

Name : Tejaswini Anil Kamble

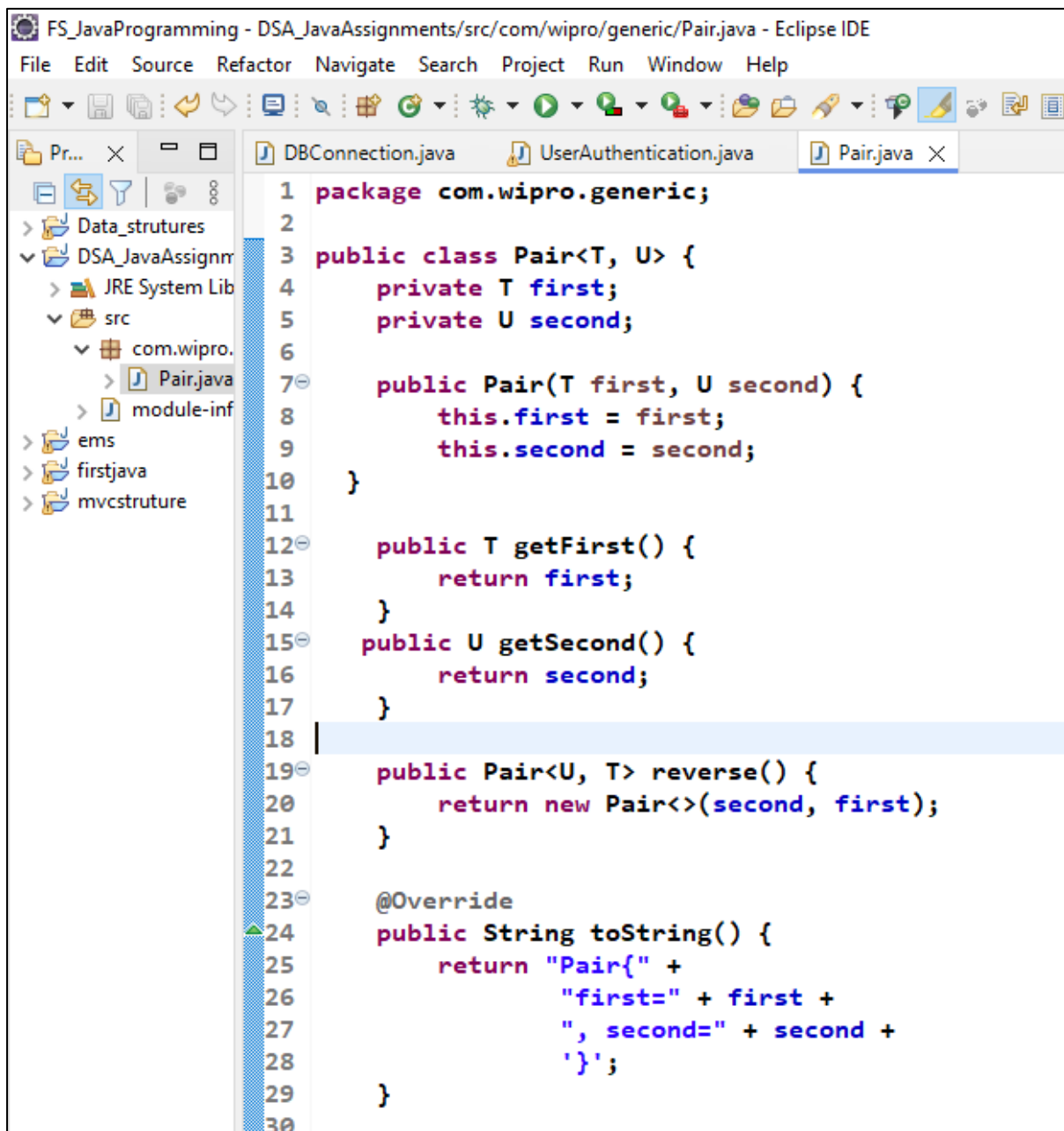
Email: teju000kamble@gmail.com

Day 19 : Assignments

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: Source Code

The image is a screenshot of the Eclipse IDE interface. The title bar at the top reads "FS_JavaProgramming - DSA_JavaAssignments/src/com/wipro/generic/Pair.java - Eclipse IDE". Below the title bar is a menu bar with options: File, Edit, Source, Refactor, Navigate, Search, Project, Run, Window, and Help. A toolbar with various icons is located below the menu bar. On the left side, there is a Project Explorer showing a tree structure of the project. The tree includes folders like "Data_structures", "DSA_JavaAssignm...", "JRE System Lib", "src", "com.wipro.", "module-inf", "ems", "firstjava", and "mvcstruture". The "Pair.java" file is selected under the "com.wipro." folder. The main editor area on the right displays the Java source code for "Pair.java". The code is as follows:

```
1 package com.wipro.generic;
2
3 public class Pair<T, U> {
4     private T first;
5     private U second;
6
