Institut für Informatik
Prof. Dr.-Ing. Elke Pulvermüller
Dipl.-Systemwiss. Mathias Menninghaus

Universität Osnabrück, 13.06.2017
http://www-lehre.inf.uos.de/~binf
Testat bis 21.06.2017, 14:00 Uhr

Übungen zu Informatik B

Sommersemester 2017

Blatt 8

Aufgabe 8.1: Testen (20 Punkte)

Bei den jährlich stattfindenden *Hungerspielen* treten 24 *Tribute* aus den 12 Distrikten des fiktiven Landes Panem in einer künstlichen Arena gegeneinander an. Bei der 75. Durchführung der Hungerspiele ist die kreisrunde Arena analog zu einer Uhr in zwölf gleich große Bereiche unterteilt. Die Mitte der Arena habe die Position $(0,0)$ und die Arena einen Durchmesser von 3 Meilen. Je nach Uhrzeit wird ein Bereich aktiv und in ihm werden tödliche Fallen freigeschaltet. Beispielsweise ist von 12 bis 1 Uhr der Bereich im Winkel von $0°$ bis $30°$ aktiv, wobei $0°$ die Linie von der Mitte bis zur Position $(0,1.5)$ anzeigt. Ein Tribut wird fast sicher umkommen, wenn er sich in einem aktiven Bereich aufhält oder die Arena verlässt.

Schreiben sie eine Java-Klasse `Arena`, die die Methode `int getArea(double x, double y)` enthält und zurück gibt, in welchem der zwölf Bereiche sich ein Tribut mit der Position (x,y) befindet. Wenn sich der Tribut außerhalb der Arena befindet, soll die Methode -1 zurück geben. Die Methoden in der Klasse `java.lang.Math` können dafür nützlich sein. Testen Sie Ihre Implementation ausführlich mit einem automatisierten Testprogramm. Wie viele Testfälle müssen Sie mindestens erzeugen, um alle möglichen Eingaben hinsichtlich einer korrekten Ausgabe geprüft zu haben? Wenn Sie separate Hilfs-Methoden implementieren, sollten Sie diese ebenfalls testen.

Aufgabe 8.2: Standard-Serialisierung (13 Punkte)

Betrachten Sie das Programm zum Berechnen der Fibonacci-Zahlen `Fibonacci.java`, das bereits berechnete Fibonacci-Zahlen in einer `HashMap` vorhält.

Verändern Sie dieses Programm dahingehend, dass die eingesetzte `java.util.HashMap` beim Start aus einer Datei gelesen und nach der Beendigung des Programms wieder in diese Datei zurückgeschrieben wird. Sollte das Programm zum ersten Mal überhaupt aufgerufen werden, soll eine neue `java.util.HashMap` erzeugt werden.

Aufgabe 8.3: Spezielle Serialisierung (25 Punkte)

Erweitern Sie den `Heap` aus der Musterlösung um das Interface `java.io.Serializable`.

Überschreiben Sie die Standard-Serialisierung soweit, dass der serialisierte `Heap` möglichst wenig Platz verbraucht. Dafür soll vor allem vermieden werden, dass `null`-Werte serialisiert werden.

Schreiben Sie anschließend eine Testklasse, in deren `main`-Methode Sie einen Beispiel-`Heap` erzeugen, serialisieren und in eine Datei schreiben, um ihn anschließend wieder aus der Datei auszulesen und zu deserialisieren. Lassen Sie die Testklasse automatisiert vergleichen, ob der Heap, den sie serialisieren, der gleiche ist, wie der, den Sie deserialisieren.

Aufgabe 8.4: Suchen und Ersetzen (22 Punkte)

Implementieren Sie ein Kommandozeilenprogramm, mit dem gemäß eines regulären Ausdrucks Zeichenketten in einer oder mehreren Dateien gesucht und ersetzt werden können.

Nutzen Sie die Klassen vom vorigen Aufgabenblatt um die Dateien zu ermitteln, in denen gesucht und ersetzt werden soll. Durchlaufen Sie jede dieser Dateien vollständig und ersetzen Sie jedes Vorkommen eines vom Nutzer gegebenen regulären Ausdrucks durch eine vom Nutzer gegebene Zeichenkette.

Da reguläre Ausdrücke auch Zeilenumbrüche enthalten können, müssen Sie die jeweilige Datei vollständig in einen `String` umwandeln. Das Zeichen, dass einen Zeilenumbruch in Ihrem Betriebssystem definiert, erhalten Sie durch den Aufruf von `System.getProperty("line.separator")`. Schreiben Sie den veränderten `String` am Ende wieder zurück in die Datei.

Die Argumente, mit denen das Programm aufgerufen werden kann, seien definiert durch die folgende Syntax:

```
[-r] Search Replacement FileOrDirectory
```

Die Argumente im Einzelnen sind wie folgt definiert:

- `-r` Flag, ob die Unterverzeichnisse rekursiv durchsucht werden sollen.
- `Search` regulärer Ausdruck, nach dem in den Dateien gesucht werden soll.
- `Replacement` Zeichenkette, mit der alle Vorkommen des regulären Ausdrucks ersetzt werden sollen.
- `FileOrDirectory` Pfadangabe zu einer Datei oder einem Verzeichnis, in der/dem gesucht und ersetzt werden soll.

Hinweis Testen Sie Ihre Implementation zuerst nur an einer einzigen (Test-)Datei um zu vermeiden, dass Sie aus Versehen wichtige Daten überschreiben.