

Day 19 Assignment

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.

The screenshot shows the Eclipse IDE with a project named 'wiproworkspace'. The Package Explorer on the left shows the project structure, including a package named 'Assignment' containing several Java files. The main editor displays the code for 'Generics.java'.

```
1 package Assignment;
2 public class Generics<T, U> {
3     private T first;
4     private U second;
5     public Generics(T first, U second) {
6         this.first = first;
7         this.second = second;
8     }
9     public T getFirst() {
10        return first;
11    }
12    public U getSecond() {
13        return second;
14    }
15    public Generics<U, T> reverse() {
16        return new Generics<>(second, first);
17    }
18    public static void main(String[] args) {
19        Generics<String, Integer> pair = new Generics<>("Hello", 123);
20        System.out.println("Original Pair: " + pair.getFirst() + ", " + pair.getSecond());
21        Generics<Integer, String> reversedPair = pair.reverse();
22        System.out.println("Reversed Pair: " + reversedPair.getFirst() + ", " + reversedPair.getSecond());
23    }
24 }
```

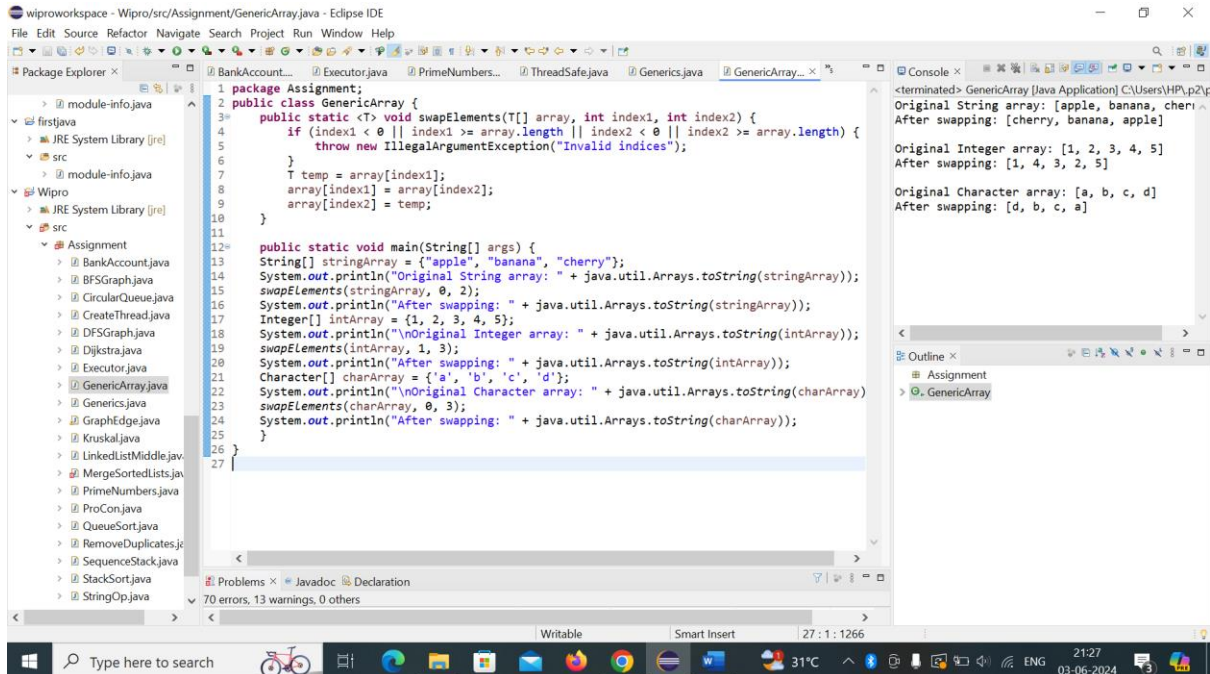
The Console window on the right shows the output of the program:

```
<terminated> Generics [Java Application] C:\Users\HP\...
Original Pair: Hello, 123
Reversed Pair: 123, Hello
```

The status bar at the bottom indicates '70 errors, 13 warnings, 0 others'.

