Technische Universität Darmstadt

Fachbereich Informatik

Prof. Dr. Andreas Koch

Allgemeine Informatik 2 im SS 2008

Übungsblatt 6

Bearbeitungszeit: 19.05. bis 25.05.2008

Aufgabe 1: Uhrzeit

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

- **`ClockTime()`**

  Standard-Konstruktor, initialisiert die Zeit mit 00:00:00.

- **`ClockTime(int hours, int minutes, int seconds)`**

  Konstruktor mit 3 **`int`**-Parametern. Zur Vereinfachung gehen wir davon aus, dass nur positive Werte übergeben werden – diese können aber beliebig groß sein. Sekunden sollen in Minuten und Minuten in Stunden überlaufen, und nach 23:59:59 soll 00:00:00 folgen. Ein Aufruf mit den Parametern **`(0, 0, 3600)`** soll also die Zeit 01:00:00 darstellen.

- **`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 getHours()`**

  Liefert die Stunden der Zeitangabe im Bereich $0 \le h \le 23$.

- **`public int getMinutes()`**

  Liefert die Minuten der Zeitangabe im Bereich $0 \le m \le 59$.

- **`public int getSeconds()`**

  Liefert die Sekunden der Zeitangabe im Bereich $0 \le s \le 59$.

- **`public boolean isSame(ClockTime ct)`**

  Akzeptiert als Argument ein anderes **`ClockTime`**-Objekt und stellt fest, ob beide die gleiche Uhrzeit darstellen. Wenn ja, wird **`true`** zurückgegeben, sonst **`false`**.

Überlegen Sie, wie Sie die Zeitinformation in **`ClockTime`** abspeichern – dafür gibt es nicht nur eine Möglichkeit!

Schreiben Sie außerdem eine Klasse **`ClockTimeTest`** mit einer Methode **`public static void test()`**, mit der Sie die Methoden der Klasse **`ClockTime`** testen.

Aufgabe 2: Josephus

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 47tägiger 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 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 zum Schluss übrig zu bleiben und so das Ritual zu überleben?

Generell lässt sich das Problem wie folgt beschreiben:

- Eine Gruppe von **n** Personen bildet einen geschlossenen Kreis.
- Von der ersten Person an wird die **k**-te Person abgezählt. Diese Person scheidet aus, der Kreis wird wieder geschlossen.
- Vom Nachfolger der ausgeschiedenen Person beginnend verfährt man wie oben.

Ihre Aufgabe ist es, ein Java-Programm zu schreiben, das herausfindet, in welcher Reihenfolge die Personen ausscheiden. Die Idee ist, dass Sie eine zirkuläre Liste von Personen konstruieren. Eine zirkuläre Liste ist eine verkettete Liste, bei dem der Nachfolgerverweis im letzten Element anstelle auf **`null`** auf das erste Element der Liste verweist.

Um die einzelnen Personen unterscheiden zu können, hat jede eine Nummer (statt eines Namens).

Gehen Sie wie folgt vor:

a) Auf unserer Webseite finden Sie auch die Datei **Circle.java**. Verwenden Sie die darin enthaltene Klassen **`Circle`** und **`Person`** (**Bearbeiten → Klasse aus Datei hinzufügen…**) und lesen Sie sich diese erst mal gut durch.

b) Implementieren Sie die Methode **`void insert(int number)`**. Sie soll bei jedem Aufruf ein neues Person-Objekt mit der Nummer **`number`** erzeugen und im Kreis vor der Startperson **`start`** einfügen – der Nachfolger der letzten Person im Kreis ist also die neue Person, der Nachfolger der neuen Person ist **`start`**. Ein Sonderfall ist der leere Kreis zu Beginn: in diesem Fall ist der Nachfolger der neuen Person diese Person selbst.

   Werden also in einen leeren Kreis die Personen 1, 2 und 3 eingefügt, soll die Liste folgende Struktur haben: 1 → 2 → 3 → 1 → …

c) Implementieren Sie die Methode **`int remove(int pos)`**. Der übergebene Parameter gibt an, an welcher Position (von der Startperson aus) die ausscheidende Person steht. Der Rückgabewert soll die Nummer der entfernten Person sein.

   Wenn der Kreis nur noch eine Person enthält (wie erkennen Sie das?), reicht es aus, die Startperson **`start`** auf **`null`** zu setzen – der Kreis ist dann leer.

   Wenn der Kreis mehr als eine Person enthält, ist es etwas komplizierter: um eine Person Y zu entfernen, die zwischen X und Z steht, muss Z der neue Nachfolger von X werden. Anschließend wird Z die neue Startperson.

d) Schreiben Sie eine Klasse **`Josephus`** mit einer Methode **`public static void test(int k, int n)`**. Erzeugen Sie darin einen neuen Kreis und fügen Sie **`n`** Personen

mit den Nummern 1 bis **n** ein. Danach entfernen Sie solange die **k**-te Person, bis der Kreis leer ist. Geben Sie jeweils die Nummer der entfernten Person auf dem Bildschirm aus.

Der Aufruf **`test(5, 9)`** sollte folgende Ausgabe hervorbringen: **`5 1 7 4 3 6 9 2 8`**.

Um herauszufinden, wie Josephus und sein Freund überleben können, müssen Sie die Methode natürlich mit den Parametern 10 und 41 aufrufen.

Hinweis: machen Sie sich Skizzen, was sich beim Einfügen bzw. Entfernen einer Person in der Listenstruktur ändert bzw. ändern soll.

Viel Spaß!