

Assignment 7 - Generics

Q1: Write a Java program to find the maximum value from the given type of elements using a generic function.

Code:

```
import java.lang.Comparable;

class MaxValue {
    //generic function to find maximum element
    public static <type extends Comparable<type>> type maximum (type[] elements) {
        type max = elements[0];
        int comparator;
        for(int i=0; i<elements.length; i++) {
            comparator = max.compareTo(elements[i]);
            if (comparator<0)
                max = elements[i];
        }
        return max;
    }

    public static void main(String arg[]) {
        //testing function for integer datatype
        Integer[] intArray= new Integer[] {21,43,51,98,14,32,64};
        System.out.print("Integer: ");
        for (int i=0; i<intArray.length; i++)
            System.out.print(intArray[i] + " ");
        System.out.println("\nMaximum value: " + maximum(intArray));

        //testing function for float datatype
        Float[] floatArray = new Float[] {1.1f,21.9f,3.2f,7f,5f,1.9f,9.8f,24.6f,0.2f,17.4f};
        System.out.print("\nFloat: ");
        for (int i=0; i<floatArray.length; i++)
            System.out.print(floatArray[i] + " ");
        System.out.println("\nMaximum value: " + maximum(floatArray));

        //testing function for string datatype
        String[] stringArray= new String[]
{"each","different","datatype","has","generic","parametric"};
        System.out.print("\nString: ");
        for(int i=0; i<stringArray.length; i++)
            System.out.print(stringArray[i] + " ");
        System.out.println("\nMaximum value: " + maximum(stringArray));
    }
}
```

Output:

```
kri@kri-ubuntu:~/workspace$ javac MaxValue.java
kri@kri-ubuntu:~/workspace$ java MaxValue
Integer: 21 43 51 98 14 32 64
Maximum value: 98

Float: 1.1 21.9 3.2 7.0 5.0 1.9 9.8 24.6 0.2 17.4
Maximum value: 24.6

String: each different datatype has generic parametric
Maximum value: parametric
kri@kri-ubuntu:~/workspace$
```

Q2: Write a Java program to create a generic stack using interface and perform the operations.

Code:

```
import java.util.ArrayList;

interface StackInterface <type> {
    public void push(type element);
    public type pop();
    public void display();
}

class Stack<type> implements StackInterface<type> {
    //data members - stack attributes
    int top;
    ArrayList<type> array;
    int size;

    //constructor - create stack
    Stack(int size) {
        this.size = size;
        top = -1;
        array = new ArrayList<type>(size);
    }

    //stack operations
    //push into stack
    public void push (type element) {
        if(top >= size-1)
            System.out.println("Stack is full! Push operation is invalid.");
        else
            array.add(++top,element);
    }

    //pop top element from stack
    public type pop() {
        if (top== -1) {
            System.out.println("Stack is empty. Pop operation is invalid.");
            return null;
        }
        else {
            type temp = array.get(top);
            array.remove(top--);
            return temp;
        }
    }
}
```

```
        //display stack contents
    public void display() {
        System.out.print("\nDisplaying stack contents: ");
        for(int i=0;i<=top;i++)
            System.out.print(array.get(i) + " ");
        System.out.println();
    }
}

class TestOperations {
    public static void main(String arg[]) {
        int n=5;
        //testing generic stack operations for different datatypes
        //integer stack
        Stack<Integer> intStack = new Stack<Integer>(n);
        System.out.println("Integer stack: pushing 2 4 6 8 10 into the stack...");
        for (int i=2; i<11; i+=2)
            intStack.push(i);
        intStack.display();
        System.out.println("Pushing 12 into stack...");
        intStack.push(12);
        System.out.println("Popping the top element..." + intStack.pop());
        intStack.display();
        System.out.println("Popping out all elements...");
        for (int i=0; i<5; i++)
            intStack.pop();
        intStack.display();

        //character stack
        Stack<Character> charStack= new Stack<Character>(n);
        System.out.println("\nCharacter stack: pushing a b c d e into the stack...");
        for(int i='a'; i<='e'; i++)
            charStack.push((char)i);
        charStack.display();
        System.out.println("Pushing f into stack...");
        charStack.push('f');
        System.out.println("Popping the top element..." + charStack.pop());
        charStack.display();
        for (int i=0; i<5; i++)
            System.out.println("Popping the top element..." + charStack.pop());
        charStack.display();

        //double stack
        Stack<Double> doubleStack= new Stack<Double>(n);
        System.out.println("\nDouble stack: pushing 0.00 1.11 2.22 3.33 4.44 5.55 into the stack...");
        doubleStack.push(0.00);
        doubleStack.push(1.11);
        doubleStack.push(2.22);
        doubleStack.push(3.33);
```

```
doubleStack.push(4.44);
doubleStack.push(5.55);
doubleStack.display();
System.out.println("Popping the top element..." + doubleStack.pop());
doubleStack.display();
for (int i=0; i<5; i++)
    System.out.println("Popping the top element..." + doubleStack.pop());
doubleStack.display();

//string stack
Stack<String> stringStack= new Stack<String>(n);
System.out.println("\nString stack: pushing first second third fourth fifth sixth into the
stack...");
stringStack.push("first");
stringStack.push("second");
stringStack.push("third");
stringStack.push("fourth");
stringStack.push("fifth");
stringStack.push("sixth");
stringStack.display();
System.out.println("Popping the top element..." + stringStack.pop());
stringStack.display();
for (int i=0; i<5; i++)
    System.out.println("Popping the top element..." + stringStack.pop());
stringStack.display();
}
}
```