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.



```
1 package Assignment;
2 public class GenericArray {
3     public static <T> void swapElements(T[] array, int index1, int index2) {
4         if (index1 < 0 || index1 >= array.length || index2 < 0 || index2 >= array.length) {
5             throw new IllegalArgumentException("Invalid indices");
6         }
7         T temp = array[index1];
8         array[index1] = array[index2];
9         array[index2] = temp;
10    }
11
12    public static void main(String[] args) {
13        String[] stringArray = {"apple", "banana", "cherry"};
14        System.out.println("Original String array: " + java.util.Arrays.toString(stringArray));
15        swapElements(stringArray, 0, 2);
16        System.out.println("After swapping: " + java.util.Arrays.toString(stringArray));
17        Integer[] intArray = {1, 2, 3, 4, 5};
18        System.out.println("\nOriginal Integer array: " + java.util.Arrays.toString(intArray));
19        swapElements(intArray, 1, 3);
20        System.out.println("After swapping: " + java.util.Arrays.toString(intArray));
21        Character[] charArray = {'a', 'b', 'c', 'd'};
22        System.out.println("\nOriginal Character array: " + java.util.Arrays.toString(charArray));
23        swapElements(charArray, 0, 3);
24        System.out.println("After swapping: " + java.util.Arrays.toString(charArray));
25    }
26 }
27 }
```

Console Output:

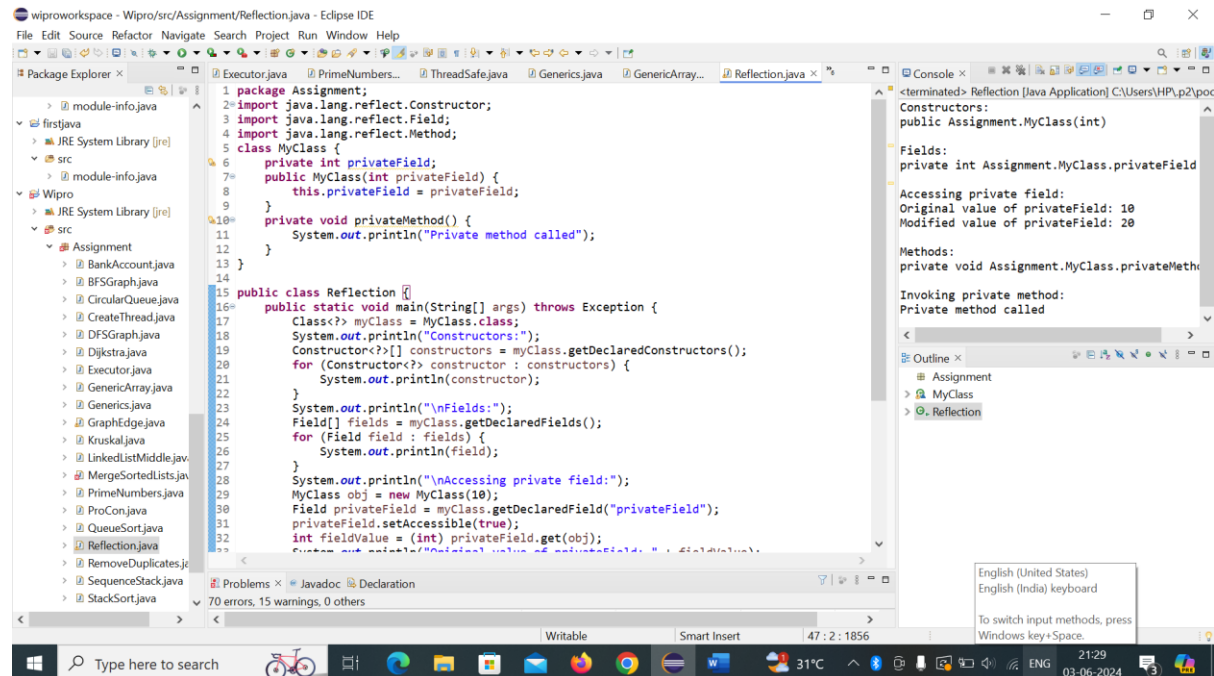
```
<terminated> GenericArray [Java Application] C:\Users\HP\AppData\Local\Temp\GenericArray\GenericArray.java
Original String array: [apple, banana, cherry]
After swapping: [cherry, banana, apple]

Original Integer array: [1, 2, 3, 4, 5]
After swapping: [1, 4, 3, 2, 5]

Original Character array: [a, b, c, d]
After swapping: [d, b, c, a]
```

