Einzelbeispiel (SEPM/JAVA), Sommersemester 2015

Bitte lesen Sie dieses Dokument aufmerksam und bis zum Ende durch, bevor Sie mit der Arbeit beginnen. Nur so können Sie sicherstellen, dass Sie die gesamte Angabe verstanden haben! Weiters existiert ein Leitfaden zum Lösen des Einzelbeispiels sowie das Bewertungsschema, die beide unbedingt zu berücksichtigen sind.

Um an der Gruppenphase der Lehrveranstaltung Software Engineering & Projektmanagement teilnehmen zu können, muss das Einzelbeispiel, welches **selbständig** zu entwickeln ist, erfolgreich gelöst werden.

Zu verwendende Technologien

Programmiersprache	Java 1.8u31
UI-Framework	JavaFX 8
Datenbank	H2
Logging	Log4j2
Test-Framework	JUnit 4.x
Versionierung	Subversion (SVN)

Als Entwicklungsumgebung empfehlen wir IntelliJ IDEA - Community Edition (https://www.jetbrains.com/idea/download/) in Kombination mit dem JavaFX SceneBuilder 2.0 (https://www.oracle.com/technetwork/java/javase/downloads/javafxscenebuilder-info-2157684.html). IntelliJ steht im Informatiklabor zur Verfügung und unsere Tutoren bieten hierfür Unterstützung an.

Tutorbetreuung während der Eingangsphase

Bei Fragen und Unklarheiten während der Eingangsphase stehen Ihnen Tutoren per TUWEL Diskussionsforum zur Verfügung. Zusätzlich gibt es betreute Laborzeiten, bei denen Tutoren im Informatiklabor anwesend sind, um vor Ort Probleme zu lösen.

Hinweis:
Je genauer Sie hier Ihre Fragen formulieren, desto schneller kann Ihnen geholfen werden.

Labor Fragestunden Termine und Anwesenheitszeiten finden Sie im TUWEL.

TUWEL Auf unserer E-Learning Plattform stellen wir ein moderiertes Diskussionsforum zur Verfügung, das von unseren Tutoren regelmäßig betreut wird.

1 Voraussetzungen

Wir gehen davon aus, dass Sie die Inhalte der Lehrveranstaltungen **PP** und **PK** (Strukturiertes Programmieren, Grundlagen der OOP), **ALGODAT** (Basiswissen der verschiedenen Datenstrukturen, Funktionsweise und Anwendung), **OOP** und **OOM** (Objektorientierte Programmierung und Objektorientierte Modellierung) sowie **Datenmodellierung** verstehen und auch praktisch anwenden können.

2 Angabe

Für *Wendy's Rennpferde* soll eine Softwarelösung zur Verwaltung und Durchführung von Pferderennsimulationen entwickelt werden. Pferde und Jockeys können angelegt werden um anschließend eine Rennsimulation durchzuführen. Derzeit werden die Pferde und Jockeys auf Papierlisten verwaltet und die Berechnungen werden händisch durchgeführt. Dies soll durch eine Softwarelösung ersetzt werden.

Die folgenden Funktionen sind zu implementieren:

- Verwaltung von Pferden und Jockeys
- Durchführen von Rennsimulationen
- Einfache statistische Auswertung der Rennen

2.1 Verwaltung von Pferden und Jockeys

Pferde und Jockeys können über eine eigene Benutzeroberfläche getrennt verwaltet werden.

2.1.1 Pferde und Jockeys erfassen

Ermöglichen Sie es, in Ihrem Programm, Pferde und Jockeys **über entsprechende Formulare in der Datenbank abspeichern** zu können. Achten Sie hier auf einen **logischen und strukturierten Aufbau** der Eingabeformulare, auf **Fehlerbehandlung** von Eingabewerten sowie auf die Darstellung wichtiger Formularelemente (z.B. Überschriften, Pflichtfelder, Datumsfelder,...). Zur besseren Orientierung soll ein Foto des Pferdes gespeichert werden, welches in mindestens einer Ansicht angezeigt wird. Die erfassten Daten (Geschwindigkeit, Können, usw.) werden später für die Durchführung der Rennsimulationen benötigt.

2.1.2 Pferde und Jockeys suchen

Erstellen Sie eine Suchfunktion, die es ermöglicht folgende Aktionen durchzuführen:

- **Anzeige aller Datensätze**
- **Einschränkung der Datensätze** über mindestens ein Attribut (nicht Primärschlüssel)

2.1.3 Pferde und Jockeys bearbeiten

Mit Ihrem Programm soll es möglich sein, bestehende **Pferde und Jockeys in der Datenbank zu verändern**. Achten Sie auf einen **strukturierten Aufbau des Formulars**, die **Fehlerbehandlung von Eingabedaten** sowie auf die Darstellung wichtiger Formularelemente, wie Pflichtfelder oder eventuell nicht veränderbare Felder.

