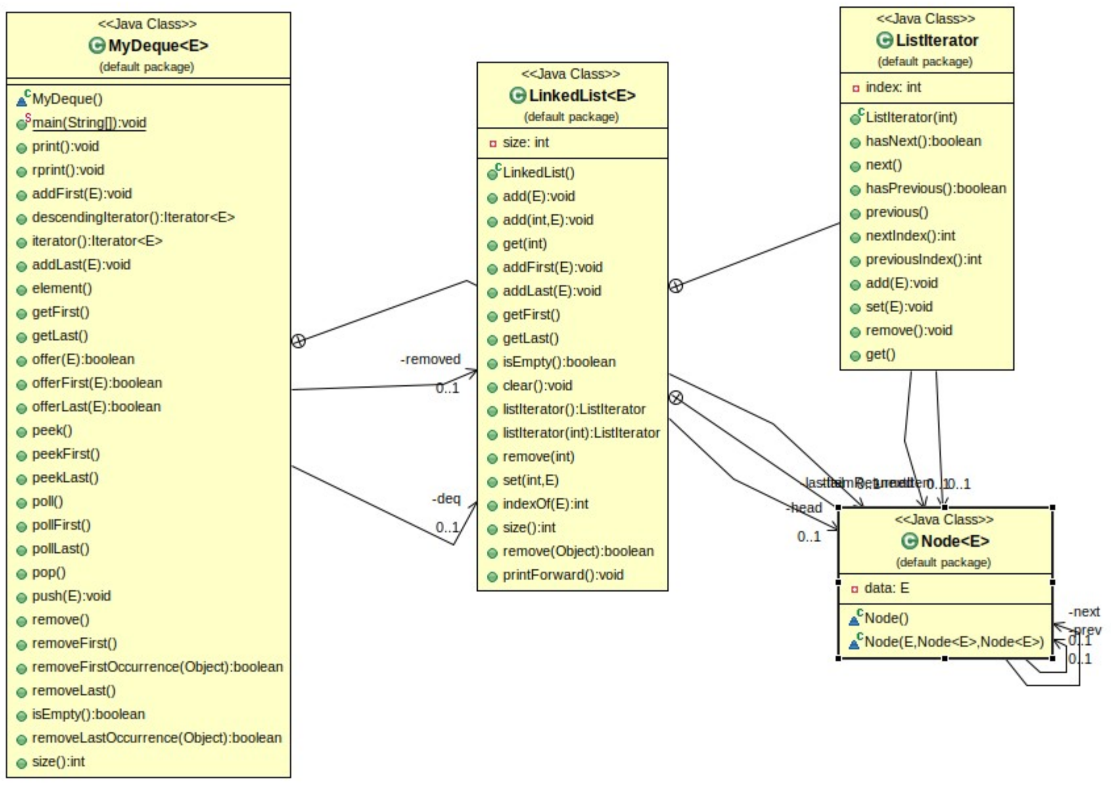
Question 2)

Class Diagram



Problem Solution Approach

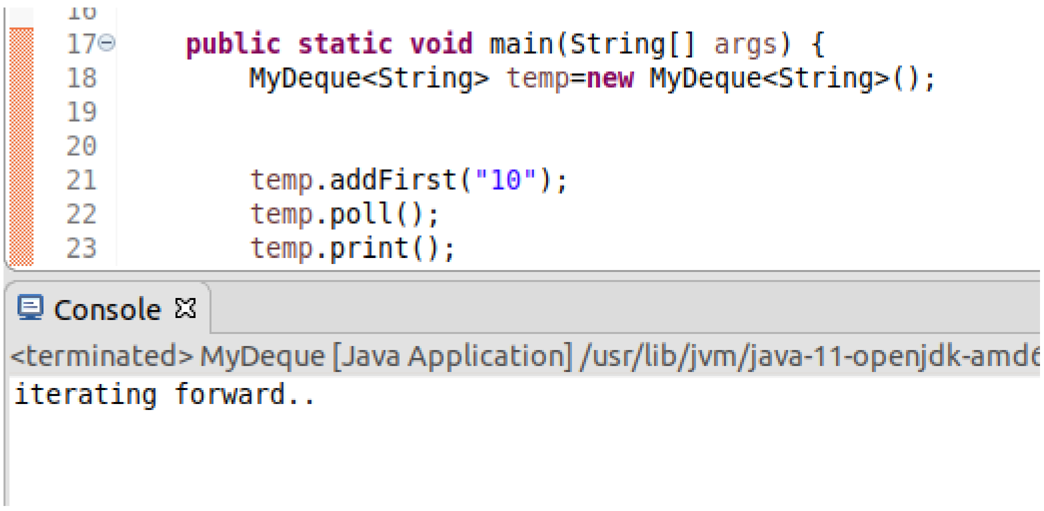
I firstly thought that If I make a linkedlist class then I can make list iterator first. Becouse Linkedlist class methods can easily implemented by using list iterator.Then I write first node class then list iterator after that linked list.I write linkedlist class becouse deque keeps linkedlists.Then I write deque methods.I write add method and poll method firstly becouse other methods mostly use these methods.My poll method is removing deque from linkedlist and then adding removed linked list by using next and prev of nodes.Then I write Iterator methods.To do this,I write own iterator class that implements Iterator<E>.So I thought that and I did.

Test Cases

	Task	Parameter	Analyze	Pass or not
1)	Use addFirst method then poll method	String “10”	Should add and poll	Pass
2)	Then use again addFirst and addLast	String “20” String “30”	Should add first and last side	Pass
3)	Use addFirst and poll then show removed list	String “10”	Should show removed list	Pass
4)	Then use addLast then show removed list becouse it must use removed one	String “20”	Should show removed list empty	Pass
5)	Use pollLast , peek, peekLast		Should do all methods	Pass
6)	Use Add a few element then Use iterator	String	Should show list	Pass
7)	Use descending iterator		Should show list descending order	Pass

Running Outputs and Commands

1)



