Klausurprüfung (Fachtheorie) aus Programmieren und Software Engineering

im Haupttermin 2021/22

für die Höhere Lehranstalt für Informatik
(SFKZ 8738)

Liebe Prüfungskandidatin,
liebe Prüfungskandidat!

Bitte füllen Sie zuerst die nachfolgenden Felder in Blockschrift aus, bevor Sie mit der Arbeit beginnen.

Maturaaccount (im Startmenü sichtbar):	
Vorname	
Zuname	
Klasse	

Klausurprüfung aus Fachtheorie

Für die 5AHIF, 5BHIF, 5CHIF, 5EHIF am 9. Mai 2022

Generelle Hinweise zur Bearbeitung

Die Arbeitszeit für die Bearbeitung der gestellten Aufgaben beträgt 6 Stunden (360 Minuten). Die 3 Teilaufgaben sind unabhängig voneinander zu bearbeiten, Sie können sich die Zeit frei einteilen. Wir empfehlen jedoch eine maximale Bearbeitungszeit von 1.5 Stunden für Aufgabe 1, 1.5 Stunden für Aufgabe 2 und 3 Stunden für Aufgabe 3. Im Beurteilungsblatt können Sie die Anforderungen für die jeweilige Beurteilungsstufe nachlesen.

Sie können die Aufgabenstellung im Gesamten entweder mit .NET oder in Java lösen.

Hilfsmittel

.NET

In der Datei *SPG_Fachtheorie.sln* befindet sich das Musterprojekt, in dem Sie Ihren Programmcode hineinschreiben. Im Labor steht Visual Studio 2022 mit der .NET Core Version 6 zur Verfügung.

Mit der Software DBeaver oder SQLite Studio können Sie sich zur generierten Datenbank verbinden und Werte für Ihre Unittests ablesen.

Zusätzlich wird ein implementiertes Projekt aus dem Unterricht ohne Kommentare bereitgestellt, wo Sie die Parameter von benötigten Frameworkmethoden nachsehen können.

Java

Ein Musterprojekt *at.spengergasse.fachtheorie* steht Ihnen zur Verfügung, in dem Sie Ihren Programmcode hineinschreiben. Im Labor steht für die Entwicklung JetBrains IntelliJ zur Verfügung. Die Pakete Lombock sowie das Spring Framework sind als Dependency installiert.

Teilaufgabe 1: Persistence - Erstellen von Modelklassen

Ein Aufnahmesystem für BewerberInnen

An unserer Schule müssen BewerberInnen für einen Schulplatz in den künstlerischen Abteilungen (Abteilung für Medien - Animation, Abteilung für Medien - Gamedesign und Abteilung für Kunst - Interiordesign) sich einem Eignungstest unterziehen. Durch die COVID Beschränkungen wird dieses Aufnahmeverfahren elektronisch durchgeführt.

Das Verfahren hat folgenden Ablauf:

- BewerberInnen melden sich im Aufnahmesystem mit ihren Daten an.
- BewerberInnen müssen Arbeiten zu drei vorgegebenen Themen hochladen.
- Nach dem Hochladen der Arbeiten werden die BewerberInnen zu einen Meeting eingeladen, um die Arbeiten mit den PrüferInnen zu diskutieren.
- Danach wird die Feststellung bestanden oder nicht bestanden getroffen.

Erforderliche Daten

Damit das System arbeiten kann, müssen folgende Daten in der Applikation abrufbar sein.

Daten zur Abteilung (Department)

Es ist der Name der Abteilung (z. B. Abteilung für Medien - Animation) zu speichern. Der Name muss eindeutig sein und darf auch nicht leer sein.

Daten zur Aufgabenstellung (Task)

Pro Abteilung werden drei Aufgabenstellungen definiert. Sie bestehen aus einem Text, einem Wert *DateFrom* und einem Wert *DateTo*. Die Datumswerte definieren, in welchem Zeitraum die BewerberInnen die Arbeiten dazu hochladen können.

Daten zum Upload

Wenn der/die BewerberIn die Dateien zu einem Bestimmten Task hochlädt (es können auch mehrere Dateien eingereicht werden), dann wird dies mit einem Zeitstempel und der URL der hochgeladenen Datei erfasst.

