

# Assignment Day-19

## Core Java with DS and Algorithms

Name: Joshnitha Rangolu

### 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 day19;

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> reversedPair() {

        return new Pair<>(second, first);

    }

    public static void main(String[] args) {

        Pair<Integer, String> myPair = new Pair<>(10, "hello");

        Pair<String, Integer> reversedPair = myPair.reversedPair();
```

```

System.out.println("Original pair: (" + myPair.getFirst() + ", " +
myPair.getSecond() + ")");

System.out.println("Reversed pair: (" + reversedPair.getFirst() + ", " +
reversedPair.getSecond() + ")");

}

}

```

The screenshot shows an IDE with several tabs open: StringCompa..., PQDemo.java, QDemo.java, SetDemo.java, MergeLinked..., ThreadSafeCo..., and Pair.java. The Pair.java file is selected and shows the following code:

```

1 package day19;
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
16    public U getSecond() {
17        return second;
18    }
19
20    public Pair<U, T> reversedPair() {
21
22        return new Pair<>(second, first);
23    }
24
25    public static void main(String[] args) {

```

Below the code editor, the Console window shows the output of the program:

```

<terminated> Pair [Java Application] C:\Users\DELL\p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_17.0.6.v20230204-17
Original pair: (10, hello)
Reversed pair: (hello, 10)

```

## 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 day19;

public class ArraySwapper {

    public static <T> void swap(T[] arr, int index1, int index2) {

        if (index1 < 0 || index1 >= arr.length || index2 < 0 || index2 >=
arr.length) {

            throw new IllegalArgumentException("Invalid index provided");

        }

        T temp = arr[index1];

```

```

arr[index1] = arr[index2];

arr[index2] = temp;

}

public static void main(String[] args) {

Integer[] intArr = {1, 5, 3, 7};

System.out.println("Original Integer array: " +
java.util.Arrays.toString(intArr));

swap(intArr, 1, 3);

System.out.println("Swapped Integer array: " +
java.util.Arrays.toString(intArr));

String[] stringArr = {"apple", "banana", "cherry"};

System.out.println("Original String array: " +
java.util.Arrays.toString(stringArr));

swap(stringArr, 0, 2);

System.out.println("Swapped String array: " +
java.util.Arrays.toString(stringArr));

}

}

```

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

```

1 package day19;
2
3 public class ArraySwapper {
4
5     public static <T> void swap(T[] arr, int index1, int index2) {
6         if (index1 < 0 || index1 >= arr.length || index2 < 0 || index2 >= arr.length) {
7             throw new IllegalArgumentException("Invalid index provided");
8         }
9         T temp = arr[index1];
10        arr[index1] = arr[index2];
11        arr[index2] = temp;
12    }
13
14    public static void main(String[] args) {
15        Integer[] intArr = {1, 5, 3, 7};
16        System.out.println("Original Integer array: " + java.util.Arrays.toString(intArr));
17        swap(intArr, 1, 3);
18        System.out.println("Swapped Integer array: " + java.util.Arrays.toString(intArr));
19
20        String[] stringArr = {"apple", "banana", "cherry"};
21        System.out.println("Original String array: " + java.util.Arrays.toString(stringArr));
22        swap(stringArr, 0, 2);

```

The console output at the bottom shows the execution results:

```

<terminated> ArraySwapper [Java Application] C:\Users\DELL\p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_17.0.6.v20230204-1
Original Integer array: [1, 5, 3, 7]
Swapped Integer array: [1, 7, 3, 5]
Original String array: [apple, banana, cherry]
Swapped String array: [cherry, banana, apple]

```

### 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.

```
private String name;

private int age;


public Person() {
}

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

public String getName() {
    return name;
}

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

public int getAge() {
    return age;
}

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

private void printPrivateMessage() {
    System.out.println("This is a private method.");
}
```

```
}
```

ReflectionExample Class:

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

public class ReflectionExample {
    public static void main(String[] args) {
        try {
            Class<?>personClass = Class.forName("Person");
            System.out.println("Constructors:");
            Constructor<?>[] constructors = personClass.getConstructors();
            for (Constructor<?>constructor : constructors) {
                System.out.println(constructor);
            }
            System.out.println("\nMethods:");
            Method[] methods = personClass.getMethods();
            for (Method method : methods) {
                System.out.println(method);
            }
            System.out.println("\nFields:");
            Field[] fields = personClass.getDeclaredFields();
            for (Field field : fields) {
                System.out.println(field);
            }
            Object personInstance = personClass.getConstructor().newInstance();
            Field nameField = personClass.getDeclaredField("name");
            nameField.setAccessible(true);
            nameField.set(personInstance, "John Doe");
            Method getNameMethod = personClass.getMethod("getName");
            String name = (String) getNameMethod.invoke(personInstance);
            System.out.println("\nModified name field: " + name);
        }
    }
}
```

