Einzelbeispiel (SEPM/JAVA), Wintersemester 2014

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 zur Umsetzung 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

Change compiler in Eclipse to 8, double check that FX 8 is also installed.

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

Als Entwicklungsumgebung (IDE) empfehlen wir Eclipse (http://www.eclipse.org) in Kombination mit dem JavaFX SceneBuilder (http://www.oracle.com/technetwork/java/javase/downloads/javafxscenebuilder-info-2157684.html). Eclipse 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 Informatik Labor 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 Reitstall* soll eine Softwarelösung zur Vereinfachung des Buchungsprozesses entwickelt werden. Kunden können eine Reitstunden mit einem bestimmten Pferd buchen. Derzeit werden die Buchungen über einen Papierkalender und Listen erfasst, welche durch eine Softwarelösung ersetzt werden sollen.

Die folgenden Funktionen sind zu implementieren.

- Verwaltung von Pferden
- Buchungsprozess
- Buchungsbestätigung

2.1 Verwaltung von Pferden

Die Pferde können über eine eigene Benutzeroberfläche erfasst werden.

2.1.1 Pferde erfassen

Ermöglichen Sie es, in Ihrem Programm, Pferde **über ein Formular 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 für die Kunden und das Personal soll ein Foto des Pferdes gespeichert werden, welches in mindestens einer Ansicht angezeigt wird. Die erfassten Daten werden später für die Buchungsbestätigung verwendet. Die Eingabe des Pferdes muss aus mindestens fünf Eingabefeldern (=Attributen; z.B.: Größe, Geburtsdatum, Gewicht,...) bestehen.

2.1.2 Pferde suchen

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

- **Anzeige aller Datensätze**
- **Einschränkung der Datensätze** über Attribute (beispielsweise mittels einer Kombination aus Textfeld und DropDown-Feldern, die eine erweiterte Suche ermöglichen)

2.1.3 Pferde bearbeiten

Mit Ihrem Programm soll es möglich sein, bestehende **Pferde 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 löschen

Achten Sie hier im Speziellen auf die **Programmlogik** und verhindern Sie **Löschanomalien** (z.B.: Ein Pferd wird gelöscht, was geschieht mit existierenden Buchungen und Buchungsbestätigungen, die dieses Pferd enthalten?).
Der User soll **Feedback über diese Aktionen**, beispielsweise durch ein **Dialogfenster**, erhalten. Beim Löschen sollte dies zweimal passieren; einmal vor (z.B.: "Wollen Sie dieses Pferd wirklich löschen?") und einmal nach dem Löschen (z.B.: "Das Pferd wurde erfolgreich gelöscht.").

2.2 Buchungsprozess

Jedes Pferd kann, sofern es zu dem Zeitpunkt noch frei ist, durch einen Kunden gebucht werden. Ein Kunde kann gleichzeitig mehrere Termine buchen. Bei jeder Buchung wird der Name des Kunden vermerkt. Die Termine einer Buchung können **bis 2 Wochen vor dem jeweiligen Termin** verändert/storniert werden. Ihre Implementierung muss die nachfolgenden Funktionen bereitstellen:

- Gleichzeitige Buchung verschiedener Termine
- Auflistung über alle Buchungen in Listenform, inkl. Möglichkeit zur Einschränkung des Anzeigezeitraums (Auflistung aller Buchungen im Zeitraum von A bis B)
- Markierung eines nicht mehr veränderbaren/stornierbaren Termins
- Einzelne Buchungen anzeigen (inkl. Auflistung von: Name des Kunden, Name der Pferde, etc.)

Achten Sie bei Ihrer Implementierung im Speziellen auf Anwendungsfall-spezifische Einschränkungen und sinnvolle Überprüfungen der Eingaben (z.B. soll es nicht möglich sein ein Pferd mehrfach zur selben Zeit zu buchen).

2.3 Buchungsbestätigung

Dem Kunden wird auf Wunsch eine Bestätigung der Buchung ausgestellt. Die Buchungsbestätigung beinhaltet das aktuelle Datum, Name des Kunden, die einzelnen gebuchten Pferde sowie die Termine. Stornierte Buchungen sollen hier auch aufgeführt und als solche ersichtlich sein. Die Anzeige der Buchungsbestätigung kann beliebig einfach gestaltet werden.
Es muss zu jedem Zeitpunkt möglich sein, eine korrekte Buchungsbestätigung auszugeben.

3 Anforderungen an die Implementierung

3.1 Datenbankstruktur

Verwenden Sie **zumindest 2 Datenbanktabellen** mit je mindestens **5 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 4 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

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

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

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.
   - **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: 27. Oktober 2014, 23:55