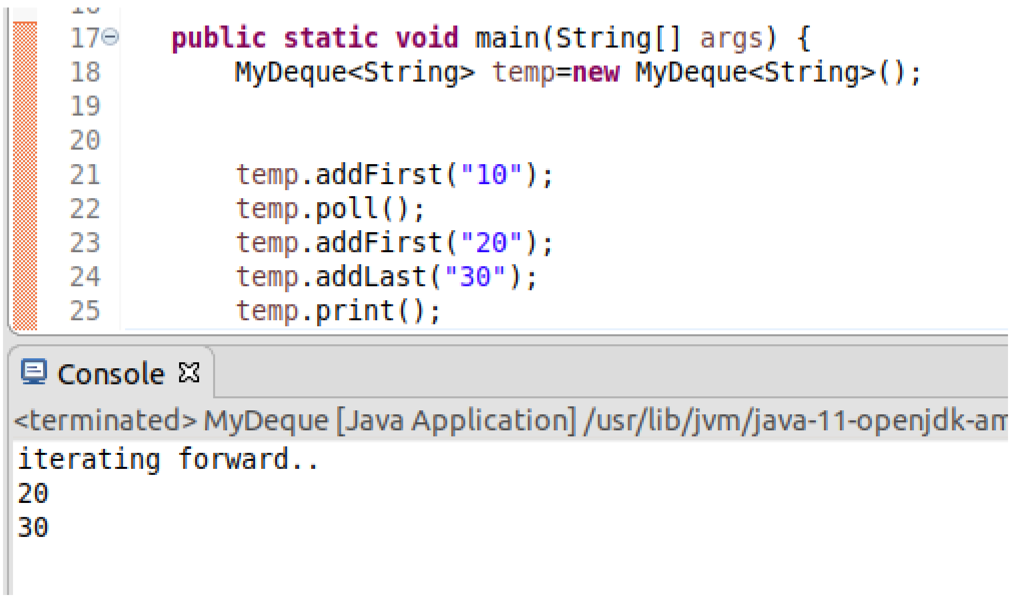
```
    public static void main(String[] args) {
        MyDeque<String> temp=new MyDeque<String>();


        temp.addFirst("10");
        temp.poll();
        temp.print();
```

Console

```
<terminated> MyDeque [Java Application] /usr/lib/jvm/java-11-openjdk-amd
iterating forward..
```

It firstly adding 10 after that it removing element .So list is empty.It works.

2)

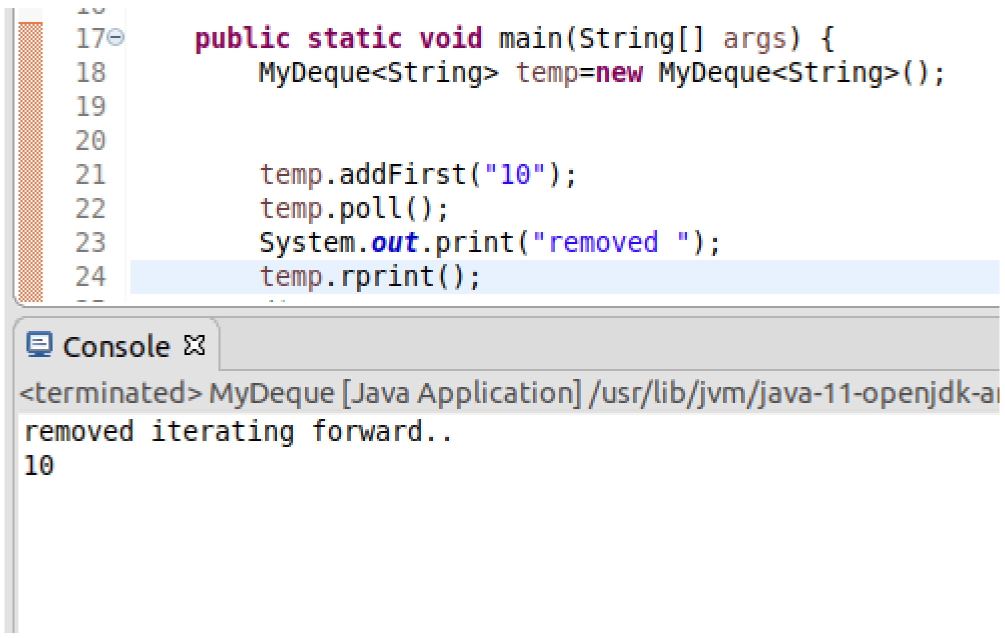


```
    public static void main(String[] args) {
        MyDeque<String> temp=new MyDeque<String>();


        temp.addFirst("10");
        temp.poll();
        temp.addFirst("20");
        temp.addLast("30");
        temp.print();
```

```
<terminated> MyDeque [Java Application] /usr/lib/jvm/java-11-openjdk-am
iterating forward..
20
30
```

It first add 10 then It removes after that It add at the first 20 and at the last 30.So it works.

3)

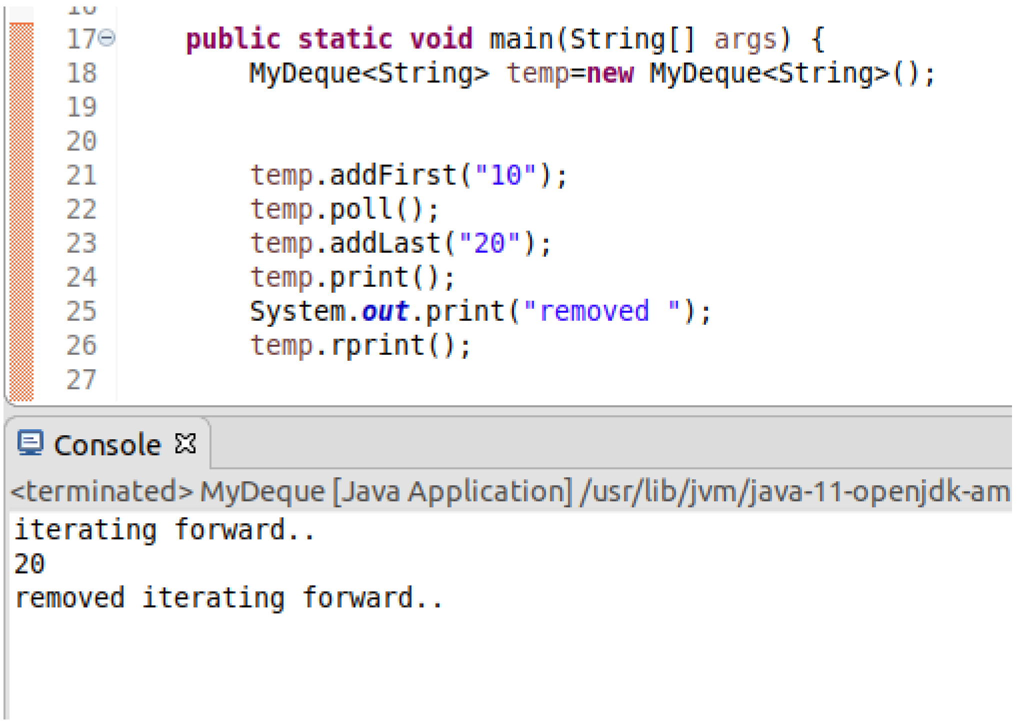


```
    public static void main(String[] args) {
        MyDeque<String> temp=new MyDeque<String>();


        temp.addFirst("10");
        temp.poll();
        System.out.print("removed ");
        temp.rprint();
```

```
<terminated> MyDeque [Java Application] /usr/lib/jvm/java-11-openjdk-a
removed iterating forward..
10
```

It firstly add 10 then it removes element.Therefore it is shown in removed list.So It works.

