Programmieren in Java

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

Betreutes Java-Programmieren 1 (ohne Wertung!)

2010-04-26

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.

Willkommen. In dieser Übung werden Sie erste Schritte mit Eclipse machen, ein erstes[1] Java-Programm schreiben und sich mit dem ganzen Ablauf vertraut machen. Weil es am Anfang meistens Reibungsverluste gibt, werden die Punkte für dieses Blatt zwar ermittelt, jedoch nicht gewertet.

Aufgabe 1 (Hallo Welt, 4 Punkte)
Kurzer Rundgang durch die Bedienung von Eclipse.

(a) Erstellen Sie ein Java-Projekt namens `abgabe1` (die Standardeinstellungen dürfen Sie beibehalten). Erstellen Sie in dem Projekt eine Java-Klasse `Hello` im Package `hello` (Rechtsklick aufs Project, "New...", "Class", dann im Dialog die Felder "Package" und "Class" ausfüllen und mit "Finish" bestätigen) Speichern Sie die Klasse.

(b) Fügen Sie folgenden Code in die Klasse (zwischen die geschweiften Klammern) ein:

```
1    public static void main(String[] args) {
2        System.out.print("Hello, ");
3        System.out.println("World!");
4    }
```

Speichern Sie mit `Ctrl+S` oder dem Speichern-Button der Toolbar. Eclipse kompiliert Ihr Java-Programm jedesmal, wenn Sie speichern. Lassen Sie das Programm laufen, indem Sie auf die Klasse (links im Baum) rechtsklicken und "Run as... Java Application" ausführen. Beobachten Sie die Console-View.

(c) Erzeugen Sie nun im Package `hello` eine Klasse `World` mit einer Methode `name()`, die immer den Wert `"Erde"` zurückliefert.

[1] Wenn Sie schon Eclipse und Java kennen, um so besser!

(d) Schreiben Sie einen Testfall für `World`, der `name()` testet: Rechtsklick aufs Project, "New...", "JUnit Test case". Im Dialog "new Junit 4 test case" anwählen, Package `hello` eintragen, Name `TestWorld` eintragen, "Finish" wählen. Wenn Eclipse Sie fragt, "Junit 4 is not on the build path", lassen Sie Junit zum Build-Path hinzufügen. Dorthinein eine Methode

```
1    @Test
2    public void testName() {
3        World w = new World();
4        assertEquals("Erde", w.name());
5    }
```

Da werden noch Namen nicht gefunden! Cursor auf die unterstrichene Stelle, Ctrl+1 drücken, Import hinzufügen lassen. Laufenlassen per "Run As.../JUnit Test".

Aufgabe 2 (Klassenmodellierung, 10(2+2+2+2+2) Punkte)
Wir modellieren Heißgetränkeautomaten. Ein Rezept besteht aus:

- Menge des Wassers (ganzzahlig, in Millilitern)
- Menge des Kaffeepulvers (reell, in Gramm)
- Menge des Milchpulvers (reell, in Gramm)

Bei Java gelten **Namenskonventionen**: Klassennamen `GrossUndZusammen`, Instanzvariablennamen `kleinUndZusammen`. Halten Sie sie bitte von nun an ein (sonst Punktabzug).

(a) Implementieren Sie dieses Design als Java-Klasse `Rezept` im Package `kaffee` im Projekt `abgabe1`.

(b) Erstellen Sie nun nach Vorbild der Hilfsklasse `EntryExample` eine Klasse `KaffeeExample`, in der Sie Rezept-Instanzen `espresso` (100ml, 7g Kaffee, 0g Milchpulver) und `latte` (180ml, 8.5g Kaffee, 20g Milchpulver) anlegen.

```
1 public class EntryExample {
2     Wurst salami = new Wurst(200, 56.5, "Salami");
3     Wurst leberwurst = new Wurst(150, 60.0, "Grobe Kalbsleberwurst");
4 }
```

(c) Zu Stammkunden wissen wir folgendes: Ein Stammkunde hat einen Namen (vom Typ `String`) und ein Stammgetränk (vom Typ `Rezept`). Implementieren Sie das in einer Klasse `Stammkunde`.

(d) Erweitern Sie die Hilfsklasse `KaffeeExample` aus (b) um die Stammkunden `arnie` ("Arnie Altbier", trinkt Espresso), `babsi` ("Babsi Broccoli", trinkt Latte), `coco` ("Coco Cramer", trinkt ein Spezialrezept mit 20g Kaffee auf 80ml Wasser und 1.5g Milchpulver).

(e) Erweitern Sie `Rezept` um eine Methode `materialKosten`, die die Materialkosten des Rezepts in Cent berechnet. Und zwar kostet jedes Gramm Kaffeepulver 1.2ct und jedes Gramm Milchpulver 0.3ct; Wasser können Sie mit konstant 1ct ansetzen (egal wie viel Wasser).

Testen Sie diese Methode mit zwei Testfällen. Beachten Sie, dass `double`s Rundungsfehler ansammeln; es gibt aber `assertEquals(x,y,delta)`, damit Sie eine Toleranz `delta` angeben können.

(f) (Bonusaufgabe für Unterforderte, 2P) Die `Rezept`-Klasse hat keine Getter, keine Setter, kein Javadoc, kein `hashCode`, kein sinnvolles `equals` und kein `toString`. Rüsten Sie nach.

Aufgabe 3 (Freie Klassenmodellierung, 6(4+2) Punkte)
Aus der Anforderungsspezifikation eines fiktiven Navigationsprogramms: *Personen haben einen Namen sowie eine dienstliche und eine private Adresse. Fahrtrouten verbinden einen Anfangsort und einen Zielort, beides gegeben durch Adressen.*

(a) Identifizieren Sie im obigen Satz Klassen, Attribute und Beziehungen. Die Anforderungen sind vage – machen Sie sinnvolle Annahmen darüber, welche Attribute für GPS-Navigation nötig sind. Implementieren Sie die so entstandenen Klassen in Java im Package `navi` Ihres Projekts.

(b) Erzeugen Sie (wieder in einer `EntryExample`-ähnlichen Hilfsklasse) Instanzen Ihrer Klassen, die zusammen Arnie Altbier und eine Fahrt von seinem Wohnort zu seinem Arbeitsplatz darstellen.