Name	**Virinchi Sadashiv Shettigar**
UID no.	**2021300118**
Experiment No.	**10**

AIM:	Programs on Packages. Write a program to demonstrate packages.
Program 1	
PROBLEM STATEMENT:	Create a package with class Reverse_String. Write a function called ReversIt() that reverses a string. It swaps the first and last characters, then the second and next-to-last characters, and so on. The string should be passed to reversit() as an argument. Write a program to exercise reversit(). Class Check get a string from the user, call reversit(), and print out the result. Use an input method that allows embedded blanks. Test the program with Napoleon's famous phrase, "Able was I ere I saw Elba."

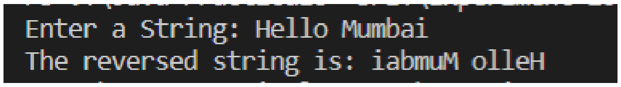
PROGRAM:

```
 package mypack;
public class reverse_string {
    public static String ReverseIt(String str) {
        int len = str.length();
        int l = len;
        if(len%2!=0) {
            len = len/2 + 1;
        }
        else {
            len = len/2;
        }
        char[] str1 = str.toCharArray();
        char temp;
        for(int i=0;i<len;i++) {
            temp = str1[i];
            str1[i] = str1[l-i-1];
            str1[l-i-1] = temp;
        }
        str = new String(str1);
        return str;
    }
}

import java.util.*;
import mypack.reverse_string;
```

```
public class Main {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        String str = new String();
        System.out.print("Enter a String: ");
        str = sc.nextLine();
        System.out.print("The reversed string is: 
"+reverse_string.ReverseIt(str));
        sc.close();
    }
}
```

RESULT:



```
Enter a String: Hello Mumbai
The reversed string is: iabmuM olleH
```

Program 2

PROBLEM STATEMENT:

A Package implements stack operations:

a.Push b. Pop

Write a user defined exception to check whether the stack is full or empty.

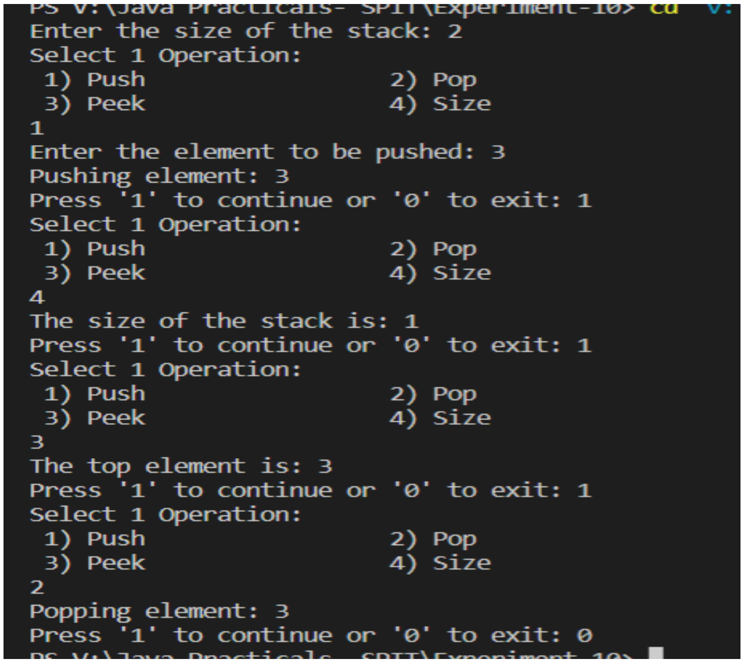
PROGRAM:

```
            import java.util.Scanner;
import mypack2.*;
public class StackC {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter the size of the stack: ");
        int size = sc.nextInt();
        Stack s = new Stack(size);
        int flag,choice;
        while(true) {
            System.out.println("Select 1 Operation:\n 1) Push\t\t 2) Pop\n 3) 
Peek\t\t 4) Size");
            choice = sc.nextInt();
            switch(choice) {
                case 1:
                    System.out.print("Enter the element to be pushed: ");
                    int e = sc.nextInt();
                    s.push(e);
                    break;
                case 2:
                    s.pop();
                    break;
                case 3:
                    if(s.peek()!=-1) {
                        System.out.println("The top element is: "+s.peek());
                    }
                    break;
                case 4:
                    System.out.println("The size of the stack is: "+s.size());
                    break;
                default:
                    System.out.println("Invalid choice");
                    break;
            }

            System.out.print("Press '1' to continue or '0' to exit: ");
            flag = sc.nextInt();
            if(flag == 0) {
                break;
            }
        }
        sc.close();
    }
}

package mypack2;
public class Stack {
    int[] stack;
    int top;
    int capacity;
    public Stack(int size) {
        stack = new int[size];
        capacity = size;
        top = -1;
    }
    public void push(int e) {
        if(isFull()) {
            System.out.println("Stack is full\nPush operation failed");
        } else {
            System.out.println("Pushing element: "+e);
            stack[++top] = e;
        }
    }
    public void pop() {
        if(isEmpty()) {
            System.out.println("Stack is empty\nPop operation failed");
        } else {
            System.out.println("Popping element: "+stack[top--]);
        }
    }
    public int peek() {
        if(!isEmpty()) {
            return stack[top];
        }
        else {
            System.out.println("Stack is empty\nPeek operation failed");
            return -1;
        }
    }
    public boolean isEmpty() {
        return top == -1;
    }
    public boolean isFull() {
        return top == capacity - 1;
    }
    public int size() {
        return top+1;
    }
}
```

RESULT:



```
PS V:\Java Practicals- SPIT\Experiment-10> cd 'V:
Enter the size of the stack: 2
Select 1 Operation:
 1) Push                    2) Pop
 3) Peek                    4) Size
1
Enter the element to be pushed: 3
Pushing element: 3
Press '1' to continue or '0' to exit: 1
Select 1 Operation:
 1) Push                    2) Pop
 3) Peek                    4) Size
4
The size of the stack is: 1
Press '1' to continue or '0' to exit: 1
Select 1 Operation:
 1) Push                    2) Pop
 3) Peek                    4) Size
3
The top element is: 3
Press '1' to continue or '0' to exit: 1
Select 1 Operation:
 1) Push                    2) Pop
 3) Peek                    4) Size
2
Popping element: 3
Press '1' to continue or '0' to exit: 0
PS V:\Java Practicals- SPIT\Experiment-10> 
```

CONCLUSION:	In this experiment, we learned about the packages and how we use packages to avoid name conflicts and to write better maintainable code.