7     public Pair(T first, U second) {
8         this.first = first;
9         this.second = second;
10    }
11
12    public T getFirst() {
13        return first;
14    }
15    public U getSecond() {
16        return second;
17    }
18
19    public Pair<U, T> reverse() {
20        return new Pair<>(second, first);
21    }
22
23    @Override
24    public String toString() {
25        return "Pair{" +
26            "first=" + first +
27            ", second=" + second +
28            '}';
29    }
30
```

```

30
31 public static void main(String[] args) {
32     Pair<Integer, String> originalPair = new Pair<>(1, "one");
33     System.out.println("Original Pair: " + originalPair);
34
35     Pair<String, Integer> reversedPair = originalPair.reverse();
36     System.out.println("Reversed Pair: " + reversedPair);
37 }
38 }
39

```

Output:

```

Console X
<terminated> Pair [Java Application] C:\Program Files\Java\jdk-17\bin\javaw.exe (Jun 1, 2024, 9:51:06 PM – 9:51:07 PM) [pid: 13576]
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.

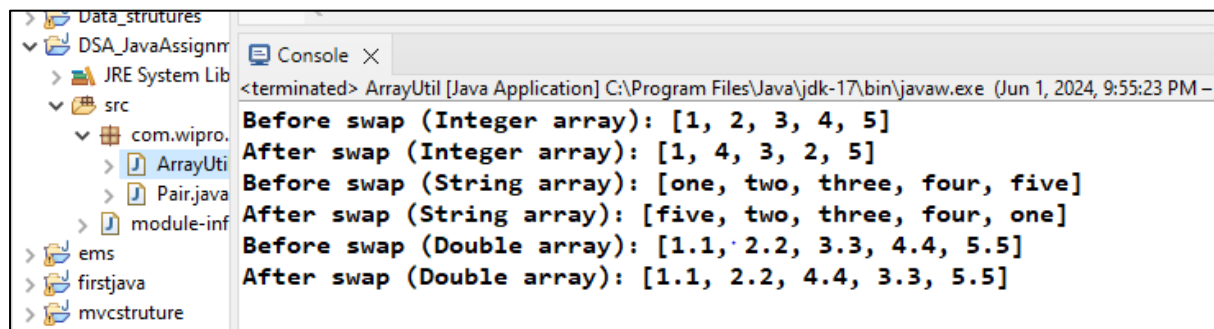
Ans: Source Code

