DSA

Module 10

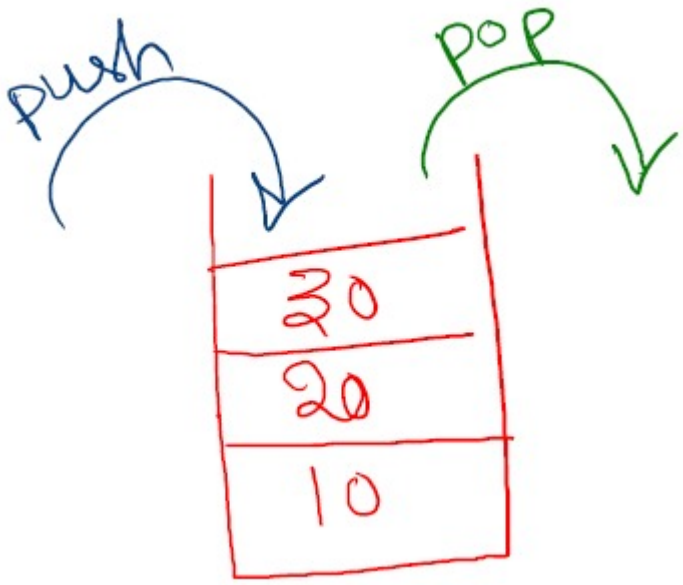
Stacks

Author
Srinivas Dande

10. Introduction to Stacks

- Stack is an ADT (Abstract Data Type).
- Stack is a **Linear Data Structure**.
- Stack follow the Strategy called **LIFO (Last In First Out)**.
- Stack is Closed at One End and Open at other end for the Operations
  i.e Bottom of the Stack is closed and Top of the Stack is open for the Operations.

- You can do the Insert and Delete Operations on the Top End of the Stack
- Insert Operation on the Stack is called as **Push**
- Delete Operation on the Stack is called as **Pop**



- Stacks can be implemented using **Arrays and LinkedLists**

> A stack is an ordered list in which insertion and deletion are done at one end, called top. The last element inserted is the first one to be deleted. Hence, it is called the Last in First out (LIFO) or First in Last out (FILO) list.

10.1. Stack Operations

a) isEmpty()
b) size()
c) push(int)
d) pop()
e) peek()

a) isEmpty()

- Returns True if the Stack is Empty otherwise false.

b) size()

- Returns the size of the stack

c) push(object)

- Inserts the Element to the Top of the Stack

d) pop()

- Returns the Top Element and Removes that from the Stack

e) peek()

- Returns the Top Element from the Stack without Remving

10.2. Corner Conditions

a) Stack Underflow
b) Stack Overflow

a) Stack Underflow

- When pop() or peek() is called on the empty Stack.

b) Stack Overflow

- When push() is called on the full Stack.

10.3. Time Complexity of Stack Operations

Operation	Arrays	Dynamic Arrays	LinkedList
push()	O(1)	Amortized O(1)	O(1)
pop()	O(1)	O(1)	O(1)
peek()	O(1)	O(1)	O(1)

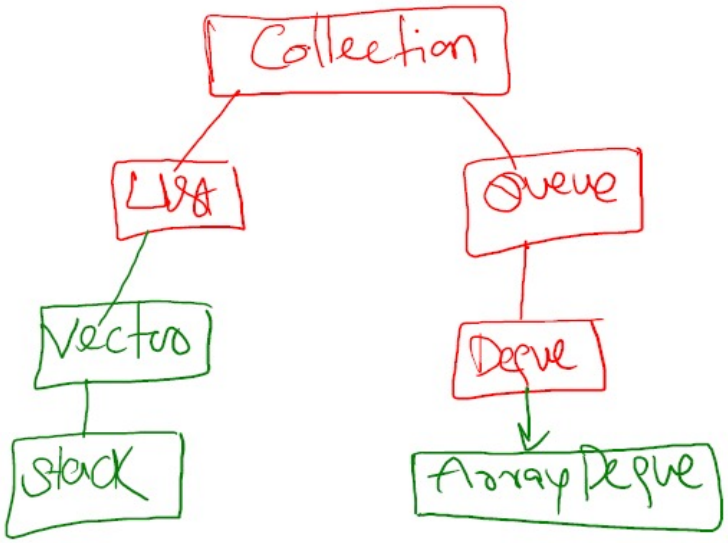
10.4. Applications of Stacks

- Function Calls
- Balanced Parenthesis
- Reversing Items
- Infix to Postfix
- Infix to Prefix
- Evaluation of Postfix
- Evaluation of Prefix
- Stock Span Problems
- Undo/Redo
- Forward/backword

etc

10.5. Stacks in Java Collection

- Java Collections has two classes for Stack use-cases.
    a) Stack
    b) ArrayDeque



a) Stack

- Stack is a legay class from java.util package.
- All the methods of Stack are synchronized means Internal synchronization is provided
- Use the Stack class in multi-threaded environment

a) ArrayDeque

- ArrayDeque is a latest class from java.util package.
- All the methods of Stack are not synchronized means Internal synchronization is not provided
- Use the ArrayDeque class in single-threaded environment

