OOS LAB

Assignment-3

Name- Swapnadeep Mishra

Section-A3, Roll-002211001115

1) Write a generic method in Java that takes an array of any data type and sorts the array in ascending order using any sorting algorithm.

```
import java.util.Arrays;


public class GenericSort {
    public static <T extends Comparable<T>> void sortArray(T[] array) {
        Arrays.sort(array);
    }


    public static void main(String[] args) {
        Integer[] intArray = {5, 3, 8, 1, 2};
        System.out.println("Original Integer Array: " +
Arrays.toString(intArray));
        sortArray(intArray);
```

```
        System.out.println("Sorted Integer Array: " +
Arrays.toString(intArray));


        String[] strArray = {"banana", "apple", "orange", "grapes"};
        System.out.println("Original String Array: " +
Arrays.toString(strArray));
        sortArray(strArray);
        System.out.println("Sorted String Array: " +
Arrays.toString(strArray));
    }
}
```

Output:-

```
[be22115@localhost Assignment3]$ javac q1.java
[be22115@localhost Assignment3]$ java GenericSort
Original Integer Array: [5, 3, 8, 1, 2]
Sorted Integer Array: [1, 2, 3, 5, 8]
Original String Array: [banana, apple, orange, grapes]
Sorted String Array: [apple, banana, grapes, orange]
```

2) Write a generic method in Java that takes any type of an array as input and finds the frequency of each data element.

```
import java.util.HashMap;
import java.util.Map;


public class FrequencyCounter {
    public static <T> Map<T, Integer> findFrequency(T[] array) {
        Map<T, Integer> frequencyMap = new HashMap<>();
```

```
        for (T element : array) {
            frequencyMap.put(element, frequencyMap.getOrDefault(element, 0) + 1);
        }

        return frequencyMap;
    }

    public static void main(String[] args) {
        Integer[] intArray = {1, 2, 3, 1, 2, 3, 4, 5, 1};
        System.out.println("Frequency of elements in Integer Array: " + findFrequency(intArray));

        String[] strArray = {"apple", "banana", "apple", "orange", "banana", "apple"};
        System.out.println("Frequency of elements in String Array: " + findFrequency(strArray));
    }
}
```

Output:-

```
[be22115@localhost Assignment3]$ javac q2.java
[be22115@localhost Assignment3]$ java FrequencyCounter
Frequency of elements in Integer Array: {1=3, 2=2, 3=2, 4=1, 5=1}
Frequency of elements in String Array: {banana=2, orange=1, apple=3}
```

3) Design a generic Java class having a method that takes an array of any data type and prints all the duplicate elements.

```
import java.util.HashMap;

import java.util.Map;


public class DuplicateFinder<T> {

   public void findDuplicates(T[] array) {

      Map<T, Integer> frequencyMap = new HashMap<>();

      for (T element : array) {

         frequencyMap.put(element,
frequencyMap.getOrDefault(element, 0) + 1);

      }

      System.out.println("Duplicate elements:");

      for (Map.Entry<T, Integer> entry : frequencyMap.entrySet()) {

         if (entry.getValue() > 1) {

            System.out.println(entry.getKey());

         }

      }

   }


   public static void main(String[] args) {

      Integer[] intArray = {1, 2, 3, 1, 2, 3, 4, 5, 1};

      System.out.println("Duplicates in Integer Array:");
```

```
        new DuplicateFinder<Integer>().findDuplicates(intArray);


        String[] strArray = {"apple", "banana", "apple", "orange", "banana",
"apple"};
        System.out.println("Duplicates in String Array:");
        new DuplicateFinder<String>().findDuplicates(strArray);
    }
}
```

Output:-

```
[be22115@localhost Assignment3]$ javac q3.java
[be22115@localhost Assignment3]$ java DuplicateFinder
Duplicates in Integer Array:
Duplicate elements:
1
2
3
Duplicates in String Array:
Duplicate elements:
banana
apple
```

4) Test the functionalities of different java reflection APIs such as getClass(), getMethods(), getConstructors(), getDeclaredMethod(), getDeclaredField(), setAccessible() etc.

```
import java.lang.reflect.*;


public class ReflectionDemo {
    public static void main(String[] args) throws
NoSuchMethodException, IllegalAccessException,
```

```
InvocationTargetException, InstantiationException,
NoSuchFieldException {
    Class<?> myClass = MyClass.class;

    System.out.println("Class name: " + myClass.getName());

    System.out.println("Public methods:");
    Method[] methods = myClass.getMethods();
    for (Method method : methods) {
        System.out.println(method.getName());
    }

    System.out.println("Constructors:");
    Constructor<?>[] constructors = myClass.getConstructors();
    for (Constructor<?> constructor : constructors) {
        System.out.println(constructor);
    }

    Method myMethod = myClass.getDeclaredMethod("myMethod",
int.class);
    myMethod.setAccessible(true);
    System.out.println("Invoking private method: ");
    myMethod.invoke(myClass.newInstance(), 10);
```

```
        Field myField = myClass.getDeclaredField("myField");
        myField.setAccessible(true);
        MyClass obj = new MyClass();
        System.out.println("Initial value of myField: " + myField.get(obj));
        myField.set(obj, "New Value");
        System.out.println("New value of myField: " + myField.get(obj));
    }
}

class MyClass {
    private String myField = "Initial Value";

    private void myMethod(int x) {
        System.out.println("Inside private method. Value: " + x);
    }
}
```

Output:-

```
[be22115@localhost Assignment3]$ javac q4.java -Xlint
q4.java:24: warning: [deprecation] newInstance() in Class has been deprecated
        myMethod.invoke(myClass.newInstance(), 10);
                                ^
  where T is a type-variable:
    T extends Object declared in class Class
1 warning
[be22115@localhost Assignment3]$ java ReflectionDemo
Class name: MyClass
Public methods:
wait
wait
wait
equals
toString
hashCode
getClass
notify
notifyAll
Constructors:
Invoking private method:
Inside private method. Value: 10
Initial value of myField: Initial Value
New value of myField: New Value
```