4)



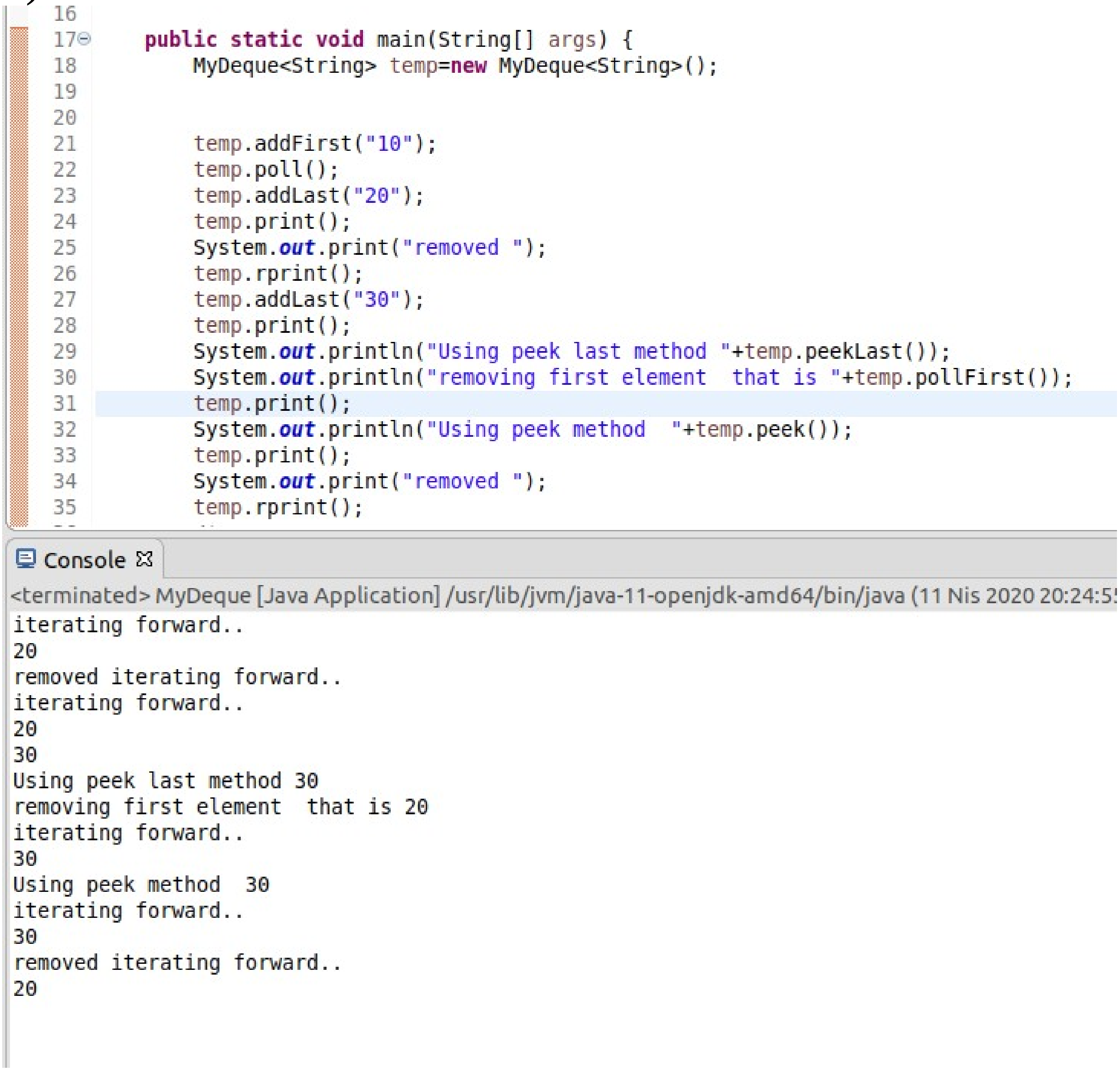
```
    public static void main(String[] args) {
        MyDeque<String> temp=new MyDeque<String>();


        temp.addFirst("10");
        temp.poll();
        temp.addLast("20");
        temp.print();
        System.out.print("removed ");
        temp.rprint();

```

Console

```
<terminated> MyDeque [Java Application] /usr/lib/jvm/java-11-openjdk-am
iterating forward..
20
removed iterating forward..
```

It adds element then it removes .After that it adds a element using removed list .So removed list must be empty and deque list is 20 .So it works.

5)

