

## Day 12 Assignment

**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.**

```
/*
    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.*/

class Pair<T, G> {
    private T obj1;
    private G obj2;

    Pair(T obj1, G obj2) {
        this.obj1 = obj1;
        this.obj2 = obj2;
    }

    @Override
    public String toString() {
        return "(" +
            obj1.getClass().getName() +
            " : " +
            obj1.toString() +
            " , " +
            obj2.getClass().getName() +
            " : " +
            obj2.toString() +
            ")";
    }

    public Pair<G, T> reversedPair() {
        return new Pair(obj2, obj1);
    }
}

public class Assignment_1 {public static
    void main(String[] args) {
        System.out.println("Creating a type safe class called Pair");
        Pair<String, Integer> pair = new Pair<String, Integer>("Hello", 1234);
```

```
System.out.println(pair);
```

```
        System.out.println("Reversed Pair");
        Pair<Integer, String> revPair = pair.reversedPair();
        System.out.println(revPair);

    }
}

package m5_core_java_programming.day_12;

/*
    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.
*/

import java.util.Scanner;

class Arr {
    private Object[] arr;
    private final int size;
    private int curr;

    Arr(int size) {
        this.arr = new Object[size];
        this.size = size; this.curr =
        0;
    }

    public int size() { return
        this.size;
    }

    public void add(Object obj) {
        if (this.curr < this.size) {
```

**Output**

```
C:\Users\coolr\.jdk\openjdk-22.0.1\bin\java.exe "-javaagent:C:\Program Files\JetBrains
Creating a type safe class called Pair
(java.lang.String : Hello , java.lang.Integer : 1234)
Reversed Pair
(java.lang.Integer : 1234 , java.lang.String : Hello)

Process finished with exit code 0
```

- 
2. 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.

```

        this.arr[this.curr] = obj;
        this.curr++;
    } else {
        System.out.println("Array is filled");
    }
}

public void replace(int a, int b) { if (a < 0 || a >= this.size || b <
    0 || b >= this.size || a == b ||
this.curr < this.size) {
    System.out.println("Please fill the array first");
} else {
    Object temp = this.arr[a];
    this.arr[a] = this.arr[b];
    this.arr[b] = temp;
}
}

@Override
public String toString() { if
    (this.curr < this.size) {
        System.out.println("Please fill the array first");
        return "Please fill the array first.";
    } else {
        String string = "( ";
        for (Object x : this.arr) {
            string += " " + x.toString();
        }
        string += " )";
        return string;
    }
}

}

public class Assignment_2 { public static
    void main(String[] args) {
        Scanner scan = new Scanner(System.in);
        System.out.println("Enter the size of the array : "); int
        n = scan.nextInt();
        System.out.println("Enter the Items");
        Arr arr = new Arr(n);

```

```
        for (int i = 0; i < n; i++) {
            System.out.println("Choose from any wrapper class type :");
System.out.println("1. Integer");
System.out.println("2. Long");
System.out.println("3. Byte");
System.out.println("4. Short");
System.out.println("5. Double");
System.out.println("6. Float");
```

```
System.out.println("7. String");
int option = scan.nextInt();
switch (option) { case 1:
    {
        System.out.println("Enter item : ");
        arr.add(scan.nextInt());
        break;
    }
    case 2: {
        System.out.println("Enter item : ");
        arr.add(scan.nextLong()); break;
    }

    case 3: {
        System.out.println("Enter item : "); arr.add(scan.nextByte());
        break;
    }

    case 4: {
        System.out.println("Enter item : "); arr.add(scan.nextShort());

        break;
    }
    case 5: {
        System.out.println("Enter item : ");
        arr.add(scan.nextDouble()); break;
    }

    case 6: {
        System.out.println("Enter item : "); arr.add(scan.nextFloat());
        break;
    }

}
```

```

        case 7: {
            System.out.println("Enter item : "); arr.add(scan.next());
            break;

        }

        default: {
            System.out.println("Wrong choice"); i--;
            ;

            break;
        }
    }
}

System.out.println(arr);

System.out.println("Enter indexes to swap item : ");
    System.out.println("Enter first index : ");
    int a = scan.nextInt();
    System.out.println("Enter second index : "); int
    b = scan.nextInt();

    arr.replace(a, b);
    System.out.println("After swapping :");
    System.out.println(arr);
}
}

