Ex-no.: 7 Name: M.Rohith
Date: 28-11-2022 3122 21 5001 085

SSN College of Engineering

Department of Computer Science and Engineering

UCS2313 – Object Oriented Programming Lab

II Year CSE - B Section (III Semester)

Academic Year 2022-23

Batch: 2021- 2025

Faculty Incharge :S.Rajalakshmi

Exercise - 7 – Generics

Objective:

1. To implement generic types – generic classes and methods

Sample Learning Outcome:

1. Learning to create a generic class and method

2. Use the generic class to store/manipulate elements of different datatypes

Best Practices:

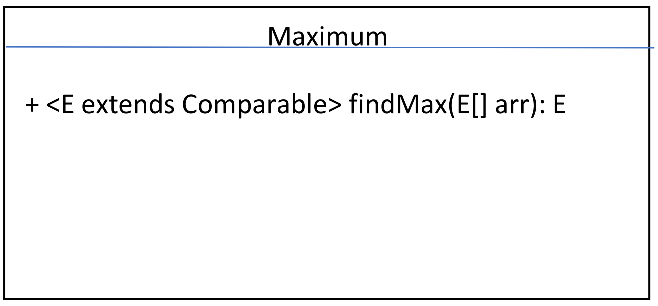
1. Class Diagram usage
2. Naming convention – for file names, variables
3. Comment usage at proper places
4. Prompt messages during reading input and displaying output
5. Incremental program development
6. Modularity
7. All possible test cases in output

Ex-no.: 7 Name: M.Rohith
Date: 28-11-2022 3122 21 5001 085

Exercises:

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

Class Diagram:

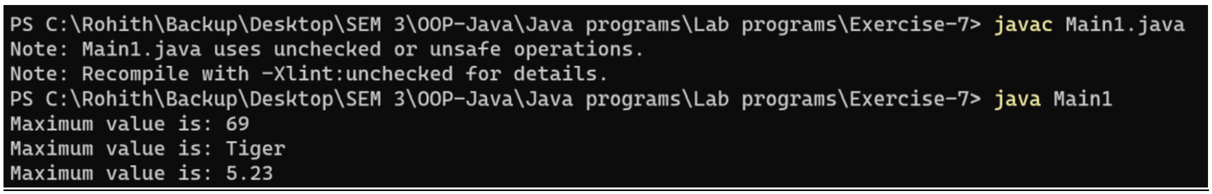


Program Code:

```
import java.util.Scanner;

class Maximum
{
    public <E extends Comparable> E findMax(E[] arr)
    {
        E max=arr[0];
        for(int i=0; i<arr.length; i++)
        {
            if(arr[i].compareTo(max)>0)
               max=arr[i];
        }
        return max;
    }
}

class Main1
{
    public static void main(String[] args)
    {
        Maximum max=new Maximum();
        Integer arr1[]={21,18,11,14,7,63,69,56};
        System.out.println("Maximum value is:
"+max.<Integer>findMax(arr1));
```

```
        String arr2[]={"Cat","Dog","Tiger","Ant"};
        System.out.println("Maximum value is: "+
max.<String>findMax(arr2));
        Double arr3[]={2.1,3.9,4.9,5.0,4.95,5.23};
        System.out.println("Maximum value is:
"+max.<Double>findMax(arr3));
    }
}
```

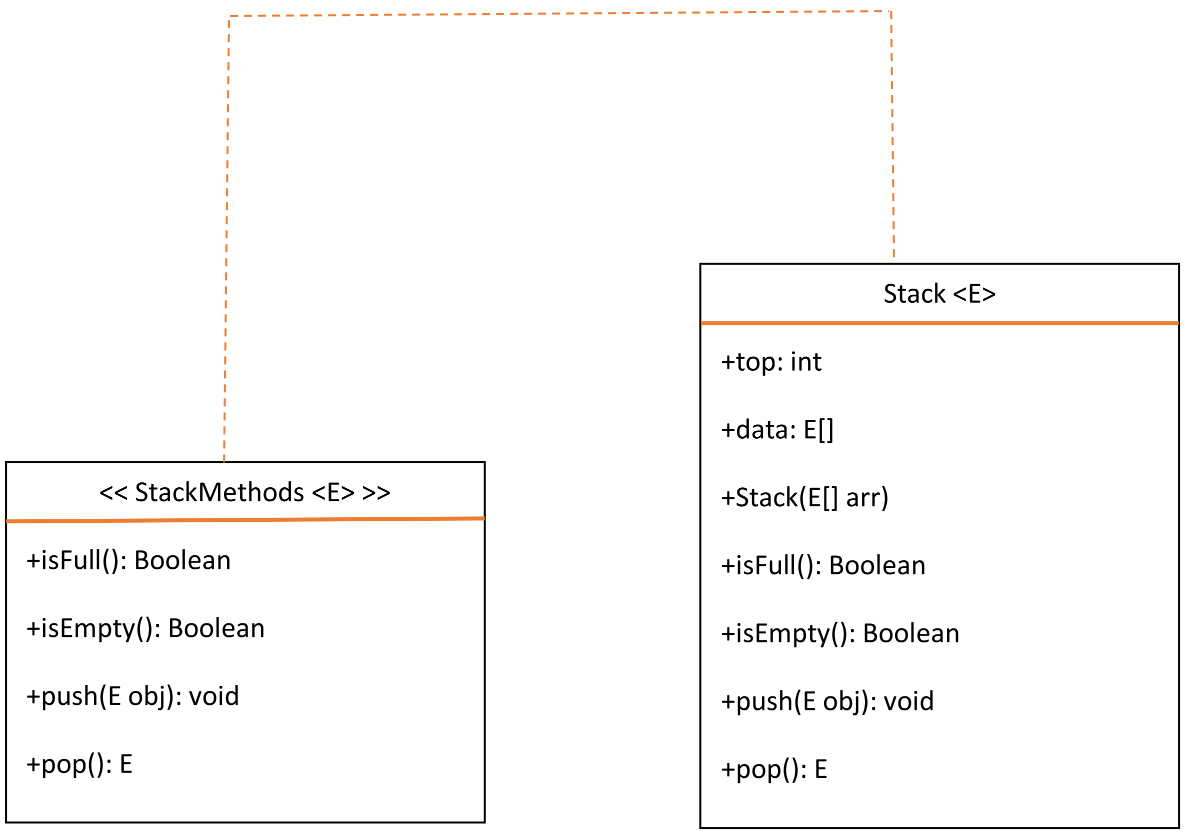
Output:



```
PS C:\Rohith\Backup\Desktop\SEM 3\OOP-Java\Java programs\Lab programs\Exercise-7> javac Main1.java
Note: Main1.java uses unchecked or unsafe operations.
Note: Recompile with -Xlint:unchecked for details.
PS C:\Rohith\Backup\Desktop\SEM 3\OOP-Java\Java programs\Lab programs\Exercise-7> java Main1
Maximum value is: 69
Maximum value is: Tiger
Maximum value is: 5.23
```

