Aufgabe 1: ListIterator

Kreieren Sie eine Klasse ListIteratorApplication mit einem Attribut vom Typ ArrayList of Strings. Füllen Sie Ihre ArrayList mit den Wörtern des Satzes "Mit dem ListIterator ist es möglich, Listen vorwärts und rückwärts zu traversieren".

```
import java.util.ArrayList;

public class ListIteratorApplication
{
    private ArrayList<String> words;

    /**
     * Constructor for objects of class ListIteratorApplication
     */
    public ListIteratorApplication()
    {
        words = new ArrayList<String>();
        setArrayList("Mit dem ListIterator ist es möglich, Listen vorwärts und rückwärts zu traversieren");
    }

    public void setArrayList(String myInput)
    {
        String[] imputStrArray = myInput.split(" ");
        for(String s : imputStrArray)
        {
            words.add(s);
        }
    }
}
```

1. Programmieren Sie eine Methode iterateDown(). Diese iteriert durch die ArrayList von "oben nach unten". Verwenden Sie dazu den Iterator.

```
private void iterateDown()
{
    Iterator<String> itr = words.iterator();
    while(itr.hasNext())
    {
        System.out.printLn(itr.next());
    }
}
```

2. Programmieren Sie nun eine Methode iterateUp(). Diese iteriert durch die ArrayList von "unten nach oben". Verwenden Sie dazu eine for-Schleife und benutzen Sie die ArrayList get() Methode.

```
public void iterateUp()
{
    for(int i = words.size()-1;i >= 0; i--)
    {
        System.out.println(words.get(i));
    }
}
```

3. Iterieren Sie nun mit einer Methode iterateBothWays() durch die Wörter, zuerst von "oben nach unten" und gleich anschliessend von "unten nach oben". Verwenden Sie dazu den ListIterator. Schlagen Sie in der API Dokumentation nach, wie er funktioniert.

```
public void iterateBothWays()
{
    ListIterator<String> itr = words.listIterator();
    while(itr.hasNext())
    {
        System.out.println(itr.next());
    }
    while(itr.hasPrevious())
    {
        System.out.println(itr.previous());
    }
}
```

4. Optional: Für das Füllen der ArrayList verwenden Sie neu die Klasse StringTokenizer. Ein Beispiel deren Anwendung finden Sie in OOP5 auf Folie 10. Verwenden Sie die Java API Dokumentation, um den Code zu verstehen.

Aufgabe 2: Algorithmus für maximale Teilsumme

Eine Folge von ganzen Zahlen wird in einem Array gespeichert.

z.B.: int[] folge = {5, -8, 3, 4, -5, 7, -2, -7, 3, 5};

Eine Teilfolge einer Folge besteht aus einem beliebig langen, zusammenhängenden Teil der Folge.

z.B.: int[] teilfolge1 = {5, -8, 3};

int[] teilfolge2 = {-5, 7, -2, -7, 3};

Natürlich sind die leere Teilfolge und die gesamte Folge ebenfalls gültige Teilfolgen der Folge folge.

Als Teilsumme bezeichnet man die Summe aller Werte einer Teilfolge.

Die Teilsumme der teilfolge1 ist 0, die Teilsumme der teilfolge2 ist -4.

Bei der leeren Teilfolge ist die Summe 0.

Gesucht ist ein Algorithmus, welcher das Maximum der Teilsummen herausfindet.

Folgender Pseudo-Code beschreibt einen intuitiven Algorithmus:

Für alle Startwerte von 0 bis Länge der Folge

Für alle Endwerte von Startwert bis Länge der Folge

Berechne Summe der Teilfolge von Startwert bis und mit Endwert

Ist es die maximale Summe?

1. Welche Ordnung hat der oben beschriebene Algorithmus? Sie müssen dazu nicht "Hohe Mathematik" betreiben!
2. Erstellen Sie eine Methode public int maxTeilsumme(int[] folge) in der Klasse Folge, welche den oben beschriebenen Algorithmus implementiert.
3. Testen Sie Ihre Implementierung mit einer Testmethode, welche verschiedenste Folgen generiert und die Methode maxTeilSumme() aufruft. z.B.: Die Folge { 5, -8, 3, 4, -5, 7, -2, -7, 3, 5 } hat die maximale Teilsumme 9 (mit der Teilfolge { 3, 4, -5, 7 }).

Aufgabe 3 (optional): Weitere Aufgaben aus dem Buch zum Random Generator

Bearbeiten Sie die Aufgaben 5.16 bis 5.17 sowie 5.19 bis 5.20.

Verwenden Sie das Projekt "tech-support2", um mit Hilfe des Debuggers zu erforschen, wie die zufällige Auswahl funktioniert.

Aufgabe 4 (optional): Zufallszahl Generator

Implementieren Sie Ihren eigene Random Generator Klasse MyRandomGenerator mit Hilfe des Algorithmus, welcher in OOP6 auf Folie 28 beschrieben ist.

Zentrale Methode dabei ist public long getNextRandom().

Verwenden Sie dazu ein Attribut, welches den Wert der letzten Berechnung festhält . (initialisieren Sie das Attribut im Konstruktor mit einem Startwert, z.B. mit der System-Zeit siehe API Dokumentation, Klasse java.lang.System)