- If you want to use ArrayDeque in multi-threaded environment then you need to provide external synchronization.
- More complete and consistent set of LIFO stack operations is provided in ArrayDeque class
- So Use ArrayDeque in preference to Stack class.
- Most ArrayDeque operations run in amortized O(1).
- This class is likely to be faster than Stack when used as a stack

10.5.1. Using Stack and ArrayDeque API

Lab1.java

```
package com.jlcindia.stacks;

/*
* @Author : Srinivas Dande
* @Company: Java Learning Center
* */

import java.util.ArrayDeque;
import java.util.Deque;

public class Lab1 {

        public static void main(String[] args) {

        Deque<Integer> mystack= new ArrayDeque<>();
        //Stack<Integer> mystack=new Stack<>();

        System.out.println(mystack);
        System.out.println(mystack.size());
        System.out.println(mystack.isEmpty());
```

```
        mystack.push(10);
        mystack.push(20);
        mystack.push(30);
        mystack.push(40);
        mystack.push(50);

        System.out.println(mystack);
        System.out.println(mystack.size());
        System.out.println(mystack.isEmpty());

        System.out.println(mystack.peek()); //50

        mystack.pop();
        mystack.pop();

        System.out.println(mystack);
        System.out.println(mystack.size());
        System.out.println(mystack.isEmpty());

        System.out.println(mystack.peek()); //30

        }
}
```

10.6. Stack Implementation using Arrays

StackFullException.java

```
package com.jlcindia.stacks.arrays1;
/*
* @Author : Srinivas Dande
* @Company: Java Learning Center
* */

public class StackFullException extends RuntimeException {

	@Override
	public String toString() {
		return "StackFullException [No Push Please ]";
	}
}
```

StackFullException.java

```
package com.jlcindia.stacks.arrays1;
/*
* @Author : Srinivas Dande
* @Company: Java Learning Center
* */

public class StackEmptyException extends RuntimeException {

	@Override
	public String toString() {
		return "StackEmptyException [No pop or peek Please ]";
	}

}
```

MyStack.java

```
package com.jlcindia.stacks.arrays1;
/*
* @Author : Srinivas Dande
* @Company: Java Learning Center
* */

public class MyStack {

	private Integer myarray[];
	private int capacity;
	private int top;

	public MyStack(int capacity) {
		this.capacity = capacity;
		this.top = -1;
		this.myarray = new Integer[capacity];
	}

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

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

	public void push(Integer element) {

		if (top == capacity - 1) {
			throw new StackFullException();
		}

		top++;
		myarray[top] = element;
	}
```

```
	public Integer pop() {

		if (top == - 1) {
			throw new StackEmptyException();
		}
		int element = myarray[top];
		myarray[top] = null;
		top--;
		return element;
	}

	public Integer peek() {

		if (top == - 1) {
			throw new StackEmptyException();
		}
		int element = myarray[top];
		return element;
	}

	public String toString() {
		String str = "[";
		if (top != -1) {
			for (Integer x : myarray) {
				if (x != null)
					str = str + x + ",";
			}

		}
		str = str + "]";
		return str;
	}
}
```

Lab2.java

```
package com.jlcindia.stacks.arrays1;
/*
* @Author : Srinivas Dande
* @Company: Java Learning Center
* */

public class Lab2 {

	public static void main(String[] args) {

	MyStack mystack=new MyStack(5);

	System.out.println(mystack);
	System.out.println(mystack.size());
	System.out.println(mystack.isEmpty());

	mystack.push(10);
	mystack.push(20);
	mystack.push(30);
	mystack.push(40);
	mystack.push(50);
	//mystack.push(99);

	System.out.println(mystack);
	System.out.println(mystack.size());
	System.out.println(mystack.isEmpty());


	System.out.println(mystack.peek()); //50

	mystack.pop();
	mystack.pop();

	System.out.println(mystack);
	System.out.println(mystack.size());
	System.out.println(mystack.isEmpty());
```

```
        System.out.println(mystack.peek()); //30

        mystack.pop();
        mystack.pop();
        mystack.pop();
        //mystack.pop();

        System.out.println(mystack);
        System.out.println(mystack.size());
        System.out.println(mystack.isEmpty());

        }
}
```

