

Name: Vishakha Avinash kale

Day19

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.

```
package com.assig.advancejavafeatureandjava8;
```

```
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();

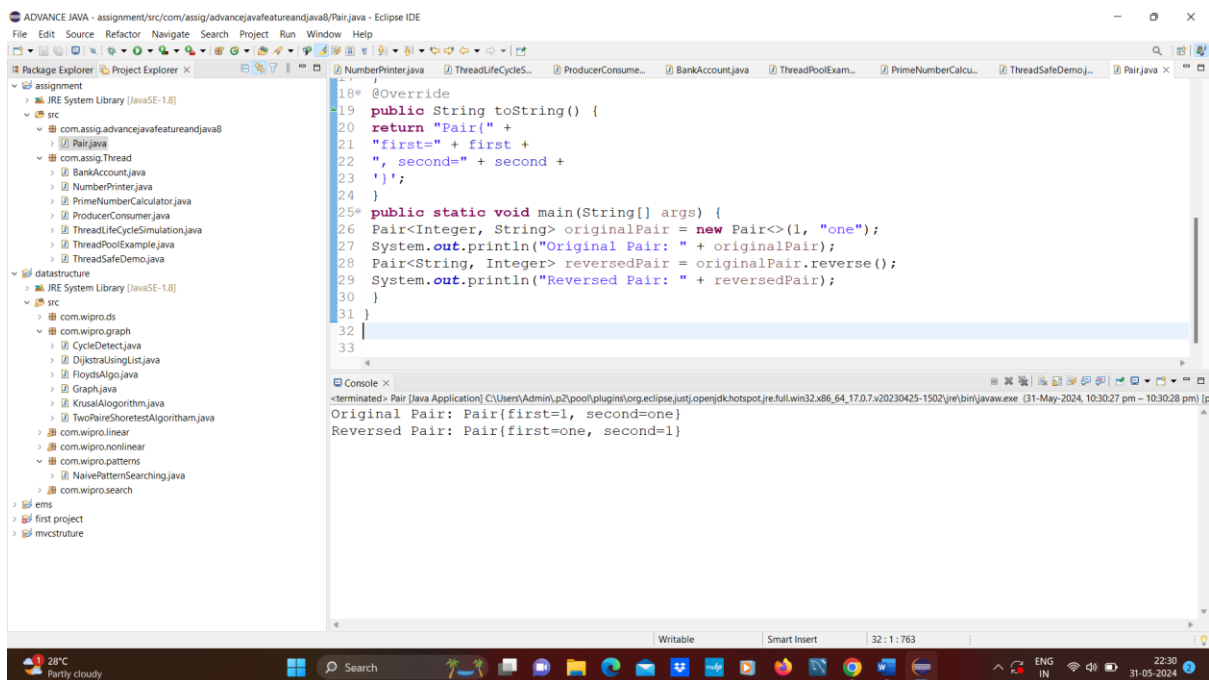
System.out.println("Reversed Pair: " + reversedPair);

}

}

```

OUTPUT:



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.

```

package com.assig.advancejavafeatureandjava8;

public class ArrayUtil {

    public static <T> void swap(T[] array, int index1, int index2) { if (index1 < 0 || index1 >=
array.length || index2 < 0 || index2 >= array.length) {

        throw new IndexOutOfBoundsException("Index out of bounds");

    }

    T temp = array[index1];