```

FS_JavaProgramming - DSA_JavaAssignments/src/com/wipro/generic/ArrayUtil.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Pair.java ArrayUtil.java X
Data_structures
DSA_JavaAssignm
JRE System Lib
src
com.wipro.
ArrayUtil
Pair.java
module-inf
ems
firstjava
mvstructure
1 package com.wipro.generic;
2
3 public class ArrayUtil {
4     public static <T> void swap(T[] array, int index1, int index2) {
5         if (index1 < 0 || index1 >= array.length || index2 < 0 || index2 >= array.length) {
6             throw new IndexOutOfBoundsException("Index out of bounds");
7         }
8
9         T temp = array[index1];
10        array[index1] = array[index2];
11        array[index2] = temp;
12    }
13
14    public static void main(String[] args) {
15
16        Integer[] intArray = {1, 2, 3, 4, 5};
17        System.out.println("Before swap (Integer array): " + java.util.Arrays.toString(intArray));
18        swap(intArray, 1, 3);
19        System.out.println("After swap (Integer array): " + java.util.Arrays.toString(intArray));
20
21        String[] strArray = {"one", "two", "three", "four", "five"};
22        System.out.println("Before swap (String array): " + java.util.Arrays.toString(strArray));
23        swap(strArray, 0, 4);
24        System.out.println("After swap (String array): " + java.util.Arrays.toString(strArray));
25
26        Double[] doubleArray = {1.1, 2.2, 3.3, 4.4, 5.5};
27        System.out.println("Before swap (Double array): " + java.util.Arrays.toString(doubleArray));
28        swap(doubleArray, 2, 3);
29        System.out.println("After swap (Double array): " + java.util.Arrays.toString(doubleArray));
30    }
31 }

```

Output:-

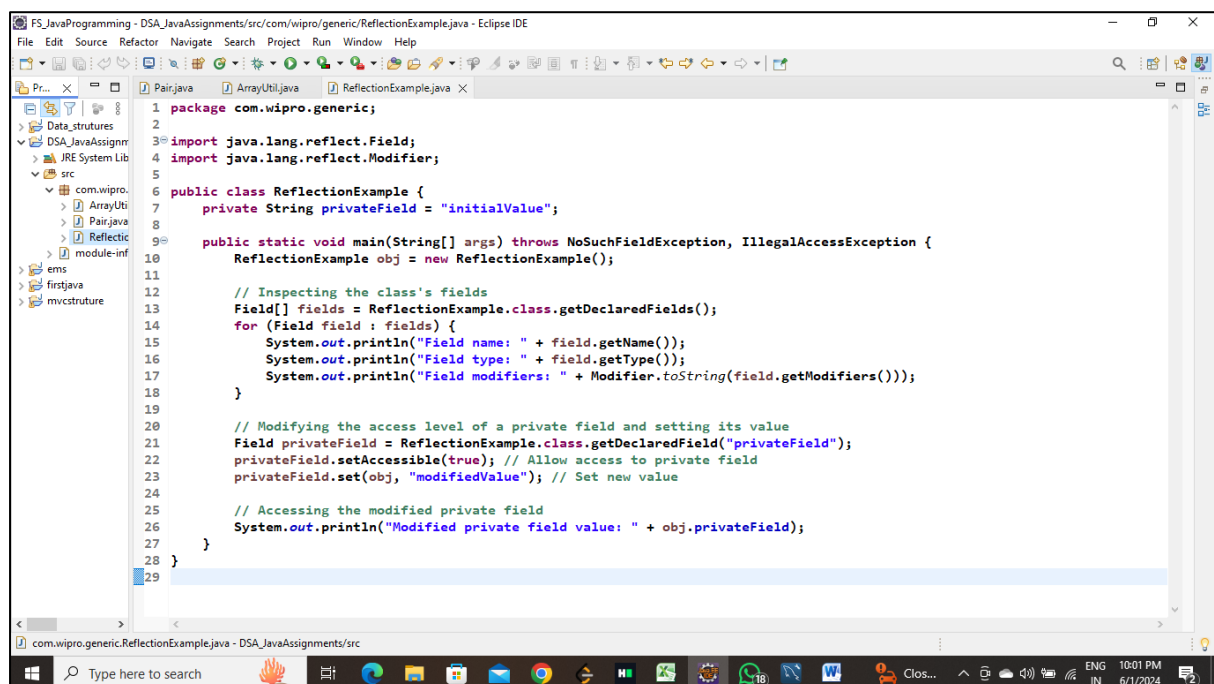


```
<terminated> ArrayUtil [Java Application] C:\Program Files\Java\jdk-17\bin\javaw.exe (Jun 1, 2024, 9:55:23 PM -
Before swap (Integer array): [1, 2, 3, 4, 5]
After swap (Integer array): [1, 4, 3, 2, 5]
Before swap (String array): [one, two, three, four, five]
After swap (String array): [five, two, 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, 4.4, 3.3, 5.5]
```

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

Ans: Source Code