Daten zu den Bewerbern (Applicant)

Die BewerberInnen werden mit Vorname, Nachname, Geburtsdatum, Email Adresse und Telefonnummer erfasst. Das Geschlecht wird in Form einer enumeration abgebildet und ist für die korrekte Anrede auf der Portalseite nötig.

Der Applicant durchläuft mehrere Stati, die zusätzliche Informationen beinhalten können (*ApplicantStatus*). Zu Beginn hat der Applicant den Status *Enrolled* (eingeschrieben). Werden die Daten

hochgeladen, wechselt der Status auf *UploadsDone*. Wird ein/e BewerberIn nach dem Meeting bewertet, so werden die Felder *RatedDate* (Datum der Bewertung) und ein Flag *Passed* (true/false) gespeichert.

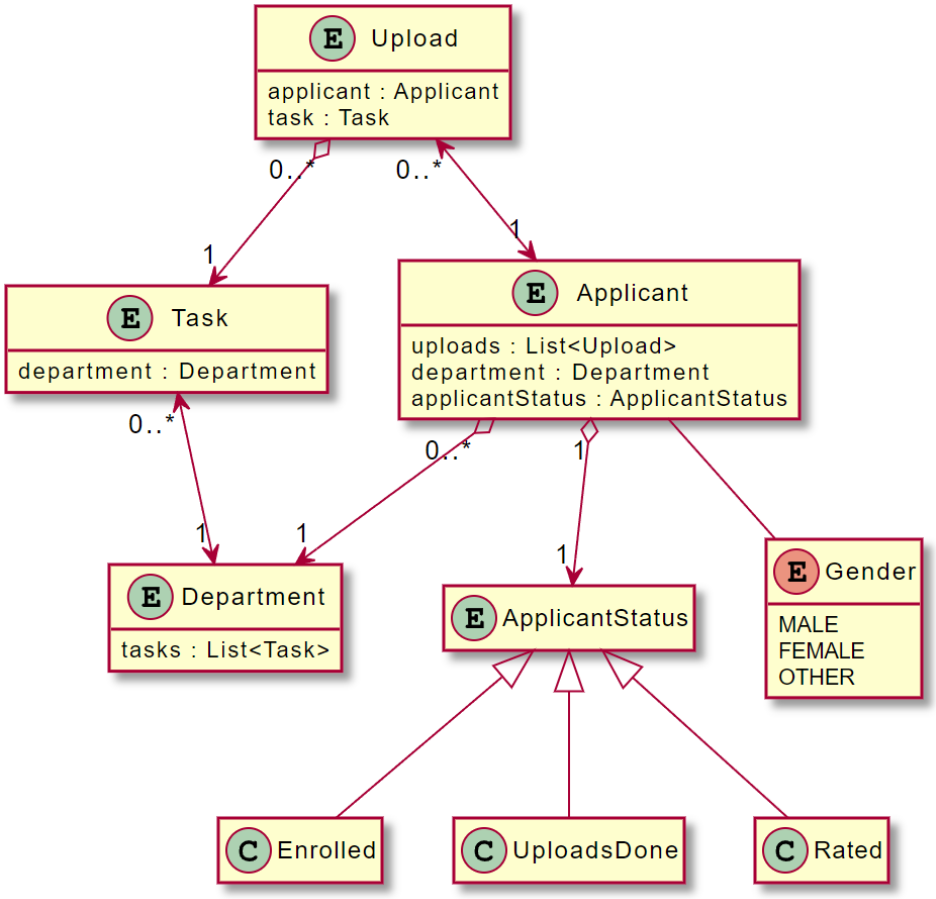
Arbeitsauftrag

Erstellen Sie Modelklassen, aus denen der OR Mapper (EF Core oder JPA) eine Datenbank erzeugen kann. Achten Sie dabei auf naming conventions der entsprechenden Programmiersprache (Großschreibung, ...).

