10. Übung **Informatik B** **SS 08**

Klaus Kriegel **Abgabe: 01.07.2008, 12:00 Uhr**

Aufgabe 1: **Ausnahmen** (2 + 4 + 3 + 2 Punkte)

Alice, Bob und Carol haben ein Spiel vereinbart, das wir mit der Klasse `ABCSpiel` implementieren wollen. Das Spiel ist in Runden organisiert, wobei jede Runde nach der folgenden Regel abläuft:
Jeder Spieler setzt einen Euro ein, wählt eine (geheime) natürliche Zahl n aus dem Bereich $0 \leq n \leq 1000$ und schreibt sie verdeckt auf einen Zettel. Dann werden alle Zettel aufgedeckt und es gewinnt der Spieler, dessen Zahl zwischen den Zahlen der Gegner liegt. Haben zwei Spieler die gleiche Zahl gewählt, gibt es keinen Sieger.
a) Zur Vereinfachung nummerieren wir die Spieler mit 0 (Alice), 1 (Bob), 2 (Carol) und bezeichnen die gewählten Zahlen mit $n0, n1, n2$. Definieren Sie eine Methode `winner`, welche aus den drei Zahlen den Sieger ermittelt und bei einem Gleichstand die Ausnahme `EqualityException` wirft. Diese Ausnahme sollte eine Information darüber enthalten, welche Spieler einen Gleichstand hatten. Sie können die folgende Codierung verwenden: 0 für $n0 \neq n1 = n2$, 1 für $n1 \neq n0 = n2$, 2 für $n2 \neq n0 = n1$ und 3 für $n0 = n1 = n2$.
b) Die Klasse `ABCSpiel` hat ein Feld, in dem Konten der 3 Spieler verwaltet werden. Jeder Spieler hat am Anfang 1000 Euro auf seinem Konto. Die Methode `updateAccounts` wertet eine Runde aus und **soll** dazu `winner` benutzen. Bei Gleichstand gilt folgende Regel: Spieler, welche die gleiche Zahl wie ein anderer Spieler gewählt haben verlieren ihren Einsatz an die Bank, alle verleibenden erhalten ihren Einsatz zurück. Setzen Sie diese Regel durch eine Fehlerbehandlung um!
c) Simulieren Sie das Spiel über 20 000 Runden, wobei alle Spieler zufällige Zahlen verwenden. Einen Zufallsgenerator finden Sie in `java.util.Random.java`. Welche Beobachtung können Sie hinsichtlich des Gewinns der Bank machen?
d) Es gibt eine bessere Strategie für Alice, wenn Sie weiß, dass beide Gegner zufällig spielen. Welche? Testen Sie diese Strategie mit dem Programm. Was passiert, wenn alle nach dieser Strategie spielen?

Aufgabe 2: **Binäre Bäume** (10 Punkte)

Bezeichnet man mit $\mathcal{T}_m$ die Menge der binären Bäume mit (genau) m Blättern und mit $t(m)$ ihre Anzahl, so kommt man durch einfache Fallunterscheidungen auf die folgenden Anfangswerte: $t(1) = 1$, $t(2) = 1$, $t(3) = 2$ und $t(4) = 5$. Die Werte für größere m lassen sich durch Rekursion bestimmen, mit der Überlegung, dass jeder Binärbaum mit m Blättern einen Baum aus $\mathcal{T}_k$ als linken Teilbaum unter der Wurzel und einen Baum aus $\mathcal{T}_{m-k}$ als rechten Teilbaum unter der Wurzel hat, wobei $1 \leq k \leq m-1$ gelten muss. Damit erhält man neben der Rekursion

$$t(n) = \sum_{k=1}^{m-1} t(k) \cdot t(m-k)$$

auch die folgende Methode zur Aufzählung der Bäume aus $\mathcal{T}_m$ als $T_{m,0}, T_{m,1}, \ldots, T_{m,t(m)-1}$: Zuerst kommt die Gruppe der $t(1) \cdot t(m-1)$ Bäume mit einem Blatt auf der linken und

$(m-1)$ Blättern auf der rechten Seite, dann die Gruppe der $t(2) \cdot t(m-2)$ Bäume mit zwei Blättern auf der linken und $(m-2)$ Blättern auf der rechten Seite, u.s.w.
Innerhalb jeder Gruppe $\mathcal{T}_k \times \mathcal{T}_{m-k}$ wird wieder rekursiv aufgezählt: Zuerst die $t(m-r)$ Bäume mit $T_{k,0}$ auf der linken Seite, dann die $t(m-r)$ Bäume mit $T_{k,1}$ auf der linken Seite, u.s.w.

a) Verwenden Sie die (auf der Homepage unter dem Übungsblatt verlinkten) Klassen `BTNode.java` und `BinTree.java` als Grundlage, um in einer Testklasse Binärbäume nach ihrem Index zu generieren. Sie sollten dabei Folgendes beachten:

- Jeder Baum ist durch seine Wurzel repräsentiert. Man kann zwei Bäume unter einer neu angelegten Wurzel zu einem neuen Baum zusammenfassen, wenn die entsprechenden Eltern/Kind–Referenzen richtig gesetzt werden.
- Die Methode `generateTree(int m, int i)` ($0 \leq i \leq t(m) - 1$ wird vorausgesetzt) soll den Baum $T_{m,i}$ erzeugen.
- Wir setzen $m \leq 15$ voraus, um im `int`–Bereich arbeiten zu können. Es ist sinnvoll, ein Feld `public int[] anzahl` als Attribut der Testklasse anzulegen, in dem alle relevanten Werte von $t(m)$ abgelegt sind und dazu eine Methode `void setAnzahl()` und ein Attribut `boolean anzahlFilled` einzuführen. Damit kann man vermeiden, dass die Werte $t(k)$ immer wieder neu berechnet werden müssen.
- Erzeugen Sie in der `main`–Methode eine Reihe von zufälligen Zahlen aus dem Bereich $\{0, 1, \ldots, t(15) - 1\}$, generieren Sie die zugehörigen Bäume aus $\mathcal{T}_{15}$ und geben Sie die Höhen dieser Bäume aus.

Hinweis: Wer mehr zu den Zahlen $t(m)$ erfahren möchte, kann nach im Netz nach dem Stichwort Catalan Zahlen suchen.