Programmieren 2 Labor - WiSe 2017/18
Fakultät Informatik - UIB
Prof. Dr. Frank Dopatka
Abgabe bis 17.11.2017

hochschule mannheim

Übungsblatt 2: Vertiefte Objektorientierung & Testen

25 Punkte möglich

Aufgabe 1: Aus dem Skript

[2 Punkte] Liefern Sie als Gruppe alle Quellcodes aus dem Skript, Kapitel 2 und 3 in einem Eclipse-Workspace als ZIP-Datei bei Prof. Dopatka ab. Der Eclipse-Workspace muss eine wohl strukturierte Package-Struktur aufweisen. Sind Quellcodes fehlerhaft oder unvollständig, so erhalten Sie keine Punkte.

Aufgabe 2: Live-Coding

Bei der Abnahme des Aufgabentyps „Live-Coding“ erstellen Sie an einem einzigen PC bei Prof. Dopatka Quellcode ohne weitere Hilfsmittel. Der Eclipse-Workspace muss geöffnet, aber leer sein. Weitere Dokumente, online oder in Papierform, sind untersagt. Prof. Dopatka prüft anhand dieses Tests, ob er Ihrem Kleinunternehmen[1] einen folgenden Großauftrag vergibt oder ob er Ihr Kleinunternehmen dazu nicht in der Lage sieht. Den geforderten Quellcode erstellen Sie zügig live während der Abgabe und erklären währenddessen Ihre Vorgehensweise. Prof. Dopatka kann alternative Vorgehens-weisen zur Lösung des Problems fordern.

2a) [3 Punkte] Ein Auto besteht aus einem Motor und 4 Rädern. Sie können dem Auto über eine separate Test-Klasse mit der main-Methode den Befehl **`fahre()`** geben. Das Auto gibt dann dem Motor den Befehl **`starte()`**, welcher für jedes Rad den Befehl **`drehe()`** absetzt.
Realisieren Sie die Abhängigkeit zwischen Auto, Motor und Rad als Komposition.
Welche Art der Komposition Sie realisieren sollen, entscheidet Prof. Dopatka zu Beginn der Abnahme durch Würfeln:

- Realisierung der Komposition:
  - 1-2: über Package-Sichtbarkeit
  - 3-4: über den Konstruktor des Teils
  - 5-6: über innere Member-Class

[1] Ihr Kleinunternehmen ist Ihre Praktikumsgruppe. Sobald Sie etwas abgeben, ist Prof. Dopatka Ihr Großkunde. Treten Sie also entsprechend freundlich, team-fähig, organisiert und sehr gut vorbereitet gegenüber dem Großkunden auf, denn die Existenz Ihres Kleinunternehmens ist von diesem Großkunden abhängig. Sie sind Dienstleister des Großkunden!

2b) [3 Punkte] Erstellen Sie in der main-Methode einen regulären Ausdruck, der eine als String vorgegebene Telefonnummer auf Gültigkeit prüft. Dabei sollen mehrere Arten der Eingabe erlaubt sein. Welche konkreten Eingabe-Arten erlaubt sein sollen, und ob der Ausdruck vorkompiliert wird oder nicht, entscheidet Prof. Dopatka zu Beginn der Abnahme durch Würfeln:

- Art des regulären Ausdrucks:
  - 1-3: regulärer Ausdruck muss vorkompiliert werden
  - 4-6: regulärer Ausdruck darf nicht vorkompiliert werden

- Ländercodes:
  - 1-3: exemplatisch
    +49 International nach Deutschland
    +43 International nach Österreich
    +41 International in die Schweiz
    +61 International nach Australien
    +1 International nach Nordamerika
  - 4-6: Ländercodes allgemein:
    + gefolgt von 1-4 Ziffern ohne führende 0

- Ortsnetzkennzahlen
  - 1-3: exemplatisch:
    030 Berlin
    0221 Köln
    0621 Mannheim
    0711 Stuttgart
    089 München
  - 4-6: allgemein:
    3 bis 5 Ziffern, immer mit führender 0

- Teilnehmerrufnummern:
  immer 3 bis 8 Ziffern ohne führende 0

Die Ländercodes, Ortsnetzkennzahlen und Teilnehmerrufnummern können aber zusätzlich noch auf verschiedene Weise angegeben werden. Die folgenden Beispiele von internationalen Rufnummern enthalten den Ländercode 49, die Ortsnetzkennzahl 30, die Teilnehmerrufnummer 12345 und die Durchwahl 67. Bedenken Sie auch, dass es zwar eine Durchwahl geben kann, aber nicht muss.