Als Hilfestellung sind die Klassen und ihre Verknüpfungen im nachfolgenden Diagramm beschrieben. Die abgebildeten Navigations müssen sich auch in Ihrem Klassenmodell wiederfinden. Fügen Sie alle notwendigen Felder hinzu, die die oben beschriebenen Daten speichern können. Definieren Sie selbst sinnvolle Datentypen und Längenbegrenzungen.

Danach implementieren Sie den Datenbankkontext (EF Core) bzw. die Repositories (JPA). Zur Kontrolle schreiben Sie pro Entityklasse einen Unittest, der einen Datensatz mit allen erforderlichen Feldern in die Datenbank einfügt. Das ergibt also 5 Unittests in Summe.

Legen Sie erforderliche Primärschlüssel sowie notwendige Konstruktoren, Annotations, ... selbst fest.



Bewertung

Für jede erfolgreich gelöste Aufgabe der nachfolgenden Liste gibt es 1 Punkt. Es sind in Summe also 15 Punkte zu erreichen.

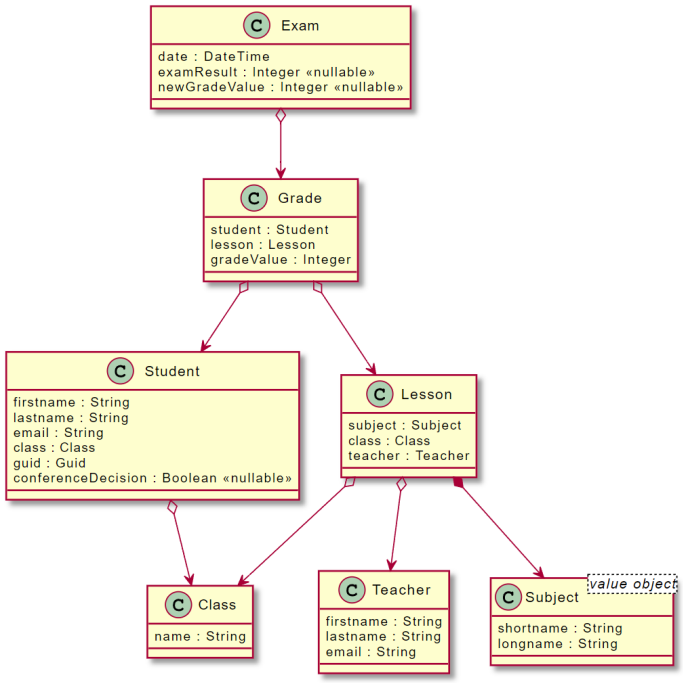
- Die Klasse Department bildet alle in der Angabe beschriebenen Daten ab.
- Die Klasse Department besitzt den notwendigen Aufbau (Annotations, Konstruktoren, ...) für die Persistierung mit dem OR Mapper.
- Ein Test *AddDepartmentSuccessTest()* beweist, dass ein Datensatz der Klasse Department in die Datenbank geschrieben werden kann.
- Die Klasse Task bildet alle in der Angabe beschriebenen Daten ab.
- Die Klasse Task besitzt den notwendigen Aufbau (Annotations, Konstruktoren, ...) für die Persistierung mit dem OR Mapper.
- Ein Test *AddTaskSuccessTest()* beweist, dass ein Datensatz der Klasse Task in die Datenbank geschrieben werden kann.
- Die Klasse Applicant bildet alle in der Angabe beschriebenen Daten ab.
- Die Klasse Applicant besitzt den notwendigen Aufbau (Annotations, Konstruktoren, ...) für die Persistierung mit dem OR Mapper.
- Ein Test *AddApplicantSuccessTest()* beweist, dass ein Datensatz der Klasse Applicant in die Datenbank geschrieben werden kann.
- Die Klasse Upload bildet alle in der Angabe beschriebenen Daten ab.
- Die Klasse Upload besitzt den notwendigen Aufbau (Annotations, Konstruktoren, ...) für die Persistierung mit dem OR Mapper.
- Ein Test *AddUploadSuccessTest()* beweist, dass ein Datensatz der Klasse Upload in die Datenbank geschrieben werden kann.
- Die Klasse ApplicantStatus sowie ihre Subklassen bilden alle in der Angabe beschriebenen Daten zum entsprechenden Bewerberstatus ab.
- Die Klasse ApplicantStatus besitzt den notwendigen Aufbau (Annotations, Konstruktoren, ...) für die Persistierung mit dem OR Mapper.
- Ein Test *AddApplicantWithApplicantStatusSuccessTest()* beweist, dass ein Datensatz der Klasse Applicant mit dem Status Rated in die Datenbank geschrieben werden kann.

