hochschule mannheim

FAKULTÄT FÜR INFORMATIK

Prof. Dr. Peter Knauber

PR1 – Programmieren 1 Bachelor-Studiengang Informatik WS2017/2018

Übungsblatt 7: Rekursion, mehrdimensionale Arrays

Ausgabe: 20.10.17
Abgabe: 27.11.17 12:00 Uhr elektronisch mittels Git

Aufgabe 1

Erstellen Sie eine Klasse `StringExtension` für verschiedene Methoden, die mit Strings arbeiten.

a) Die Methode

```
String toUpper( String text )
```

akzeptiert einen String als Parameter und soll einen neuen String zurückgeben, in dem alle Buchstaben des ursprünglichen Strings in Großbuchstaben umgewandelt enthalten sind. Gewandelt werden sollen die Buchstaben a bis z und die Umlaute ä, ö und ü.
Beispiel:

```
"Viel Erfolg bei der Aufgabe!"
```

soll umgewandelt werden in

```
"VIEL ERFOLG BEI DER AUFGABE!".
```

b) Entwickeln Sie eine Methode

```
String toUpperRecursive( String text )
```

die dasselbe tut wie die Methode `toUpper`, die aber rekursiv arbeitet. Das bedeutet, dass Sie in dieser Methode wie auch in eventuellen Hilfsmethoden keinerlei Schleifen verwenden dürfen!

c) Die Methode

```
int scan( String text, String needle )
```

soll den String-Parameter `text` nach einem Teil-String `needle` durchsuchen und dessen Position innerhalb von `text` zurückgeben (0 gilt dabei als erste Position).
Sollte `needle` mehr als einmal innerhalb von `text` vorkommen, wird die Position des ersten Treffers zurückgeliefert. Wenn `needle` nicht auffindbar oder ein Leerstring ist, wird eine `PRException` mit einer entsprechenden Fehlermeldung ausgelöst.
Beispiel:
In dem String

```
"Hello, world! This world is cool!"
```

soll das Wort `"world"` aufgespürt werden; Ergebnis ist die Position 7.

d) Die Methode

```
String[] split( String text, char delimiter )
```

soll eine Zeichenkette, die als Parameter an `text` übergeben wird, in Teil-Strings aufteilen, die in einem passend langen String-Array zurückgegeben werden. Getrennt wird jeweils am Trennzeichen `delimiter`.
Beispiel:
Der String

```
"Banane;Erdbeere;Apfel"
```

soll über das Semikolon als Trennzeichen aufgetrennt werden in die Strings `"Banane"`, `"Erdbeere"` und `"Apfel"`, die als Array der Länge drei an den Aufrufer zurückgegeben werden.

e) Erstellen Sie eine `main`-Methode, in der man Eingaben machen kann, um die drei Funktionen zu Testzwecken aufzurufen.
alternativ:
Erstellen Sie eine Klasse `StringExtensionTest` mit *guten* und *sinnvollen* JUnit-Tests für die angegebenen Methoden. Sie sollten eine Erklärung bereit haben, warum Sie diesen oder jenen Test geschrieben haben.

Hinweis
Von der Klasse `String` dürfen lediglich die Methoden `length`, `charAt`, `equals` und `toCharArray` verwendet werden, *keine* anderen!

Aufgabe 2

Erstellen Sie ein Java-Programm `Maze`, das von einem gegebenen Eingang ausgehend einen (beliebigen) Weg durch ein rechteckiges Labyrinth findet.

Eine Methode

```
static void findWay(char[][] field)
```

erhält ein zweidimensionales Array von Zeichen, die ein Labyrinth beschreiben. Geeignete Zeichen stellen die Wände (`'#'`), die Wege (`' '`) und den Eingang (`'S'`) in das Labyrinth dar. Die Methode sucht zunächst den Eingang, der mit dem Start-Zeichen (`'S'`) gekennzeichnet ist. Dann ruft sie die rekursive Hilfsmethode

```
static boolean findWayRecursive(char[][] field, int row, int col)
```

mit den Koordinaten der gefundenen Startposition auf. Diese Methode sucht einen Weg durch das Labyrinth bis zu einem Ausgang (ungleich dem Eingang). Dabei markiert sie den Eingang sowie (nur) den gefundenen Weg mit Punkten (`'.'`) und den gefundenen Ausgang (`'E'`). Nach Beendigung der Methode ist *nur der direkte Weg* im Array mit Punkten bzw. 'E' markiert; eventuelle Irrwege und Sackgassen sind wieder als Weg (`' '`) dargestellt.

Beispiel: So sieht das Labyrinth vor und nach der Ausgangssuche aus:

```
## #######
#       #
# ###### #
#     ## #
##### ## S
#   # ####
# # #    #
# # #### #
# #      #
# ########
```

```
##E#######
# .......#
# ######.#
#     ##.#
##### ##..
#   # ####
# # #    #
# # #### #
# #      #
# ########
```

Hinweise zum Programm

- Es kann zu verschiedenen Fehlersituationen kommen. In allen diesen Situationen soll eine `PRException` ausgelöst werden.
- Ein einfaches Testprogramm mit einem Test-Labyrinth finden Sie im Wiki.
- Sie können davon ausgehen, dass alle Labyrinthe rechteckig sind, d.h. alle Zeilen bestehen aus mindestens drei Zeichen und sind gleich lang.

Allgemeines

- Die Aufgaben sind in Eclipse zu bearbeiten und beim Testat vorzuführen.
- Legen Sie für die Bearbeitung dieses Übungsblattes ein Paket namens `pr1.uebung07` an, in dem Sie Ihre Klassen anlegen.
- Erlaubt sind `MakeItSimple`-Funktionen (keine nicht besprochene Funktionalität aus der Java-Standard-Bibliothek) und das bisher erworbene Wissen aus den PR1-Vorlesungen. Sie müssen alle(!) von Ihnen verwendeten Konstrukte der Sprache sowie alle verwendeten Methoden, die nicht aus der Hilfsbibliothek `MakeItSimple` stammen, gut erklären und nötigenfalls im Testat selbst programmieren können!
- Sie geben ab, indem Sie vor Ende der Abgabefrist Ihr Projekt mit den lauffähigen Programmen des Übungsblattes in das Repository auf *ovid* pushen. Achten Sie darauf, ob Sie die Kontroll-E-Mail bekommen! Die letzte hochgeladene Version Ihres Projekts wird gewertet. Andere Abgaben, ob elektronisch oder auf Papier, zählen als **nicht abgegeben**!