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



```
1 package Assignment;
2 import java.lang.reflect.Constructor;
3 import java.lang.reflect.Field;
4 import java.lang.reflect.Method;
5
6 class MyClass {
7     private int privateField;
8     public MyClass(int privateField) {
9         this.privateField = privateField;
10    }
11    private void privateMethod() {
12        System.out.println("Private method called");
13    }
14 }
15
16 public class Reflection {
17     public static void main(String[] args) throws Exception {
18         Class<?> myClass = MyClass.class;
19         System.out.println("Constructors:");
20         Constructor<?>[] constructors = myClass.getDeclaredConstructors();
21         for (Constructor<?> constructor : constructors) {
22             System.out.println(constructor);
23         }
24         System.out.println("\nFields:");
25         Field[] fields = myClass.getDeclaredFields();
26         for (Field field : fields) {
27             System.out.println(field);
28         }
29         System.out.println("\nAccessing private field:");
30         MyClass obj = new MyClass(10);
31         Field privateField = myClass.getDeclaredField("privateField");
32         privateField.setAccessible(true);
33         int fieldValue = (int) privateField.get(obj);
34         System.out.println("Original value of privateField: " + fieldValue);
35     }
36 }
```

Console Output:

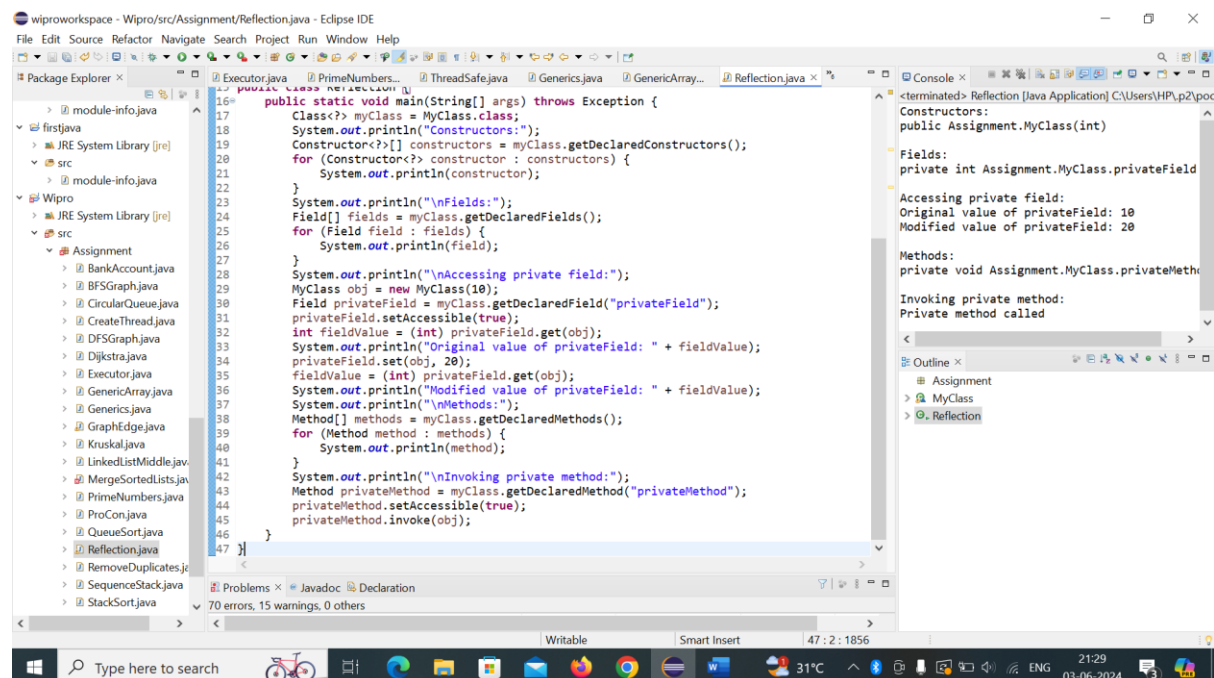
```
<terminated> Reflection [Java Application] C:\Users\HP\AppData\Local\Temp\
Constructors:
public Assignment.MyClass(int)

Fields:
private int Assignment.MyClass.privateField