Teilaufgabe 2: Services

Eine Verwaltung für Wiederholungsprüfungen

Wird ein/e SchülerIn im Jahreszeugnis negativ beurteit, so hat er bzw. sie das Recht auf eine Wiederholungsprüfung im Herbst. Dies ist bei einem oder zwei Nicht genügend im Jahreszeugnis möglich. In den letzten Jahren wurde jedoch eine Sonderregelung ausgegeben, wonach sich SchülerInnen mit mehr als zwei Nicht genügend zu bis zu zwei Wiederholungsprüfungen anmelden konnten.

Bei einer Schule unserer Größe ist hierfür eine elektronische Lösung nötig. Das Klassenmodell zur Umsetzung dieses Problems ist nachfolgend abgebildet. Es befindet sich bereits fertig implementiert im Musterprojekt. Diese Klassen sind also nicht von Ihnen selbst zu implementieren, Sie müssen sie nur verwenden.



Klasse Lesson

Pro Klasse werden nur bestimmte Gegenstände unterrichtet. Die Klasse *Lesson* speichert jeden unterrichteten Gegenstand und welche Lehrkraft diesen Unterrichtet. In unserem Modell unterrichtet nur eine Lehrkraft einen Gegenstand pro Klasse. Diese Lehrkraft ist auch der/die PrüferIn bei der Wiederholungsprüfung.

Klasse Student

Wie der Name schon sagt ist diese Klasse für die SchülerInnendaten da. Das
Feld *conferenceDecision* gibt an, ob die Klassenkonferenz für einen Aufstieg trotz negativer Noten gestimmt hat. Es ist NULL, wenn (noch) kein Beschluss gefasst wurde.

Klasse Grade

Speichert die Zeugnisnote. *gradeValue* kann 1, 2, 3, 4 oder 5 annehmen. Eine negative Beurteilung ist dann gegeben, wenn die Note gleich 5 ist. Ansonsten ist die Beurteilung positiv.

Klasse Exam

Speichert die Anmeldung zu einer Prüfung. Die Felder *examResult* und *newGradeValue* werden erst nach der Prüfung befüllt und haben zum Zeitpunkt der Anmeldung den Wert NULL. *examResult* speichert die Note auf die Prüfung (1 - 5), *newGradeValue* die neu festzusetzende Zeugnisnote (3 - 5).

Die Datenbank speichert nur ein Schuljahr und wird jedes Jahr neu befüllt.

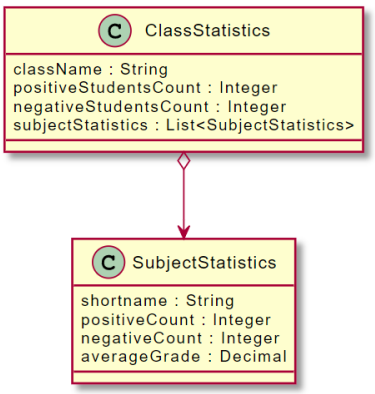
Arbeitsauftrag

Im Projekt wird ein Mockup der jeweiligen Repositories mit Musterdaten (Java) bzw. ein DbContext mit einer Seed Methode zur Verfügung gestellt. Sie können daher auf diese Musterdaten zugreifen.

Erstellen Sie eine Klasse *GradeService*. Die Klasse *GradeService* soll die nachfolgenden Abfragen bzw. Operationen als Methoden beinhalten.

