Technische Universität Darmstadt
Fachbereich Informatik
Prof. Johannes Fürnkranz

Allgemeine Informatik II im SS 2007

Übungsblatt 6

Bearbeitungszeit: 06.06. bis 12.06.2006

Aufgabe 1: Uhrzeit

Zeitangaben bestehen aus Stunden (0-23), Minuten (0-59) und Sekunden (0-59). Definieren Sie eine Klasse **`ClockTime`**, die eine Zeitangabe mit diesen Komponenten repräsentiert. Die Klasse soll die folgenden Methoden anbieten:

- **`public ClockTime()`**:Standard-Konstruktor
  initialisiert die Zeit mit **`00:00:00`**.
- **`public ClockTime(int hours, int minutes, int seconds)`**:
  Konstruktor mit 3 **`int`**-Parametern
  Zur Vereinfachung gehen wir davon aus, dass nur positive Werte übergeben werden. Sie können aber beliebig groß sein. Sekunden sollen in Minuten und Minuten in Stunden überlaufen können. Und nach **`23:59:59`** soll **`00:00:00`** folgen.
- **`public ClockTime(ClockTime ct)`**: Kopier-Konstruktor
  bekommt als Parameter ein **`ClockTime`**-Objekt übergeben und initialisiert das neue Objekt mit der Zeit des übergebenen Objekts.
- **`public int getSeconds()`**
  liefert die Sekunden der Zeitangabe im Bereich 0 < **`s`** < 59.
- **`public int getMinutes()`**
  liefert die Minuten der Zeitangabe im Bereich 0 < **`s`** < 59.
- **`public int getHours()`**
  liefert die Stunden der Zeitangabe im Bereich 0 < **`s`** < 23
- **`public boolean isSame(ClockTime ct)`**
  akzeptiert als Argument ein anderes **`ClockTime`**-Objekt und stellt fest, ob beide Zeiten gleich sind. Dem Vergleich entsprechend wird **`true`** oder **`false`** zurückgeliefert.

Überlegen Sie, wie die Zeitinformation in **`ClockTime`** abgespeichert wird.

Schreiben Sie außerdem eine Klasse **`ClockTimeTest`** mit der Sie die Methoden der Klasse **`ClockTime`** testen können. (main-Methode nicht vergessen).

Hinweis: In der Vorlesung haben Sie das Konzept der Überladung kennen gelernt. Überladene Methoden oder Konstruktoren haben denselben vollständigen Namen. Sie unterscheiden sich aber in der Anzahl bzw. dem Typ der Parameter.

Aufgabe 2: Das Josephus-Problem

Im Jahr 67 n. Chr. wurde die galiläische Stadt Jotapata, die unter der Führung des späteren jüdischen Geschichtsschreibers Josephus (37 - 100) ein Zentrum des antirömischen Widerstandes war, nach 47-tägiger Belagerung von Kaiser Vespasian eingenommen. Josephus und 40 Soldaten versteckten sich in einer Zisterne. Um der Versklavung zu entgehen, wollten die Soldaten sich selbst töten. Josephus, der sie vergeblich beschwor, von ihrem Vorhaben Abstand zu nehmen, wollte wenigstens sich und einen Freund retten. Deshalb schlug er als Tötungsritual den beliebten römischen Brauch des 'decimatio' vor. Dabei bilden die Soldaten einen Kreis, es wird jeweils jeder zehnte Soldat ausgezählt und umgebracht. An welche Stellen des Kreises sollten sich Josephus und sein Freund stellen, um das Ritual zu überleben?

Allerdings stellen auch die Auszählreime von Kindern genau diese Situation - zum Glück unblutig dar. Die Anzahl der Silben oder Betonungen eines Zählreimes wird zum Zählen im Kreis verwendet. Ein Kind nach dem anderen scheidet aus, bis nur mehr eines übrigbleibt.

Generell lässt sich das Problem wie folgt beschreiben:

- Eine Gruppe von **`N`** Personen bildet einen geschlossenen Kreis.
- Von der ersten Person an wird jeweils bis zur **`K`**-ten Person weitergezählt. Diese Person scheidet aus, der Kreis wird wieder geschlossen.
- Vom Nachbarn (Nachfolger) der ausgeschiedenen Person beginnend verfährt man wie oben

Schreiben Sie ein Java-Programm, welches herausfindet, in welcher Reihenfolge die Personen ausgeschieden werden. Die Idee ist, dass sie eine zirkuläre Liste von Personen konstruieren. Eine Zirkuläre Liste sei eine Liste, bei dem der Verweis im letzten Knoten anstelle auf **`null`** auf das erste Element dieser Liste verweist. Eine Person soll durch eine Nummer charakterisiert werden.

Nachdem in diese Liste **`N`** Personen eingefügt wurden, soll jede **`K`**-te Person aus der Liste entfernt werden. Danach ist die erste Person (das heißt, das nun erste Element der Liste) die Person, die auf die gelöschte folgt.

Gehen Sie wie folgt vor:

1. Verwenden Sie die vorgegebene Klasse **`Circle`**.
2. Implementieren Sie die Methode **`insert`**. Sie soll bei jedem Aufruf ein **`Person`**-Objekt erzeugen und in die Liste nach dem zuletzt eingefügten Objekt einfügen. Der übergebene Parameter ist die Nummer der Person, die neu erzeugt wird. Werden nacheinander die Personen 1, 2, 3 eingefügt, soll die Liste folgende Struktur haben: **`1->2->3->x-> ... ->x->1`**
3. Implementieren Sie die Methode **`remove`**. Der übergebene Parameter gibt an, an welcher Position die aus der Liste zu entfernende Person steht. Der Rückgabewert soll die Nummer der entfernten Person sein.
4. Schreiben Sie die Klasse **`Josephus`**. Es reicht, wenn diese eine **`Main`**-Methode besitzt. Erwarten Sie von dem Eingabefenster zwei Zahlen als Parameter. Die erste Zahl soll das **`K`** und die zweite das **`N`** sein. Erzeugen Sie dann die Klasse **`Circle`** und fügen Sie **`N`** Personen ein. Danach entfernen Sie solange die **`K`**-te Person bis die Liste leer ist. Geben Sie jeweils die Nummer der entfernten Person auf dem Bildschirm aus. Siehe folgendes Beispiel: Der Aufruf der **`main`**-Methode mit den Zahlen **`5`** und **`9`** sollte folgende Ausgabe hervorbringen:

```
> 5 1 7 4 3 6 9 2 8
```

Beachten Sie bei der Implementierung folgendes:

- Das Einfügen einer neuen Person in eine leere Liste. (Dieses Element verweist auf sich selbst).
- Das Entfernen des letzten Elements. Wie kann es erkannt werden. Die Variable **`start`** sollte danach **`null`** sein.
- Machen Sie sich Skizzen, was sich beim Einfügen bzw. Entfernen einer Person in der Listenstruktur ändert bzw. ändern soll.

Viel Spaß!