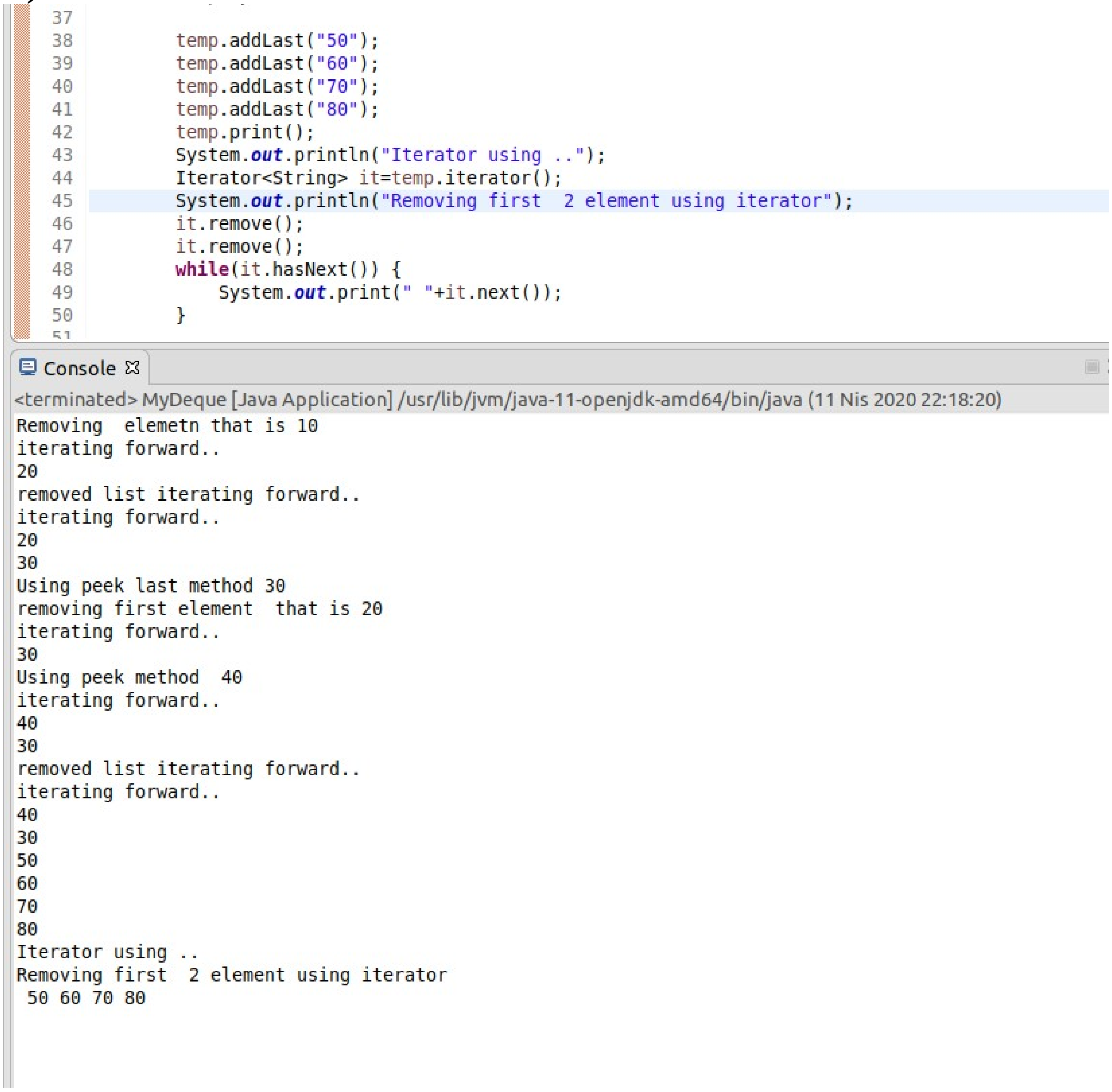


```java
    public static void main(String[] args) {
        MyDeque<String> temp=new MyDeque<String>();


        temp.addFirst("10");
        temp.poll();
        temp.addLast("20");
        temp.print();
        System.out.print("removed ");
        temp.rprint();
        temp.addLast("30");
        temp.print();
        System.out.println("Using peek last method "+temp.peekLast());
        System.out.println("removing first element  that is "+temp.pollFirst());
        temp.print();
        System.out.println("Using peek method  "+temp.peek());
        temp.print();
        System.out.print("removed ");
        temp.rprint();
```

```
Console
<terminated> MyDeque [Java Application] /usr/lib/jvm/java-11-openjdk-amd64/bin/java (11 Nis 2020 20:24:5
iterating forward..
20
removed iterating forward..
iterating forward..
20
30
Using peek last method 30
removing first element  that is 20
iterating forward..
30
Using peek method  30
iterating forward..
30
removed iterating forward..
20
```

Doing test 4 and adding element then using peeklast to show the last element.Then it use pollfirst then it prints screen.After that it uses peek method.Then shows list and removed list.So it works.

6)

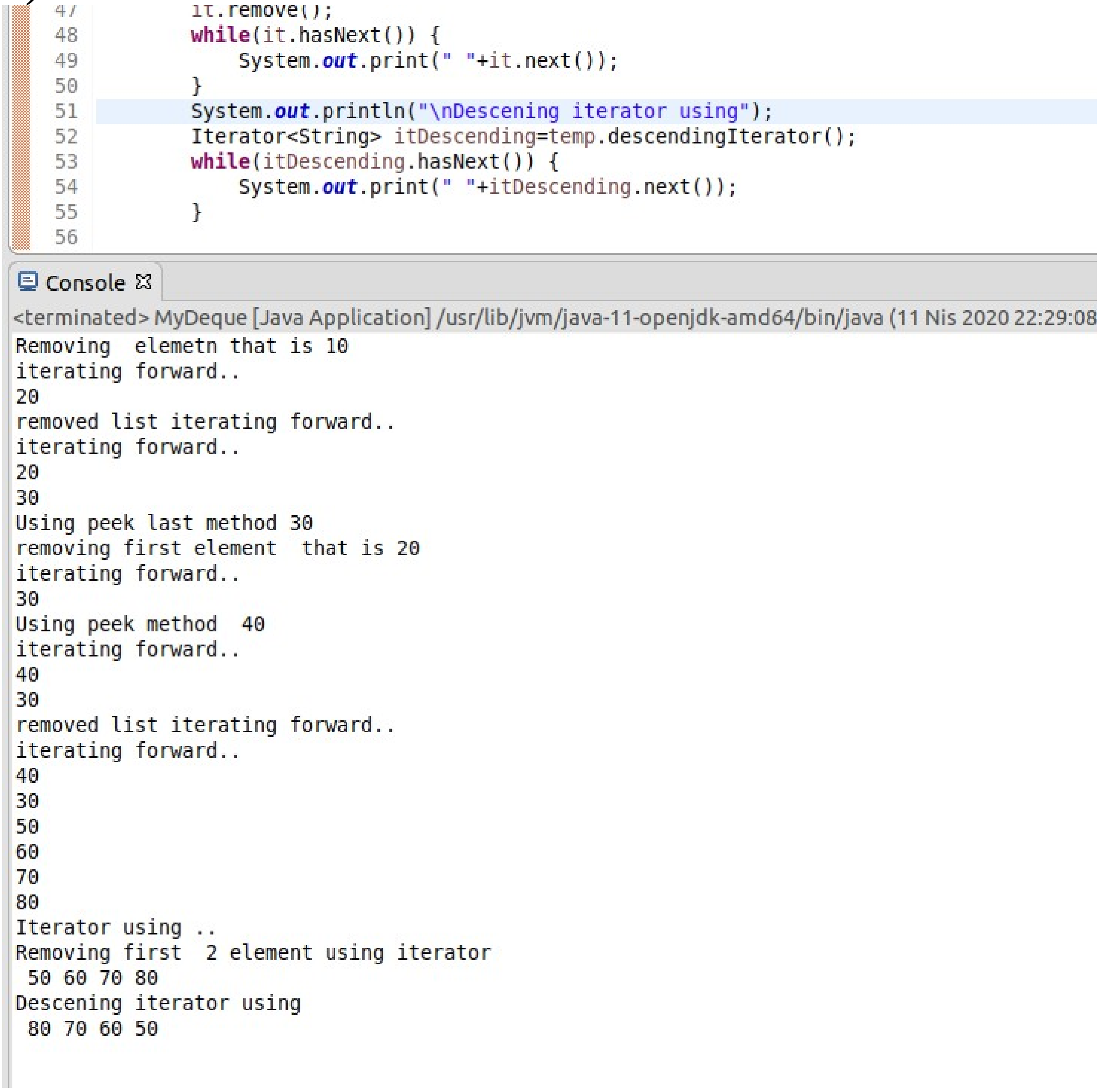


```
        temp.addLast("50");
        temp.addLast("60");
        temp.addLast("70");
        temp.addLast("80");
        temp.print();
        System.out.println("Iterator using ..");
        Iterator<String> it=temp.iterator();
        System.out.println("Removing first  2 element using iterator");
        it.remove();
        it.remove();
        while(it.hasNext()) {
            System.out.print(" "+it.next());
        }
```

```
<terminated> MyDeque [Java Application] /usr/lib/jvm/java-11-openjdk-amd64/bin/java (11 Nis 2020 22:18:20)
Removing  elemetn that is 10
iterating forward..
20
removed list iterating forward..
iterating forward..
20
30
Using peek last method 30
removing first element  that is 20
iterating forward..
30
Using peek method  40
iterating forward..
40
30
removed list iterating forward..
iterating forward..
40
30
50
60
70
80
Iterator using ..
Removing first  2 element using iterator
 50 60 70 80
```