Klassenstatistik ermitteln: GetClassStatistics

Schreiben Sie eine Methode *ClassStatistics GetClassStatistics(string class)*. Diese Methode soll eine Statistik über die Leistungen einer Klasse liefern. Der Rückgabetyp *ClassStatistics* ist nach folgendem Muster zu definieren:



positiveStudentsCount gibt an, wie viele SchülerInnen der Klasse keine Note gleich 5 in der *Grade* Tabelle eingetragen haben. *negativeStudentsCount* gibt an, wie viele SchülerInnen der Klasse mindestens eine Note gleich 5 in der *Grade* Tabelle eingetragen haben.

subjectStatistics ist eine Berechnung pro Gegenstand der Klasse. Die Gegenstände einer Klasse sind in der *Lessons* Tabelle aufgelistet. *positiveCount* gibt an, wie viele SchülerInnen in diesem Gegenstand in der angegebenen Klasse eine positive Note (kleiner als 5) haben. Bei *negativeCount* werden die Anzahl der Noten gleich 5 gewählt. *averageGrade* gibt den Mittelwert aller Noten der angegebenen Klasse in diesem Gegenstand an.

Es ist aus Zeitgründen *kein Unittest* für diese Methode zu schreiben.

Eine Anmeldung zur Prüfung einfügen: TryAddRegistration

Schreiben Sie eine Methode *bool TryAddRegistration(Student student, string subjectShortname, DateTime date)*. Diese Methode soll einen Exam Datensatz ohne *examResult* und *newGradeValue* in die Datenbank schreiben. Die Methode soll sich so verhalten:

1. Liefert *false*, wenn der Gegenstand in der Klasse des Schülers nicht vorhanden ist.
2. Liefert *false*, wenn der Gegenstand positiv beurteilt wurde.
3. Liefert *false*, wenn der Gegenstand bereits als Exam eingetragen wurde.
4. Liefert *false*, wenn an diesem Tag bereits eine Prüfung eingetragen wurde.
5. Liefert *true*, wenn die obigen Prüfungen erfolgreich waren und der Datensatz in die *Exam* Tabelle eingetragen werden konnte.

Die Richtigkeit soll mittels Unittest belegt werden. Fügen Sie dafür einen Schülerdatensatz in die Datenbank ein. Der Schüler soll Beurteilungen in 2 Gegenständen besitzen (eine positive und eine negative Beurteilung). Danach schreiben Sie folgende Unittests:

- TryAddRegistrationReturnsFalseIfSubjectDoesNotExist prüft, ob (1) in der obigen Liste erfüllt wurde.
- TryAddRegistrationReturnsFalseIfSubjectIsNotNegative prüft, ob (2) in der obigen Liste erfüllt wurde.
- TryAddRegistrationReturnsFalseIfExamExists prüft, ob (3) in der obigen Liste erfüllt wurde.
- TryAddRegistrationReturnsFalseOnDateConflict prüft, ob (4) in der obigen Liste erfüllt wurde.
- TryAddRegistrationReturnsSuccessTest prüft, ob (5) in der obigen Liste erfüllt wurde.

Bewertung

Für jeden korrekt gelösten Eintrag in der nachfolgenden Liste gibt es einen Punkt. In Summe sind also 16 Punkte zu erreichen.

GetClassStatistics (6 P)

- Die Klasse *ClassStatistics* wurde gemäß der Vorgabe korrekt erstellt.
- Die Methode ermittelt korrekt die positiven SchülerInnen der übergebenen Klasse.
- Die Methode ermittelt korrekt die negativen SchülerInnen der übergebenen Klasse.
- Die Methode ermittelt korrekt die positiven SchülerInnen jedes Gegenstandes der Klasse.
- Die Methode ermittelt korrekt die negativen SchülerInnen jedes Gegenstandes der Klasse.
- Die Methode ermittelt korrekt den Mittelwert der Note jedes Gegenstandes der Klasse.

TryAddRegistration (10 P)

- Die Methode prüft, ob der Gegenstand in der Klasse des Schülers vorhanden ist.
- Die Methode prüft, ob der Gegenstand auch negativ beurteilt wurde.
- Die Methode prüft, ob der Gegenstand nicht schon als Exam eingetragen wurde.
- Die Methode prüft, ob der Tag nicht schon vergeben wurde.