```

**Output**

Enter the size of the array :

4

Enter the Items

Choose from any wrapper class type :

1. Integer
2. Long
3. Byte
4. Short
5. Double
6. Float
7. String

1

Enter item :

656

Choose from any wrapper class type :

1. Integer
2. Long
3. Byte
4. Short
5. Double
6. Float
7. String

7

Enter item :

Sayan

Choose from any wrapper class type :

1. Integer

```
2. Long
3. Byte
4. Short
5. Double
6. Float
7. String
5
Enter item :
232.5
Choose from any wrapper class type :
1. Integer
2. Long
3. Byte
4. Short
5. Double
6. Float
7. String
4
Enter item :
12
( 656 Sayan 232.5 12 )
Enter indexes to swap item :
Enter first index :
1
Enter second index :
3
After swapping :
( 656 12 232.5 Sayan )
```

---

### 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 m5_core_java_programming.day_12;

/*
    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
*/

import java.lang.reflect.*;

class RefExample {
    private int num1;
```



```
protected int num2;  
public String name;
```

```
private RefExample() {  
    this.num1 = 0;  
    this.num2 = 0;  
    this.name = "Default Name.";  
}  
  
public RefExample(int num1, int num2) {  
    this.num1 = num1;  
    this.num2 = num2;  
    this.name = "Default Name.";  
}  
  
public RefExample(int num1, int num2, String name) {  
    this.num1 = num1;  
    this.num2 = num2;  
    this.name = name;  
}  
  
public void publicMethod() {  
    System.out.println("I am public.");  
}  
  
private void privateMethod(String str) {  
    System.out.println("I am private.");  
    System.out.println("But they have invoked me with this value " + str);  
}  
  
protected void protectedMethod() {  
    System.out.println("I am protected.");  
}  
  
public int getNum1() {
```

```

        return num1;
    }

    public void setNum1(int num1) {
        this.num1 = num1;
    }

    public int getNum2() {
        return num2;
    }

    public void setNum2(int num2) {
        this.num2 = num2;
    }

    public String getName() {

```

```

        return name;
    }

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

public class Assignment_3 {
    public static void main(String[] args) {
        try {
            Class c =
Class.forName("m5_core_java_programming.day_12.RefExample");

            System.out.println("All constructors of : " + c.getTypeName());
            Constructor[] constructor = c.getDeclaredConstructors();
            for (Constructor cons : constructor) {

System.out.println("-----"
_"
+ "\n" +
                                "Modifier : " + Modifier.toString(cons.getModifiers()) +
"\n" +

```

```

        "Constructor Name : " + cons.getName() + "\n" +
        "Parameter count : " + cons.getParameterCount()
    );
}

System.out.println(".....
..
...");

```

```

    System.out.println("All the methods of : " +
c.getTypeName()); Method[] method = c.getDeclaredMethods();
    for (Method meth : method) {

System.out.println("-----
"
+ "\n" +
        "Modifier : " + Modifier.toString(meth.getModifiers()) +
"\n" +
        "Method Name : " + meth.getName() + "\n" +
        "Parameter count : " + meth.getParameterCount() + "\n" +
        "Return type : " + meth.getReturnType()
    );
}

System.out.println(".....
..
...");

    System.out.println("All the Fields of : " + c.getTypeName());
    Field[] fields = c.getDeclaredFields();
    for (Field field : fields) {