10.7. Stack Implementation using Arrays

MyStack.java

```
package com.jlcindia.stacks.arrays2;
/*
* @Author : Srinivas Dande
* @Company: Java Learning Center
* */

import java.util.EmptyStackException;

public class MyStack {

	private Integer myarray[];
	private int capacity;
	private int top;

	public MyStack(int capacity) {
		this.capacity = capacity;
		this.top = -1;
		this.myarray = new Integer[capacity];
	}
	public int capacity() {
		return capacity;
	}
	public int size() {
		return top + 1;
	}

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

```
public void push(Integer element) {

        if (top == capacity - 1) {
                //Resize the Array
                int newCapacity= this.capacity * 2;
                Integer mynewArray[] = new Integer[newCapacity];
                System.arraycopy(myarray, 0, mynewArray, 0, capacity);
                myarray=mynewArray;
                capacity=newCapacity;
        }

        top++;
        myarray[top] = element;

}

public Integer pop() {

        if (top == - 1) {
                throw new EmptyStackException();
        }
        int element = myarray[top];
        myarray[top] = null;
        top--;
        return element;
}

public Integer peek() {

        if (top == - 1) {
                throw new EmptyStackException();
        }
        int element = myarray[top];
        return element;
}
```

```
	public String toString() {
		String str = "[";
		if (top != -1) {
			for (Integer x : myarray) {
				if (x != null)
					str = str + x + ",";
			}

		}
		str = str + "]";
		return str;
	}
}
```

Lab3.java

```
package com.jlcindia.stacks.arrays2;
/*
* @Author : Srinivas Dande
* @Company: Java Learning Center
* */
public class Lab3 {
public static void main(String[] args) {

	MyStack mystack=new MyStack(5);

	System.out.println(mystack);
	System.out.println(mystack.size());
	System.out.println(mystack.isEmpty());
	System.out.println(mystack.capacity());//5

	mystack.push(10);
	mystack.push(20);
	mystack.push(30);
	mystack.push(40);
	mystack.push(50);
```

```
        System.out.println(mystack);
        System.out.println(mystack.size());
        System.out.println(mystack.isEmpty());
        System.out.println(mystack.capacity());//5

        mystack.push(60);
        System.out.println(mystack);
        System.out.println(mystack.size());
        System.out.println(mystack.isEmpty());
        System.out.println(mystack.capacity()); //10

        }
}
```

10.8. Stack Implementation using LinkedLists

Node.java

```
package com.jlcindia.stacks.linkedlist;
/*
* @Author : Srinivas Dande
* @Company: Java Learning Center
* */

public class Node {

        int data;
        Node next;

        Node(int data) {
                this.data = data;
                this.next = null;
        }
}
```

MyStack.java

```
package com.jlcindia.stacks.linkedlist;

/*
* @Author : Srinivas Dande
* @Company: Java Learning Center
* */

import java.util.EmptyStackException;

public class MyStack {

	private Node headNode;
	private int size;

	public MyStack() {
		this.headNode = null;
		this.size = 0;
	}

	public int size() {
		return this.size;
	}

	public boolean isEmpty() {
		return this.headNode == null;
	}

	public void push(int element) {
		Node temp = new Node(element);
		temp.next=headNode;
		headNode=temp;
		size++;
	}
```

```
public Integer pop() {

        if(headNode==null) {
                throw new EmptyStackException();
        }

        int element = headNode.data;

        headNode = headNode.next;
        size--;

        return element;
}

public Integer peek() {

        if(headNode==null) {
                throw new EmptyStackException();
        }

        int element = headNode.data;

        return element;
}

public String toString() {
        if (this.headNode == null) {
                return "[]";
        }

        String str = "[";
        Node currentNode = this.headNode;
        while (currentNode != null) {
                str = str + "" + currentNode.data + " , ";
                currentNode = currentNode.next;
        }
```

```
                str = str.substring(0, str.length() - 2);
                str = str + "]";

                return str;
        }

}
```

Lab4.java

```
package com.jlcindia.stacks.linkedlist;
/*
* @Author : Srinivas Dande
* @Company: Java Learning Center
* */

public class Lab4 {

        public static void main(String[] args) {

        MyStack mystack=new MyStack();