```

```

array[index1] = array[index2];
array[index2] = temp;
}

public static void main(String[] args) {

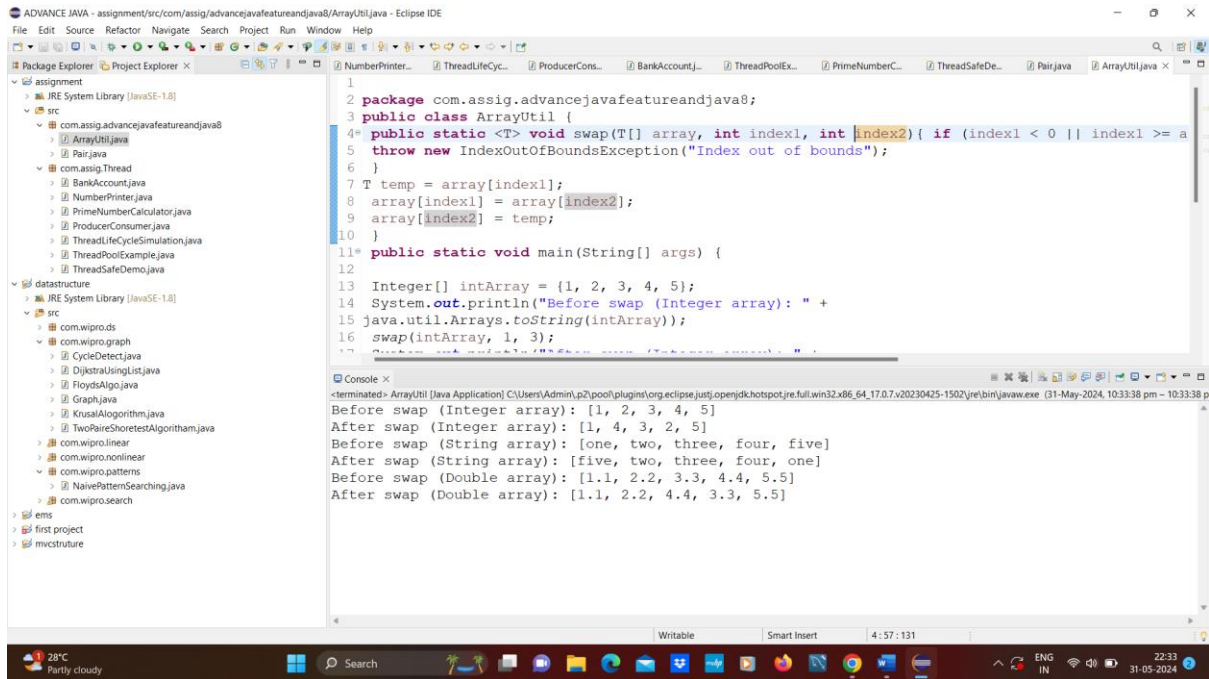
    Integer[] intArray = {1, 2, 3, 4, 5};
    System.out.println("Before swap (Integer array): " +
        java.util.Arrays.toString(intArray));
    swap(intArray, 1, 3);
    System.out.println("After swap (Integer array): " +
        java.util.Arrays.toString(intArray));

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

    Double[] doubleArray = {1.1, 2.2, 3.3, 4.4, 5.5};
    System.out.println("Before swap (Double array): " +
        java.util.Arrays.toString(doubleArray));
    swap(doubleArray, 2, 3);
    System.out.println("After swap (Double array): " +
        java.util.Arrays.toString(doubleArray));
}
}

```

OUTPUT:



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 com.assig.advancejavafeatureandjava8;
```

```
import java.lang.reflect.Field;
```

```
import java.lang.reflect.Modifier;
```

```
public class ReflectionExample {
```

```
    private String privateField = "initialValue";
```

```
    public static void main(String[] args) throws NoSuchFieldException,
```

```
    IllegalAccessException {
```

```
        ReflectionExample obj = new ReflectionExample();
```

```
        // Inspecting the class's fields
```

```
        Field[] fields = ReflectionExample.class.getDeclaredFields();
```

```
        for (Field field : fields) {
```

```
            System.out.println("Field name: " + field.getName());
```

```
            System.out.println("Field type: " + field.getType());
```

```
            System.out.println("Field modifiers: " +
```

```

Modifier.toString(field.getModifiers()));
}

// Modifying the access level of a private field and setting its
value

Field privateField =
ReflectionExample.class.getDeclaredField("privateField");
privateField.setAccessible(true); // Allow access to private field
privateField.set(obj, "modifiedValue"); // Set new value

// Accessing the modified private field

System.out.println("Modified private field value: " +
obj.privateField);
}
}

```

OUTPUT:

The screenshot shows the Eclipse IDE with the ReflectionExample.java file open. The code in the file is as follows:

```

12 for (Field field : fields) {
13     System.out.println("Field name: " + field.getName());
14     System.out.println("Field type: " + field.getType());
15     System.out.println("Field modifiers: " +
16         Modifier.toString(field.getModifiers()));
17 }
18 // Modifying the access level of a private field and setting its value
19 Field privateField =
20 ReflectionExample.class.getDeclaredField("privateField");
21 privateField.setAccessible(true); // Allow access to private field
22 privateField.set(obj, "modifiedValue"); // Set new value
23 // Accessing the modified private field
24 System.out.println("Modified private field value: " +
25     obj.privateField);
26 }
27 }

```

The console output shows the following results:

```

<terminated> ReflectionExample [Java Application] C:\Users\Admin\p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_17.0.7.v20230425-1502\jre\bin\javaw.exe (31-May-2024, 10:36:55 pm)
Field name: privateField
Field type: class java.lang.String
Field modifiers: private
Modified private field value: modifiedValue

```

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..

```

package com.assig.advancejavafeatureandjava8;

import java.util.ArrayList;

```

```
import java.util.Comparator;
import java.util.List;

public class PersonComparators {

    private String name;

    private int age;

    public PersonComparators(String name, int age) {
        this.name = name;
        this.age = age;
    }

    public String getName() {
        return name;
    }

    public int getAge() {
        return age;
    }

    public static void main(String[] args) {
        List<PersonComparators> personList = new ArrayList<>();
        personList.add(new PersonComparators("Alice", 25));
        personList.add(new PersonComparators("Bob", 30));
        personList.add(new PersonComparators("Charlie", 20));
        // Sorting the list by age using a lambda expression

        personList.sort(Comparator.comparingInt(PersonComparators::getAge));

        // Printing the sorted list
        for (PersonComparators person : personList) {
            System.out.println("Name: " + person.getName() + ", Age: " +
                person.getAge());
        }
    }
}
```

```
}  
  
}
```

OUTPUT:

```
1  
2 package com.assig.advancejavafeatureandjava8;  
3 import java.util.ArrayList;  
4  
5  
6 public class PersonComparators {  
7     private String name;  
8     private int age;  
9     public PersonComparators(String name, int age) {  
10        this.name = name;  
11        this.age = age;  
12    }  
13    public String getName() {  
14        return name;  
15    }  
16    public int getAge() {  
17        return age;  
18    }  
19 }  
20  
21  
22  
23  
24  
25  
26  
27  
28  
29  
30  
31  
32  
33  
34  
35  
36  
37  
38  
39  
40  
41  
42  
43  
44  
45  
46  
47  
48  
49  
50  
51  
52  
53  
54  
55  
56  
57  
58  
59  
60  
61  
62  
63  
64  
65  
66  
67  
68  
69  
70  
71  
72  
73  
74  
75  
76  
77  
78  
79  
80  
81  
82  
83  
84  
85  
86  
87  
88  
89  
90  
91  
92  
93  
94  
95  
96  
97  
98  
99  
100
```

```
<terminated> PersonComparators [Java Application] C:\Users\Admin\p2\pool\plugins\org.eclipse.justi.openjdk.hotspot.jre.full.win32.x86_64.17.0.7.v20230425-1502\jre\bin\javaw.exe (31-May-2024, 10:42:59 pm -  
Name: Charlie, Age: 20  
Name: Alice, Age: 25  
Name: Bob, Age: 30
```

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.

```
package com.assig.advancejavafeatureandjava8;
```

```
import java.util.function.Consumer;
```

```
import java.util.function.Function;
```

```
import java.util.function.Predicate;
```

```
import java.util.function.Supplier;
```

```
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;
}

public void setName(String name) {
    this.name = name;
}

public void setAge(int age) {
    this.age = age;
}

public static void processPerson(Person person,
    Predicate<Person> predicate,
    Function<Person, String> function,
    Consumer<String> consumer,
    Supplier<Integer> supplier) {
    if (predicate.test(person)) {
        String result = function.apply(person);
        consumer.accept(result);
        int newAge = supplier.get();
        person.setAge(newAge);
    }
}