2.1.4 Pferde und Jockeys löschen

Achten Sie hier im Speziellen auf die **Programmlogik** und verhindern Sie **Löschanomalien** (z.B.: Wenn ein Pferd gelöscht wird, was geschieht dann mit den bereits existierenden Rennen des Pferdes?).
Der Benutzer soll **Feedback über diese Aktionen**, beispielsweise durch ein **Dialogfenster**, erhalten. Beim Löschen sollte zudem eine Bestätigung vom Benutzer eingefordert werden (z.B.: Wollen Sie dieses Pferd wirklich löschen?).

2.2 Durchführen von Rennsimulationen

Bei einem Rennen können beliebig viele Pferd-Jockey-Kombinationen teilnehmen. Natürlich können ein Pferd und ein Jockey jeweils nur einmal gleichzeitig an einem Rennen teilnehmen. Sobald die Simulation gestartet wird, wird für jede am Rennen teilnehmende Pferd-Jockey-Kombinationen die Durchschnittsgeschwindigkeit berechnet. Die Pferd-Jockey-Kombination mit der höchsten Durchschnittsgeschwindigkeit ist somit der Sieger des Rennens. Eine einmal durchgeführte Rennsimulation soll in der Datenbank gespeichert werden und lässt sich nach der Durchführung nicht mehr ändern. Jede Rennsimulation besteht aus einer eindeutigen Nummer und den Simulationsdaten. Sonst sollen in einer Rennsimulation keine weiteren Daten gespeichert werden.

Die Durchschnittsgeschwindigkeit einer Pferd-Jockey-Kombination berechnet sich aus den Daten des Pferdes (Maximal- und Minimalgeschwindigkeit eines Pferdes), dem Können des Jockeys und einem gewissen Glücksfaktor.

Die Formel zur Berechnung der Durchschnittsgeschwindigkeit d sieht folgendermaßen aus:

$p : p \in \mathbb{R},\ p \geq 50,\ p \leq 100$
: Zufallszahl zwischen min. und max. Geschwindigkeit des Pferdes in km/h.
Wobei gilt: min. und max. Geschwindigkeit müssen zwischen 50 und 100 km/h liegen.

$k : k \in \mathbb{R}$
: Können des Jockeys zwischen $-\infty$ und $+\infty$. 0 ist Ausgangswert.

$g : g \in \mathbb{R},\ g \geq 0.95,\ g \leq 1.05$
: Glücksfaktor, Zufallszahl zwischen 0.95 und 1.05.

$$k_{berechnet} = 1 + \left(0.15 * \frac{1}{\pi} * \arctan\left(\frac{1}{5} * k\right)\right)$$

$$d = p * k_{berechnet} * g$$

Das Ergebnis der Rennsimulation soll in Form einer, nach Rang sortierten, Liste ausgegeben werden. Jeder Listeneintrag soll dabei mindestens den Namen des Pferdes sowie des Jockeys, die zufällige Geschwindigkeit des Pferdes p, den zufälligen Glücksfaktor g, das berechnete Können des Jockeys $k_{berechnet}$ sowie die finale Durchschnittsgeschwindigkeit enthalten.

Die korrekten Ergebnisse einer Rennsimulation müssen jederzeit abrufbar sein. Außerdem soll es möglich sein, alle Rennsimulationen aufzulisten und diese Liste nach Jockey, Pferd und Rennnummer zu filtern.

Achten Sie bei Ihrer Implementierung im Speziellen auf anwendungsfallspezifische Einschränkungen und sinnvolle Überprüfungen der Eingaben.

2.3 Statistische Auswertung

Für die Analyse der Pferde und Jockeys soll es möglich sein, eine einfache statistische Auswertung zu machen. Dabei soll man die Statistik für ein Pferd, einen Jockey sowie die Kombination aus Pferd und Jockey erstellen lassen können. Aus der Statistik soll hervorgehen, welcher Platz jeweils wie oft belegt wurde.

Die Ausgabe der Statistik muss mindestens textuell erfolgen. Natürlich ist auch eine grafisch Ausgabe, z.B. mit Charts http://docs.oracle.com/javafx/2/charts/jfxpub-charts.htm möglich.

Wichtig: Die berechnete Statistik muss nicht in der Datenbank gespeichert werden!

3 Anforderungen an die Implementierung

3.1 Datenbankstruktur

Verwenden Sie **3 Datenbanktabellen** mit je mindestens **4 Attributen** und den dazu sinnvoll passenden Datentypen. Weiters sind **sinnvolle Primär- und Fremdschlüssel** zu definieren. Beachten Sie hierbei eine **sinnvolle Namensgebung** und die Abdeckung aller zu speichernden Informationen.

