Institut für Informatik
Praktische Informatik und Bioinformatik
Prof. Dr. Ralf Zimmer

Sommersemester 2017

Übungen zum Bioinformatik-Tutorium

Blatt 4

Termin:Donnerstag, 20.07.2015, 12 Uhr

In diesem Übungsblatt wollen wir anfangen eine eigene Bibliothek anzulegen, die Funktionen und Klassen für uns bereitstellt, die wir immer wieder gebrauchen können. Dazu fangen wir mit dem alt bekannten Beispiel des Einlesens von Dateien an.
Legt zur Bearbeitung des Übungsblatt ein neues Projekt in eurem workspace an (z.B. MyUtils). In diesem Projekt soll nun ein package angelegt werden (z.B.: utils), dass Klassen enthält, die bestimmte Aufgaben erfüllen können (z.B.: FileUtils). Mit der Zeit entsteht daraus eine Bibliothek, die euren Source Code besser lesbar macht und die Programmiereffektivität erhöht.

Ein paar generelle Dinge über Utils:

- **Klassen** sollten final sein: `public final class ExampleUtils`
  - ... und der Name sollte beschreiben was die Klasse kann.
- **Methoden** sind statisch und public: `public static int exampleMethod(int a)`
  - ... und können überladen werden: `public static int exampleMethod(int a, int b)`
- **Kommentiert** euren Code so kurz wie möglich, dass man weiß was die einzelnen Methoden machen. Verwendet die Javadoc Kommentare, damit die Beschreibung in der IDE angezeigt wird: `/** Javadoc Kommentar */`
- Links: (nur im PDF sichtbar)
  - Erstellen von Utils Klassen
  - Namen von Utils Klassen

Aufgabe 1 - FileUtils

(a) Schreibe eine Utils Klasse die für das Filehandling verantwortlich ist. Die Klasse soll zunächst eine Methoden zur Verfügung stellen, die eine Datei in Tabellenformat (*.tsv) einlesen kann und den Inhalt in einem beliebigen Format speichert und zurückgibt.

(b) Erweitere die Klasse mit einer Methode, die den gesplitteten Inhalt jeder Zeile in einem String Array abspeichert und am Ende eine `ArrayList<String[]>` (oder ein Vector) zurückgegeben wird.

(c) Da solche Dateien oft mehrere Spalten haben, wobei zum Beispiel in der ersten Spalte eine ID angegeben ist, macht es Sinn eine Methoden in der Klasse anzubieten, die eine Hashmap (key:String → value:String) zurückgibt. Die Methode soll unter anderem als Parameter die Spaltennummer des Keys und des Values übergeben bekommen.

- *Erweiterung*: Biete eine Methode an, die auch mehrere Spalten als Value für einen Key abspeichern kann (z.B: key:String → value:ArrayList<String>)

(d) Initialisiere zwei neue Methoden, die die Methode aus Aufgabe 1b und 1c erweitern, indem sie ein Seperator Zeichen entgegen nehmen. Der Seperator soll dann genutzt werden, um die Zeile der Datei zu splitten. Die Methoden sollen genauso heißen, wie in den Aufgaben 1b und 1c (Überladen von Methoden).

(e) **Erweiterung:** Um Methoden flexibler zu definieren, können Interfaces und Lamdas verwendet werden. Dies bietet sich zum Beispiel dann an, wenn man einen Wert in der Zeile, die man gerade einliest, in einen Integer umwandeln will. Ein geeignetes Interface wäre dann zum Beispiel auch in der Lage einen Double zu parsen, ohne dass man eine neue Methode dafür schreiben muss.
Füge nun einen neue Methode in die Klasse hinzu, die ein Interface und ein File als Parameter entgegen nimmt. Die Methode soll in der Lage sein, mit Hilfe des selbst definierten Interfaces, die eingelesenen Zeilen in Zahlen (int, double, float, ...) umzuwandeln. Zur Vereinfachung reicht es, wenn eine ArrayList ohne Typ zurückgegeben wird.