Accessing private field:
Original value of privateField: 10
Modified value of privateField: 20

Methods:
private void Assignment.MyClass.privateMethod()

Invoking private method:
Private method called
```



```
16 public static void main(String[] args) throws Exception {
17     Class<?> myClass = MyClass.class;
18     System.out.println("Constructors:");
19     Constructor<?>[] constructors = myClass.getDeclaredConstructors();
20     for (Constructor<?> constructor : constructors) {
21         System.out.println(constructor);
22     }
23     System.out.println("\nFields:");
24     Field[] fields = myClass.getDeclaredFields();
25     for (Field field : fields) {
26         System.out.println(field);
27     }
28     System.out.println("\nAccessing private field:");
29     MyClass obj = new MyClass(10);
30     Field privateField = myClass.getDeclaredField("privateField");
31     privateField.setAccessible(true);
32     int fieldValue = (int) privateField.get(obj);
33     System.out.println("Original value of privateField: " + fieldValue);
34     privateField.set(obj, 20);
35     fieldValue = (int) privateField.get(obj);
36     System.out.println("Modified value of privateField: " + fieldValue);
37     System.out.println("\nMethods:");
38     Method[] methods = myClass.getDeclaredMethods();
39     for (Method method : methods) {
40         System.out.println(method);
41     }
42     System.out.println("\nInvoking private method:");
43     Method privateMethod = myClass.getDeclaredMethod("privateMethod");
44     privateMethod.setAccessible(true);
45     privateMethod.invoke(obj);
46 }
47 }
```

