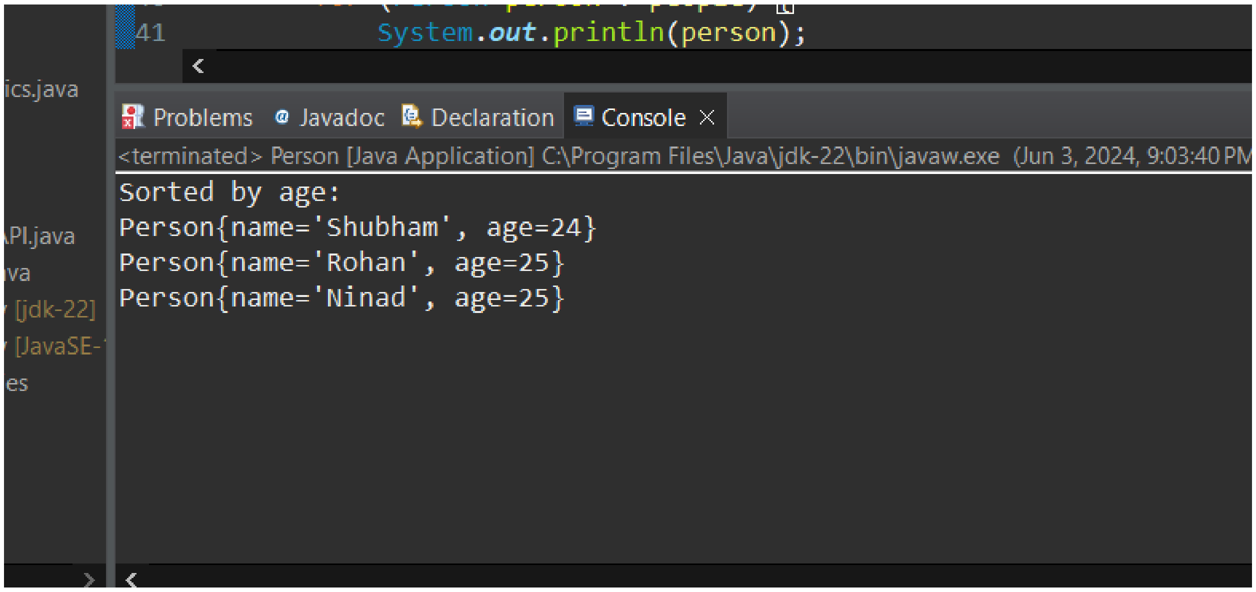
Ans:

```
package Practice3;


import java.util.ArrayList;

import java.util.Comparator;

import java.util.List;


public 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 "Person{name='" + name + "', age=" + age + '}';

}


public static void main(String[] args) {


    List<Person> people = new ArrayList<>();

    people.add(new Person("Shubham", 24));

    people.add(new Person("Rohan", 25));

    people.add(new Person("Ninad", 25));




    people.sort(Comparator.comparingInt(Person::getAge));
```

```
        System.out.println("Sorted by age:");
        for (Person person : people) {
            System.out.println(person);
        }
    }
}
```

Output:



```
41            System.out.println(person);
```

Problems Javadoc Declaration Console

```
<terminated> Person [Java Application] C:\Program Files\Java\jdk-22\bin\javaw.exe  (Jun 3, 2024, 9:03:40 PM
Sorted by age:
Person{name='Shubham', age=24}
Person{name='Rohan', age=25}
Person{name='Ninad', age=25}
```

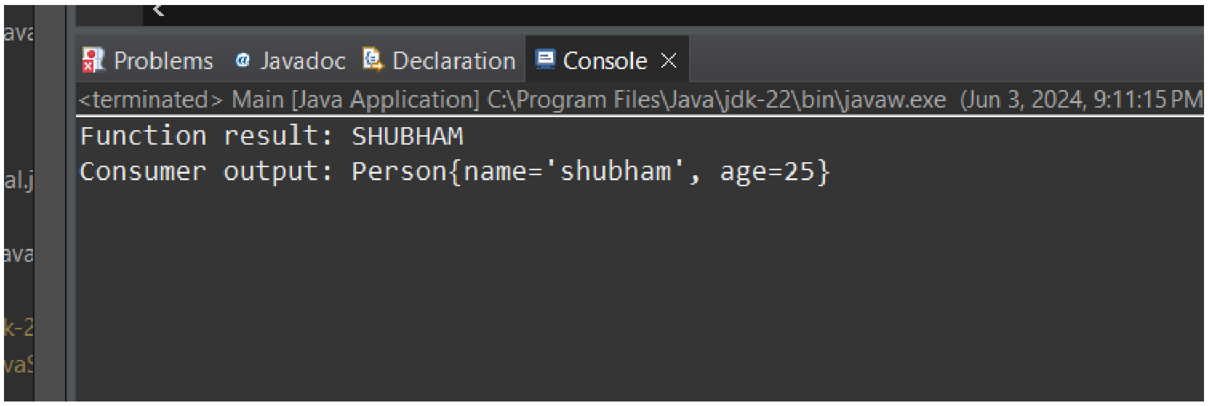
Task 5: Functional Interfaces

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

Ans:

```
package Practice3;

import java.util.function.Consumer;
import java.util.function.Function;
import java.util.function.Predicate;

import java.util.function.Supplier;


class Person1 {

    String name;

    int age;


    public Person1(String name, int age) {

        this.name = name;

        this.age = age;

    }


    @Override

    public String toString() {

        return "Person{" +

                "name='" + name + '\'' +

                ", age=" + age +

                '}';

    }

}


public class Main {


    public static void operateOnPerson(Person1 person,

                                       Predicate<Person1> predicate,

                                       Function<Person1, String> function,

                                       Consumer<Person1> consumer,
                                    Supplier<Person1> supplier) {
        if (predicate.test(person)) {
            String result = function.apply(person);
            System.out.println("Function result: " + result);
            consumer.accept(person);
        } else {
            Person1 newPerson = supplier.get();
            System.out.println("Supplier created: " + newPerson);
        }
    }

    public static void main(String[] args) {
        Person1 person = new Person1("shubham", 25);

        operateOnPerson(
                person,
                p -> p.age > 18,
                p -> p.name.toUpperCase(),
                p -> System.out.println("Consumer output: " + p),
                () -> new Person1("New Person", 20)
        );
    }
}
```

Ans:



```
<terminated> Main [Java Application] C:\Program Files\Java\jdk-22\bin\javaw.exe  (Jun 3, 2024, 9:11:15 PM
Function result: SHUBHAM
Consumer output: Person{name='shubham', age=25}
```