```
1 package com.wipro.generic;
2
3 import java.lang.reflect.Field;
4 import java.lang.reflect.Modifier;
5
6 public class ReflectionExample {
7     private String privateField = "initialValue";
8
9     public static void main(String[] args) throws NoSuchFieldException, IllegalAccessException {
10         ReflectionExample obj = new ReflectionExample();
11
12         // Inspecting the class's fields
13         Field[] fields = ReflectionExample.class.getDeclaredFields();
14         for (Field field : fields) {
15             System.out.println("Field name: " + field.getName());
16             System.out.println("Field type: " + field.getType());
17             System.out.println("Field modifiers: " + Modifier.toString(field.getModifiers()));
18         }
19
20         // Modifying the access level of a private field and setting its value
21         Field privateField = ReflectionExample.class.getDeclaredField("privateField");
22         privateField.setAccessible(true); // Allow access to private field
23         privateField.set(obj, "modifiedValue"); // Set new value
24
25         // Accessing the modified private field
26         System.out.println("Modified private field value: " + obj.privateField);
27     }
28 }
29
```

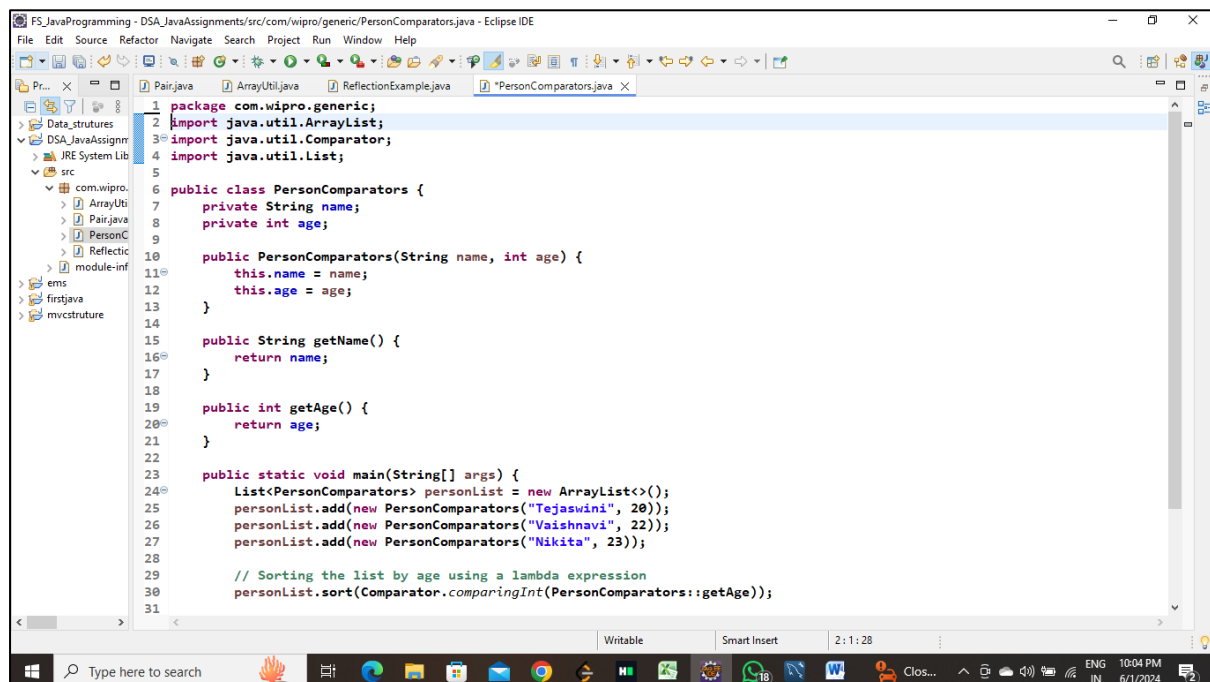
Output:-

```
ents Console X
<terminated> ReflectionExample [Java Application] C:\Program Files\Java\jdk-17\bin\javaw.exe (Jun 1, 2024, 10:00:37 PM - 10:00:37 PM) [pid: 816]
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..