```

        Field ageField = personClass.getDeclaredField("age");
        ageField.setAccessible(true);
        ageField.setInt(personInstance, 30);

        Method getAgeMethod = personClass.getMethod("getAge");
        int age = (int) getAgeMethod.invoke(personInstance);

        System.out.println("Modified age field: " + age);

        Method privateMethod = personClass.getDeclaredMethod("printPrivateMessage");
        privateMethod.setAccessible(true);
        privateMethod.invoke(personInstance);
    } catch (Exception e) {
        e.printStackTrace();
    }
}
}

```

Output:

Constructors:

```

public Person()
public Person(java.lang.String,int)

```

Methods:

```

public java.lang.String Person.getName()
public void Person.setName(java.lang.String)
public int Person.getAge()
public void Person.setAge(int)
public final void java.lang.Object.wait(long,int) throws java.lang.InterruptedException
public final void java.lang.Object.wait() throws java.lang.InterruptedException
public final native void java.lang.Object.wait(long) throws java.lang.InterruptedException
public boolean java.lang.Object.equals(java.lang.Object)
public java.lang.String java.lang.Object.toString()
public native int java.lang.Object.hashCode()
public final native java.lang.Class<?> java.lang.Object.getClass()

```

```
public final native void java.lang.Object.notify()
public final native void java.lang.Object.notifyAll()
```

Fields:

```
private java.lang.String Person.name
private int Person.age
```

Modified name field: John Doe

Modified age field: 30

This is a private method.

#### 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 day19;

import java.util.ArrayList;
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("Alice", 30));

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

people.add(new Person("Charlie", 38));

System.out.println("Original list: " + people);

// Sort by age using lambda expression

people.sort((p1, p2) -> p1.getAge() - p2.getAge());

System.out.println("Sorted by age: " + people);

}

}

```

```

7      private String name;
8      private int age;
9
10     public Person(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     @Override
24     public String toString() {
25         return "Person [name=" + name + ", age=" + age + "]";
26     }
27
28     public static void main(String[] args) {

```

Console ×

```

<terminated> Person [Java Application] C:\Users\DELL\p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_17.0.6.v20230204-1729\jre\bin\javaw.exe
Original list: [Person [name=Alice, age=30], Person [name=Bob, age=25], Person [name=Charlie, age=38]]
Sorted by age: [Person [name=Bob, age=25], Person [name=Alice, age=30], Person [name=Charlie, age=38]]

```



## 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 day19;

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

public class Person1 {

    private String name;

    private int age;

    public Person1(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 FunctionalPersonOperations {
```

```

static Predicate<Person> isAdult = person -> person.getAge() >= 18;

static Function<Person, String> getFullName = person ->
(isAdult.test(person) ? "Mr./Ms. " : "") + person.getName();

static Consumer<Person> printDetails = person ->
System.out.println("Name: " + person.getName() + ", Age: " +
person.getAge());

static Supplier<Person> createPerson = () -> new Person("John Doe", 30);

public static void main(String[] args) {

    // Create a Person object
    Person person = createPerson.get();

    // Check if adult
    if (isAdult.test(person)) {

        System.out.println("Adult");

    } else {

        System.out.println("Not an adult");

    }

    // Get full name
    String fullName = getFullName.apply(person);

    System.out.println("Full Name: " + fullName);

    // Print details
    printDetails.accept(person);

}

}

```

```
StringCompa... PQDemo.java ThreadSafeCo... Pair.java ArraySwappe... ReflectionT... Person.java *Person1.java × 40
7
8 public class Person1 {
9
10     private String name;
11     private int age;
12
13     public Person1(String name, int age) {
14         this.name = name;
15         this.age = age;
16     }
17
18     public String getName() {
19         return name;
20     }
21
22     public int getAge() {
23         return age;
24     }
25
26     @Override
27     public String toString() {
28         return "Person [name=" + name + ", age=" + age + "]";
29     }
30 }
```

Console ×

<terminated> Person1 [Java Application] C:\Users\DELL\p2\pool\plugins\org.eclipse.justi.openjdk.hotspot.jre.full.win32.x86\_64\_17.0.6.v20230204-1729\jre\bin\javaw.exe (0  
Adult  
Full Name: Mr./Ms. John Doe  
Name: John Doe, Age: 30