CSE 2001: Data Structure & Algorithms

Programming Assignment-VI

(Stack)

Stack is an ordered set of elements in which insertion and deletion are from one end of the stack called the **top** of the stack. It is a Last In First Out structure (**LIFO**).

PART-I

Static Implementation (Array Implementation)

A stack is implemented statically by using an array of size **MAX** to hold stack elements and an integer **top** storing the position of top of the stack. The stack elements can be integers, characters, strings or user defined data types.

The operations to be performed on a stack are

`public static int push(int S[],int top)` – adding an element x to the stack S

`public static int pop(int S[],int top)`– deletes and returns the top element from the stack S

`public static void display(int S[],int top)-` display all the elements of Stack S

`public static boolean isEmpty(int top)` – check if the stack is empty

`public static boolean isFull(int top)` – check if the stack is full when top equals MAX -1

Write a menu driven Java Program using class, methods and array, to construct a **Stack and implement the above five operations.**

The template for menu driven java program to use the above Stack and invoke the required methods to perform different operations is given below.

```
import java.util.Scanner;
public class StackDemo1 {

	public static int push(int S[],int top)
	{

	 -----
	 ----
	}

/* Write the code for remaining user defined methods*/



	public static final int MAX=10;
	public static void main(String[] args) {
		Scanner sc=new Scanner(System.in);
		 int stack[]=new int[MAX];
		 int top=-1;
		  while(true)
		  {

		    System.out.println("***MENU***");
		      System.out.println("0: Exit");
		      System.out.println("1: Push");
		      System.out.println("2: Pop");
		      System.out.println("3: Display");
		      System.out.println("Enter your choice");
		      int choice=sc.nextInt();
		      switch(choice)
		       {
		            case 0:
		                        System.exit(0);
		            case 1:
		                        top=push(stack,top);
		                        break;
		            -----
		            -----

		            default:
		                  System.out.println("Invalid choice");
		       }
		  }
	}

}
```

PART-II

Dynamic Implementation (Linked List Implementation)

A stack is implemented dynamically by using a Linked list where each node in the linked list has two parts, the data element and the reference to the next element of the stack. The class defintion of Node is given below.

```
class Node
{
      int info;
      Node next;
}
```

The stack elements can be integers, characters, strings or user defined data types. There is no restriction on how big the stack can grow.

The operations to be performed on a stack are

`public static Node push(Node top) -` Add an element x to the stack S requires creation of node containing x and putting it in front of the top node pointed by S.

`public static Node pop(Node top)-` Delete the top node from the stack S so that next element becomes the top.

`public static void display(Node top)-` Display all the elements of Stack S.

Write a menu driven Java Program using class, methods and list, to construct a **Stack and implement the above three operations.**

The code template for constructing the above Stack and performing the required operation is given below.

```
public class StackDemo2 {

	public static Node push(Node top)
	{

		----
		----
	}

	/* Write the code for remaining user defined methods*/



	public static void main(String[] args) {

		Scanner sc=new Scanner(System.in);
		Node top;
		top=null;

		while(true)
		{
			System.out.println("****MENU****");
			System.out.println("0:Exit");
			System.out.println("1:Push");
			System.out.println("2:Pop");
			System.out.println("3:Display");
			System.out.println("Enter your choice");
			int choice=sc.nextInt();
			switch(choice)
			{
			case 0:
				System.exit(0);

			case 1:
				top=push(top);

			-----
			-----

			default:
				System.out.println("Wrong choice");


			}
		}

	}

}
```

****