Ορισμός της Στοίβας μέσω Διασύνδεσης

```
public interface Stack {
    public int size();
        // Returns the size of the Stack
    public boolean isEmpty();
        // Returns true if the Stack is empty
    public boolean isFull();
        // Returns true if the Stack is full
    public Object top() throws StackEmptyException;
        // Returns the top item of the Stack
    public void push(Object item) throws StackFullException;
        // Adds a new item into the Stack
    public Object pop() throws StackEmptyException;
        // Removes the top item of the Stack
}
```

```
public class ArrayStack implements Stack {
   ...
```


Δημιουργία Στοίβας από φοιτητές

```
public class StackManagement {

    public static void main(String[] args) {
    ArrayStack st = new ArrayStack(10);
    st.push(new Student(181001, "Nikos"));
    st.push(new Student(181015, "Anna"));
    st.push(new Student(181032, "Kostas"));

    System.out.println("The size of stack is "+st.size());
    System.out.println(st.isEmpty());

    int size=st.size();
    for(int i=0; i<size; i++){
        Student S= (Student)st.pop();
        System.out.println("Student: "+ S.getOnoma()+ ", AM: "+S.getAM());
        }

    System.out.println("The size of stack is "+st.size());
    System.out.println(st.isEmpty());
    }
}
```

Μετάπτωση τύπου Student σε Object (up-casting)

Μετάπτωση τύπου Object σε Student (down-casting)

Έλεγχος Παρενθέσεων με τη βοήθεια Στοίβας

```
public class Brackets {

    public static void main(String args[]) {
        String sentence = "(x+y))*((a+(b+c)-z))";
        System.out.println(simpleBracketsCheck(sentence));
    }

    public static boolean simpleBracketsCheck(String expression) {
        ArrayStack CheckStack = new ArrayStack(10);
        char charToCheck;

        for(int i = 0; i < expression.length(); i++)
        {
           charToCheck = expression.charAt(i);
           if (charToCheck=='(') CheckStack.push('(');
           if (charToCheck==')') {
               if (CheckStack.isEmpty())
                  { System.out.println("Error at position : " + (i));
                    return false;
                  }
               else CheckStack.pop( );
           }
```


Έλεγχος Παρενθέσεων με τη βοήθεια Στοίβας

```
    if(CheckStack.isEmpty())
    {
        System.out.println("No error ");
        return true;
    }
    else
    {
          System.out.println("Error at position : " + expression.length());
          return false;
    }
}
```

Έλεγχος Παρενθέσεων με τη βοήθεια Στοίβας

```
public static void main(String args[ ])
 {
      String sentence = "{[(a+(b+c))]}";
  //  String sentence = "{[(a+(b]+c))]}";
  //  String sentence = "{[(a+(b+c))}";
      System.out.println(bracketsCheck(sentence));
 }

public static boolean bracketsCheck(String expression)
 {
      ArrayStack CheckStack = new ArrayStack(10);
      char charToCheck;
```


Έλεγχος Παρενθέσεων με τη βοήθεια Στοίβας

```
for(int i = 0; i < expression.length(); i++)
    {
      charToCheck = expression.charAt(i);
      switch(charToCheck) {
        case '(':
          CheckStack.push('('); break;
        case '{':
          CheckStack.push('{'); break;
        case '[':
          CheckStack.push('['); break;
        case ')':
          if(CheckStack.isEmpty() || !(CheckStack.pop().equals('(')))
            {  System.out.println("Error at position : " + (i));
               return false;
            }
          break;
```

Έλεγχος Παρενθέσεων με τη βοήθεια Στοίβας

```
case '}':
        if(CheckStack.isEmpty() || !(CheckStack.pop().equals('{')))
          {   System.out.println("Error at position : " + (i));
              return false;
           }
        break;
 case ']':
        if(CheckStack.isEmpty() || !(CheckStack.pop().equals('[')))
          {   System.out.println("Error at position : " + (i));
              return false;
           }
        break;
  default:
      System.out.println("character " + charToCheck +" not used");
   } //end of switch
} //end of for
```


Έλεγχος Παρενθέσεων με τη βοήθεια Στοίβας

```
if(CheckStack.isEmpty())
      {
         System.out.println("No error ");
         return true;
      }
      else
      {
          System.out.println("Error at position: "+
                                              expression.length());
          return false;
      }
   }
}
```

Ορισμός της Ουράς μέσω Διασύνδεσης

```
public interface Queue {

    public int size( );
  // επιστρέφει το μέγεθος (αριθμός στοιχείων) της ουράς
    public boolean isEmpty( );
  // αληθεύει εάν η ουρά είναι κενή
    public Object front( ) throws QueueEmptyException;
  // επιστρέφει το στοιχείο που βρίσκεται στο εμπρός
  // μέρος της ουράς
    public void enqueue(Object item) throws QueueFullException;
  // εισάγει ένα νέο στοιχείο στο πίσω μέρος της ουράς
    public Object dequeue( ) throws QueueEmptyException;
  // εξάγει και επιστρέφει το στοιχείο που βρίσκεται
  // στο εμπρός μέρος της ουράς
}
```

```
public class ArrayQueue implements Queue {
  ...
```


Δημιουργία Ουράς από φοιτητές

```
public class QueueApp {

    public static void main(String[] args) {
    ArrayQueue queue1 = new ArrayQueue(10);
    queue1.enqueue(new Student(181001, "Nikos"));
    queue1.enqueue(new Student(181015, "Anna"));
    queue1.enqueue(new Student(181032, "Kostas"));

    int size=queue1.size();
    for(int i=0; i<size; i++){
        Student S= (Student)queue1.dequeue();
        System.out.println("Student: "+ S.getOnoma()+ ", AM: "+S.getAM());
        queue1.enqueue(S);
        }
    System.out.println("The queue has "+queue1.size()+ " items");
    }
}
```

Μετάπτωση τύπου Student σε Object (up-casting)

Μετάπτωση τύπου Object σε Student (down-casting)