Student ID: 202112047

IT623 - Lab Assignment 4

1. **Create stack class with the following properties**.

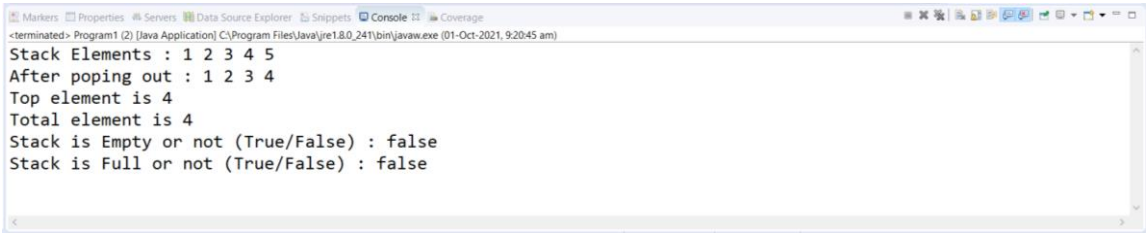
   Code:

```
public class Program1 {
	int arr[];
	int top;
	int capacity;

	Program1(int size) {
		arr = new int[size];
		capacity = size;
		top = -1;
	}

	public void push(int data) {
		if (top == capacity) {
			System.out.println("Stack Overflow");
		} else {
			arr[++top] = data;
		}

	}

	public int pop() {
		if (top == -1) {
			System.out.println("Stack Underflow");
		}
		return top--;
```

```
}

public int peek() {
        if (top == -1) {
                System.out.println("Stack Underflow");
        } else {
                return arr[top];
        }
        return 0;
}

public int size() {
        return top + 1;
}

public Boolean isEmpty() {
        return top == -1;
}

public Boolean isFull() {
        return top == capacity - 1;
}

public void print() {
        for (int i = 0; i <= top; i++) {
                System.out.print(arr[i] + " ");
        }
}

public static void main(String args[]) {
        Program1 p1 = new Program1(5);
        p1.push(1);
        p1.push(2);
        p1.push(3);
```

```
            p1.push(4);
            p1.push(5);

            System.out.print("Stack Elements : ");
            p1.print();

            p1.pop();
            System.out.print("\nAfter poping out : ");
            p1.print();

            System.out.print("\nTop element is " + p1.peek());

            System.out.print("\nTotal element is " + p1.size());

            System.out.print("\nStack is Empty or not (True/False) : " +
            p1.isEmpty());

            System.out.print("\nStack is Full or not (True/False) : " + p1.isFull());
        }
}
```

Output Snapshot:



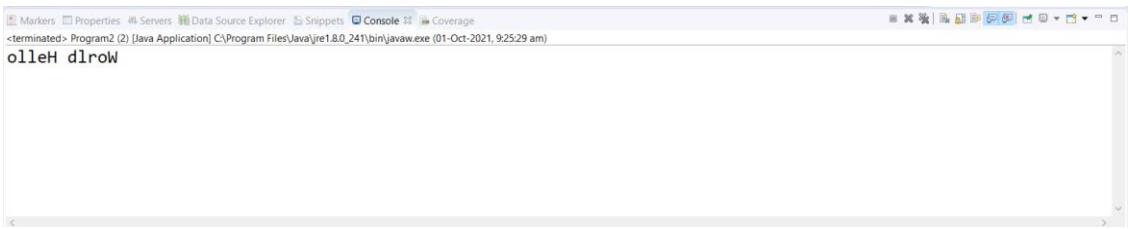
2. **Given string str, we need to print reverse of individual words.**

Code:

```
class Stack1 {
	char a[] = new char[100];
	int top = -1;

	void push(char c) {
		a[++top] = c;
	}

	char pop() {
		return a[top--];
	}

	boolean isEmpty() {
		return (top == -1) ? true : false;
	}

	char peek() {
		return a[top];
	}

}

public class Program2 {
	static Stack1 st = new Stack1();

	static void reverseWords(String str)
	{
	    for (int i = 0; i < str.length(); ++i) {
	        if (str.charAt(i) != ' ')
	            st.push(str.charAt(i));
```

```
            else {
                while (st.isEmpty() == false) {
                    System.out.print(st.pop());

                }
                System.out.print(" ");
            }
        }

        while (st.isEmpty() == false) {
            System.out.print(st.pop());

        }
    }

    public static void main(String[] args) {
            String str = "Hello World";

            reverseWords(str);
    }
}
```

