Stack:

- The stack is a linear data structure that is used to store the collection of objects.
- It is based on Last-In-First-Out (LIFO).
- The stack data structure has the two most important operations that are push and pop.
- The push operation inserts an element into the stack and pop operation removes an element from the top of the stack.
- Stack only defines the default constructor, which creates an empty stack.
- Stack includes all the methods defined by Vector, and adds several of its own.

```
import java.util.*;

class Collection6
{
   public static void main(String[] args)
   {

      // Only no argument constructor for Stack
      Stack<Integer> s1 = new Stack<>();

      System.out.println("Default capacity of Stack: "+s1.capacity());
      //Default capacity of Stack: 10

 // 1. Object push(Object element) Pushes the element onto the stack. Element is also returned.

      System.out.println("Element 1 added is: "+s1.push(12));
      System.out.println("Element 2 added is: "+s1.push(10));
      System.out.println("Element 3 added is: "+s1.push(14));
      System.out.println("Element 3 added is: "+s1.push(17));
      System.out.println("Element 3 added is: "+s1.push(22));

      System.out.println("s1 is: "+s1);//s1 is: [12, 10, 14, 17, 22]


// 2. Object pop( )- Returns the element on the top of the stack, removing it in the process.
```

```
        System.out.println("Top most element removed is: "+s1.pop());
        //Top most element removed is: 22

        System.out.println("Updated s1 after pop() is: "+s1);
        //Updated s1 after pop() is: [12, 10, 14, 17]

// 3. Object peek( ) - Returns the element on the top of the stack, but does not remove it.

        System.out.println("Top most element is: "+s1.peek());
        //Top most element is: 17

        System.out.println("Updated s1 after peek() is: "+s1);
        //Updated s1 after peek() is: [12, 10, 14, 17]

// 4. int search(Element e) - Searches for element in the stack. If found, its offset from the top of stack is returned, else -1is returned.

        System.out.println("Position of element 17 from top is: "+s1.search(17));
        //Position of element 17 from top is: 1

        System.out.println("Position of element 10 from top is: "+s1.search(+10));
       // Position of element 10 from top is: 3

        System.out.println("Position of element 21 from top is: "+s1.search(21));
        //Position of element 21 from top is: -1

        // 5. Iterator iterator() - Fetching value of element

        Iterator itr = s1.iterator();
        while(itr.hasNext())
        {
           System.out.println(itr.next());
        }
    }
}
```