Day - 19

1] 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.

Solution:-

Code:-

GenericNTypeSafety.java

```java
package com.assignments;

public class GenericNTypeSafety <T,U>{

        private T first;
        private U second;

        public GenericNTypeSafety (T first, U second) {
            this.first = first;
            this.second = second;
        }

        public T getFirst() {
            return first;
        }

        public void setFirst(T first) {
            this.first = first;
        }

        public U getSecond() {
            return second;
        }

        public void setSecond(U second) {
            this.second = second;
        }

        public GenericNTypeSafety<U, T> reverse() {
            return new GenericNTypeSafety<>(second, first);
```

GenericNTypeSafety.java

```java
        public U getSecond() {
            return second;
        }

        public void setSecond(U second) {
            this.second = second;
        }

        public GenericNTypeSafety<U, T> reverse() {
            return new GenericNTypeSafety<>(second, first);
        }

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

        // Example usage
        public static void main(String[] args) {
        GenericNTypeSafety<Integer, String> pair = new GenericNTypeSafety<>(10, "Hello");
            System.out.println("Original Pair: " + pair);

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

Output :-

Console

```
<terminated> GenericNTypeSafety [Java Application] C:\Users\Skynet\.p2\pool\plugins\org.eclipse
Original Pair: (10, Hello)
Reversed Pair: (Hello, 10)
```

2] 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.

Solution :-

Code :-

GenericClassnMethods.java

```java
package com.assignments;

import java.util.Arrays;

public class GenericClassnMethods {



public static <T> void swap(T[] array, int index1, int index2) {
     if (index1 < 0 || index1 >= array.length || index2 < 0 || index2 >= array.length) {
                throw new IllegalArgumentException("Invalid indices provided");
            }

            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("Original Integer array: " + Arrays.toString(intArray));
            swap(intArray, 1, 3);
            System.out.println("After swapping: " + Arrays.toString(intArray));


            String[] strArray = {"apple", "banana", "cherry"};
            System.out.println("Original String array: " + Arrays.toString(strArray));
            swap(strArray, 0, 2);
            System.out.println("After swapping: " + Arrays.toString(strArray));
```

GenericClassnMethods.java

```java


            String[] strArray = {"apple", "banana", "cherry"};
            System.out.println("Original String array: " + Arrays.toString(strArray));
            swap(strArray, 0, 2);
            System.out.println("After swapping: " + Arrays.toString(strArray));


            Double[] doubleArray = {1.5, 2.5, 3.5};
            System.out.println("Original Double array: " + Arrays.toString(doubleArray));
            swap(doubleArray, 0, 1);
            System.out.println("After swapping: " + Arrays.toString(doubleArray));


            Person[] personArray = {
                    new Person("Shweta", 25),
                    new Person("Purva", 30),
                    new Person("Suyog", 28)
            };
            System.out.println("Original Person array: " + Arrays.toString(personArray));
            swap(personArray, 0, 2);
            System.out.println("After swapping: " + Arrays.toString(personArray));
        }
    }


    class Person {
        private String name;
        private int age;
```

GenericClassnMethods.java

```java
                    new Person("Shweta", 25),
                    new Person("Purva", 30),
                    new Person("Suyog", 28)
            };
            System.out.println("Original Person array: " + Arrays.toString(personArray));
            swap(personArray, 0, 2);
            System.out.println("After swapping: " + Arrays.toString(personArray));
        }
    }


    class Person {
        private String name;
        private int age;

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

        @Override
        public String toString() {
            return "Person{" +
                    "name='" + name + '\'' +
                    ", age=" + age +
                    '}';
        }
    }
```

Output :-

```
Console
<terminated> GenericClassnMethods [Java Application] C:\Users\Skynet\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_22.0.1.v20240426-1149\jre\bin\javaw.exe (Jun 22, 2024, 12:03
Original Integer array: [1, 2, 3, 4, 5]
After swapping: [1, 4, 3, 2, 5]
Original String array: [apple, banana, cherry]
After swapping: [cherry, banana, apple]
Original Double array: [1.5, 2.5, 3.5]
After swapping: [2.5, 1.5, 3.5]
Original Person array: [Person{name='Shweta', age=25}, Person{name='Purva', age=30}, Person{name='Suyog', age=28}]
After swapping: [Person{name='Suyog', age=28}, Person{name='Purva', age=30}, Person{name='Shweta', age=25}]
```

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

Solution :-

Code:-

```java
ReflectionAPI.java
package com.assignments;

import java.lang.reflect.*;
public class ReflectionAPI {

public static void main(String[] args) throws NoSuchFieldException,
          IllegalAccessException, NoSuchMethodException,
               InvocationTargetException, InstantiationException {

            Class<Person1> personClass = Person1.class;


            System.out.println("Methods of Person class:");
            Method[] methods = personClass.getDeclaredMethods();
            for (Method method : methods) {
                System.out.println(method.getName());
            }


            System.out.println("\nFields of Person class:");
            Field[] fields = personClass.getDeclaredFields();
            for (Field field : fields) {
                System.out.println(field.getName() + " (Type: "
            + field.getType().getSimpleName() + ")");
            }