Output:

```
kri@kri-ubuntu:~/workspace$ javac TestOperations.java
kri@kri-ubuntu:~/workspace$ java TestOperations
Integer stack: pushing 2 4 6 8 10 into the stack...

Displaying stack contents: 2 4 6 8 10
Pushing 12 into stack...
Stack is full! Push operation is invalid.
Popping the top element...10

Displaying stack contents: 2 4 6 8
Popping out all elements...
Stack is empty. Pop operation is invalid.

Displaying stack contents:

Character stack: pushing a b c d e into the stack...

Displaying stack contents: a b c d e
Pushing f into stack...
Stack is full! Push operation is invalid.
Popping the top element...e

Displaying stack contents: a b c d
Popping the top element...d
Popping the top element...c
Popping the top element...b
Popping the top element...a
Stack is empty. Pop operation is invalid.
Popping the top element...null

Displaying stack contents:
```

```
Double stack: pushing 0.00 1.11 2.22 3.33 4.44 5.55 into the stack...
Stack is full! Push operation is invalid.
```

```
Displaying stack contents: 0.0 1.11 2.22 3.33 4.44
Popping the top element...4.44
```

```
Displaying stack contents: 0.0 1.11 2.22 3.33
Popping the top element...3.33
Popping the top element...2.22
Popping the top element...1.11
Popping the top element...0.0
Stack is empty. Pop operation is invalid.
Popping the top element...null
```

```
Displaying stack contents:
```

```
String stack: pushing first second third fourth fifth sixth into the stack...
Stack is full! Push operation is invalid.
```

```
Displaying stack contents: first second third fourth fifth
Popping the top element...fifth
```

```
Displaying stack contents: first second third fourth
Popping the top element...fourth
Popping the top element...third
Popping the top element...second
Popping the top element...first
Stack is empty. Pop operation is invalid.
Popping the top element...null
```

```
Displaying stack contents:
```

Q3: Write a Java program to perform a sorting operation on various types of elements using a generic method.