2. **Write a Java program to create a generic stack using interface and perform the operations.**

Class Diagram:



Ex-no.: 7 Name: M.Rohith
Date: 28-11-2022 3122 21 5001 085

Program code:

```
import java.util.*;

interface StackMethods <E>
{
    Boolean isFull();
    Boolean isEmpty();
    void push(E obj);
    E pop();
}

class Stack<E> implements StackMethods <E>
{
    int top;
    E[] data;
    public Stack(E[] arr)
    {
        data=arr;
    }

    public Boolean isFull()
    {
        if(top==data.length)
            return true;
        else
            return false;
    }

    public Boolean isEmpty()
    {
        if(top==0)
            return true;
        else
            return false;
    }

    public void push(E element)
    {
        if(isFull()==false)
        {
            data[top]=element;
            top++;
        }
        else
        {
            System.out.println("Stack overflow. Cannot push the
element "+element);
        }
    }

    public E pop()
    {
        if(isEmpty()==false)
            return data[--top];
        else
        {
            System.out.println("Stack underflow. Cannot pop an
element");
            return null;
        }
    }
}

class Main2
{
    public static void main(String[] args)
    {
        Scanner scanner=new Scanner(System.in);
        int choice1,choice2;
        do
        {
            System.out.println("Choices: ");
            System.out.println("1.Integer datatype\n2.Double
datatype\n3.String datatype\n4.Exit\n");
            System.out.println("Enter the choice: ");
            choice1=scanner.nextInt();
            switch(choice1)
            {
                case 1:
                    Integer a1[]=new Integer[2];
                    Stack<Integer> s1=new Stack<Integer>(a1);
                    do
                    {
```

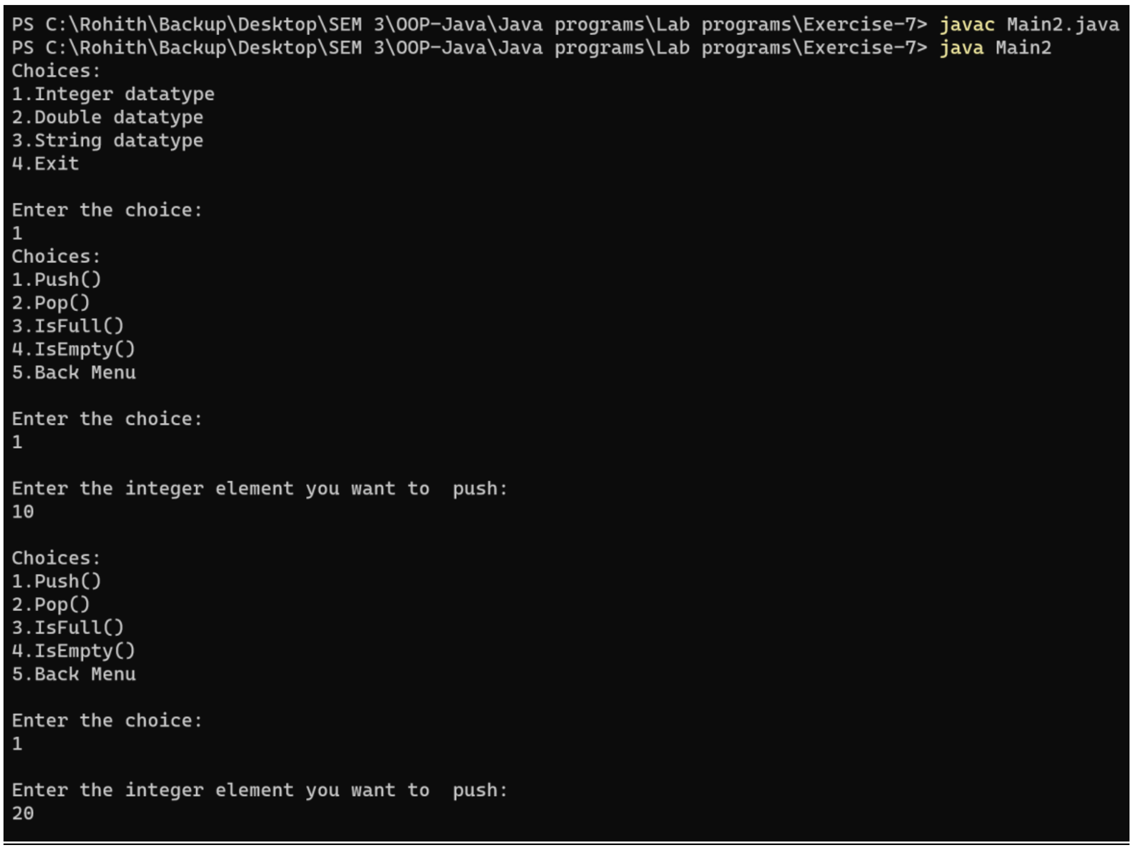
```
            System.out.println("Choices: ");
            System.out.println("1.Push()\n2.Pop()\n3.IsF
ull()\n4.IsEmpty()\n5.Back Menu\n");
            System.out.println("Enter the choice: ");
            choice2=scanner.nextInt();
            switch(choice2)
            {
              case 1:
                System.out.println("\nEnter the
integer element you want to  push: ");
                int data=scanner.nextInt();
                s1.push(data);
                System.out.println();
                break;
              case 2:
                System.out.println("Popped element
is: "+s1.pop()+"\n");
                break;
              case 3:
                System.out.println("\nIsFull?
"+s1.isFull()+"\n");
                break;
              case 4:
                System.out.println("\nIsEmpty?
"+s1.isEmpty()+"\n");
                break;
              case 5:
                System.out.println("\nBack menu\n");
                break;
              default:
                System.out.println("\nInvalid
choice...\n");
            }
          }while(choice2!=5);
          break;
        case 2:
          Double a2[]=new Double[2];
          Stack<Double> s2=new Stack<Double>(a2);
          do
          {
            System.out.println("Choices: ");
```

```
                        System.out.println("1.Push()\n2.Pop()\n3.IsF
ull()\n4.IsEmpty()\n5.Back Menu\n");
                        System.out.println("Enter the choice: ");
                        choice2=scanner.nextInt();
                        switch(choice2)
                        {
                            case 1:
                                System.out.println("\nEnter the
double element you want to  push: ");
                                double data=scanner.nextDouble();
                                s2.push(data);
                                System.out.println();
                                break;
                            case 2:
                                System.out.println("Popped element
is: "+s2.pop()+"\n");

                                break;
                            case 3:
                                System.out.println("\nIsFull?
"+s2.isFull()+"\n");

                                break;
                            case 4:
                                System.out.println("\nIsEmpty?
"+s2.isEmpty()+"\n");

                                break;
                            case 5:
                                System.out.println("\nBack menu\n");
                                break;
                            default:
                                System.out.println("\nInvalid
choice...\n");
                        }
                    }while(choice2!=5);
                    break;
                case 3:
                    String a3[]=new String[2];
                    Stack<String> s3=new Stack<String>(a3);
                    do
                    {
                        System.out.println("Choices: ");
                        System.out.println("1.Push()\n2.Pop()\n3.IsF
ull()\n4.IsEmpty()\n5.Back\n");
```

