P. Thiemann und K. Anton Sommersemester 2010

Programmieren in Java

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

Betreutes Java-Programmieren 3

2010-05-10

Anfang Loggen Sie sich in der normalen Linux-Umgebung ein.

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 Erzeugen Sie ein ZIP- oder tgz-Archiv Ihres Eclipse-Projects: Rechtsklick aufs Project, "Export...", im Ast "General" die Zeile "Archive File" auswählen, prüfen, dass links oben genau Ihr Project angekreuzt ist, im Feld "To archive file" einen Dateinamen eingeben, der Ihren Namen enthaelt: zB `/home/mustermm/Desktop/abgabe3-mustermm.zip` für Max Mustermann, "Finish" klicken. Dabei sollte ein ZIP-Archiv enstehen, das Sie am Ende dem Tutor mailen.

Dieses Blatt geben Sie bitte den Tutoren zurück.

Eclipse-Project. Heute brauchen Sie das Eclipse-Projekt nicht selbst anzulegen – Sie starten mit einem halbfertigen Projekt. Laden Sie zunächst `http://nonopapa:8080/teaching/java/2010/supp/skel3.zip` herunter. Dann Kontextmenü im Package-Explorer aufmachen, "Import...", "General" aufklappen, "Existing projects into workspace" auswählen, "Next" klicken, oben "Select Archive File" auswählen, "Browse" drücken, das runtergeladene `skel3.zip` suchen, dann "Finish" klicken. Sie sollten jetzt ein Project "abgabe3" haben.

Aufgabe 1 ((2+4+3+4+4+3) Punkte)

Heute basteln wir an einem Aufbaustrategiespiel. Es gibt eine Landschaft, die in Felder (`ITerrainTile`) aufgeteilt ist. Auf Feldern können Gebäude stehen (`IBuilding`). Der Spieler bekommt für jedes seiner Gebäude Prestigepunkte, die für die einzelnen Gebäudearten unterschiedlich ausgerechnet werden. In jeder Runde des Spiels produzieren oder konsumieren Gebäude die Ressourcen Eisen und Kohle und erwirtschaften oder verbrauchen Geld.

Wir werden zunächst das Kohlebergwerk (`CoalMine`), die Eisenhütte (`Ironworks`) und die Universität (`University`) modellieren. Alle Klassen kommen in das **Package** `world`.

(a) Das Interface `IBuilding` ist schon fertig. Implementieren Sie die Klasse `University`, die dieses Interface implementiert. Eine Universität bringt 50 Prestigepunkte. Testen Sie das.

(b) Bei allen Produktionsgebäuden berechnet sich das Prestige aus der Kohleproduktion und der Eisenproduktion, aber nur der positiven – wenn ein Gebäude eine Ressource netto *verbraucht*, wird diese Ressource nicht betrachtet. Für jede produzierte Kohleeinheit gibt es 2 Prestigepunkte, für jede produzierte Eiseneinheit 3 Prestigepunkte.

Erstellen Sie eine *abstrakte* Klasse `AProductionBuilding`, die zwei abstrakte Methoden `int ironProduction()` und `int coalProduction()` hat. Konkrete Unterklassen sollen darin ihre Eisen- und Kohleeinheiten berechnen und zurückliefern (positiv: produziert, negativ: konsumiert). `AProductionBuilding` implementiert obendrein `IBuilding` und berechnet das Prestige nach der oben genannten Regel.
Der Mechanismus mit den fixen Kohle-/Eisenproduktionswerten geht nur Produktionsgebäude etwas an und soll ohne großen Aufwand geändert werden können. Verhindern Sie mit geeigneten Maßnahmen, dass die Entwickler anderer Komponenten direkt den Kohle-/Eisenproduktionswert auslesen.

(c) Eisenhütten konsumieren immer 2 Kohleeinheiten und produzieren 5 Eiseneinheiten. Implementieren Sie die Klasse `Ironworks` als Subklasse von `AProductionBuilding` und testen Sie die Prestigeberechnung.

(d) Kohlebergwerke produzieren 3 Kohleeinheiten, wenn sie in der Ebene stehen, und 10, wenn sie auf einem Berg stehen. Das können Kohlebergwerke herausfinden, weil sie in einem Feld eine Referenz auf ihr Terrain haben (vom Typ `ITerrainTile`). Implementieren Sie `CoalMine` als Subklasse von `AProductionBuilding`.

(e) Um das Kohlebergwerk zu testen, brauchen Sie eine Implementation von `ITerrainTile`. Die "große" Implementation von `ITerrainTile` kennt vielleicht noch Straßennetz und so, was uns nur im Weg herumstehen würde. Stattdessen bauen wir eine Stub[1]-Implementation `TerrainTileStub` von `ITerrainTile`, die einfach ein Feld für Berge-ja/nein hat. Testen Sie damit `CoalMine`.

(f) Eigentlich war das Spiel friedlich gedacht – egal, jetzt soll es doch noch Krieg geben. Jedes Gebäude hat einen Verteidigungswert, der angibt, wie schwierig es zu stürmen ist. Machen Sie an `IBuilding` noch eine Methode `int defenseValue()` dran, die den Verteidigungswert in Verteidigungspunkten zurückliefern soll. Alle Zivilgebäude (dazu zählen Unis, Bergwerke und Eisenhütten) haben 0 Verteidigungspunkte. Machen Sie es sich leicht. Testen Sie es aber für jede der drei bekannten Gebäudeklassen.

[1] d.h. Stumpf