OOS LAB
ASSIGNMENT - 3

Name - Anuska Nath

Department - Information Technology

Roll No - 002311001003

Section - A1

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.

Source Code:

```
import java.util.Comparator;

class GenericSorter {
    public static <T extends Comparable<T>> void sortArray(T[] array) {
        // Bubble Sort algorithm
        for (int i = 0; i < array.length - 1; i++) {
            for (int j = 0; j < array.length - i - 1; j++) {
                if (array[j].compareTo(array[j + 1]) > 0) {
                    // Swap the elements
                    T temp = array[j];
                    array[j] = array[j + 1];
                    array[j + 1] = temp;
                }
            }
        }
    }

    public static void main(String[] args) {
        // Test the sortArray method with an Integer array
        Integer[] intArray = {4, 2, 7, 1, 9, 3};
        sortArray(intArray);
        System.out.println("Sorted Integer Array: ");
        for (Integer num : intArray) {
            System.out.print(num + " ");
        }
         System.out.println();

        // Test the sortArray method with a String array
        String[] strArray = {"apple", "orange", "banana", "kiwi"};
        sortArray(strArray);
        System.out.println("Sorted String Array: ");
        for (String str : strArray) {
            System.out.print(str + " ");
        }
         System.out.println();
    }
}
```

Output:

```
[be2303@localhost a3]$ javac q1.java
[be2303@localhost a3]$ java GenericSorter
Sorted Integer Array:
1 2 3 4 7 9
Sorted String Array:
apple banana kiwi 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.

Source Code:

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

class FrequencyCounter {
    public static <T> void findFrequency(T[] array) {
        // Create a Map to store element frequencies
        Map<T, Integer> frequencyMap = new HashMap<>();

        // Iterate through the array to count the frequency of each element
        for (T element : array) {
            frequencyMap.put(element, frequencyMap.getOrDefault(element, 0) + 1);
        }

// Print the frequency of each element
        for (Map.Entry<T, Integer> entry : frequencyMap.entrySet()) {
                System.out.println(entry.getKey() + " : " + entry.getValue() + " time(s)");
        }
    }

    public static void main(String[] args) {
        // Test the findFrequency method with an Integer array
        Integer[] intArray = {1, 2, 2, 3, 3, 3, 4};
        System.out.println("Frequency of elements in Integer array : ");
        findFrequency(intArray);
        System.out.println();

        // Test the findFrequency method with a String array
        String[] strArray = {"apple", "banana", "apple", "kiwi", "banana", "apple"};
        System.out.println("Frequency of elements in String array:");
        findFrequency(strArray);
    }
}
```

Output:

```
[be2303@localhost a3]$ javac q2.java
[be2303@localhost a3]$ java FrequencyCounter
Frequency of elements in Integer array :
1 : 1 time(s)
2 : 2 time(s)
3 : 3 time(s)
4 : 1 time(s)

Frequency of elements in String array:
banana : 2 time(s)
apple : 3 time(s)
kiwi : 1 time(s)
```

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

Source Code:

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

class DuplicateFinder {

    // Generic method to find and print duplicate elements
    public static <T> void printDuplicates(T[] array) {
        // Create a Map to store the frequency of each element
        Map<T, Integer> elementCount = new HashMap<>();

        // Count occurrences of each element
        for (T element : array) {
            elementCount.put(element, elementCount.getOrDefault(element, 0) + 1);
        }

         // Set to store duplicates
        Set<T> duplicates = new HashSet<>();

        // Iterate over the map and find elements with a count greater than 1
(duplicates)
        for (Map.Entry<T, Integer> entry : elementCount.entrySet()) {
            if (entry.getValue() > 1) {
                duplicates.add(entry.getKey());
            }
         }

         // Print the duplicate elements
        if (duplicates.isEmpty()) {
            System.out.println("No duplicates found.");
        } else {
            for (T duplicate : duplicates) {
                System.out.println(duplicate);
            }
         }
    }

    public static void main(String[] args) {
        // Test with an Integer array
        Integer[] intArray = {1, 2, 2, 3, 4, 5, 3};
        System.out.println("Duplicates in Integer array:");
        printDuplicates(intArray);

        System.out.println();
```

```
        // Test with a String array
        String[] strArray = {"apple", "banana", "apple", "kiwi", "banana", "orange"};
        System.out.println("Duplicates in String array:");
        printDuplicates(strArray);
    }
}
```

Output:

```
[be2303@localhost a3]$ javac q3.java
[be2303@localhost a3]$ java DuplicateFinder
Duplicates in Integer array:
2
3

Duplicates in String array:
banana
apple
```

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

Source Code:

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

class Person {
    private String name;
    private int age;

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

    // Public method
    public void introduce() {
        System.out.println("Hello, my name is " + name + " and I am " + age + " years
old.");
    }

    // Private method
    private void setName(String name) {
        this.name = name;
    }

    // Private field
    private String address;
```

```
}

class ReflectionTest {

    public static void main(String[] args) throws Exception {
        // Step 1: Create an object of the Person class
        Person person = new Person("John", 30);

        // Step 2: Using getClass() to get the Class object
        Class<?> personClass = person.getClass();
        System.out.println("Class Name: " + personClass.getName());

        // Step 3: Using getMethods() to get all public methods
        System.out.println("\nPublic Methods:");
        Method[] methods = personClass.getMethods();
        for (Method method : methods) {
            System.out.println(method.getName());
        }

            // Step 4: Using getConstructors() to get all public constructors
        System.out.println("\nPublic Constructors:");
        Constructor<?>[] constructors = personClass.getConstructors();
        for (Constructor<?> constructor : constructors) {
            System.out.println(constructor.getName());
        }

            // Step 5: Using getDeclaredMethod() to get a private method
        System.out.println("\nPrivate Method (using getDeclaredMethod):");
        Method privateMethod = personClass.getDeclaredMethod("setName",
String.class);
        System.out.println("Method Name: " + privateMethod.getName());

        // Step 6: Using setAccessible() to access a private method
        privateMethod.setAccessible(true);
        privateMethod.invoke(person, "Alice");
        System.out.println("Private method invoked successfully.");

        // Step 7: Using getDeclaredField() to access a private field
        System.out.println("\nPrivate Field (using getDeclaredField):");
        Field privateField = personClass.getDeclaredField("address");
        privateField.setAccessible(true);  // Set the field accessible
        privateField.set(person, "123 Main St");
        System.out.println("Private field 'address' value: " + privateField.get(person));

        // Step 8: Modify a private field using reflection
        System.out.println("\nModifying private field 'name':");
        Field nameField = personClass.getDeclaredField("name");
        nameField.setAccessible(true);
        System.out.println("Initial name: " + nameField.get(person));
        nameField.set(person, "Bob");
        System.out.println("Modified name: " + nameField.get(person));
```

```
        // Step 9: Call a public method using reflection
        System.out.println("\nCalling public method 'introduce':");
        Method introduceMethod = personClass.getMethod("introduce");
        introduceMethod.invoke(person);
    }
}
```

Output:

```
[be2303@localhost a3]$ javac q4.java
[be2303@localhost a3]$ java ReflectionTest
Class Name: Person

Public Methods:
introduce
wait
wait
wait
equals
toString
hashCode
getClass
notify
notifyAll

Public Constructors:
Person

Private Method (using getDeclaredMethod):
Method Name: setName
Private method invoked successfully.

Private Field (using getDeclaredField):
Private field 'address' value: 123 Main St

Modifying private field 'name':
Initial name: Alice
Modified name: Bob

Calling public method 'introduce':
Hello, my name is Bob and I am 30 years old.
```