public static void main(String[] args) {
    Person person = new Person("vishakha", 24);
    // Example usage of the processPerson method
    processPerson(

```


person,

p -> p.getAge() >= 18, // Predicate to check if person is an

adult

p -> "Name: " + p.getName() + ", Age: " + p.getAge(), //

Function to get person details as string

System.out::println, // Consumer to print the person details

() -> 30 // Supplier to provide a new age for the person

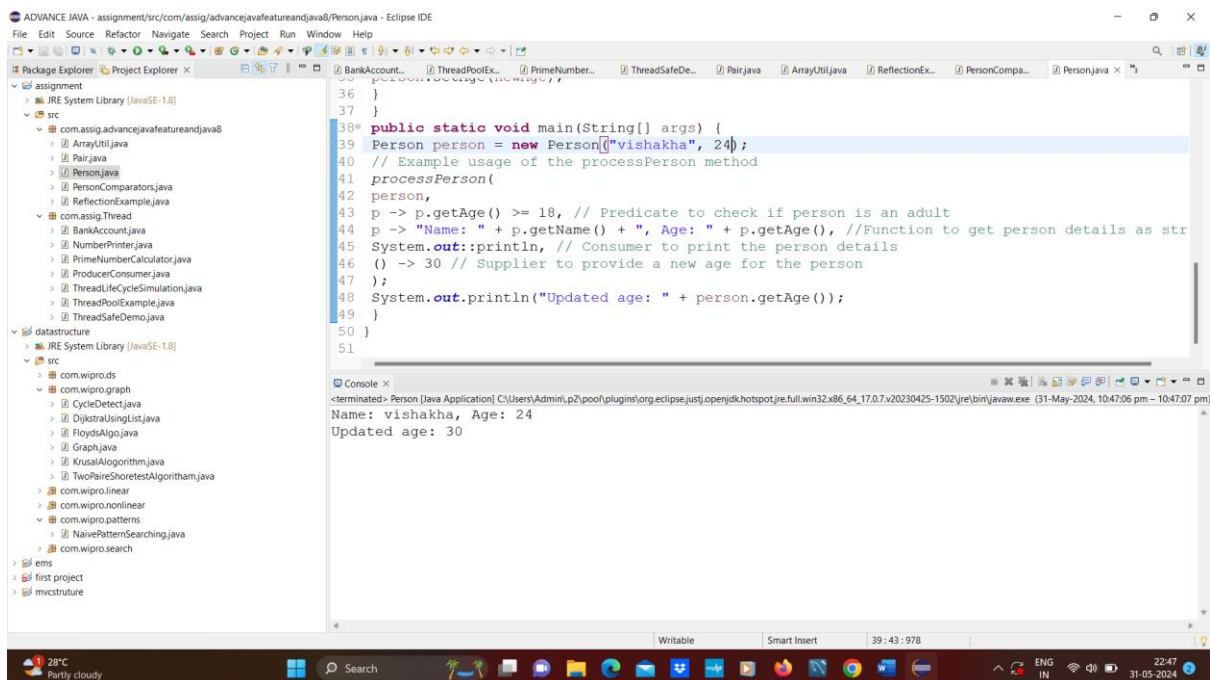
);

System.out.println("Updated age: " + person.getAge());

}

}

OUTPUT:



The screenshot shows the Eclipse IDE with a Java project named 'ADVANCE JAVA'. The 'Project Explorer' on the left shows the project structure, including a 'src' folder with various Java files. The 'Main' editor displays the following code:

```
36 }
37 }
38 public static void main(String[] args) {
39     Person person = new Person("vishakha", 24);
40     // Example usage of the processPerson method
41     processPerson(
42         person,
43         p -> p.getAge() >= 18, // Predicate to check if person is an adult
44         p -> "Name: " + p.getName() + ", Age: " + p.getAge(), //Function to get person details as str
45         System.out::println, // Consumer to print the person details
46         () -> 30 // Supplier to provide a new age for the person
47     );
48     System.out.println("Updated age: " + person.getAge());
49 }
50 }
51 }
```

The 'Console' window at the bottom shows the output of the program:

```
<terminated> Person [Java Application] C:\Users\Admin\p2\pool\plugins\org.eclipse.justi.openjdk.hotspot.jre.full.win32.x86_64.17.0.7.v20230425-1502\jre\bin\javaw.exe (31-May-2024, 10:47:06 pm - 10:47:07 pm)
Name: vishakha, Age: 24
Updated age: 30
```