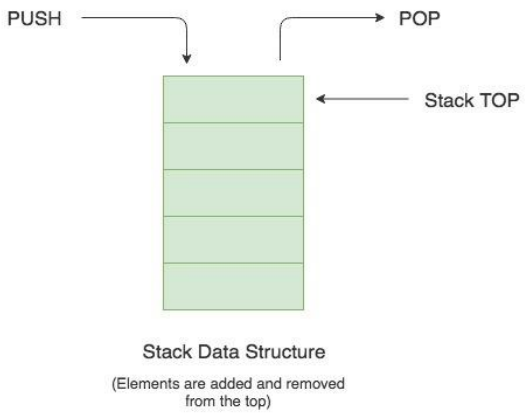
Opgave 3: Gebruik van een stack

Het doel van de opdracht was het leren gebruiken van een stack. We moesten een programma schrijven die nakijk wanneer haakjes opengingen in een teskt/code en of deze wel in de juiste volgorde werden gesloten.

Het gebruiken van een stack

Om het programma te schrijven moeten we eerst weten hoe we een stack gebruiken. Hier waren er 4 methode vooral belangrijk. De eerste is empty, deze method zegt simpelweg via een boolean of de stack leeg is of niet. De volgende method is push. Push wordt gebuikt om een element boven aan de stack toe te voegen. Met de method peek krijgen we terug wat er bovenaan de stack staat. De laatste method die voor ons van belang was was pop. Deze doet het tegenovergestelde als push en zal het element bovenaan de stack er uitgooien. De afbeelding hieronder geeft deze 2 methods duidelijk weer.



Het programma

De klasse die ik eerst schreef was de parser, deze zal nakijken of we een haakje tegenkomen en deze toevoegen of van de stack halen.

```java
package com.company;

import java.util.Stack;

public class Parser {
    private Stack<Character> StackBracket;

    public Parser ()
    {
        StackBracket = new Stack();

    }

    public boolean Parse(String testString)
    {

        for(char c : testString.toCharArray())
        {
            if(c=='(' || c=='[' || c=='{')
                StackBracket.push(c);

            else if (c==')' || c==']' || c=='}') {

                switch(c)
                {
                    case ')':

                        if( StackBracket.peek()=='(')
                            StackBracket.pop();
                        break;

                    case ']':
                        if ( StackBracket.peek()=='[')
                            StackBracket.pop();
                        break;

                    case '}':
                        if (StackBracket.peek()=='}')
                            StackBracket.peek();
                        break;

                    default:
                        return false;

                }

            }

        }
        return StackBracket.empty();
    }

}
```

Eerst creëren we een nieuwe stack met new Stack. De method Parse zal true of false teruggeven naargelang alle haakjes juist sluiten.

Het eerste wat we tegenkomen is een for each lus die over elk karakter van de string gaat. Daarin kijken we met een if statement of het karakter gelijk is aan een opengaand haakje. Wanner dit het geval zou zijn wordt deze bovenaan de stack toegevoegd.

De else if statement daarna gaat kijken of het karakter toevallig een sluitend haakje is. Is dit het geval dan gaan we verder met een switch waarbij elke case een ander soort sluitend haakje is.

We kijken met peek of er bovenaan de stack het tegengestelde haakje staat, als dit waar is gooien we dit er uit met pop.

Wanneer we op een of andere manier in de switch komen zonder het karakter een van de sluitende haakjes is wordt een return false teruggestuurd.

Het laatste deel van method gaat kijken of de stack leeg is nadat we elk karakter hebben overlopen. Wanneer dit het geval is is het return statement true, elk ander geval is het false.

```java
public class Main {

    public static void main(String[] args)
    {
    checkString( testString: "(3+2*{AB-C}/[35°])");
    checkString( testString: "(Dit is een [testString) ");
    }

    public static void checkString(String testString)
    {
        Parser parser= new Parser();
        if (parser.parse(testString))
        {
            System.out.println("Text was correct.");
        }
        else if(!parser.parse(testString))
        {
            System.out.println("The text was incorrect.");
        }
        else
        {
            System.out.println("Something went wrong.");
        }
    }
}
```

In de main staat checkString, hierbij word een nieuwe Parser gemaakt en deze zal de method parse oproepen. We kijken of de boolean van de method parse true is, dan printen we af dat het correct is gelukt. Krijgen we false terug dan wordt er afgeprint dan de tekst incorrect is. Wanneer er op een of andere manier geen van beide terugkrijgen dan drukken we af dat er iets mis is gelopen.