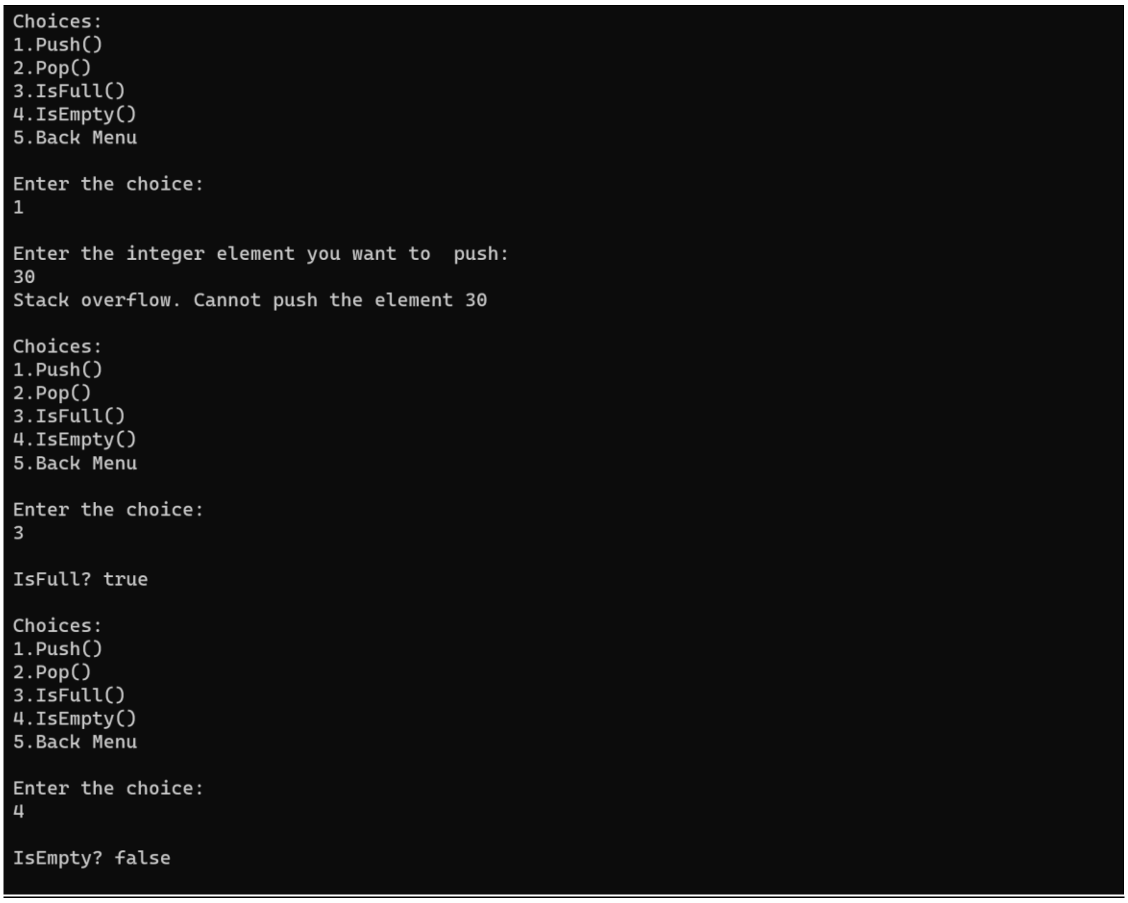
```
                        System.out.println("Enter the choice: ");
                        choice2=scanner.nextInt();
                        switch(choice2)
                        {
                            case 1:
                                System.out.println("\nEnter the 
string element you want to  push: ");
                                scanner.nextLine();
                                String data=scanner.nextLine();
                                s3.push(data);
                                System.out.println();
                                break;
                            case 2:
                                System.out.println("Popped element 
is: "+s3.pop()+"\n");
                                break;
                            case 3:
                                System.out.println("\nIsFull? 
"+s3.isFull()+"\n");
                                break;
                            case 4:
                                System.out.println("\nIsEmpty? 
"+s3.isEmpty()+"\n");
                                break;
                            case 5:
                                System.out.println("\nBack menu\n");
                                break;
                            default:
                                System.out.println("\nInvalid 
choice...\n");
                        }
                    }while(choice2!=5);
                    break;
                case 4:
                    System.exit(1);
                default:
                    System.out.println("\nInvalid choice...\n");
            }
        }while(choice1!=4);
    }
}
```

