Name	**Hatim Yusuf Sawai**
UID no.	**2021300108**
Experiment No.	**10**

AIM:	**To make packages in java, implement them in a program and observe different types of Access Modifiers in java.**
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 Code:**
	package *mypackage;*
	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;
    }
}
```

Program Code:

```
import java.util.*;
import mypackage.Reverse_string;
public class Check {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        String str = new String();
        System.out.println("Enter a String: ");
        str = sc.nextLine();
        System.out.println("The reversed string is:
"+Reverse_string.ReverseIt(str));
        sc.close();
    }
}
```

RESULT:

```
PS D:\Java Practicals\Experiment_10> cd "d:\Java Pr
Enter a String:
Able was I ere I saw Elba
The reversed string is: ablE was I ere I saw elbA
PS D:\Java Practicals\Experiment_10>
```

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:	**Package Code:**
	package *mypack*;
	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;
    }
}
```

Program Code:

```
import java.util.Scanner;
import mypack.*;
public class StackCheck {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.println("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:\n1. Push\t\t2. Pop\n3.
Peek\t\t4. Size");
            choice = sc.nextInt();
            switch(choice) {
                case 1:
                    System.out.println("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.println("Do you want to continue?\n1. Yes\t2. No");
        flag = sc.nextInt();
        if(flag == 2) {
            break;
        }
    }
  }
}
```

RESULT:

Push & Pop:

```
Enter the size of the stack:
4
Select 1 Operation:
1. Push         2. Pop
3. Peek         4. Size
1
Enter the element to be pushed:
532
Pushing element: 532
Do you want to continue?
1. Yes  2. No
1
Select 1 Operation:
1. Push         2. Pop
3. Peek         4. Size
2
Popping element: 532
```

Exception Test Cases:

```
Do you want to continue?
1. Yes  2. No
1
Select 1 Operation:
1. Push         2. Pop
3. Peek         4. Size
2
Stack is empty
Pop operation failed
Do you want to continue?
1. Yes  2. No
1
Select 1 Operation:
1. Push         2. Pop
3. Peek         4. Size
1
Enter the element to be pushed:
567
Stack is full
Push operation failed
Do you want to continue?
1. Yes  2. No
```

CONCLUSION:	In this experiment, we learnt how to build our own package and use access modifiers to differentially access methods and classes outside the package. We also learnt how to implement a Stack in java using arrays with push & pop operations.