It uses iterator to remove first two element and then shows list using iterator. It tests iterator methods.

7)



```
        it.remove();
        while(it.hasNext()) {
            System.out.print(" "+it.next());
        }
        System.out.println("\nDescening iterator using");
        Iterator<String> itDescending=temp.descendingIterator();
        while(itDescending.hasNext()) {
            System.out.print(" "+itDescending.next());
        }

```

Console

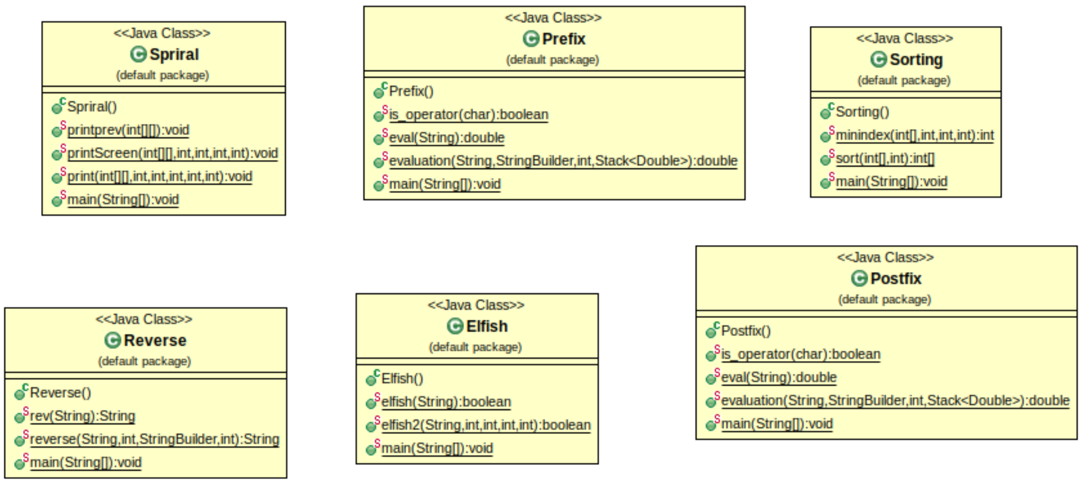
<terminated> MyDeque [Java Application] /usr/lib/jvm/java-11-openjdk-amd64/bin/java (11 Nis 2020 22:29:08

```
Removing  elemetn that is 10
iterating forward..
20
removed list iterating forward..
iterating forward..
20
30
Using peek last method 30
removing first element  that is 20
iterating forward..
30
Using peek method  40
iterating forward..
40
30
removed list iterating forward..
iterating forward..
40
30
50
60
70
80
Iterator using ..
Removing first  2 element using iterator
 50 60 70 80
Descening iterator using
 80 70 60 50
```

Uses descening iterator to show list.It works.

Q3)

Class Diagram



Problem Solution Approach

Before I start to write recursive codes , I thought base cases of recursive functions.Then I write base cases.After that I write recusive function to think recursively.So I did these for all.So , This is my solution approach.

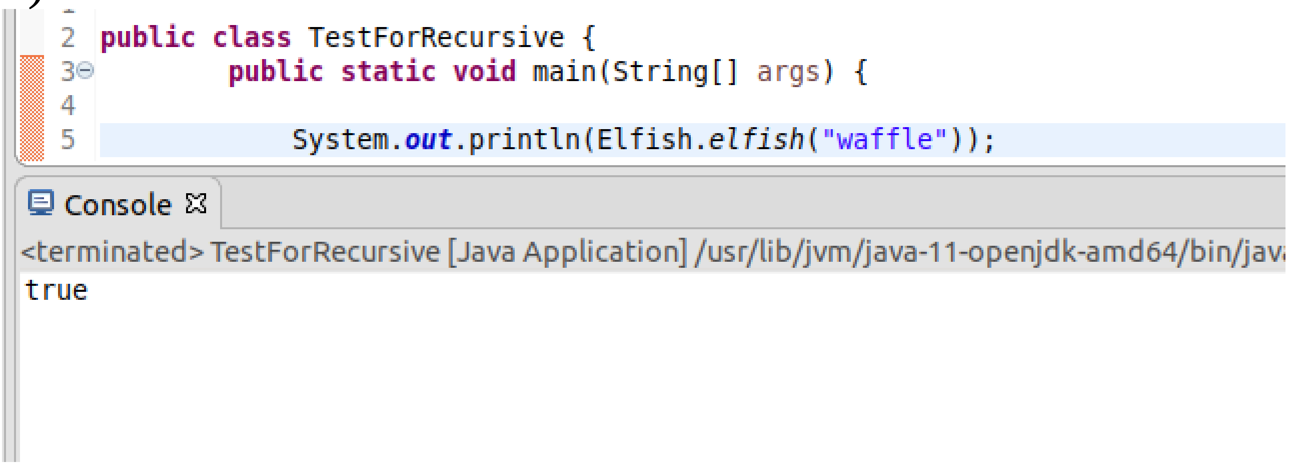
Test Cases

	Tasks	Parameter	Expected result	Pass or not
1)	Test elfish method with word waffle	“waffle”	Should return true	Pass
2)	Test elfish method with word mustafa	“mustafa”	Should return false	Pass
3)	Test postfix method with 2 3 4 * + 5 +	2 3 4 * + 5 +	Should return 19	Pass
4)	Test postfix method with 2 30 4 * + 5 +	2 30 4 * + 5 +	Should return 127	Pass
5)	Test prefix method with + + 2 / 4 4 5	+ + 2 / 4 4 5	Should return 8	Pass

