
Programmieren in Java

<http://proglang.informatik.uni-freiburg.de/teaching/java/2013/>

Java-Übung Blatt 13 (Mini-Projekt)

2013-07-23

Hinweise

- Schreiben Sie Identifier *genau so*, wie sie auf dem Blatt stehen (inklusive Groß- und Kleinschreibung), nicht nur ungefähr.
- Identifier und Kommentare bitte auf *englisch*!
- Schreiben Sie *sinnvolle* Kommentare
- Laden Sie Ihre Lösungen mit subversion (svn) ins Übungssystem hoch. Den entsprechenden Pfad finden Sie online.
- Das Übungssystem kann überprüfen, ob Sie Ihr Quelltext den Anforderungen genügt und ob Sie alle Klassen erstellt haben, etc. Nutzen Sie dies!
- Sollte das Übungssystem Ihre Lösungen ablehnen, dann werden sie *nicht* korrigiert! Akzeptanzkriterien:
 - Compiliert erfolgreich
 - Checkstyle bringt keine Fehler
 - Alle Packages, Klassen, Interfaces, Methoden, Typen, Argumente sind exakt wie auf dem Übungsblatt gefordert.
- Ihr korrigierender Tutor wird die Korrektur Ihrer Abgabe in Ihr svn-Repository unter dem Namen `Feedback-<login>-ex<XX>.txt` comitten. (<login> ist dabei Ihr myAccount name und <XX> die Kennziffern des Übungsblatts)
- Zusätzlich können Sie Ihre Gesamtpunktzahl im Übungsportal einsehen.

Abgabe: Freitag, 16. August 2013, um 23.59 Uhr.

Hinweis: Bitte unbedingt Blatt komplett durchlesen und dann erst arbeiten, da spätere Teilaufgaben die Herangehensweise ggf. beeinflussen können.

Tutorat: Falls Sie Fragen haben nutzen Sie das Forum, oder kommen Sie bei den Tutoraten vorbei. Es werden folgende zwei Tutorate stattfinden: 30.07. 14-16 Uhr und 06.08. 14-16 Uhr jeweils im Pool.

In diesem Übungsblatt programmieren wir das Kartenspiel „Autoquartett“ (engl. „Supercars“). Die Regeln sind wie folgt:

Kartendeck Ein Deck besteht aus mehreren Karten, welche jeweils verschiedene Automobile anhand ihrer technischen Daten beschreiben. Auf allen Karten eines Decks ist die gleiche Art technischer Daten (wie Hubraum, Höchstgeschwindigkeit, usw.) angegeben.

Startkonfiguration Zu Beginn werden die Karten eines Decks Karten gemischt und dann den Spielern gleichmäßig verteilt. Jeder Spieler hat dann einen Stapel von Karten. Sollte die Anzahl der Karten nicht gleichmäßig verteilt werden können, werden nur so viele Karten vom Kartendeck verwendet, dass es gerade noch aufgeht. Die übrigen Karten werden nicht genutzt. Das Los entscheidet einen Startspieler, der als erster am Zug ist.

Ablauf

1. Der Spieler, der am Zug ist, sucht sich eine Eigenschaft (z.B. Hubraum) und nennt den angegebenen Wert auf seiner obersten Karte (z.B. 2,3 Liter).
2. Die anderen Spieler nennen die Werte derselben Eigenschaft ihrer obersten Karte.
3. Der Spieler mit dem höchsten Wert gewinnt die Runde und bekommt die obersten Karten aller anderen Spieler und fügt sie, zusammen mit seiner Gewinner-Karte, seinem Kartenstapel am Ende an. Der Rundengewinner ist dann als nächstes am Zug.

4. Gibt es keinen eindeutigen Maximalwert, dann wählt der aktuelle Spieler eine neue Eigenschaft aus bis es einen eindeutigen Maximalwert gibt (wir gehen davon aus, dass das immer möglich ist).
5. Hat ein Spieler keine Karten mehr auf der Hand, hat er verloren und steigt aus dem Spiel aus.

Spielende Gewonnen hat der Spieler, der am Ende als Einziger übrig bleibt.

Achtung: Sie brauchen nur **eine** der folgenden Aufgaben zu bearbeiten.

Aufgabe 1 (Swing Supercars, 33 Punkte)

Projekt: `ex13_1`. Package: `supercars.swing`.

Implementieren Sie ein Supercars Spiel basierend auf einer Swing-Benutzeroberfläche. Benutzen Sie dazu ein fest programmiertes Kartendeck mit mindestens 10 Autos. Das Spiel soll wahlweise mit 2 oder 3 Teilnehmern ablaufen; dies soll der Benutzer nach dem Starten des Spiels in einem Dialog festlegen können. Die Darstellung des Spielzustands und der Karten, sowie Eingabe der Spielerentscheidungen soll mit Swing-GUIs geschehen. Sie können davon ausgehen, dass immer nur der Spieler am Computer sitzt, dessen Name gerade als aktueller Spieler angezeigt wird.

Achten Sie auf gutes Design. Extrahieren Sie aus den oben gegebenen Regeln ein Datenmodell und implementieren Sie dann die Spiellogik. Versuchen Sie die Darstellung modular und getrennt von der Spiellogik zu implementieren.

Aufgabe 2 (Custom Supercars, 33 Punkte)

Projekt: `ex13_2`. Package: `supercars.custom`.

Implementieren Sie ein Konsolen-basiertes Supercars Spiel, das benutzerdefinierte Kartendecks unterstützt. Das Spiel besteht aus 3 festen Spielern mit festen Namen und beginnt mit einem zufällig ausgewählten Spieler. Die Darstellung des Spielzustandes und der obersten Karte des aktuellen Spieler soll auf der Konsole erfolgen. Auch die Eingabe der Spielentscheidung (welche Eigenschaft gewählt werden soll) soll über die Konsole abgefragt werden.

Das zu verwendende Kartendeck soll aus einer Datei gelesen werden, die als Kommandozeilenargument der `main`-Methode angegeben wird. Eine Beispieldatei in zwei Formaten (XML oder Plaintext) finden Sie in der Projektvorlage. Wie sie das Lesen der Karten implementieren bleibt Ihnen überlassen. Vorschläge:

- XStream zur Deserialisierung verwenden. Dafür muss Ihre Datenstruktur in etwa der Hierarchie in der XML-Datei entsprechen. Mit den `alias` Methoden der XStream Klasse können Sie kleine Änderungen vornehmen.
- Sie nutzen die Java IO Klassenhierarchie (z.B. `BufferedReader`), `String.split` oder die `Scanner` Klasse und lesen die Felder direkt ein.

Achten Sie auf gutes Design. Extrahieren Sie aus den oben gegebenen Regeln ein Datenmodell und implementieren Sie dann die Spiellogik. Versuchen Sie die Darstellung auf der Konsole modular und getrennt von der Spiellogik zu implementieren.