            System.out.println("\nConstructors of Person class:");
            Constructor<?>[] constructors = personClass.getDeclaredConstructors();
            for (Constructor<?> constructor : constructors) {
```

ReflectionAPI.java

```java
            System.out.println("\nConstructors of Person class:");
            Constructor<?>[] constructors = personClass.getDeclaredConstructors();
            for (Constructor<?> constructor : constructors) {
                System.out.println(constructor.toString());
            }


            Field ageField = personClass.getDeclaredField("age");
            ageField.setAccessible(true);


            Constructor<Person1> constructor =
                    personClass.getDeclaredConstructor(String.class, int.class);
            constructor.setAccessible(true);
            Person1 person = constructor.newInstance("Shweta", 24);


            ageField.setInt(person, 31);


            System.out.println("\nModified Age: " + ageField.getInt(person));
        }
    }

    class Person1 {
        private String name;
        private int age;

        private Person1(String name, int age) {
            this.name = name;
```

```java
ReflectionAPI.java
            Person1 person = constructor.newInstance("Shweta", 24);


            ageField.setInt(person, 31);


            System.out.println("\nModified Age: " + ageField.getInt(person));
        }
    }

    class Person1 {
        private String name;
        private int age;

        private Person1(String name, int age) {
            this.name = name;
            this.age = age;
        }


        public String getName() {
            return name;
        }

        public int getAge() {
            return age;
        }
    }

```

Output :-

```
Console
<terminated> ReflectionAPI [Java Application] C:\Users\Skynet\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_22.0.1.v2024
Methods of Person class:
getName
getAge

Fields of Person class:
name (Type: String)
age (Type: int)

Constructors of Person class:
private com.assignments.Person1(java.lang.String,int)

Modified Age: 31
```

4] 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.

Solution :-

Code :-

LambdaExpression.java

```java
package com.assignments;

import java.util.ArrayList;
import java.util.Collections;
import java.util.Comparator;
import java.util.List;


public class LambdaExpression {

        public static void main(String[] args) {

            List<Person2> persons = new ArrayList<>();
            persons.add(new Person2("Shweta", 25));
            persons.add(new Person2("Purva", 27));
            persons.add(new Person2("Saee", 20));


            System.out.println("Before sorting:");
            printPersons(persons);


            Collections.sort(persons, Comparator.comparingInt(Person2::getAge));


            System.out.println("\nAfter sorting by age:");
            printPersons(persons);
        }


```

LambdaExpression.java

```java
        }


        private static void printPersons(List<Person2> persons) {
            for (Person2 person : persons) {
                System.out.println(person);
            }
        }
    }
    class Person2 {
        private String name;
        private int age;

        public Person2(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 + '\'' +
```

LambdaExpression.java

```java
        }
    }
    class Person2 {
        private String name;
        private int age;

        public Person2(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 +
                    '}';
        }
    }


```

Output :-

Console

<terminated> LambdaExpression [Java Application] C:\Users\Skynet\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_22.0.

```
Before sorting:
Person{name='Shweta', age=25}
Person{name='Purva', age=27}
Person{name='Saee', age=20}

After sorting by age:
Person{name='Saee', age=20}
Person{name='Shweta', age=25}
Person{name='Purva', age=27}
```

5] 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.

Solution :-

Code :-

FunctionalInteface.java

```java
package com.assignments;

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

public class FunctionalInteface {


        public static void main(String[] args) {

            Person5 person = new Person5("Saee", 30);


            processPerson(
                    person,
                    p -> p.getAge() > 25,
                    p -> "Name: " + p.getName(),
                    System.out::println,
                    () -> new Person5("NewPerson", 25)
            );
        }


        public static void processPerson(Person5 person,
                                         Predicate<Person5> predicate,
                                         Function<Person5, String> function,
                                         Consumer<String> consumer,
                                         Supplier<Person5> supplier) {
```

FunctionalInteface.java

```java
                                        Predicate<Person5> predicate,
                                        Function<Person5, String> function,
                                        Consumer<String> consumer,
                                        Supplier<Person5> supplier) {

            if (predicate.test(person)) {

                String result = function.apply(person);

                consumer.accept(result);
            } else {

                Person5 newPerson = supplier.get();
                consumer.accept("New Person created: " + newPerson.getName());
            }
        }
    }

    
    class Person5 {
        private String name;
        private int age;

        public Person5(String name, int age) {
            this.name = name;
            this.age = age;
        }

        public String getName() {
            return name;
```

FunctionalInteface.java

```java
                consumer.accept(result);
            } else {

                Person5 newPerson = supplier.get();
                consumer.accept("New Person created: " + newPerson.getName());
            }
        }
    }


    class Person5 {
        private String name;
        private int age;

        public Person5(String name, int age) {
            this.name = name;
            this.age = age;
        }

        public String getName() {
            return name;
        }

        public int getAge() {
            return age;
        }
    }


```

Output :-

```
Console
<terminated> FunctionalInteface [Java Application] C:\Users\Skynet\.p2\pool\plugins\org.eclipse.justj.openjdk.
Name:  Saee
```