Ex-no.: 7 Name: M.Rohith
Date: 28-11-2022 3122 21 5001 085

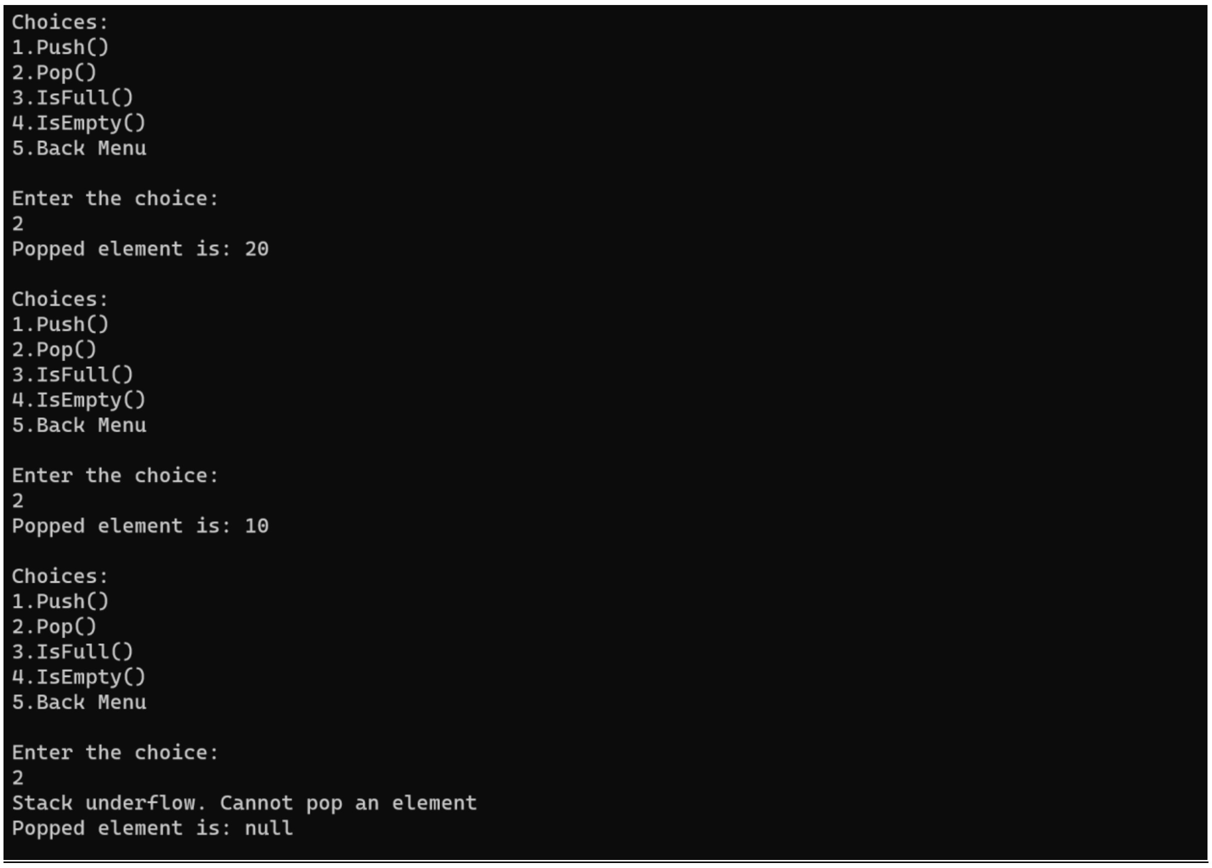
Output:



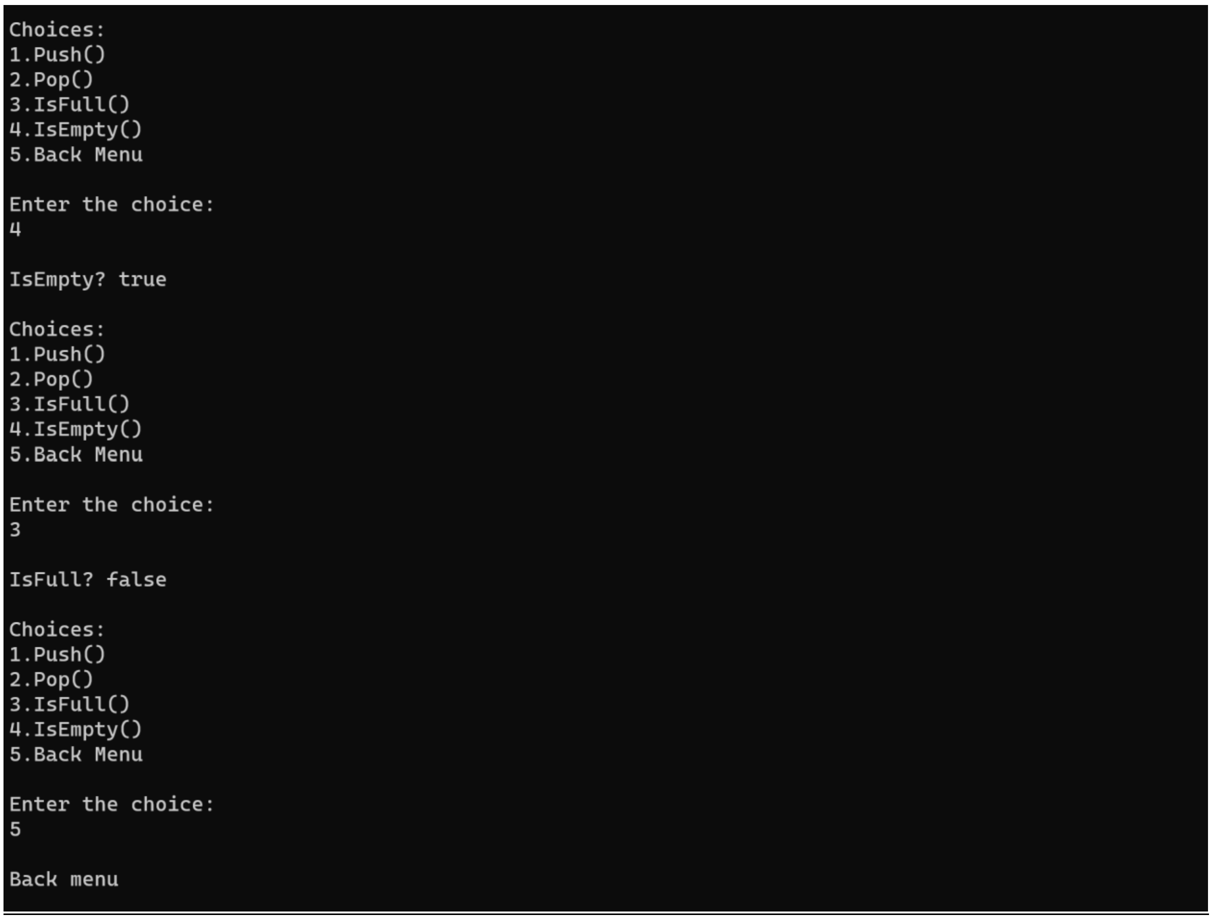
```
PS C:\Rohith\Backup\Desktop\SEM 3\OOP-Java\Java programs\Lab programs\Exercise-7> javac Main2.java
PS C:\Rohith\Backup\Desktop\SEM 3\OOP-Java\Java programs\Lab programs\Exercise-7> java Main2
Choices:
1.Integer datatype
2.Double datatype
3.String datatype
4.Exit

Enter the choice:
1
Choices:
1.Push()
2.Pop()
3.IsFull()
4.IsEmpty()
5.Back Menu

Enter the choice:
1

Enter the integer element you want to  push:
10

Choices:
1.Push()
2.Pop()
3.IsFull()
4.IsEmpty()
5.Back Menu

Enter the choice:
1

Enter the integer element you want to  push:
20
```