Console Output:

```
<terminated> Reflection [Java Application] C:\Users\HP\AppData\Local\Temp\
Constructors:
public Assignment.MyClass(int)

Fields:
private int Assignment.MyClass.privateField

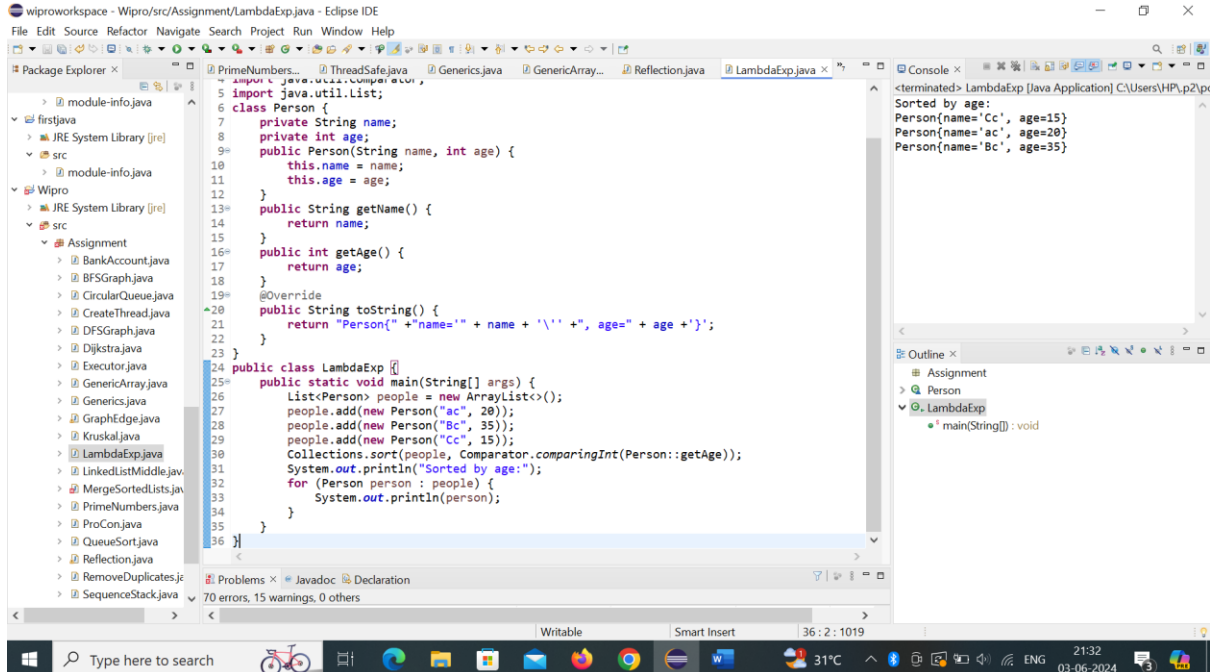
Accessing private field:
Original value of privateField: 10
Modified value of privateField: 20

Methods:
private void Assignment.MyClass.privateMethod()

Invoking private method:
Private method called
```

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.



The screenshot shows the Eclipse IDE with the following components:

