Vorlesung „Java I“ SS 04 6. Übung (31. 5. 2004)

http://www.coli.uni-sb.de/cl/courses/java-04/

---

1 Maschinelle Übersetzung (3+1 Punkte)

Schreibe ein Programm, das englische Zahlwörter im Zahlenraum von 1 bis 100 in deutsche Zahlwörter übersetzt und umgekehrt. Die Zielsprache (`de` oder `en`) und das zu übersetzende Zahlwort werden auf der Kommandozeile übergeben.

Beispiel (Eingaben sind *kursiv* gedruckt):
`koller@cicero:~$` *java Uebersetzung en fuenfundzwanzig*
`twenty-five`
`koller@cicero:~$` *java Uebersetzung de thirteen*
`dreizehn`

Hinweis: Man könnte ein Array (mit Initialisierern) oder eine Map verwenden, um zwischen den Zahlwörtern und den `int`-Werten hin- und herzukonvertieren.

Bonuspunkt: Ändere das Programm so ab, dass es automatisch die Quellsprache erkennt und das Zahlwort dann in die andere Sprache übersetzt.

2 Effizienz von Listen-Implementierungen (3 Punkte)

Im Java Collections Framework gibt es verschiedene Implementierungen für das Interface `List`. Diese Implementierungen haben verschiedene Vor- und Nachteile. Insbesondere unterstützen sie bestimmte Operationen mehr oder weniger effizient; beim Programmieren denkt man deshalb darüber nach, welche Operationen das Programm häufig verwenden wird, und wählt die beste Implementierung aus.

Schreibe ein Programm, das die Effizienz der Methoden `insert`, `remove(Integer)` und `get` in den beiden Klassen `ArrayList<Integer>` und `LinkedList<Integer>` vergleicht. Für Listen mit 20.000 Elementen sollte man einen Laufzeitunterschied bemerken, der für größere Listen noch dramatischer wird.

Unter welchen Umständen würdest Du eine der beiden Implementierungen der anderen vorziehen?

3 Prioritäts-Warteschlangen (3+1 Punkte)

Eine Prioritäts-Warteschlange (engl. *priority queue*) ist eine überaus nützliche Datenstruktur, deren Einträge mit einer numerischen *Priorität* versehen sind. Eine Prioritäts-Warteschlange unterstützt die folgenden Operationen:

- `add(entry, priority)`: Fügt der Warteschlange einen neuen Eintrag mit der angegebenen Priorät hinzu.

- `removeMin()`: Gibt den Eintrag mit der kleinsten Priorität zurück und löscht ihn aus der Warteschlange.
- `isEmpty()`: Prüft, ob die Warteschlange leer ist.
- `size()`: Gibt die Anzahl der Einträge zurück.

Definiere ein Interface `IntPriorityQueue`, das Prioritäts-Warteschlangen mit Einträgen vom Typ `String` und Prioritäten vom Typ `int` unterstützt. Schreibe eine Klasse, die das Interface `IntPriorityQueue` implementiert, sowie ein Hauptprogramm, das die Klasse testet.

Hinweis: Im Java Collections Framework gibt es eine Variante des `Map`-Interfaces, die hier nützlich sein könnte.

Bonuspunkt: Zur Datenstruktur "Prioritäts-Warteschlange" gehört normalerweise auch eine Methode `decreasePriority(entry, newPriority)`, die die Priorität eines bestehenden Eintrags reduziert. Füge dem Interface und der implementierenden Klasse eine solche Methode hinzu.

Abgabe bis 7. 6. 2004, 9 Uhr
(java-uebungen@coli.uni-sb.de)