3.2 Fehlerbehandlung

Behandeln Sie mögliche **Eingabefehler und interne Fehler** folgendermaßen:

- abfangen und behandeln
- das **Programm terminiert nicht** unerwartet
- es gibt **geringstmögliche Einschränkungen** im Programmablauf
- in **verständlicher Form** anzeigen (etwa **Dialogfenster**)

3.3 JUnit Tests

Erstellen Sie zu **einem** Ihrer **DAOs** für **jede Methode** (*create*, *read*, *update* und *delete*, ...) **mindestens 2 Tests**, die die korrekte Funktionsweise dieser Methode, sowohl im **Normalfall** als auch im **Fehlerfall** (gewünschtes Abbruchverhalten), sicherstellen.
Erstellen Sie weiters **mindestens 2 Tests**, welche Methoden überprüfen, die Teil der **Serviceschicht** sind. Hier soll Funktionalität getestet werden, die nicht mit dem Speichern, Laden, Aktualisieren oder Löschen einer Entität zu tun hat. Achten Sie auch hier auf die Erstellung von Tests für den **Normalfall** als auch den **Fehlerfall**.

3.4 Layout

Wichtige Merkmale einer guten GUI sind einerseits eine **geeignete Anordnung** der Komponenten und andererseits **klare Beschriftungen**. Achten Sie beim Design auf eine möglichst **einheitliche Linie**. Finden Sie **passende GUI-Elemente** zu den jeweiligen Datentypen (z.B.: boolesche Werte mit Checkboxen).

4 Dokumentation

4.1 Codedokumentation

Die **Codedokumentation** erfolgt im Code selbst (JavaDoc). Dokumentieren Sie mindestens alle Interfaces sowie komplexere Algorithmen und Formeln.

*Wichtig: Dokumentation im **Code** (JavaDoc) bitte **nicht ausdrucken!***

4.2 Produktdokumentation

Die **Produktdokumentation** ist bei der Abgabe in **ausgedruckter, gehefteter** und strukturierter Form abzugeben und soll nachfolgende Bestandteile beinhalten:

- **Titelseite** mit Name, Matrikelnummer, E-Mail und dem aktuellen Datum
- **Stundenliste** mit Datum, Start, Dauer, Tätigkeit und **abschließender Summe**
- **Domänenmodell** in UML Klassendiagrammnotation (konzeptionelle Sichtweise)
- **UML Anwendungsfalldiagramm** der Anforderungen
- **Anwendungsfallbeschreibungen** zum Anwendungsfalldiagramm aus Anwendersicht

5 Ablauf der Abgabe

5.1 Vorbedingungen

- Sie sind im Anmeldetool (SAT) zu einem **Abgabegespräch angemeldet**.
- Ihr Projekt ist vollständig (Dokumente, Programmdateien und Quelltext; kein Sourcecode von Libraries!) im SVN vorhanden.
  Nur Code und Dokumentation, die im SVN liegen, werden für die Bewertung herangezogen.
- Zur Sicherheit ist auch ein Backup des Projekts zur Abgabe mitzubringen (am eigenen Laptop und/oder USB Stick).

5.2 Ablauf des Abgabegesprächs

1. Live-Beispiel (Programmieraufgabe)
   - **Selbständiges Lösen einer Programmieraufgabe** (max. 60min)
   - **Automatisierte Bewertung**
2. Abgabe des Einzelbeispiels
   - **Produktcheck**: Der Tutor überprüft die **Produktdokumentation** und lässt sich das Programm samt Datenbank ausführlich erklären und vorführen.
     Hinweis: Gerne können Sie Ihr Produkt auf Ihrem eigenen Laptop präsentieren.
   - **Verständnisfragen**: Der Tutor überprüft das Verständnis des Studierenden zum vorliegenden Produkt und zum Entwicklungsprozess.
   - **Prüfen der Kenntnisse der Entwicklungsumgebung**
3. Nach dem Abgabegespräch
   - **Bewertung des Einzelbeispiels** durch den Tutor
   - Erklärung der **weiteren Vorgehensweise** durch den Tutor
   - Klärung etwaiger **Fragen**

5.3 Wichtige Hinweise zur Abgabe

- **Plagiate werden ausnahmslos negativ beurteilt!**
- Achten Sie auf **Vollständigkeit** der Funktionalität Ihres Programms.
- Das **grafische Interface** Ihrer Implementierung muss **nicht händisch programmiert** werden, wir raten Ihnen zur Verwendung des JavaFX Scene Builder.
- Während der Abgabe müssen Sie den **Sourcecode** Ihres Programms an beliebigen Stellen jederzeit und ohne Vorbereitungszeit flüssig **erklären** können.
- Das Programm muss am Abgaberechner ausführbar sein.
- Die **Datenbank** muss mit einer entsprechenden Anzahl an realistischen **Testdatensätzen** befüllt sein (**zumindest 10 Stück** pro Tabelle)

Allgemeiner Hinweis Beginnen Sie **möglichst frühzeitig mit der Bearbeitung** des Beispiels und melden Sie sich auch **rechtzeitig für ein Abgabegespräch** an. Nutzen Sie auch die **Hilfestellungen** durch das **moderierte Forum im TUWEL** und durch die **Tutoren**.

Viel Spaß und Erfolg beim Einzelbeispiel aus der SE&PM Laborübung!

Deadline: Sonntag, 12. April 2015, 23:55