- Package Explorer:** Shows the project structure with 'Wipro' as the main package, containing 'Assignment' and 'src' sub-packages. The 'src' package contains 'LambdaExp.java'.
- Editor:** Displays the code for 'LambdaExp.java'. The code defines a 'Person' class with attributes 'name' and 'age', and methods 'getName()', 'getAge()', and 'toString()'. It also defines a 'LambdaExp' class with a 'main' method that creates a list of 'Person' objects, sorts them using a lambda expression as a comparator, and prints the sorted list.
- Console:** Shows the output of the program, which is 'Sorted by age:' followed by the details of the sorted 'Person' objects: 'Person(name='Cc', age=15)', 'Person(name='ac', age=20)', and 'Person(name='Bc', age=35)'.
- Outline:** Shows the class hierarchy, including 'Assignment', 'Person', and 'LambdaExp'.
- Problems:** Shows 70 errors and 15 warnings, likely due to the IDE's configuration or the use of lambda expressions.

```
import java.util.*;
import java.util.Comparator;

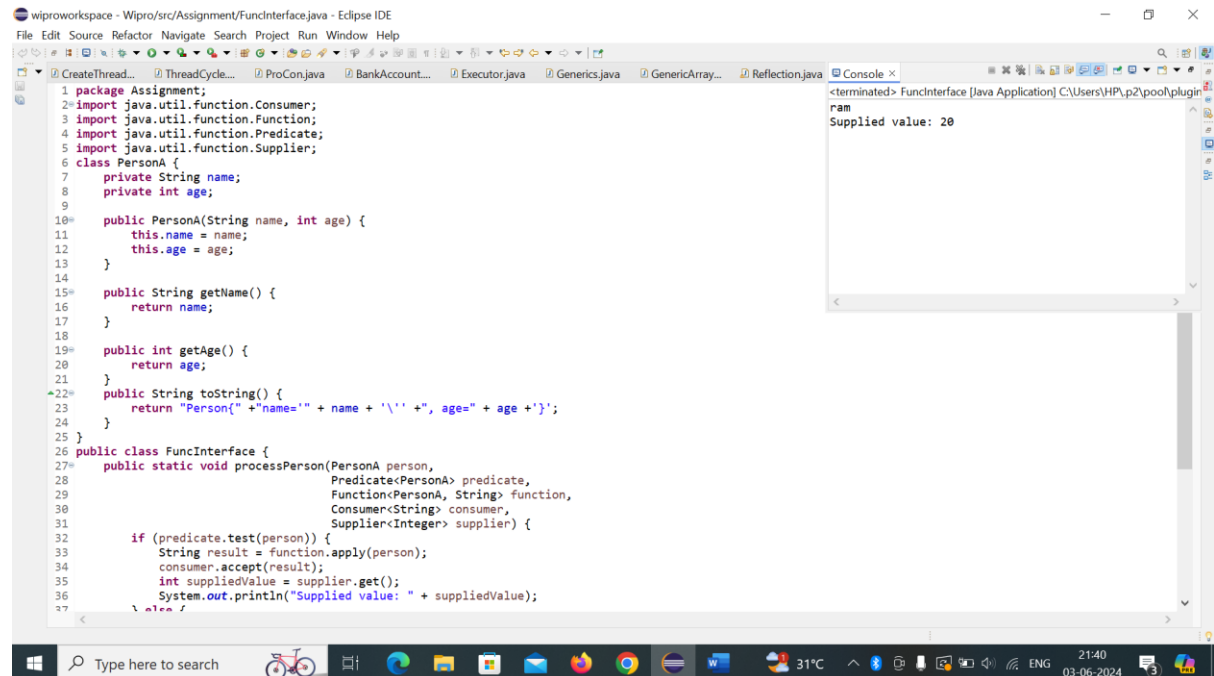
class Person {
    private String name;
    private int age;
    public Person(String name, int age) {
        this.name = name;
        this.age = age;
    }
    public String getName() {
        return name;
    }
    public int getAge() {
        return age;
    }
    @Override
    public String toString() {
        return "Person{" + "name=" + name + '\'' + ", age=" + age + '\'' + '}';
    }
}

public class LambdaExp {
    public static void main(String[] args) {
        List<Person> people = new ArrayList<>();
        people.add(new Person("ac", 20));
        people.add(new Person("Bc", 35));
        people.add(new Person("Cc", 15));
        Collections.sort(people, Comparator.comparingInt(Person::getAge));
        System.out.println("Sorted by age:");
        for (Person person : people) {
            System.out.println(person);
        }
    }
}
```

Sorted by age:
Person(name='Cc', age=15)
Person(name='ac', age=20)
Person(name='Bc', age=35)