- Angabe der Syntax
  - 1-2: gemäß DIN 5008
    +49 30 12345-67
  - 3: gemäß E.123:
    +49 30 1234567
  - 4: gemäß RFC 3966:
    +49-30-1234567
  - 5-6: weit verbreitet, aber nicht standardkonform:
    +49 (0)30 12345-67

Aufgabe 3: Das Spiel

3a) [2 Punkte] Richten Sie das Projekt MADN in Eclipse so ein, dass es über GIT mit allen Teilnehmern Ihrer Praktikumsgruppe synchronisiert wird. Jeder Teilnehmer hat zuvor ein persönliches GIT-Konto für sich selbst einzurichten. Händigen Sie dem Kunden Ihren Namen, Ihren GIT-Profilnahmen sowie Ihren Repository-Link aus. Diese Daten sind gegenüber den anderen Gruppen strikt geheimzuhalten! Weisen Sie dem Kunden nach, dass Sie GIT im Team beherrschen incl. Merging. **Vor allen folgenden Abgaben** muss die aktuelle Version Ihrer Software in GIT vorhanden sein, der **Kunde schaut sich Ihren Code ab jetzt nur noch im GIT an**!

3b) [3 Punkte] Erstellen Sie für jede Basisklasse **`Würfel`**, **`Spielfigur`**, **`Spielbrett`**, **`Spielfeld`** und **`Spieler`** aus dem letzten Übungsblatt jeweils 5 sinnvolle JUnit-Tests. Unter welchen Voraussetzungen ist der TDD-Ansatz erfüllt?

3c) [2 Punkte] Die Klasse **`Spiel`** implementiert nun ein Interface **`iBediener`**, welches Sie sinnvoll definieren müssen. Es werden ausschließlich primitive Typen sowie String-Objekte übergeben. Alle Interaktionen der Benutzer müssen ab jetzt über dieses Interface erfolgen. Testen Sie über das Interface mit 10 sinnvollen JUnit-Tests.

3d) [2 Punkte] Überschreiben Sie die Methode **`toString()`** der **`Spiel`**-Klasse so, dass sie das aktuelle Spielbrett mit der Belegung der Figuren so als Zeichenkette zurückgibt, dass der aktuelle Spielstatus graphisch erkennbar ist, vgl. [1]. Außerdem soll ausgegeben werden, welcher Spieler als Nächstes an der Reihe ist. Weisen Sie die Funktionalität über den Debugger nach.

Aufgabe 4: Die Künstliche Intelligenz (KI)

4a) [5 Punkte] Erstellen Sie eine abstrakte Klasse **`KI`**, die ein spezieller Spieler ist. Leiten Sie von dieser abstrakten Klasse die Klassen **`KI_Aggressiv`** und **`KI_Defensiv`** ab. Objekte von beiden Klassen sollen sich so verhalten wie menschliche Spieler. Die beiden künstlichen Intelligenzen sollen etwas unterschiedliche Prioritäten im Spiel haben:

- KI_Aggressiv (1 hat stets höchste Priorität):
  1. In jedem Fall einen Gegner schlagen, wenn es möglich ist.
  2. Alle Figuren von den Startfeldern ins Spiel bringen, wenn eine 6 gewürfelt wurde.
  3. Seine eigenen Figuren auf die Endfelder zu bringen.
- KI_Defensiv (1 hat stets höchste Priorität):
  1. Seine eigenen Figuren möglichst schnell auf die Endfelder zu bringen.
  2. Alle Figuren von den Startfeldern ins Spiel bringen, wenn eine 6 gewürfelt wurde.
  3. Einen Gegner schlagen, wenn es möglich ist.

Die Vorgehensweise der KI ist über konkrete Testklassen und Testszenarien nachzuweisen.

4b) [2 Punkte] Erstellen Sie ein präzises UML Aktivitätsdiagramm für die beiden konkreten KIs. Zeigen Sie insbesondere auf, wie Entscheidungen zustande kommen.

4c) [1 Punkt] Dokumentieren Sie die fertige Klasse **`KI`** und deren beide abgeleitete Klassen vollständig in JavaDoc. Jeder Parameter und jede Funktion ist ordentlich und vollständig zu beschreiben. Generieren Sie die JavaDoc bei der Abnahme live neu und präsentieren Sie das Ergebnis Ihrem Kunden.

[1] http://www.linux-community.de/var/ezwebin_site/storage/images/internal/artikel/print-artikel/linuxuser/2003/12/ascii-grafiken-erstellen-mit-cadubi-figlet-und-boxes/abbildung-4/1239793-1-ger-DE/Abbildung-4_large.png