System.out.println("-----
_"
+ "\n" +
        "Modifier : " + Modifier.toString(field.getModifiers())
+ "\n" +
        "Field Name : " + field.getName() + "\n" +
        "Field type : " + field.toGenericString()
    );
}

```

```

System.out.println(".....
..
...");
        System.out.println("Setting private privateMethod method to
public");
        Method privToPublic = c.getDeclaredMethod("privateMethod",
String.class);
        privToPublic.setAccessible(true);
        RefExample ref = new RefExample(1, 10);
        privToPublic.invoke(ref, "Set to Public");

    } catch (ClassNotFoundException e) {
        throw new RuntimeException(e);
    } catch (NoSuchMethodException e) { throw
        new RuntimeException(e);
    } catch (InvocationTargetException e) {
        throw new RuntimeException(e);
    } catch (IllegalAccessException e) { throw
        new RuntimeException(e);
    }
}
}

```

**Output**

```
C:\Users\c00tr\jaks\openjdk-22.0.1\bin\java.exe -javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2
All constructors of : m5_core_java_programming.day_12.RefExample
```

```
-----
Modifier : public
Constructor Name : m5_core_java_programming.day_12.RefExample
Parameter count : 3
-----
```

```
Modifier : public
Constructor Name : m5_core_java_programming.day_12.RefExample
Parameter count : 2
-----
```

```
Modifier : private
Constructor Name : m5_core_java_programming.day_12.RefExample
Parameter count : 0
-----
```

```
.....
All the methods of :m5_core_java_programming.day_12.RefExample
```

```
-----
Modifier : public
Method Name : getName
Parameter count : 0
Return type : class java.lang.String
-----
```

```
Modifier : public
Method Name : setName
Parameter count : 1
Return type : void
-----
```

```
Modifier : public
Method Name : publicMethod
Parameter count : 0
Return type : void
-----
```

```
Modifier : private
Method Name : privateMethod
Parameter count : 1
Return type : void
-----
```

```
Modifier : protected
Method Name : protectedMethod
Parameter count : 0
Return type : void
-----
```

```
Modifier : public
Method Name : getNum2
Parameter count : 0
Return type : int
-----
```

```
Modifier : public
Method Name : getNum1
Parameter count : 0
Return type : int
-----
```

```

Modifier : public
Method Name : setNum1
Parameter count : 1
Return type : void
-----
Modifier : public
Method Name : setNum2
Parameter count : 1
Return type : void
-----
All the Fields of :m5_core_java_programming.day_12.RefExample
-----
Modifier : private
Field Name : num1
Field type : private int m5_core_java_programming.day_12.RefExample.num1
-----
Modifier : protected
Field Name : num2
Field type : protected int m5_core_java_programming.day_12.RefExample.num2
-----
Modifier : public
Field Name : name
Field type : public java.lang.String m5_core_java_programming.day_12.RefExample.name
-----
Setting private privateMethod method to public
I am private.
But they have invoked me with this value Set to Public

```

---

#### 4. Implement a Comparator for a Person class using a lambda expression, and sort a list of Person objects by their age.

```

package m5_core_java_programming.day_12;

/*
    Implement a Comparator for a Person class using a lambda expression,
    and sort a list of Person objects by their age.
*/

import java.util.LinkedList;
import java.util.Scanner;

class Person {
    private String name;
    private int age;

    public Person(String name, int age) {

```

```
        this.name = name;
        this.age = age;
    }
```

```
public int getAge() {
    return age;
}
```

```
}
```

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

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

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

```
@Override
```

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

```
}
```

```
public class Assignment_4 {
    public static void main(String[] args) {
        Scanner scan = new Scanner(System.in);
        LinkedList<Person> ls = new LinkedList<Person>();
        System.out.println("Number of people to add : ");
        int n = scan.nextInt();
        for (int i = 0; i < n; i++) {
            System.out.println("Add name :");
            String name = scan.next();
        }
    }
}
```