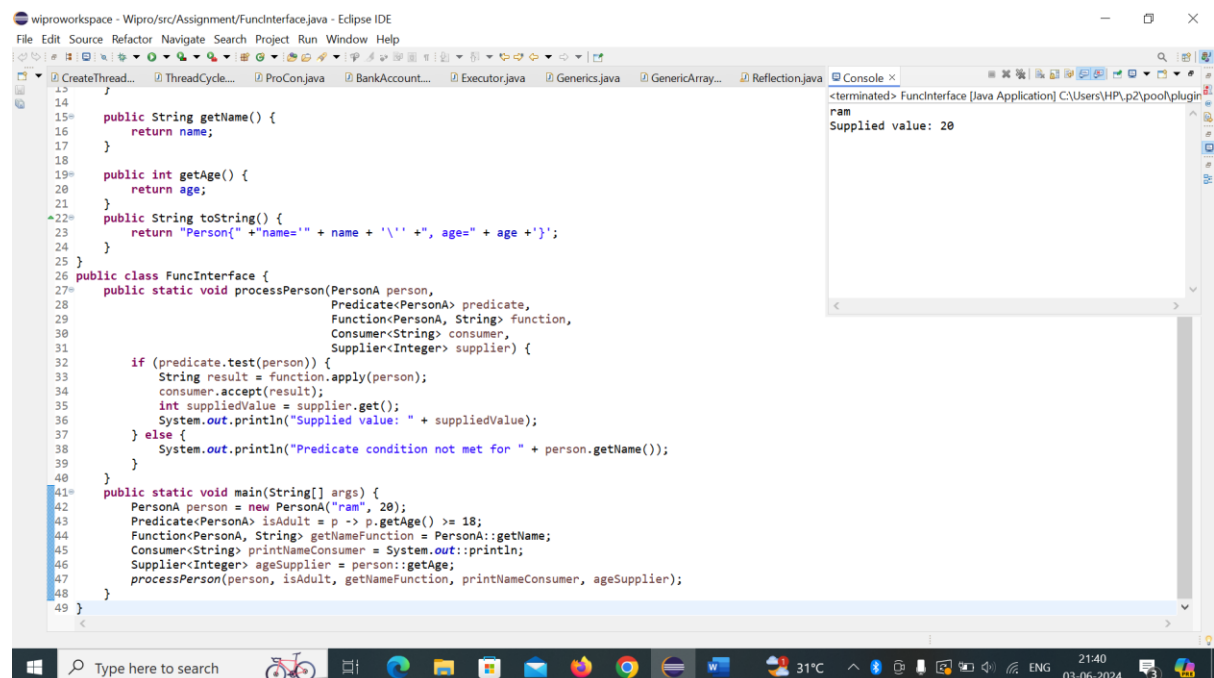
Task 5: Functional Interfaces

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



```
1 package Assignment;
2 import java.util.function.Consumer;
3 import java.util.function.Function;
4 import java.util.function.Predicate;
5 import java.util.function.Supplier;
6 class PersonA {
7     private String name;
8     private int age;
9
10    public PersonA(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    public String toString() {
23        return "Person{" + "name='" + name + '\'' + ", age=" + age + '}';
24    }
25 }
26 public class FuncInterface {
27     public static void processPerson(PersonA person,
28                                     Predicate<PersonA> predicate,
29                                     Function<PersonA, String> function,
30                                     Consumer<String> consumer,
31                                     Supplier<Integer> supplier) {
32         if (predicate.test(person)) {
33             String result = function.apply(person);
34             consumer.accept(result);
35             int suppliedValue = supplier.get();
36             System.out.println("Supplied value: " + suppliedValue);
37         }
38     }
39 }
```

Console output: ram
Supplied value: 20



```
14 }
15 public String getName() {
16     return name;
17 }
18
19 public int getAge() {
20     return age;
21 }
22 public String toString() {
23     return "Person{" + "name='" + name + '\'' + ", age=" + age + '}';
24 }
25 }
26 public class FuncInterface {
27     public static void processPerson(PersonA person,
28                                     Predicate<PersonA> predicate,
29                                     Function<PersonA, String> function,
30                                     Consumer<String> consumer,
31                                     Supplier<Integer> supplier) {
32         if (predicate.test(person)) {
33             String result = function.apply(person);
34             consumer.accept(result);
35             int suppliedValue = supplier.get();
36             System.out.println("Supplied value: " + suppliedValue);
37         } else {
38             System.out.println("Predicate condition not met for " + person.getName());
39         }
40     }
41     public static void main(String[] args) {
42         PersonA person = new PersonA("ram", 20);
43         Predicate<PersonA> isAdult = p -> p.getAge() >= 18;
44         Function<PersonA, String> getNameFunction = PersonA::getName;
45         Consumer<String> printNameConsumer = System.out::println;
46         Supplier<Integer> ageSupplier = person::getAge;
47         processPerson(person, isAdult, getNameFunction, printNameConsumer, ageSupplier);
48     }
49 }
```

Console output: ram
Supplied value: 20