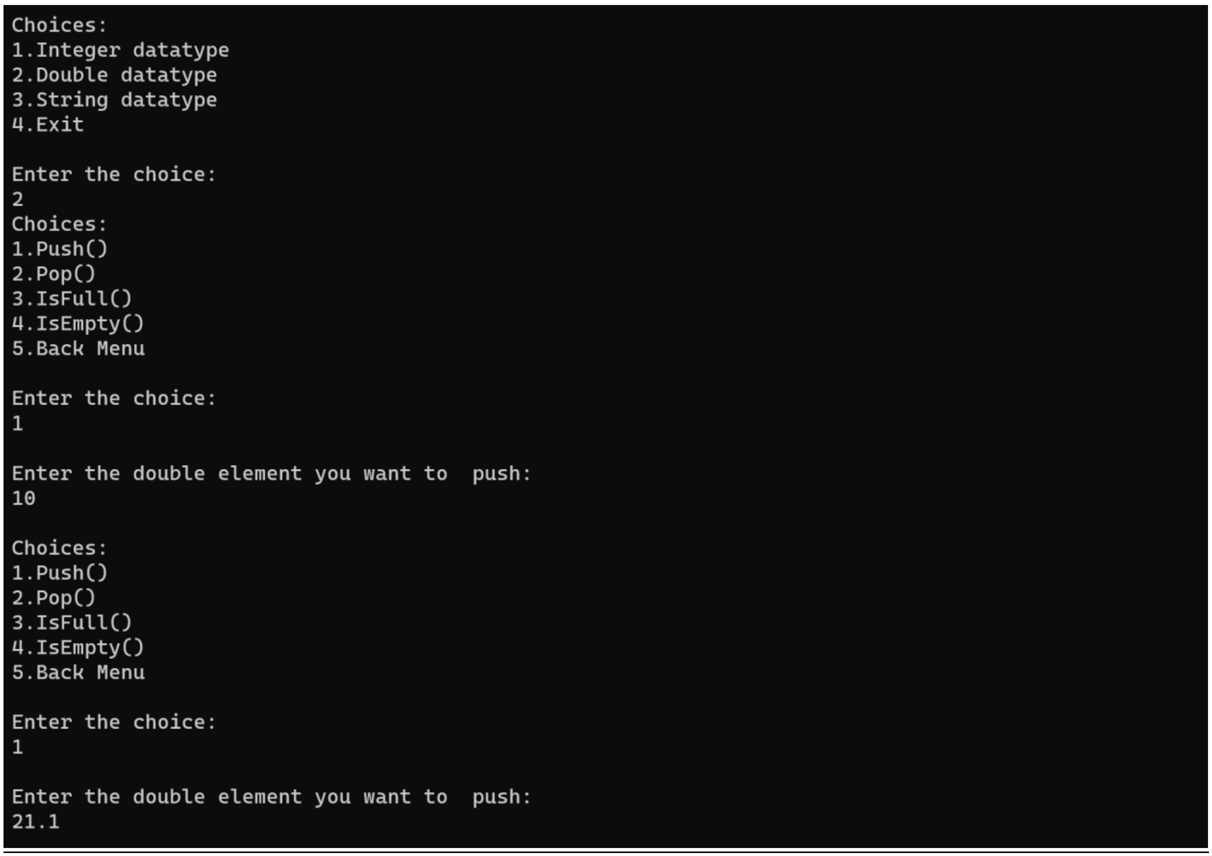
```
Choices:
1.Push()
2.Pop()
3.IsFull()
4.IsEmpty()
5.Back Menu

Enter the choice:
1

Enter the integer element you want to  push:
30
Stack overflow. Cannot push the element 30

Choices:
1.Push()
2.Pop()
3.IsFull()
4.IsEmpty()
5.Back Menu

Enter the choice:
3

IsFull? true

Choices:
1.Push()
2.Pop()
3.IsFull()
4.IsEmpty()
5.Back Menu

Enter the choice:
4

IsEmpty? false
```