6)	Test Sort method with array has a lot of number	Array has a lot of number	Should sort array	Pass
7)	Test Spiral with 2D array in the homework	2D array	Should print spiral	Pass
8)	Test reverse method with string	String	Should print reverse	Pass

Running Outputs

1)

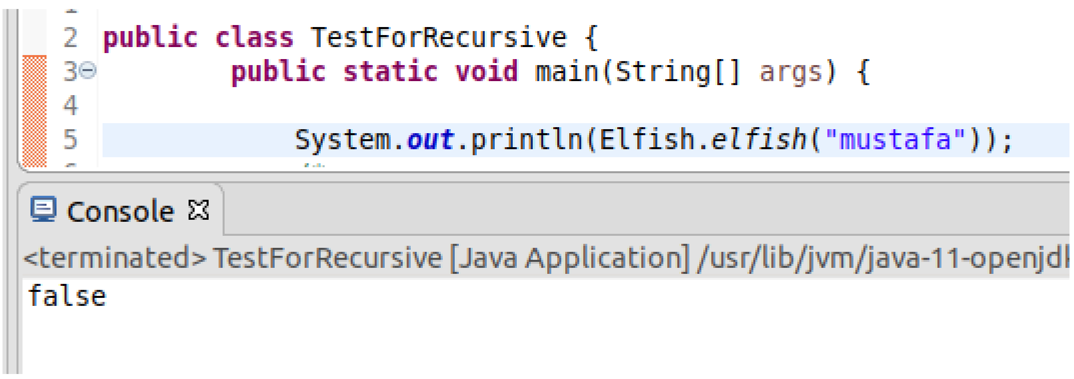


```
public class TestForRecursive {
        public static void main(String[] args) {

                System.out.println(Elfish.elfish("waffle"));
```

Console

```
<terminated> TestForRecursive [Java Application] /usr/lib/jvm/java-11-openjdk-amd64/bin/jav
true
```

2)

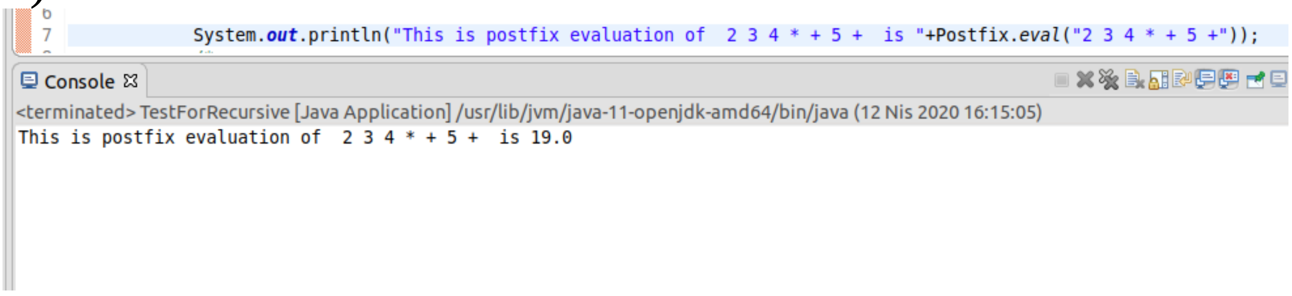


```
public class TestForRecursive {
        public static void main(String[] args) {

                System.out.println(Elfish.elfish("mustafa"));
```

Console

```
<terminated> TestForRecursive [Java Application] /usr/lib/jvm/java-11-openjdl
false
```

3)

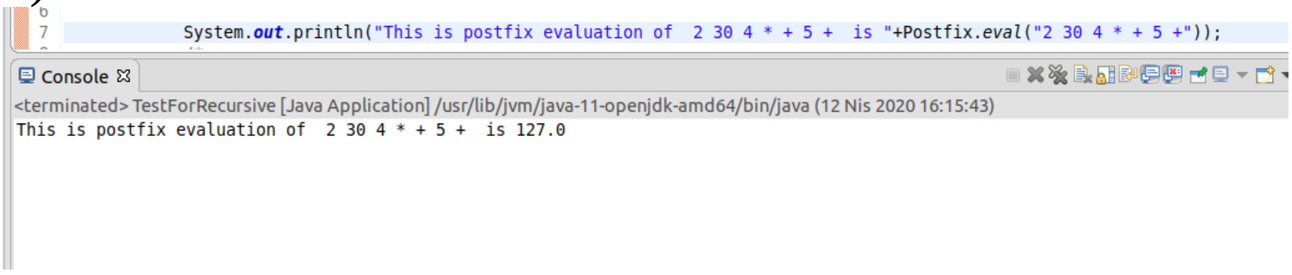


```
                System.out.println("This is postfix evaluation of  2 3 4 * + 5 +  is "+Postfix.eval("2 3 4 * + 5 +"));
```

Console

```
<terminated> TestForRecursive [Java Application] /usr/lib/jvm/java-11-openjdk-amd64/bin/java (12 Nis 2020 16:15:05)
This is postfix evaluation of  2 3 4 * + 5 +  is 19.0
```

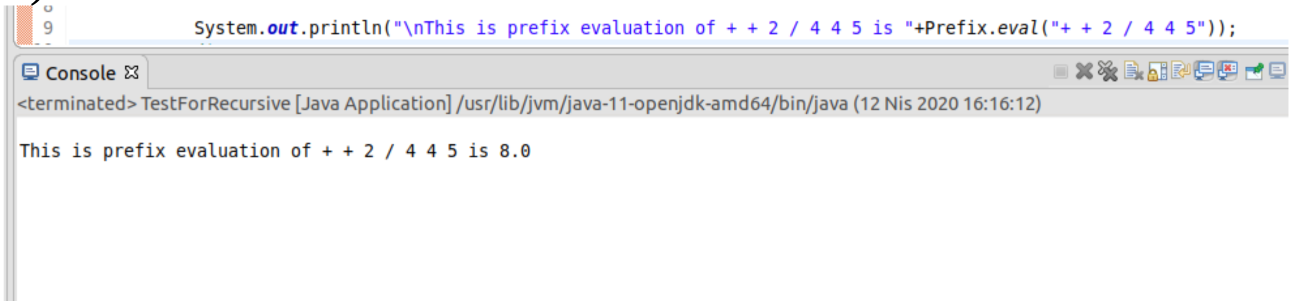
4)



```
System.out.println("This is postfix evaluation of  2 30 4 * + 5 +  is "+Postfix.eval("2 30 4 * + 5 +"));
```

```
<terminated> TestForRecursive [Java Application] /usr/lib/jvm/java-11-openjdk-amd64/bin/java (12 Nis 2020 16:15:43)
This is postfix evaluation of  2 30 4 * + 5 +  is 127.0
```

