Methodisches Programmierung

Übungen zur Vorlesung

Übungsblatt 2 (zu besprechen in Vorlesungswoche 3)

Beachten Sie diese Anweisungen bei allen folgenden Programmierübungen:

- *Achten Sie bei allen Funktionen, Werten und Variablen auf eine möglichst gute Benennung.*
- *Schreiben Sie für alle Methoden Tests. Schreiben Sie diese bevor Sie die Methoden implementieren.*
- *Dokumentation: Ist der Name einer Methode nicht selbsterklärend, sollten Sie zunächst überlegen, ob Sie nicht einen besseren Namen vergeben können (siehe Punkt 1). Wenn das nicht möglich ist, erklären Sie möglichst präzise in einem Code-Kommentar, was die Methode tut.*
- Fragen Sie in der A-Übung nach, wenn die Aufgabenstellung missverständlich ist!

1 Schleifen und Strings

1. Schreiben Sie eine Funktion, die eine Binärzahl in eine Dezimalzahl umwandelt. Die Eingabe soll eine positiver Long-Wert sein, der nur die Ziffern 0 und 1 enthält. Die Ausgabe soll ein Int-Wert sein.

a. Schreiben Sie eine Variante der Funktion, die als Eingabe einen String akzeptiert, der eine positive Binärzahl enthält, also eine Folge von 1 und 0, die mit einer 1 beginnt, außer es ist „0“ selbst.

2. Schreiben Sie eine Funktion, die für eine gegebene Zahl überprüft, ob sie eine Primzahl ist oder nicht. Recherchieren Sie ggf. online nach Lösungsansätzen.

3. Schreiben Sie eine Funktion, die einen String umkehrt, z.B. „String“ in „gnirtS“. Verwenden Sie dabei nicht die String-Methode `.reversed()`, sondern entwickeln Sie eine eigene Lösung.

4. Schreiben Sie eine Funktion, die die Anzahl von Worten in einem String zurückgibt.

5. Schreiben Sie eine Funktion

```
checkParenthesis(s : String) : Boolean
```

die überprüft, ob ein String bestehend aus Klammern und anderen Zeichen richtig geklammert ist. Beispiele:

- `checkParenthesis("()")` ergibt `true`
- `checkParenthesis("(())()")` ergibt `true`
- `checkParenthesis("(()))()")` ergibt `false`
- `checkParenthesis("(foo)(bar())")` ergibt `true`
- `checkParenthesis("(foo))((bar())")` ergibt `false`

6. Schreiben Sie eine Funktion

`caesarEncrypt(s:String, offset : Int) : String`

die eine einfache Variante der Caesar-Verschlüsselung[1] eines Strings durchführt. Dabei sollen Kleinbuchstaben (a-z) und Großbuchstaben (A-Z) um ein gegebenes `offset` im Alphabet verschoben werden. Andere Satzzeichen sowie Umlaute sollen nicht manipuliert werden. Beispiele:

`caesarEncrypt("Hello!", 3)` ergibt `"Khoor!"`
`caesarEncrypt("Hello!", 4)` ergibt `"Lipps!"`
`caesarEncrypt("Hello!", -3)` ergibt `"Ebiil!"`

Dort, wo das Verschieben im Alphabet über das Ende oder den Anfang des Alphabets hinausgeht, sollte die Verschiebung wieder von vorn bzw. hinten im Alphabet anfangen. Beispiele:

`caesarEncrypt("Abc", -2)` ergibt `"Yza"`
`caesarEncrypt("Lizzy", 3)` ergibt `"Olccb"`

(*Hinweis*: Einen Character-Wert können Sie mit den Methoden `plus()` und `minus()` im Unicode-Alphabet verschieben. z.B. `'a'.plus(3)` ergibt `'d'`)

+ "Weihnachtsbaum" - Aufgabe, siehe Folien

[1] https://de.wikipedia.org/wiki/Caesar-Verschl%C3%BCsselung