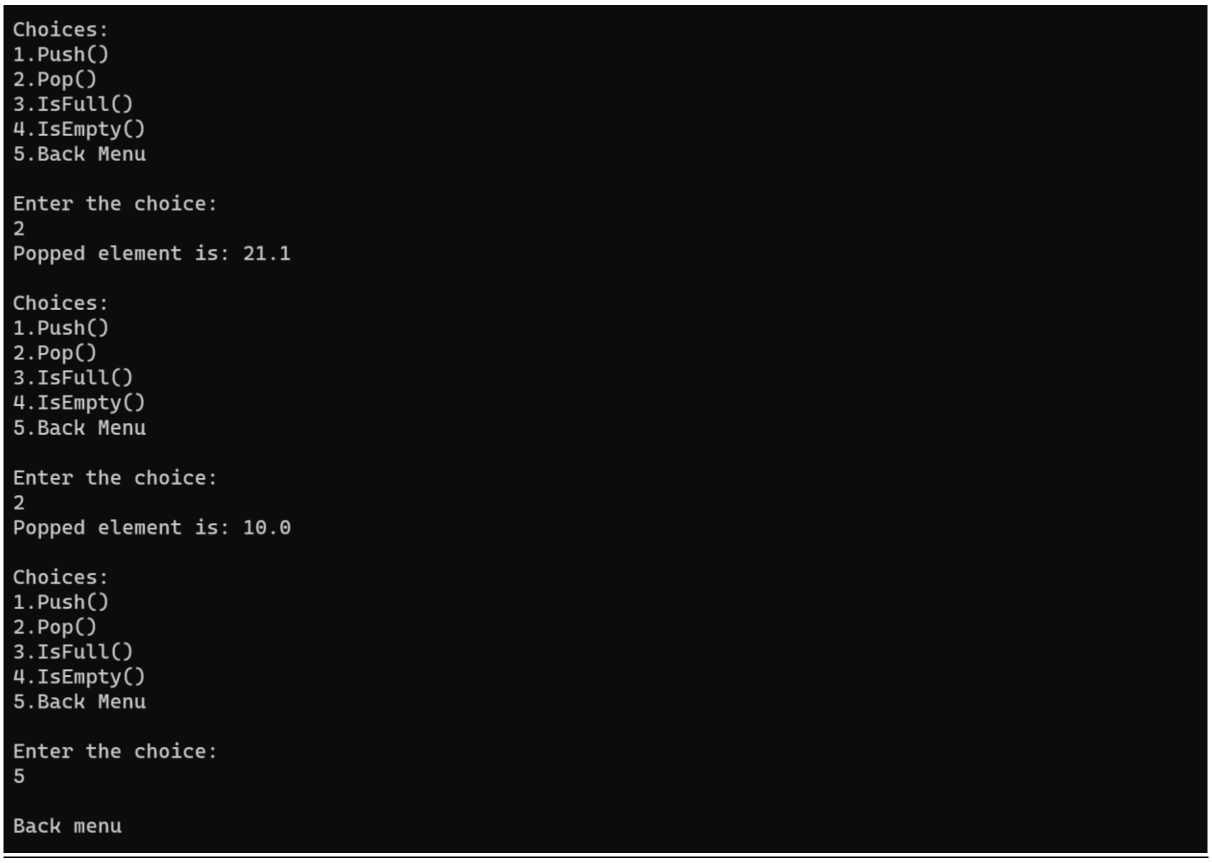
```
Choices:
1.Push()
2.Pop()
3.IsFull()
4.IsEmpty()
5.Back Menu

Enter the choice:
2
Popped element is: 20

Choices:
1.Push()
2.Pop()
3.IsFull()
4.IsEmpty()
5.Back Menu

Enter the choice:
2
Popped element is: 10

Choices:
1.Push()
2.Pop()
3.IsFull()
4.IsEmpty()
5.Back Menu

Enter the choice:
2
Stack underflow. Cannot pop an element
Popped element is: null
```



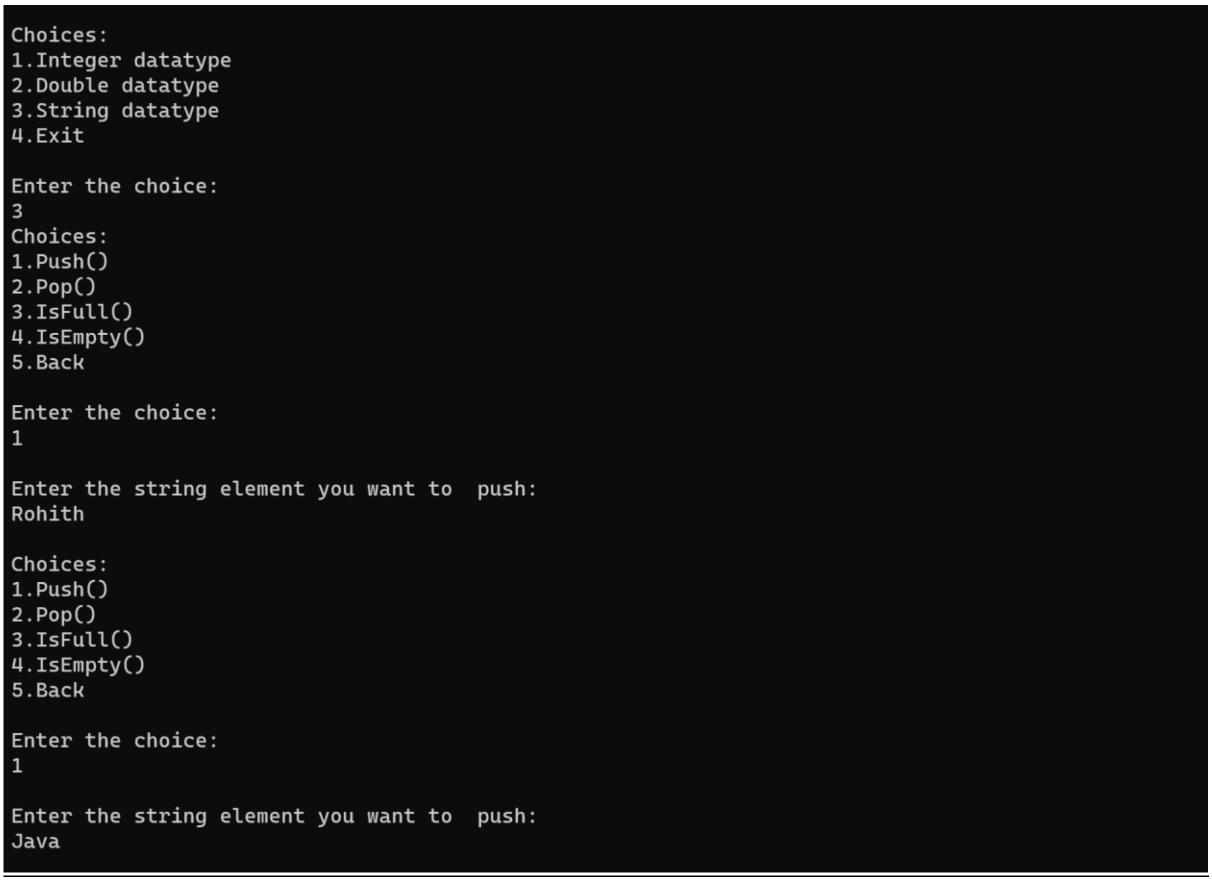
```
Choices:
1.Push()
2.Pop()
3.IsFull()
4.IsEmpty()
5.Back Menu

Enter the choice:
4

IsEmpty? true

Choices:
1.Push()
2.Pop()
3.IsFull()
4.IsEmpty()
5.Back Menu

Enter the choice:
3

IsFull? false

Choices:
1.Push()
2.Pop()
3.IsFull()
4.IsEmpty()
5.Back Menu

Enter the choice:
5

Back menu
```