- Die Methode schreibt den Datensatz korrekt in die Datenbank.
- Der Unittest *TryAddRegistrationReturnsFalseIfSubjectDoesNotExist* ist richtig gestaltet (Arrange - Act - Assert).
- Der Unittest *TryAddRegistrationReturnsFalseIfSubjectIsNotNegative* ist richtig gestaltet (Arrange - Act - Assert).
- Der Unittest *TryAddRegistrationReturnsFalseIfExamExists* ist richtig gestaltet (Arrange - Act - Assert).
- Der Unittest *TryAddRegistrationReturnsFalseOnDateConflict* ist richtig gestaltet (Arrange - Act - Assert).
- Der Unittest *TryAddRegistrationReturnsSuccessTest* ist richtig gestaltet (Arrange - Act - Assert).

Teilaufgabe 3: Webapplikation

Für die Wiederholungsprüfungen in Aufgabe 2 soll ein Front End für die Klassenvorstände geschrieben werden. Dabei sind 2 Themen abzudecken: Die Klassenkonferenzen und die Eingabe der Anmeldung.

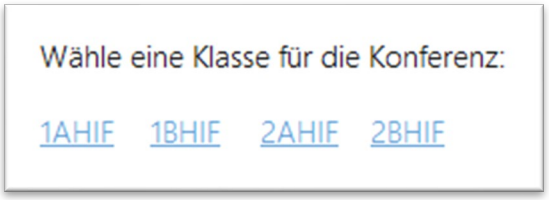
Es wird auf das Klassenmodell samt Musterdaten aus Aufgabe 2 verwiesen. Da dies bereits implementiert vorgegeben ist, ist diese Aufgabe unabhängig von Aufgabe 2 zu lösen.

Klassenkonferenzen

In den letzten 2 Jahren konnten SchülerInnen mit mehr als einem Nicht genügend auf Beschluss der Konferenz auch ohne Wiederholungsprüfung aufsteigen. Dieser Sachverhalt soll implementiert werden. Es werden folgende Seiten definiert:

Seite /Konferenz/Index

Diese Seite listet alle Klassen im System auf. Die entsprechenden Links verweisen auf die Seite */Konferenz/Klasse/(klasse)*



Seite /Konferenz/Klasse/(klasse)

Die Klassenansicht zeigt eine SchülerInnenliste. Die ersten 3 Spalten werden aus der *Student* Klasse befüllt (*lastname*, *firstname*, *conferenceDecision*). *conferenceDecision* kann NULL sein. Ist der Wert true, so ist bei *Beschluss?* der Wert JA anzuzeigen. Ist er false, so ist NEIN anzuzeigen. Bei einem NULL Wert ist nichts anzuzeigen.

In dieser Liste wird auf 2 Aktionen verwiesen:

Beschluss eintragen

Falls der/die SchülerIn *mehr als ein Nicht genügend* besitzt (und nur dann), soll ein Link *Beschluss eintragen* angeboten werden. Dieser Link verweist auf die Seite */Konferenz/Beschluss/(guid)*, wobei *guid* die GUID des Schülers ist.

Prüfung anmelden

Da die SchülerInnen meist minderjährig sind, gibt der Klassenvorstand aufgrund einer unterschriebenen Anmeldung die Daten selbst ein. Wenn ein/e SchülerIn Gegenstände mit Nicht genügend besitzt (und

nur dann), wird ein Link *Prüfung anmelden* angeboten. Er soll auf die Seite */Pruefung/Anmeldung/(guid)* verweisen, wobei *guid* die GUID des Schülers ist.