Code:

```
import java.lang.Comparable;

public class TestSort {
    //generic function for sorting
    public static <type extends Comparable<type>> void sort (type[] array) {
        for (int i=0; i<array.length; i++) {
            type temp = array[i];
            int j=i;
            while(j>0 && array[j-1].compareTo(temp)>=0) {
                array[j] = array[j-1];
                j--;
            }
            array[j]=temp;
        }
    }

    //function to display array
    public static <type> void display (type[] array) {
        System.out.print("Array: ");
        for (int i=0; i<array.length; i++)
            System.out.print(array[i]+" ");
        System.out.println();
    }

    public static void main(String arg[]) {
        //testing sort function for different datatypes
        //integer array
        Integer[] array = new Integer[] {42,101,75,61,9,23};
        System.out.println("Original integer array: ");
        display(array);
        System.out.println("Sorted integer array: ");
        sort(array);
        display(array);

        //character array
        Character[] array2 = new Character[] {'d','a','b','e','c'};
        System.out.println("\nOriginal character array: ");
        display(array2);
        System.out.println("Sorted character array: ");
        sort(array2);
        display(array2);

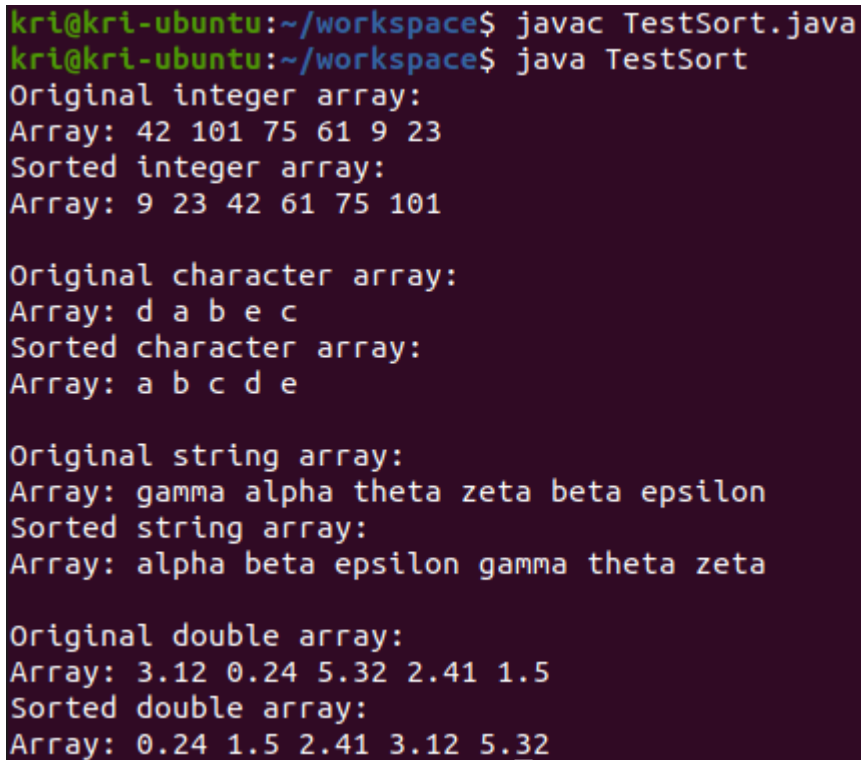
        //string array
```



```
String[] array3 = new String[] {"gamma","alpha","theta","zeta","beta","epsilon"};
System.out.println("\nOriginal string array: ");
display(array3);
System.out.println("Sorted string array: ");
sort(array3);
display(array3);

//double array
Double[] array4 = new Double[] {3.12,0.24,5.32,2.41,1.50};
System.out.println("\nOriginal double array: ");
display(array4);
System.out.println("Sorted double array: ");
sort(array4);
display(array4);
}
```

Output:



```
kri@kri-ubuntu:~/workspace$ javac TestSort.java
kri@kri-ubuntu:~/workspace$ java TestSort
Original integer array:
Array: 42 101 75 61 9 23
Sorted integer array:
Array: 9 23 42 61 75 101

Original character array:
Array: d a b e c
Sorted character array:
Array: a b c d e

Original string array:
Array: gamma alpha theta zeta beta epsilon
Sorted string array:
Array: alpha beta epsilon gamma theta zeta

Original double array:
Array: 3.12 0.24 5.32 2.41 1.5
Sorted double array:
Array: 0.24 1.5 2.41 3.12 5.32
```