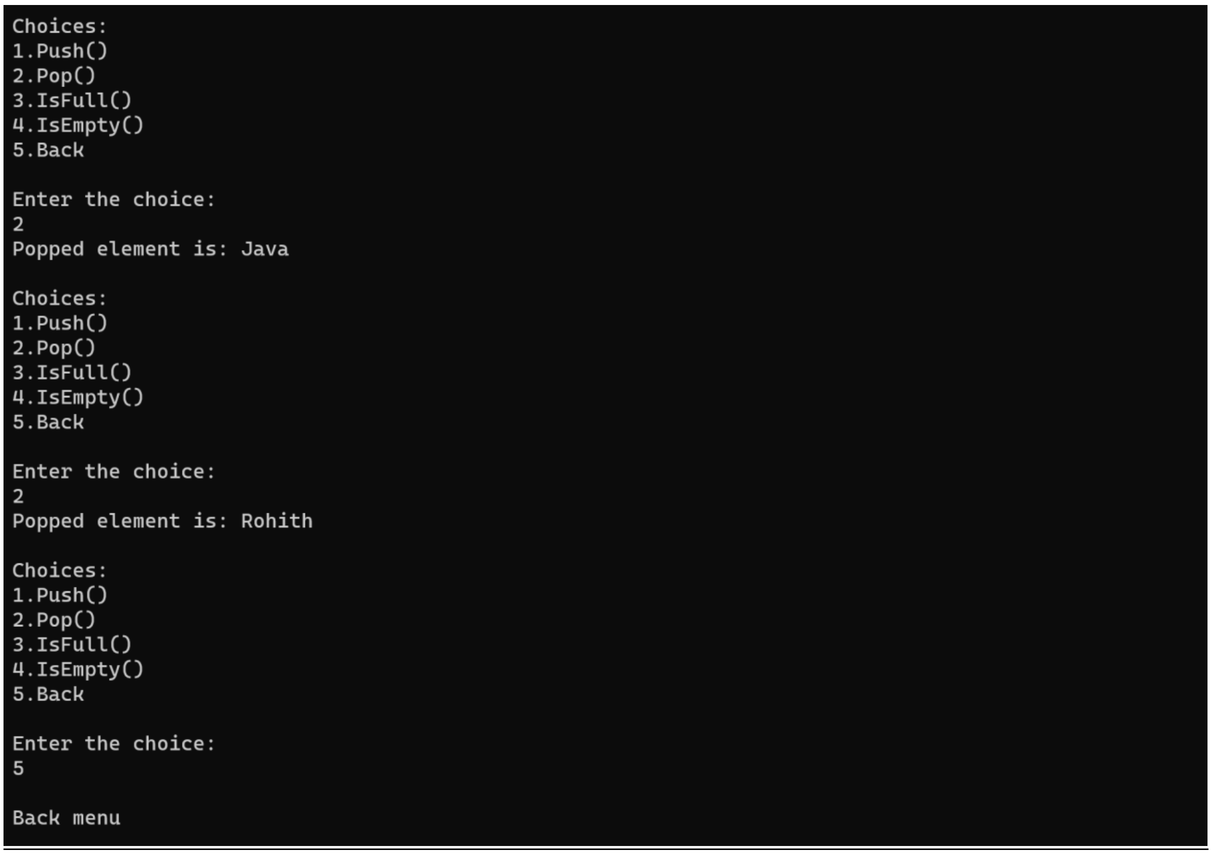
```
Choices:
1.Integer datatype
2.Double datatype
3.String datatype
4.Exit

Enter the choice:
2
Choices:
1.Push()
2.Pop()
3.IsFull()
4.IsEmpty()
5.Back Menu

Enter the choice:
1

Enter the double element you want to  push:
10

Choices:
1.Push()
2.Pop()
3.IsFull()
4.IsEmpty()
5.Back Menu

Enter the choice:
1

Enter the double element you want to  push:
21.1
```



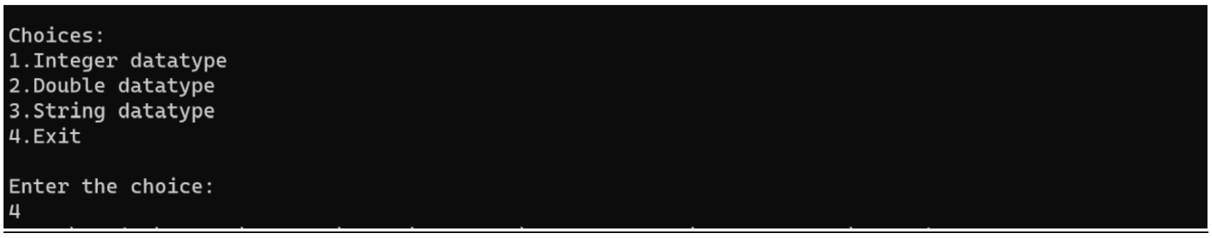
```
Choices:
1.Push()
2.Pop()
3.IsFull()
4.IsEmpty()
5.Back Menu

Enter the choice:
2
Popped element is: 21.1

Choices:
1.Push()
2.Pop()
3.IsFull()
4.IsEmpty()
5.Back Menu

Enter the choice:
2
Popped element is: 10.0

Choices:
1.Push()
2.Pop()
3.IsFull()
4.IsEmpty()
5.Back Menu

Enter the choice:
5

Back menu
```



```
Choices:
1.Integer datatype
2.Double datatype
3.String datatype
4.Exit

Enter the choice:
3
Choices:
1.Push()
2.Pop()
3.IsFull()
4.IsEmpty()
5.Back

Enter the choice:
1

Enter the string element you want to  push:
Rohith

Choices:
1.Push()
2.Pop()
3.IsFull()
4.IsEmpty()
5.Back

Enter the choice:
1

Enter the string element you want to  push:
Java
```