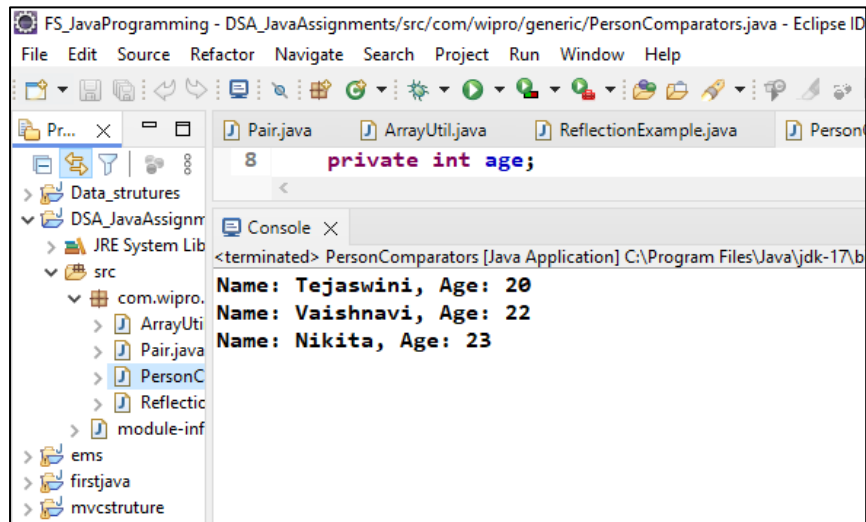
Ans: Source Code



```
FS_JavaProgramming - DSA_JavaAssignments/src/com/wipro/generic/PersonComparators.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Pair.java ArrayUtil.java ReflectionExample.java *PersonComparators.java X
Data_structures
DSA_JavaAssignm
JRE System Lib
src
com.wipro.
ArrayUtil
Pair.java
PersonC
Reflectic
module-inf
ems
firstjava
mvcstruture
1 package com.wipro.generic;
2 import java.util.ArrayList;
3 import java.util.Comparator;
4 import java.util.List;
5
6 public class PersonComparators {
7     private String name;
8     private int age;
9
10    public PersonComparators(String name, int age) {
11        this.name = name;
12        this.age = age;
13    }
14
15    public String getName() {
16        return name;
17    }
18
19    public int getAge() {
20        return age;
21    }
22
23    public static void main(String[] args) {
24        List<PersonComparators> personList = new ArrayList<>();
25        personList.add(new PersonComparators("Tejaswini", 20));
26        personList.add(new PersonComparators("Vaishnavi", 22));
27        personList.add(new PersonComparators("Nikita", 23));
28
29        // Sorting the list by age using a lambda expression
30        personList.sort(Comparator.comparingInt(PersonComparators::getAge));
31    }
32}
```

```
20
29 // Sorting the list by age using a lambda expression
30 personList.sort(Comparator.comparingInt(PersonComparators::getAge));
31
32 // Printing the sorted list
33 for (PersonComparators person : personList) {
34     System.out.println("Name: " + person.getName() + ", Age: " + person.getAge());
35 }
36 }
37 }
38
```

Output :

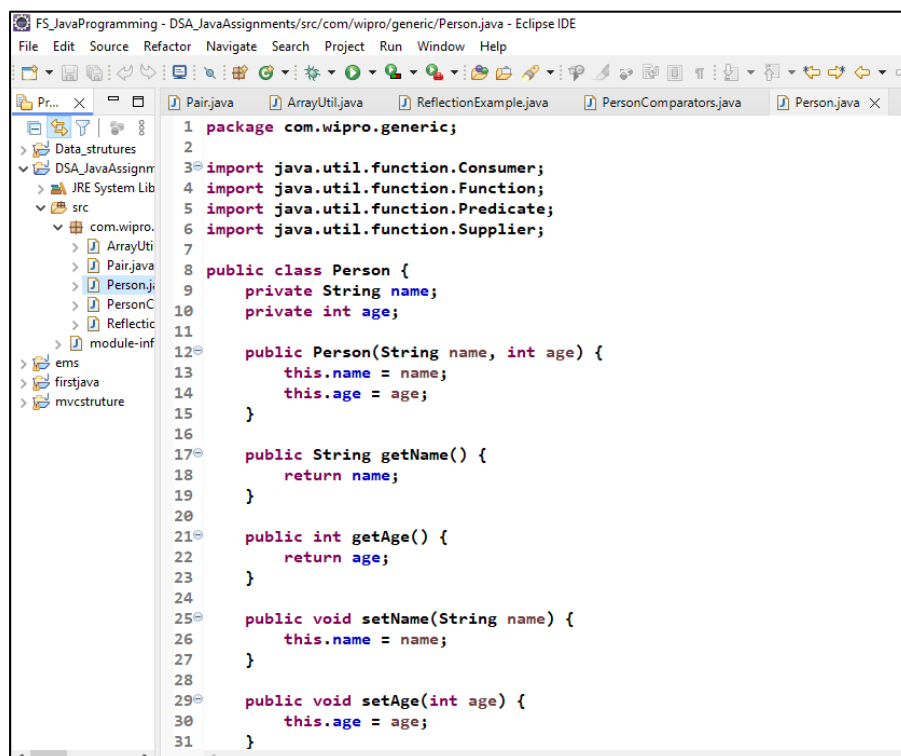


```
FS_JavaProgramming - DSA_JavaAssignments/src/com/wipro/generic/PersonComparators.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Pr... X
Data_structures
DSA_JavaAssignm
JRE System Lib
src
com.wipro.
ArrayUti
Pair.java
PersonC
Reflectic
module-inf
ems
firstjava
mvcstruture
Pair.java
ArrayUtil.java
ReflectionExample.java
Person
8 private int age;
Console X
<terminated> PersonComparators [Java Application] C:\Program Files\Java\jdk-17\bin
Name: Tejaswini, Age: 20
Name: Vaishnavi, Age: 22
Name: Nikita, Age: 23
```

