SE Praktikum Aufgabe 3: Torben Haug, Tim Hagemann

# Komponentendiagram



User Story

Student kann sich am System anmelden

Use case

Titel: Single-Choice-Lernkarten beantworten  
Akteur: Student  
Ziel: Überprüfung des Wissens

Auslöser:

Ein Student will eine Lernkarte bearbeiten.

Vorbedingungen:

Student besitzt ein HAW-Konto.  
Es wurde mindestens eine Lernkarte von einem Professor eingetragen.

Nachbedingungen:

Student hat eine Lernkarte bearbeitet.

Erfolgsszenario:

1. Student öffnet die SoLe-App.
2. System zeigt die Anmeldeseite.
3. Student meldet sich mit seiner HAW-Kennung an.
4. System prüft die Anmeldedaten. -> IAccountingRegistry
5. System zeigt das Hauptfenster an.
6. Student wählt das Lernen aus.
7. System zeigt die Lernseite an.
8. Student wählt ein Fach und Fragen aus.
9. System speichert die Auswahl von Fach und Fragen. -> IPruefungControl
10. Student startet die Übung.
11. System zeigt die nächste Frage an. -> GUI
12. Student gibt sein Antwort an und bestätigt diese.
13. System gibt die Richtigkeit dieser aus. ->IUebung
14. Student wählt „Next“.
15. Solange noch Fragen sind, gehe zu 11.

Fehlerfälle:

4a) Das System lehnt die Anmeldung ab. Der Student wird aufgefordert seine Anmeldedaten noch einmal einzugeben. Zurück zu Punkt 2.

# Abgedeckte Anforderungen:

* Generell: A06 a)
* 4 -> A21
* 8, 9 -> A17
* 13 -> A16, A08

# Sequenzdiagramm



# TestFile

package se1\_aufgabe3;

public class UebungTest

{

IAccountingRegistry m\_accountingRegistry;

ICardArchive m\_cardArchive;

IAnswerRegistry m\_answerRegistry;

IPruefungControl m\_pruefungControl;

@Before

public void setup()

{

DatabaseConnection.getInstance().connect("jdbc:mysql://localhost:3306/sole", "root", "");

this.m\_accountingRegistry = new AccountingRegistry();

this.m\_cardArchive = new CardArchive(this.m\_accountingRegistry);

this.m\_answerRegistry = new AnswerRegistry(this.m\_accountingRegistry, this.m\_cardArchive);

this.m\_pruefungControl = new PruefungControl(this.m\_answerRegistry);

}

@Test

public void testDurchlauf()

{

IFach fach = this.m\_cardArchive.getFachByName("Software Engineering");

IUebung uebung = this.m\_pruefungControl.createFor(this.m\_accountingRegistry.getStudentMitKennung("abq264"), fach, fach.getLernkarten());

int[] results = { 0, 0, 0 };

while(uebung.canContinue())

{

uebung.getNextLernkarte();

AnswerResult result = uebung.setAnswer(0);

results[result.ordinal()]++;

}

Assert.assertTrue(results[0] == 3);

Assert.assertTrue(results[1] == 2);

Assert.assertTrue(results[2] == 0);

}

}

# Komponenteninterfaces

/\*\*

\* Zentrale Lernkartenverwaltungskomponente

\*

\*/

**public** **interface** ICardArchive {

/\*\*

\* gibt ein Fach anhand seines Namens zurueck

\*

\* **@param** inName

\* **@return** IFach

\*/

IFach getFachByName(String inName);

/\*\*

\* Gibt alle registrierten Fächer zurueck

\*

\* **@return** Collection<IFach>

\*/

Collection<IFach> getFaecher();

/\*\*

\* Erstellt und speichert ein Fach

\* **@param** inName

\* **@return**

\*/

IFach createFachWithName(String inName);

/\*\*

\* Gibt alle Lernkarten zurueck

\*

\* **@return** Collection<ILernkarte<? extends IAntwort>>

\*/

Collection<ILernkarte<? **extends** IAntwort>> getLernkarten();

/\*\*

\* Erstellt eine neue Antwort

\*

\* **@param** type

\* **@return** IAntwort

\*/

IAntwort createAntwort(Antwort.AntwortType type);

/\*\*

\* Erstellt aus einer Collection von Lernkarten CardSelection

\*

\* **@param** inFach - Fach das dieser Auswahl zugeordnet werden soll

\* **@param** inSelection - Liste von lernkarten die der Selection zugeordnet werden soll.