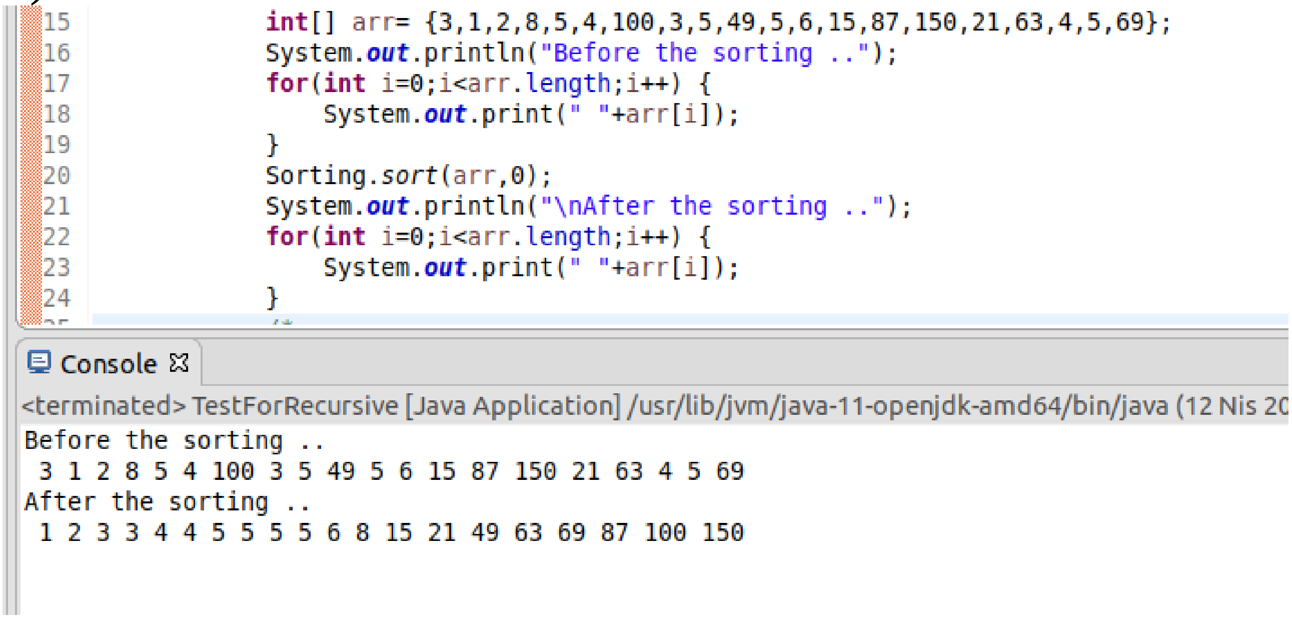
5)



```
System.out.println("\nThis is prefix evaluation of + + 2 / 4 4 5 is "+Prefix.eval("+ + 2 / 4 4 5"));
```

```
<terminated> TestForRecursive [Java Application] /usr/lib/jvm/java-11-openjdk-amd64/bin/java (12 Nis 2020 16:16:12)

This is prefix evaluation of + + 2 / 4 4 5 is 8.0
```

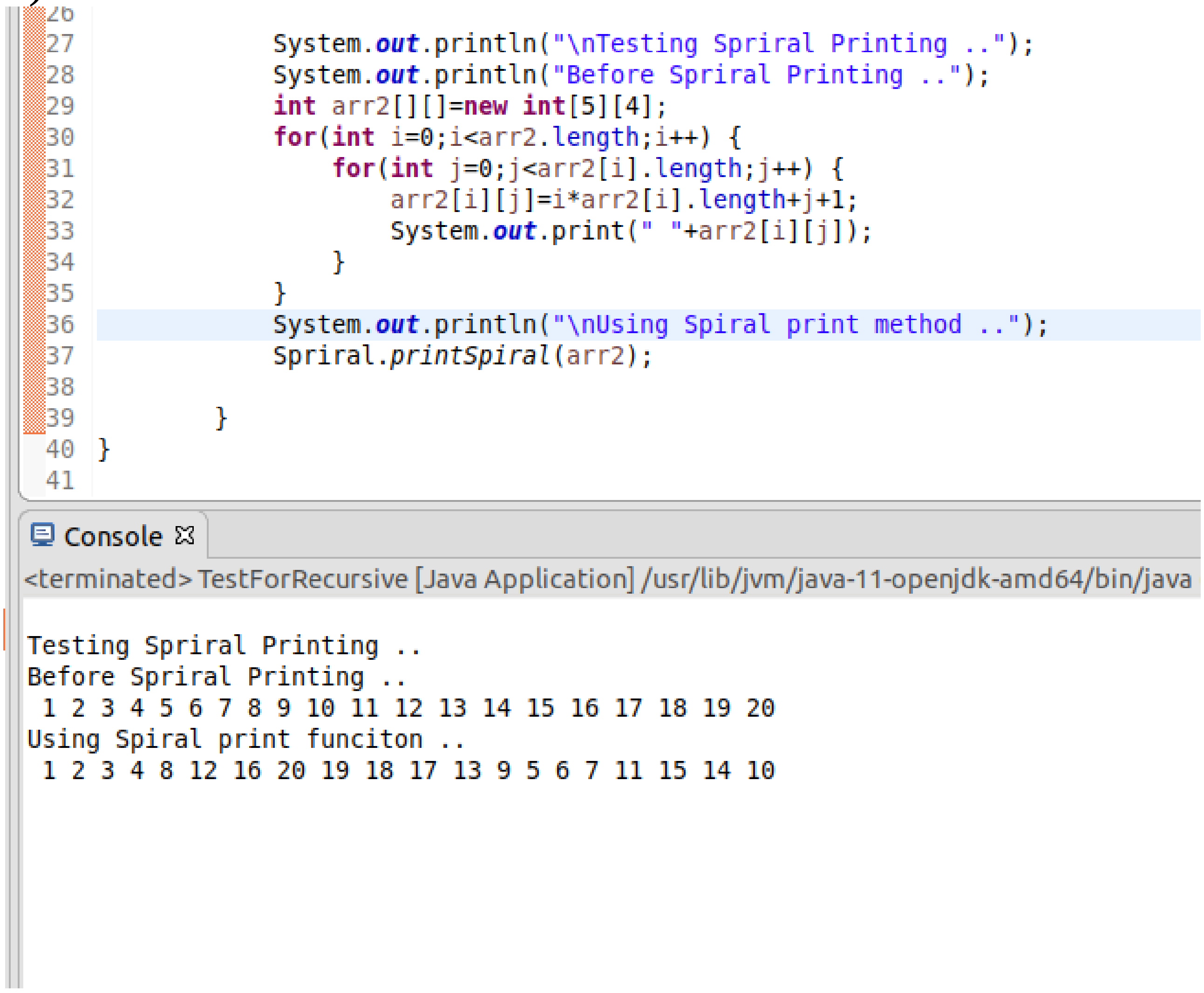
6)



```
int[] arr= {3,1,2,8,5,4,100,3,5,49,5,6,15,87,150,21,63,4,5,69};
System.out.println("Before the sorting ..");
for(int i=0;i<arr.length;i++) {
    System.out.print(" "+arr[i]);
}
Sorting.sort(arr,0);
System.out.println("\nAfter the sorting ..");
for(int i=0;i<arr.length;i++) {
    System.out.print(" "+arr[i]);
}
```

```
<terminated> TestForRecursive [Java Application] /usr/lib/jvm/java-11-openjdk-amd64/bin/java (12 Nis 20
Before the sorting ..
 3 1 2 8 5 4 100 3 5 49 5 6 15 87 150 21 63 4 5 69
After the sorting ..
 1 2 3 3 4 4 5 5 5 5 6 8 15 21 49 63 69 87 100 150
```