        System.out.println(mystack);
        System.out.println(mystack.size());
        System.out.println(mystack.isEmpty());

        mystack.push(10);
        mystack.push(20);
        mystack.push(30);
        mystack.push(40);
        mystack.push(50);

        System.out.println(mystack);
        System.out.println(mystack.size());
        System.out.println(mystack.isEmpty());
        System.out.println(mystack.peek()); //50
```

```
        mystack.pop();
        mystack.pop();

        System.out.println(mystack);
        System.out.println(mystack.size());
        System.out.println(mystack.isEmpty());
        System.out.println(mystack.peek()); //30

        }
}
```

10.9. Two Stacks Implementation in One Array

MyStack.java

```
package com.jlcindia.twostacks;
/*
* @Author : Srinivas Dande
* @Company: Java Learning Center
* */

public class TwoStacks {

        private Integer myarray[];
        private int capacity;
        private int top1;
        private int top2;

        public TwoStacks(int capacity) {
                this.capacity = capacity;
                this.top1 = -1;
                this.top2 = capacity;
                this.myarray = new Integer[capacity];
        }
```

```
public boolean push1(Integer element) {

        if (top1 < top2 - 1) {
                top1++;
                myarray[top1] = element;
                return true;
        }

        return false;
}

public boolean push2(Integer element) {

        if (top1 < top2 - 1) {
                top2--;
                myarray[top2] = element;
                return true;
        }

        return false;
}

public Integer pop1() {
        if (top1 > -1) {
                int element = myarray[top1];
                myarray[top1] = null;
                top1--;
                return element;
        }
        return null;
}

public Integer pop2() {
        if (top2 < capacity) {
                int element = myarray[top2];
                myarray[top2] = null;
                top2++;
```

```
                return element;
            }
            return null;
        }

        public Integer peek1() {
            if (top1 > -1) {
                int element = myarray[top1];
                return element;
            }
            return null;
        }

        public Integer peek2() {
            if (top2 < capacity) {
                int element = myarray[top2];
                return element;
            }
            return null;
        }

        public int size1() {
            return top1 + 1;
        }

        public int size2() {
            return capacity - top2;
        }

        public void printStacks() {
            for (int i = 0; i < capacity; i++) {
                System.out.print(myarray[i] + " ");
            }
            System.out.println();
        }
}
```

Lab5.java

```
package com.jlcindia.twostacks;
/*
* @Author : Srinivas Dande
* @Company: Java Learning Center
* */

public class Lab5 {

	public static void main(String[] args) {

	TwoStacks mystack=new TwoStacks(5);

	mystack.printStacks();
	System.out.println(mystack.size1());
	System.out.println(mystack.size2());

	mystack.push1(10);
	mystack.push1(20);
	mystack.push2(99);
	mystack.push2(88);
	mystack.push2(77);

	mystack.push1(30);

	System.out.println("-------------------------");
	mystack.printStacks();
	System.out.println(mystack.size1());
	System.out.println(mystack.size2());
	System.out.println(mystack.peek1());
	System.out.println(mystack.peek2());

	mystack.pop1();
	mystack.pop2();
	mystack.pop2();
```

```
        System.out.println("-------------------------");
        mystack.printStacks();
        System.out.println(mystack.size1());
        System.out.println(mystack.size2());
        System.out.println(mystack.peek1());
        System.out.println(mystack.peek2());
        }

}
```

10.10. Palindome with Stacks

Lab6.java

```
package com.jlcindia.stacks;
/*
* @Author : Srinivas Dande
* @Company: Java Learning Center
* */
import java.util.ArrayDeque;
import java.util.Deque;

public class Lab6 {

        static boolean isPalimdrome(String str) {
                Deque<Character> mystack= new ArrayDeque<>();
                StringBuilder sb=new StringBuilder(str.length());
                str = str.toLowerCase();
                for(int i=0;i<str.length();i++) {
                        char ch= str.charAt(i);
                                if(ch>='a' && ch<='z') {
                                        sb.append(ch);
                                        mystack.push(ch);
                                }
                }
```

```
            StringBuilder revSb=new StringBuilder(str.length());
            while(!mystack.isEmpty()) {
                  revSb.append(mystack.pop());
            }

            return sb.toString().equals(revSb.toString());

      }

      public static void main(String[] args) {
            String str = "ab@$cb%3a";
            boolean x= isPalimdrome(str);
            System.out.println(x);
      }

}
```