```

        System.out.println("Add age :");
        int age = scan.nextInt();
        Person person = new Person(name, age); ls.add(person);

    }

    System.out.println("Before Sorting :");
    System.out.println(ls);
    System.out.println("After sorting :");
    ls.sort((a, b) -> a.getAge() - b.getAge());
    System.out.println(ls);

}
}

```

## Output

```

Number of people to add :
4
Add name :
Sayan
Add age :
25
Add name :
Ashlesha
Add age :
22
Add name :
Sumit
Add age :
24
Add name :
Prantika
Add age :
27
Before Sorting :
[Person{name='Sayan', age=25}, Person{name='Ashlesha', age=22}, Person{name='Sumit', age=24}, Person{name='Prantika', age=27}]
After sorting :
[Person{name='Ashlesha', age=22}, Person{name='Sumit', age=24}, Person{name='Sayan', age=25}, Person{name='Prantika', age=27}]

Process finished with exit code 0

```

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

```

package m5_core_java_programming.day_12;

```



```

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

import java.util.Scanner; import
java.util.function.Consumer;

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

class SpecialFunction { public static boolean predicate(Predicate<Person>
    p, Person person) { return p.test(person);
    }

    public static Person supplier(Supplier<Person> s, String name, int age) {
        return s.get();
    }

    public static int function(Function<Person, Integer> f, Person person) {
        return f.apply(person);
    }

```

```

    public static void consumer(Consumer<Person> c, Person person) {
        c.accept(person);
    }
}

```

```

public class Assignment_5 {

    public static void main(String[] args) {
        Scanner scan = new Scanner(System.in);
        System.out.println("Creating a person object :");
        System.out.println("Enter name :");
        String name = scan.next();
        System.out.println("Enter age :");
        int age = scan.nextInt();
        Person person = new Person(name, age);
    }
}

```

```

System.out.println("
");
    System.out.println("Calling predicate functional interface :");
    Predicate<Person> p = (Person per) -> per.getAge() > 18;
    System.out.println("This Predicate if the person is adult " +
SpecialFunction.predicate(p, person));

System.out.println("
");
    System.out.println("Calling supplier functional interface
:"); System.out.println("Enter name :"); name = scan.next();

    System.out.println("Enter age :");
    age = scan.nextInt();
    String finalName = name;
    int finalAge = age;

    Supplier<Person> s = () -> new Person(finalName, finalAge);
    person = SpecialFunction.supplier(s, finalName, finalAge);
    System.out.println("This Supplier will create a new Person object " +
    System.out.println("Enter new age :");
    age = scan.nextInt();
    int finalAge1 = age;
    Consumer<Person> c = (Person per) -> per.setAge(finalAge1);
    SpecialFunction.consumer(c, person);
    System.out.println("This Consumer will set new age in the person
object
" + person);
    }
}
person);

System.out.println("
");
    System.out.println("Calling function functional interface :");
    Function<Person, Integer> f = (Person per) -> per.getAge();
    System.out.println("This Function will return the age of the person " +
SpecialFunction.function(f, person));

System.out.println("
");
    System.out.println("Calling consumer functional interface :");
    System.out.println("Current Age of person " + person.getAge());

```

## Output

```
C:\Users\coolr\.jdk\openjdk-22.0.1\bin\java.exe "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2024.1\lib\idea
Creating a person object :
Enter name :
Sayan
Enter age :
25
-----
Calling predicate functional interface :
This Predicate if the person is adult true
-----
Calling supplier functional interface :
Enter name :
Ashlesha
Enter age :
22
This Supplier will create a new Person object Person{name='Ashlesha', age=22}
-----
Calling function functional interface :
This Function will return the age of the person 22
-----
Calling consumer functional interface :
Current Age of person 22
Enter new age :
23
This Consumer will set new age in the person object Person{name='Ashlesha', age=23}

Process finished with exit code 0
```

---