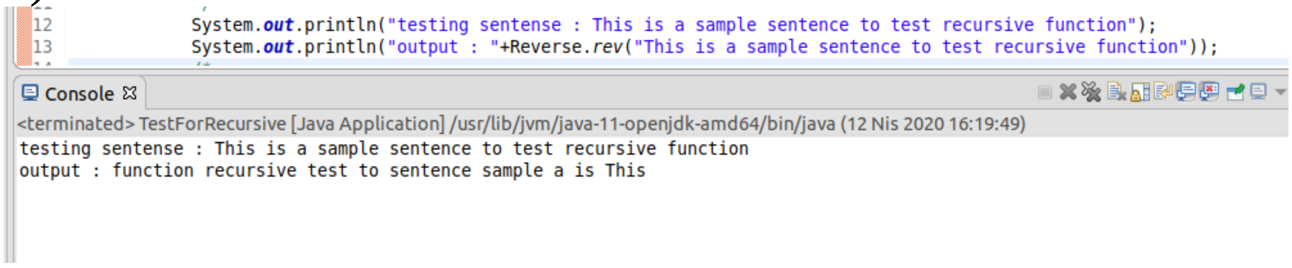
7)



```
            System.out.println("\nTesting Spriral Printing ..");
            System.out.println("Before Spriral Printing ..");
            int arr2[][]=new int[5][4];
            for(int i=0;i<arr2.length;i++) {
                for(int j=0;j<arr2[i].length;j++) {
                    arr2[i][j]=i*arr2[i].length+j+1;
                    System.out.print(" "+arr2[i][j]);
                }
            }
            System.out.println("\nUsing Spiral print method ..");
            Spriral.printSpiral(arr2);

        }
}
```

```
<terminated> TestForRecursive [Java Application] /usr/lib/jvm/java-11-openjdk-amd64/bin/java

Testing Spriral Printing ..
Before Spriral Printing ..
 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
Using Spiral print funciton ..
 1 2 3 4 8 12 16 20 19 18 17 13 9 5 6 7 11 15 14 10
```

8)



```
        System.out.println("testing sentense : This is a sample sentence to test recursive function");
        System.out.println("output : "+Reverse.rev("This is a sample sentence to test recursive function"));
```

```
<terminated> TestForRecursive [Java Application] /usr/lib/jvm/java-11-openjdk-amd64/bin/java (12 Nis 2020 16:19:49)
testing sentense : This is a sample sentence to test recursive function
output : function recursive test to sentence sample a is This
```

Explanation of Recursive Methods

Problem 1)

Base Case :

returns String
stops when incremented number and lenght of the given string

Small Problems :

To find character at given index
To insert a character at given index to new string

Combination :

I use charAt(int) method to find character one by one. Then If it equals space.I use insert(0,char) method to add at first and I make index to 0. Else I insert(index,char) ,Then I increment index. I traverse the sentence by incrementing a number that is firstly 0. If number is equal to sentence length It returns sb.toString() . Hence, I find reverse sentence from given sentence by using Stringbuilder.

Problem 2)

Base Cases :

return true If checks are 1.
returns false If length of word is equal to incremented number.

Small Problems :

To find character at given index

Combination :

I use charAt(int) method to check that It is equal to e,l,f or not. To do this I send 3 parameter to check 3 character and I increments number one by one.If all checks are 1 then it returns true. If length of word is equal to number,It returns false.

Problem 3)

Base Case :

returns array if array length is equal to incremented number.

Small Problems :

To find index of minimum number

Combination :

To find index of minimum number , I write a method that finds it recursively.After that I write sort method recursivly.I control all elements whether It is small the minimum element or not , If it is then I swap the numbers.After that I increment the number to traverse array.When number is equal to array length, It stops and returns array.

Problem 4)

Base Case :

returns double stack.pop() when if number is equal to -1

Small Problems :

To find a character at given index
To determine if a character is operator or not

Combination :

I send a number that is array length -1 to method.Then I decrement every call method.I check if a character is operator or not.If it is then I pop twice time from stack then I push number that is calculated.Then If character is space then I change string to double value.If character is not space then number occurs by using StringBuilder .If checking number is -1 then it returns result that is stack.pop().

Problem 5)

Base Case :

returns double stack.pop() when if number is equal to array length

Small Problems :

To find a character at given index
To determine if a character is operator or not
Changing String to Double

Combination :

I send a number that is 0 to method.Then I increment
every call method.I check if a character is operator or not.If it is then I pop twice time from stack then I push number that is calculated.Then If character is space then I change string to double value.If character is not space then number occurs by using StringBuilder .If checking number is array length then it returns result that is stack.pop().

Problem 6)

Base Case :

Stops when if top number is bigger than bottom number and left number is bigger than right number

printScreen method stops if start is equal to end+1 or start+1 is equal to end if selection is differ to 1 or 2.

Small Problems :

To print a point to another point

Combination :

Firstly,Direction is left to right so I send direction as a 0 .Then It uses printScreen method that is recursively printing numbers one way to another.It prints left to the right.Then I increment direction to traverse top to the bottom and increment top number we traverse all top up side.After that It prints top to the bottom.I decrement right number becouse we traverse all last right side.I increment direction also to traverse right to the left.Then it prints right to the left and decrement bottom number becouse we traverse all bottom down side.Then I increment direction to traverse bottom to the up.Then I did direction 0.So It prints spiral recursivly.The Reason Why I increment top and left number and decrement bottom and right number is that I traverse same number.I send 0 as top and as left and I send array length -1 as bottom and right.So , It will close to each other.When top is bigger than bottom and left is bigger than right it will return and stop.Also I use printScreen method to prints elements as a row or coulumn.

Mustafa Tokgöz
171044077