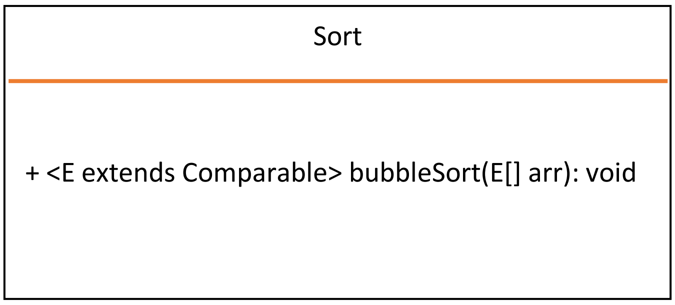
```
Choices:
1.Push()
2.Pop()
3.IsFull()
4.IsEmpty()
5.Back

Enter the choice:
2
Popped element is: Java

Choices:
1.Push()
2.Pop()
3.IsFull()
4.IsEmpty()
5.Back

Enter the choice:
2
Popped element is: Rohith

Choices:
1.Push()
2.Pop()
3.IsFull()
4.IsEmpty()
5.Back

Enter the choice:
5

Back menu
```



```
Choices:
1.Integer datatype
2.Double datatype
3.String datatype
4.Exit

Enter the choice:
4
```

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

Class Diagram:

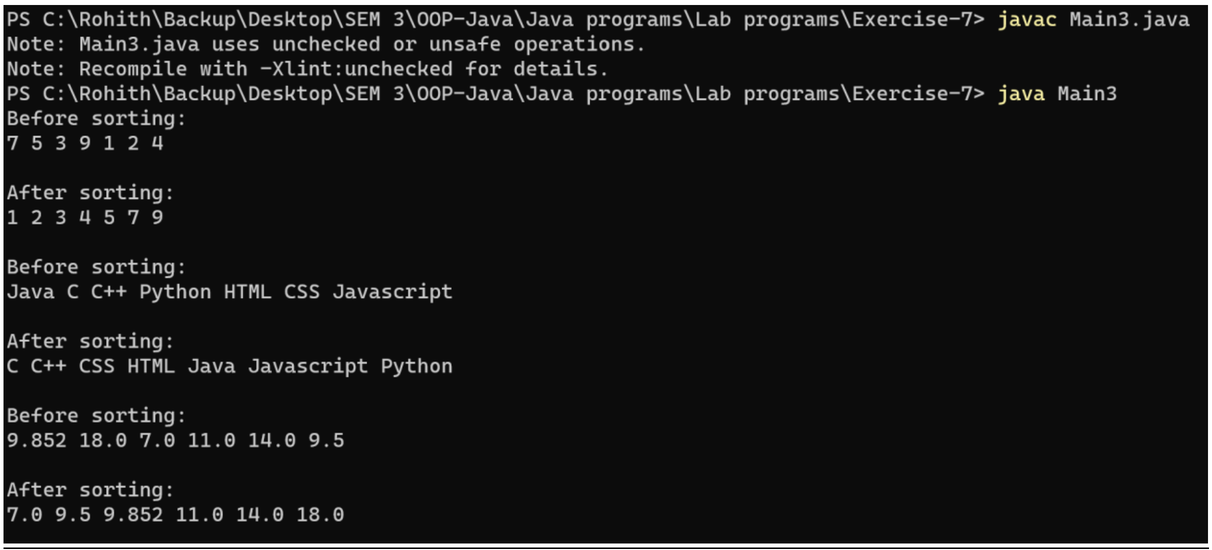


Program code:

```java
import java.util.ArrayList;

class Sort
{
    public <E extends Comparable> void bubbleSort(E[] arr)
    {
        E temp;
        System.out.println("Before sorting: ");
        for(E ele:arr)
            System.out.print(ele+" ");
        System.out.println("\n");

        for(int i=0; i<arr.length; i++)
        {
            for(int j=0; j<arr.length-i-1; j++)
            {

                if(arr[j].compareTo(arr[j+1]) > 0)
                {
                    temp=arr[j];
                    arr[j]=arr[j+1];
                    arr[j+1]=temp;
                }
            }
        }

        System.out.println("After sorting: ");
        for(E ele:arr)
            System.out.print(ele+" ");
        System.out.println("\n");
    }
}

class Main3
{
    public static void main(String args[])
    {
        Sort sort=new Sort();
        Integer arr1[]={7,5,3,9,1,2,4};
        sort.<Integer>bubbleSort(arr1);

        String
arr2[]={"Java","C","C++","Python","HTML","CSS","Javascript"};
        sort.<String>bubbleSort(arr2);

        Double arr3[]={9.852,18.0,7.0,11.0,14.0,9.5};
        sort.<Double>bubbleSort(arr3);
    }
}
```

Ex-no.: 7 Name: M.Rohith
Date: 28-11-2022 3122 21 5001 085

Output:



```
PS C:\Rohith\Backup\Desktop\SEM 3\OOP-Java\Java programs\Lab programs\Exercise-7> javac Main3.java
Note: Main3.java uses unchecked or unsafe operations.
Note: Recompile with -Xlint:unchecked for details.
PS C:\Rohith\Backup\Desktop\SEM 3\OOP-Java\Java programs\Lab programs\Exercise-7> java Main3
Before sorting:
7 5 3 9 1 2 4

After sorting:
1 2 3 4 5 7 9

Before sorting:
Java C C++ Python HTML CSS Javascript

After sorting:
C C++ CSS HTML Java Javascript Python

Before sorting:
9.852 18.0 7.0 11.0 14.0 9.5

After sorting:
7.0 9.5 9.852 11.0 14.0 18.0
```

Learning Outcomes:

Hence Java Programs have been implemented to create generic classes and methods and to use the generic classes to store or manipulate the elements of different datatypes have been performed successfully.