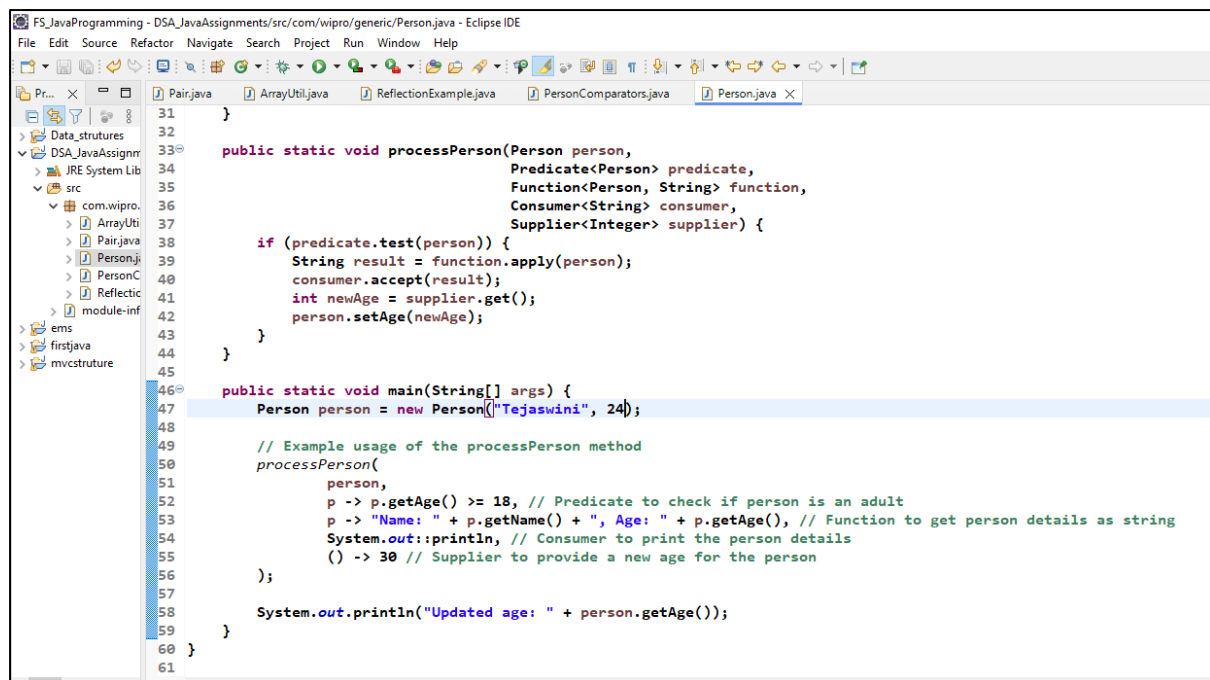
Task 5: Functional Interfaces

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

Ans: Source Code

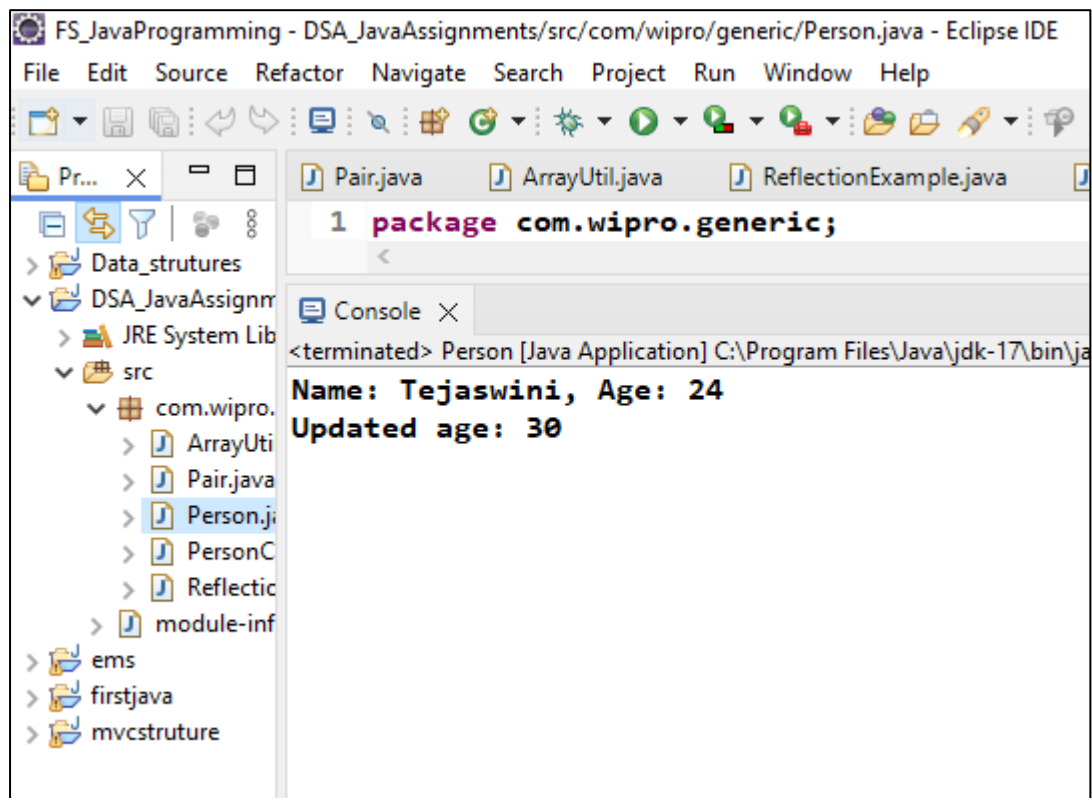


```
FS_JavaProgramming - DSA_JavaAssignments/src/com/wipro/generic/Person.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Pr... X
Data_structures
DSA_JavaAssignm
JRE System Lib
src
com.wipro.
ArrayUti
Pair.java
Person.j
PersonC
Reflectic
module-inf
ems
firstjava
mvcstruture
Pair.java
ArrayUtil.java
ReflectionExample.java
PersonComparators.java
Person.java X
1 package com.wipro.generic;
2
3 import java.util.function.Consumer;
4 import java.util.function.Function;
5 import java.util.function.Predicate;
6 import java.util.function.Supplier;
7
8 public class Person {
9     private String name;
10    private int age;
11
12    public Person(String name, int age) {
13        this.name = name;
14        this.age = age;
15    }
16
17    public String getName() {
18        return name;
19    }
20
21    public int getAge() {
22        return age;
23    }
24
25    public void setName(String name) {
26        this.name = name;
27    }
28
29    public void setAge(int age) {
30        this.age = age;
31    }
}
```



```
31 }
32
33 public static void processPerson(Person person,
34                                 Predicate<Person> predicate,
35                                 Function<Person, String> function,
36                                 Consumer<String> consumer,
37                                 Supplier<Integer> supplier) {
38     if (predicate.test(person)) {
39         String result = function.apply(person);
40         consumer.accept(result);
41         int newAge = supplier.get();
42         person.setAge(newAge);
43     }
44 }
45
46 public static void main(String[] args) {
47     Person person = new Person("Tejaswini", 24);
48
49     // Example usage of the processPerson method
50     processPerson(
51         person,
52         p -> p.getAge() >= 18, // Predicate to check if person is an adult
53         p -> "Name: " + p.getName() + ", Age: " + p.getAge(), // Function to get person details as string
54         System.out::println, // Consumer to print the person details
55         () -> 30 // Supplier to provide a new age for the person
56     );
57
58     System.out.println("Updated age: " + person.getAge());
59 }
60
61 }
```

Output:



```
1 package com.wipro.generic;
```

<terminated> Person [Java Application] C:\Program Files\Java\jdk-17\bin\ja

Name: Tejaswini, Age: 24

Updated age: 30