\* **@return** ICardSelection

\*/

ICardSelection createSelection(IFach inFach, Collection<ILernkarte<? **extends** IAntwort>> inSelection);

/\*\*

\* Prototype Generator fÜr Antworten

\*

\* **@param** inKarte - Bearbeitete Lernkarte

\* **@param** inValue - Antwort wert des Users

\* **@return**

\*/

<T **extends** IAntwort> T prepareAnswer(ILernkarte<T> inKarte, Object inValue);

}

/\*\*

\*Antwort Verwaltungskomponente

\*

\*/

**public** **interface** IAnswerRegistry

{

/\*\*

\* Gibt alle Antworten zurück, die ein Student jemals abgegeben hat

\*

\* **@param** inStudent

\* **@return**

\*/

**public** Collection<IAbgegebeneAntwort> getAntwortenVonStudent(IStudent inStudent);

/\*\*

\* Gibt die nächste Abgegebene Antwort, die noch nicht durch das System oder einen

\* Mitarbeiter geprüft wurde, zurück

\*

\* **@return**

\*/

**public** IAbgegebeneAntwort getNaechsteUngepruefteAntwort();

/\*\*

\* Gibt eine Collection von Antworten zurück, die von einem bestimmten Mitarbeiter geprüft wurden

\*

\* **@param** inMitarbeiter

\* **@return**

\*/

**public** Collection<IAbgegebeneAntwort> getGepruefteAntwortenVon(IMitarbeiter inMitarbeiter);

/\*\*

\* Speichert eine gegebene Antwort.

\*

\* **@param** inStudent - Der die lernkarte bearbeitet hat

\* **@param** inLernkarte - Lernkarte die bearbeitet worden ist

\* **@param** inAntwort - Antwort, die gegeben wurde

\*/

**public** <T **extends** IAntwort> **void** saveAnswer(IStudent inStudent, ILernkarte<T> inLernkarte, IAntwort inAntwort);

}

/\*\*

\* Zentrale Benuterverwaltung.

\* Interagiert mit LDAP der HAW

\*/  
**public** **interface** IAccountingRegistry

{

/\*\*

\* Prüft die Logindaten und gibt den dazugehörigen User zurück

\* **@param** inUser - Username

\* **@param** inPassword - Password

\* **@return** IPerson

\* **@throws** InvalidCredentialsException

\*/

**public** IPerson loginAs(String inUser, String inPassword);

}

/\*\*

\* Verwaltet laufende Pruefungen und Uebungen von Studenten

\*/

**public** **interface** IPruefungControl

{

/\*\*

\* Erstellt eine neue Übung/PrÜfung fÜr einen Studenten

\*

\* **@param** inStudent - Eingelogter Student

\* **@param** inFach - Ausgwähltes Fach

\* **@param** inLerkarten - Ausgewählte Lernkarten

\* **@return** IUebung

\*/

IUebung createFor(IStudent inStudent, IFach inFach, Collection<ILernkarte<? **extends** IAntwort>> inLerkarten);

}

# Entity Interfaces

/\*\*

\* UbungsKlasse, stellt die Funktionen für eine laufende Übung bereit

\*

\*/

**public** **interface** IUebung

{

**void** addLernkarten(ICardSelection inCardSelection);

IStudent getStudent();

**boolean** isOver();

/\*\*

\* Prueft ob noch weitere unbeantwortete Lernkarten vorhanden sind.

\*

\* **@return**

\*/

**boolean** canContinue();

**boolean** isUebung();

/\*\*

\* Gibt die nächste noch nicht beantwortete Lernkarte zurÜck

\*

\* **@return**

\*/

ILernkarte<? **extends** IAntwort> getNextLernkarte();

Collection<IAntwort> getAntworten();

/\*\*

\* Überträgt die Antwort des Nutzers an die Übung

\*

\* **@param** inAntwort - Antwort des Nutzers

\* **@return**

\*/

AnswerResult setAnswer(Object inAntwort);

Map<ILernkarte<? **extends** IAntwort>, IAntwort> getAnswersForLernkarten();

}

/\*\*

\* Lernkartenklasse

\*/  
**public** **interface** ILernkarte<T **extends** IAntwort>

{

String getFrage();

IProfessor getErsteller();

/\*\*

\* Gibt die Musterlösung zurück

\* **@return**

\*/

T getAntwort();

AntwortType getType();

ArrayList<IAnhang> getAnhaenge();

UUID getId();

IFach getFach();

/\*\*

\* Breiten Konversionsmethode von Object auf Antwort

\* Mögliche Eingaben sind:

\* - ISingleChoice

\* - IMultiChoice

\* - Freitext

\* - Bild

\* - Audio

\* **@param** inValue - gegebene Antwortmöglichkeit

\* **@return** - IAntwort oder Null, wenn das Object keine gültige Antwortmöglichkeit war

\*/

T prepareAnswer(Object inValue);

}

/\*\*

\* SingleChoiceAntwortKlasse

\*/

**public** **interface** ISingleChoiceAntwort

{

/\*\*

\* Gibt eine Liste der moeglichen Antworten zurueck

\* **@return**

\*/

List<IAntwortMoeglichkeit> getAntwortMoeglichkeiten();

**void** addAntwortMoeglichkeit(String inAntwort, **boolean** inIstRichtig);

**void** removeAntwortMoeglichkeit(IAntwortMoeglichkeit inMoeglichkeit);

IAntwortMoeglichkeit getRichtigeAntwort();

}