PT2: Serie 3

Abgabetermin	**22.05.2018, 21:00 Uhr**
Übungstermin	17.08.2018

Minimum einer Folge finden

Gegeben sei eine Folge ganzer, nicht negativer Zahlen. Die Anzahl der Zahlen ist im Voraus nicht bekannt. Diese Zahlen werden vom Benutzer online über die Konsole eingegeben.

Gesucht ist das Minimum der Zahlen. Das Programm wird durch Eingabe einer negativen Zahl beendet. Es soll dann das gefundene Minimum auf dem Bildschirm ausgeben.

Implementieren Sie `Minimum.java`, in dem die `main`-Methode dieses Problem löst. Zur Eingabe von Zahlen können Sie die Klasse `In.java` verwenden. Sie enthält eine Klassenmethode `getInt()`, welche eine neue Zahl von Standardeingabe einliest und als `int`-Wert zurückgibt.

Abstrakte Klasse

Erstellen Sie eine abstrakte Klasse `Conversion` mit einer Funktion `abstract double convert(double val)`.

Leiten Sie von der `Conversion`-Klasse zwei Klassen `Fahrenheit` und `Celsius` ab, in denen `convert()` so implementiert wird, dass `Fahrenheit.convert()` einen Celsius-Wert ergibt und `Celsius.convert()` einen Fahrenheit-Wert erzeugt.

Die Umrechungsformel lautet:

```
Fahrenheit = 9/5 * Celsius + 32.
```

Legen Sie die drei Klassen in separaten Quelldateien in einem Paket `TempConv` an. Schreiben Sie zwei Programme `F2C.java` und `C2F.java`, die die Kommandozeilenargumente in die jeweils andere Einheit konvertieren und zeilenweise auf die Standardausgabe schreiben.

Nutzen Sie dazu die oben genannten Klassen. Ihre `main`-Methoden sollten sich nur an einem Token unterscheiden.

Benutzerdatenbank

Gegeben sei eine Liste von Nutzernamen `passwd` (Password-Datei, latin-1-kodiert), die einen Nutzer pro Zeile enthält und die Nutzer in der Form

```
login_name:password:UID:GID:user_name:directory:shell
```

Entwickeln Sie eine Datensatzklasse `Account` in Java mit gleich benannten Feldern. Folgendes soll gelten:

- Alle Felder sind `public`. Es werden keine Getter und Setter benötigt.
- `UID` und `GID` sind ganze Zahlen; alle anderen Felder Zeichenketten.
- `Account` enthält die Methode `public static Account[] open(String filename)`. Diese liest eine Passwort-Datei, und gibt eine Liste von Account-Datensätzen in der Reihenfolge zurück, wie sie auch in der Datei stehen. Bei Scheitern des Einlesens soll `null` zurückgeben werden.
- Eine Methode `public static Account find_account(Account[] liste, String login_name);` soll das zu `login_name` passende Account-Exemplar liefern, oder `null` wenn kein Datensatz gefunden wurde.

Hinweise:

- Zum zeilenweisen Einlesen können Sie die Klasse `BufferedReader` verwenden.
- Fügen Sie die Strings/Account-Objekte zunächst in einen Container ein, etwa `Vector`, um dann ein Ergebnis-Array der passenden Größe zu produzieren.

Tests

Testen Sie Ihr Modul aus der letzten Aufgabe, indem Sie ein Programm `AccountTest.java` schreiben, welches in seiner Hauptmethode `Account.open` und `Account.find_account` aufruft und mittels `assert` folgende Testfälle prüft:

- Die UID des ersten Nutzers ist 5316.
- Der Name des letzten Nutzers ist Steffen Kensy.
- Es gibt genau einen Nutzer mit der UID 5131.
- `find_account(liste, "fwesack")` liefert einen Nutzer, dessen Verzeichnis `/home/stud/2005/fwesack` ist.
- `find_account(liste, "billg")` liefert kein Ergebnis.

Abgabe

Reichen Sie Ihre Lösung in Form eines einzelen gzip-komprimierten Tarfiles ein. Dieses soll im Wurzelverzeichnis Ihre Quelldateien, sowie ein Makefile enthalten, das mit dem Ziel `all` alle Programme zu Bytecode übersetzt.

Es sollen keine Binärdateien, Validationsskripte, Testdateien, Editorbackups, lokale Repositories, o.Ä. enthalten sein.

Die `passwd` -Datei wird bei der Verifikation automatisch in richtiger Version und Encoding angelegt.

```
/
|--Account.java
|--AccountTest.java
|--C2F.java
|--F2C.java
|--In.java
|--Makefile
|--Minimum.java
|--TempConv
   |--Celsius.java
   |--Conversion.java
   |--Fahrenheit.java
```