Output Snapshot:



3. **Given an array, print the Next Greater Element (NGE) for every element. The Next greater Element for an element x is the first greater element on the right side of x in the array. Elements for which no greater element exists, consider the next greater element as -1.**

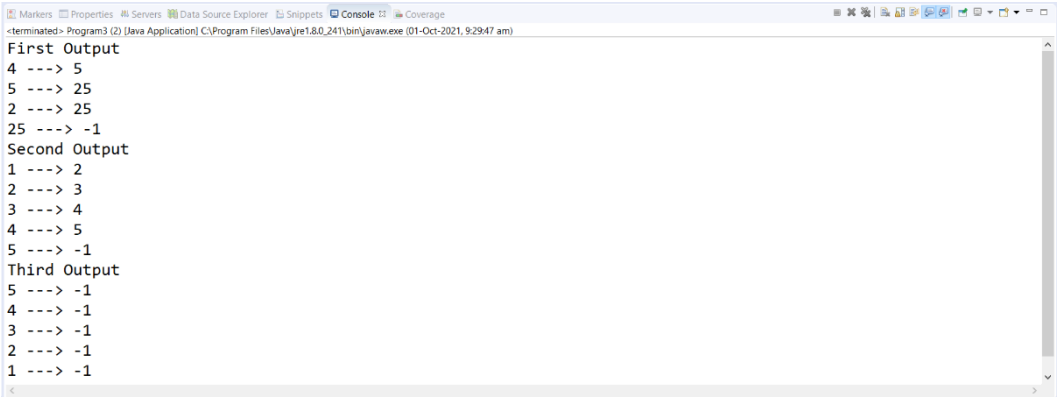
Code:

```
public class Program3 {
	static class stack {
		int top;
		int items[] = new int[50];

		void push(int x) {
			if (top == 49) {
				System.out.println("Stack Overflow");
			} else {
				items[++top] = x;
			}
		}

		int pop() {
			if (top == -1) {
				System.out.println("Stack Underflow");
				return -1;
			} else {
				int element = items[top];
				top--;
				return element;
			}
		}

		int peek() {
			return items[top];
		}
```

```
        boolean isEmpty() {
                return (top == -1) ? true : false;
        }
}

static void print(int arr[], int n) {

        stack s = new stack();
        s.top = -1;

        int arr1[] = new int[n];

        for (int i = n - 1; i >= 0; i--) {
                while (!s.isEmpty() && s.peek() <= arr[i])
                        s.pop();

                if (s.isEmpty())
                        arr1[i] = -1;
                else
                        arr1[i] = s.peek();

                s.push(arr[i]);
        }

        for (int i = 0; i < n; i++)
                System.out.println(arr[i] + " ---> " + arr1[i]);
}

public static void main(String[] args) {
        int arr1[] = { 4, 5, 2, 25 };
        int n1 = arr1.length;
        System.out.println("First Output");
        print(arr1, n1);
```

```
            int arr2[] = { 1, 2, 3, 4, 5 };
            int n2 = arr2.length;
            System.out.println("Second Output");
            print(arr2, n2);

            int arr3[] = { 5, 4, 3, 2, 1 };
            int n3 = arr3.length;
            System.out.println("Third Output");
            print(arr3, n3);
        }
}
```

Output Snapshot:



4. Write a program to convert infix expression to postfix expression.

Code:

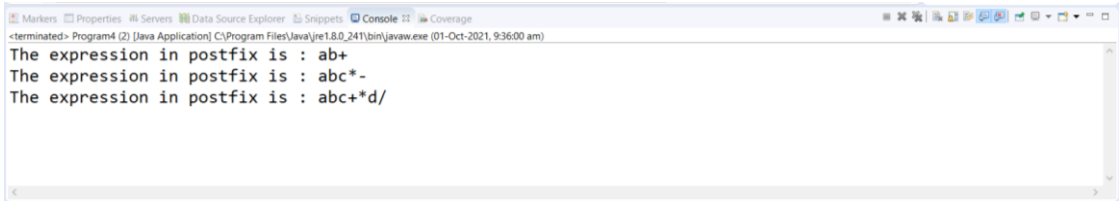
```
class Stack {
        char a[] = new char[100];
        int top = -1;

        void push(char c) {
                a[++top] = c;
```

```
	}

	char pop() {
		return a[top--];
	}

	boolean isEmpty() {
		return (top == -1) ? true : false;
	}

	char peek() {
		return a[top];
	}

}

public class Program4 {

	static Stack operators = new Stack();

	public static String toPostfix(String infix) {
		char symbol;
		String postfix = "";

		for (int i = 0; i < infix.length(); ++i) {
			symbol = infix.charAt(i);

			if (Character.isLetter(symbol))
				postfix += symbol;
			else if (symbol == '(') {
				operators.push(symbol);
			} else if (symbol == ')') {
				while (operators.peek() != '(') {
					postfix += operators.pop();
```

```
                }
                operators.pop();
            } else {
                while (!operators.isEmpty() && !(operators.peek() ==
                '(') && prec(symbol) <= prec(operators.peek()))
                    postfix += operators.pop();

                operators.push(symbol);
            }
        }
        while (!operators.isEmpty())
            postfix += operators.pop();
        return postfix;
}

static int prec(char x) {
        if (x == '+' || x == '-')
            return 1;
        if (x == '*' || x == '/' || x == '%')
            return 2;
        return 0;
}

public static void main(String argv[]) {
        String infix1 = "a+b";
        System.out.println("The expression in postfix is : " +
        toPostfix(infix1));

        String infix2 = "a-b*c";
        System.out.println("The expression in postfix is : " +
        toPostfix(infix2));

        String infix3 = "a*(b+c)/d";
```

```
            System.out.println("The expression in postfix is : " +
            toPostfix(infix3));

        }
}
```

Output Snapshot:



5. **Write a program to implement two stacks with a single array. Create separate push() and pop() for both the stacks and perform a set of push and pop such that overflow, underflow, successful push, as well as successful pop operation, takes place for both the stacks. Note: Utilize the entire space of the array.**

Code:

```
class Program5 {
        int size;
        int top1, top2;
        int arr[];

        Program5(int n) {
                arr = new int[n];
                size = n;
                top1 = -1;
```

```
        top2 = size;
}

void push1(int x) {
        if (top1 < top2 - 1) {
                top1++;
                arr[top1] = x;
        } else {
                System.out.println("Stack 1 Overflow");
        }
}


void push2(int x) {
        if (top1 < top2 - 1) {
                top2--;
                arr[top2] = x;
        } else {
                System.out.println("Stack 2 Overflow");
        }
}

int pop1() {
        if (top1 >= 0) {
                int x = arr[top1];
                top1--;
                return x;
        } else {
                System.out.println("Stack 1 Underflow");
                System.exit(0);
        }
        return 0;
}
```

```
int pop2() {
        if (top2 < size) {
                int x = arr[top2];
                top2++;
                return x;
        } else {
                System.out.println("Stack 2 Underflow");
                System.exit(0);
        }
        return 0;
}

void print_stack1() {
        int i;
        System.out.print("Stack 1 : ");
        for (i = top1; i >= 0; --i) {
                System.out.print(arr[i]+" ");
        }
}

void print_stack2() {
        int i;
        System.out.print("\nStack 2 : ");
        for (i = top2; i < size; ++i) {
                System.out.print(arr[i]+" ");
        }
}

public static void main(String args[]) {
        Program5 p5 = new Program5(10);

        p5.push1(5);
        p5.push1(10);
```

```
            p5.push1(11);
            p5.push1(13);
            p5.push1(20);

            p5.push2(7);
            p5.push2(12);
            p5.push2(15);
            p5.push2(22);
            p5.push2(27);
            p5.push2(40);

            p5.print_stack1();
            p5.print_stack2();

            System.out.println("\nPopped element from" + " stack 1 is " +
            p5.pop1());
            System.out.println("Popped element from" + " stack 2 is " +
            p5.pop2());
        }
}
```

Output Snapshot: