P. Thiemann und K. Anton Sommersemester 2010

Programmieren in Java

`http://proglang.informatik.uni-freiburg.de/teaching/java/2010/`

Betreutes Java-Programmieren 2

2010-05-03

Anfang Führen Sie folgende Schritte durch, um den PC in der Javakurs-Umgebung zu starten.

1. Rechner neu starten:
   - bei "Ankermann"-PCs Reset-Schalter am Gehäuse drücken
   - bei "Dell"-PCs Powerschalter lang drücken (aus!) dann nochmal (an!)
2. Unterste Bildschirmzeile verfolgen: da steht bei manchen "press (Taste) for boot menu". (Dell: meist F8 oder F12, Ankermann: F8 oder Esc, je nach Kaufdatum). Angegebene Taste im Sekundenrhythmus immer wieder drücken. Wenn da nichts steht, drücken Sie abwechselnd F12 und F8.
3. Im Bootmenü den Eintrag "Nvidia Boot Age" oder "Realtek Boot Age" (Ankermann) oder "Onboard Network Controller" (Dell) auswählen. Rechner sollte etwas von DHCP und TFTP erzählen.
4. Im nächsten Menü den Eintrag "Java-Programmieren (SS2010)" wählen.
5. Wenn Sie in Windows oder der gewohnten Poolumgebung landen, nochmal versuchen.

Starten und Speichern Starten Sie Eclipse, indem Sie in einem Terminalfenster `eclipse` eingeben. Wenn Eclipse sie fragt, wo der Workspace hin soll, antworten Sie bitte `/home/`*`ihrloginname`*`/workspace`, was normalerweise auch schon voreingestellt sein sollte.

Drumherum Unter der URL `http://nonopapa:8080/teaching/java/2010/` steht Ihnen das Vorlesungsmaterial zur Verfügung. Die Sun-Java-Doku gibt es unter `http://nonopapa:8080/javadoc/`

Bei Fragen zur Rechnerbenutzung, zu den Aufgaben oder zu diesen Hinweisen wenden Sie sich bitte sofort an einen der Tutoren.

Ende Sie werden nach Ablauf der Übungszeit automatisch ausgeloggt. Werden Sie früher fertig, dann loggen Sie sich bitte selbst aus.

Drücken Sie nach dem Ausloggen `Ctrl+Alt+F1`, um auf eine Textkonsole zu kommen, und dann `Ctrl+Alt+Del`, um neu zu booten.

Dieses Blatt geben Sie bitte den Tutoren zurück.

Ja, Doku. Letzte Woche war Dokumentation noch optional. Ab heute legen wir Wert auf sauber dokumentierten Code. Das heißt:

- Schreiben Sie Javadoc-Kommentare an jede Klasse, an jede Methode und an jedes Feld – sonst Punktabzug für die Teilaufgabe (und zwar saftig).
- Klassenkommentare sollen erzählen, wozu die Klasse da ist und ggf. was sie genau repräsentiert, wenn das unklar ist (Unterschied z.B. zwischen "eine Fahrt auf der Strecke von A nach B" und "die Strecke von A nach B")
- Methodenkommentare sollen erzählen, *was* die Methode erreicht ("berechnet die Fahrtkosten für diese Fahrt in Cent"), weniger, wie sie es tut ("multipliziert bla mit blub")
- Feldkommentare sollen erzählen, wozu das Feld dient und was man dabei beachten muss (z.B. Einheiten!) ("die Länge der Strecke in Zehntelkilometern").
- Konstruktorkommentare sollen alles erzählen, was man zum Verwenden des Konstruktors wissen muss (bei unseren einfachen Klassen nur: dass er eine neue Instanz anlegt).

Eclipse-Project. Legen Sie schon mal ein Projekt `abgabe2` an, in dem Sie alle Lösungen dieser Übung ablegen.

Aufgabe 1 (HD-Video, 20 Punkte(5+1+2+4+4+4)
Vergessen Sie BluRay – heute schaut man GrayRay^{TM}. Alle Klassen dieser Aufgabe gehören ins Package `grayray`.

(a) GrayRay-Filme werden nicht einmalig bezahlt, sondern pro Minute. Eine Minute kostet nur 11ct! Zudem müssen Sie bei Filmen, die länger als 120 min sind, nur die ersten 120 min bezahlen!
Schreiben Sie eine Klasse `PayPerMinuteFilm` mit einem Feld für den Namen (String) und einem für die Länge in Minuten (ganzzahlig). Lagern Sie den Preis pro Minute in ein (konstantes) Feld aus. Die Klasse soll auch eine Methode `int pricePerViewing()` enthalten, die den Preis für einmaliges Betrachten des ganzen Films ausrechnet. Testen Sie diese Methode mit drei Testfällen[1].

(b) Diese blöden Konsumenten wollen nicht pro Minute bezahlen. Der Filmvertrieb führt daher Filme ein, die einen Namen und einen festen monatlichen Mietpreis haben. Implementieren Sie das in der Klasse `MonthlyFilm`.

(c) Wir wollen Filme monatlich abrechnen: der Player erfasst, wie oft jeder Film angeschaut wurde (teilweise Anschauen kostet so viel wie ganz Anschauen, daher brauchen wir keine Minuten zu zählen) und berechnet so, wieviel für jeden Film zu zahlen ist.
Schreiben Sie ein Interface `IFilm` für Filme, die ihren Preis ausrechnen können. Es soll eine Methode `double totalPrice(int numViewings)` haben. Dieser Methode übergibt man die Anzahl der Betrachtungen des Films und bekommt den Preis in Cent zurück, der für diesen Film in diesem Monat zu zahlen ist.

(d) Lassen Sie `PayPerMinuteFilm` das Interface `IFilm` implementieren. Testen Sie die Ihre Implementation mit zwei Testfällen.

(e) Lassen Sie auch `MonthlyFilm` das Interface `IFilm` implementieren. Testen Sie die Ihre Implementation mit zwei Testfällen.

(f) Diese blöden Konsumenten finden die Filme immer noch zu teuer. Der Filmvertrieb führt daraufhin werbefinanzierte Filme ein. Man kann jeden Film auch mit einer unüberspringbaren Werbung davor kaufen, das kostet dann 20% weniger als er normal kosten würde.
Schreiben Sie eine Klasse `AdSupportedFilm`, die `IFilm` implementiert. `AdSupportedFilm` besitzt einen inneren `IFilm`, und wenn man den `AdSupportedFilm` nach seinem Preis fragt, antwortet er 80% vom Preis des inneren Films.
Testen Sie die Methoden von `AdSupportedFilm` mit zwei Testfällen.

Erinnerung: Javadoc? Überall? Gut.

[1] Also z.B. einer Testklasse mit drei Methoden.