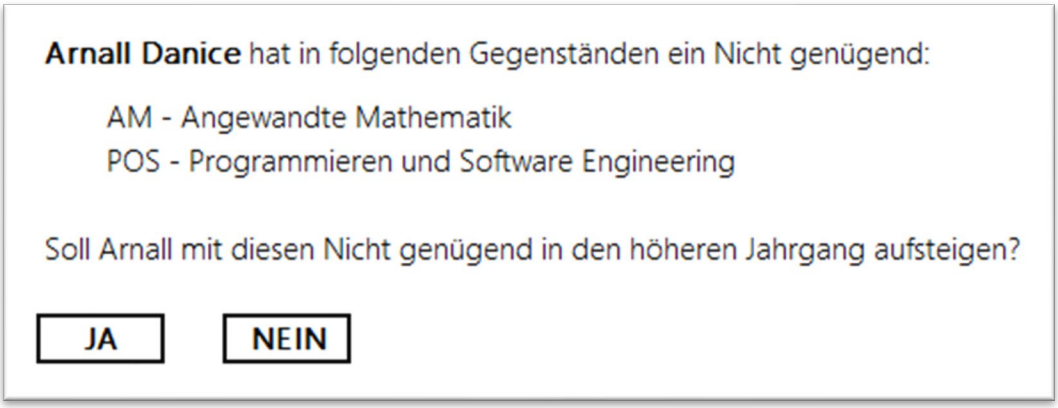
Das folgende Bild zeigt das grobe Layout der Seite. Die angezeigten Daten können von den Musterdaten in der Datenbank abweichen.

SchülerInnen der Klasse 1AHIF

Zuname	Vorname	Beschluss?	Aktionen	
Arnall	Danice	JA	Beschluss eintragen	Prüfung anmelden
Borsay	Fredek			Prüfung anmelden
Brumbie	Julita	NEIN	Beschluss eintragen	Prüfung anmelden
Flarity	Belicia			Prüfung anmelden
Glencros	Dorolice			
Goretti	Hastie			
Sarney	Gwendolyn			
Snape	Kissiah		Beschluss eintragen	Prüfung anmelden
Thring	Malachi			
Yakolev	Austin			Prüfung anmelden

Seite /Konferenz/Beschluss/(guid)

Wie bereits beschrieben kann der Klassenvorstand für SchülerInnen, die mehr als ein Nicht genügend haben, einen Aufstieg eintragen. Dies soll mit folgender Maske geschehen:



Druckt der Klassenvorstand auf *Ja*, wird Student Datensatz das Feld *conferenceDecision* auf *true* gesetzt. Danach wird auf die Seite */Konferenz/Klasse/(klasse)* zurückverwiesen.

Druckt der Klassenvorstand auf *Nein*, wird Student Datensatz das Feld *conferenceDecision* auf *false* gesetzt. Danach wird auf die Seite */Konferenz/Klasse/(klasse)* zurückverwiesen.

Eingabe einer Anmeldung

Seite /Pruefung/Anmeldung/(guid)

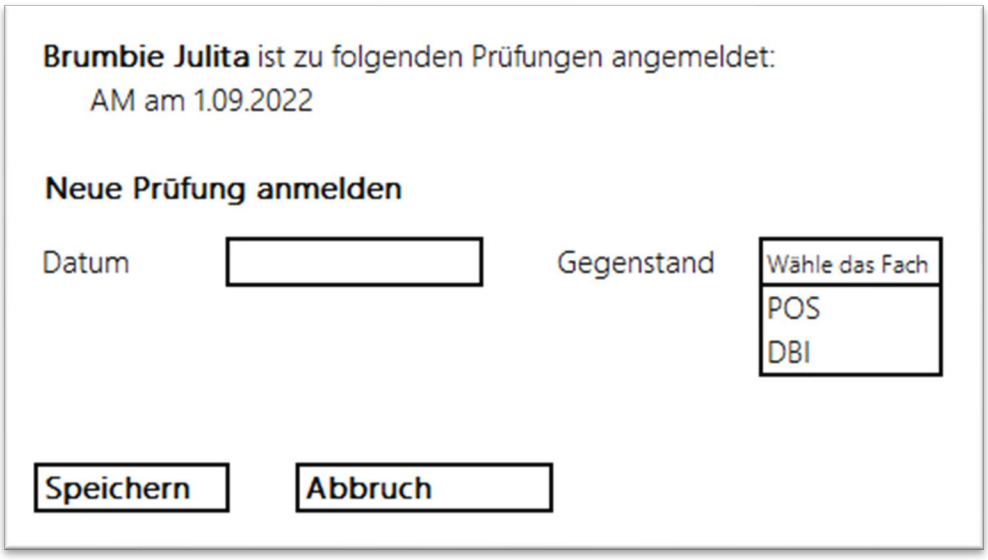
Auf der Klassenseite führt der Link *Prüfung anmelden* zur Eingabeseite der Prüfungsdaten. Sie soll folgende Features besitzen:

- Bereits angemeldete Prüfungen werden mit Gegenstandsname (Kurzname) und dem Datum angezeigt.
- Bei einer neuen Prüfung kann das Datum eingegeben werden. Es sind keine Prüfungen des Datums zu implementieren.
- Als Gegenstand soll eine Dropdownliste mit den negativen Gegenständen befüllt werden, *die noch nicht angemeldet wurden*.

Nach dem Drücken auf *Speichern* soll der neue Exam Datensatz angelegt werden. Danach wird auf die Seite */Konferenz/Klasse/(klasse)* zurückverwiesen.

Beim Drücken auf *Abbruch* wird nichts gespeichert und auf die Seite */Konferenz/Klasse/(klasse)* zurückverwiesen.

Ein mögliches Layout kann so aussehen:



Arbeitsauftrag

Implementieren Sie die oben angeführten Seiten mit den geforderten Funktionalitäten. Das Layout kann abweichen, muss aber die geforderten Funktionalitäten abbilden.

Bewertung

Für jeden korrekt gelösten Eintrag in der nachfolgenden Liste gibt es einen Punkt. In Summe sind also 19 Punkte zu erreichen.

/Konferenz/Index (2 P)

- Die Seite zeigt eine Liste aller Klassen im System an.
- Die Links verweisen auf die Detailseite der entsprechenden Klasse.

/Konferenz/Klasse/(klasse) (6 P)

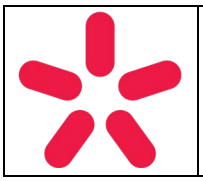
- Die Seite zeigt eine Liste aller SchülerInnen der Klasse an.
- Die Seite zeigt den Konferenzbeschluss mit JA oder NEIN an.
- Die Seite zeigt den Punkt *Prüfung anmelden* nur an, wenn negative Noten vorliegen.
- Die Seite zeigt den Punkt *Beschluss eintragen* nur an, wenn mindestens 2 negative Noten vorliegen.
- Der Link *Prüfung anmelden* verweist auf die korrekte Anmeldeseite des Schülers.
- Der Link *Beschluss eintragen* verweist auf die korrekte Eintragungsseite des Schülers.

/Konferenz/Beschluss/(guid) (4 P)

- Die Seite zeigt den SchülerInnennamen und die negativen Gegenstände an.
- Der Button JA speichert den Beschluss in der Datenbank.
- Der Button JA verweist nach dem Speichern auf die Klassenseite zurück.
- Der Button NEIN verweist nach dem Speichern auf die Klassenseite zurück.

/Pruefung/Anmeldung/(guid) (7 P)

- Die Seite zeigt den SchülerInnennamen und die negativen Gegenstände an.
- Das Dropdownfeld der Gegenstände ist mit den negativen Gegenständen gefüllt.
- Das Dropdownfeld der Gegenstände schließt angemeldete Gegenstände aus.
- Das Datumsfeld ist vom korrekten Typ (*date*).
- Der Button *Speichern* speichert die Anmeldung in der Datenbank.
- Der Button *Speichern* verweist nach dem Speichern auf die Klassenseite zurück.
- Der Button *Abbruch* verweist ohne Speichern auf die Klassenseite zurück.



Haupttermin Mai 2022

PROGRAMMIEREN UND SOFTWARE ENGINEERING

Höhere Lehranstalt für Informatik (SFKZ 8728)

Bewertungsblatt (vom Prüfer auszufüllen)

Für jede erfüllte Teilaufgabe gibt es 1 Punkt. In Summe sind also 50 Punkte zu erreichen. Für eine Berücksichtigung der Jahresnote müssen mindestens 30 % der Gesamtpunkte erreicht werden. Für eine positive Beurteilung der Klausur müssen mindestens 50 % der Gesamtpunkte erreicht werden.

Beurteilungsstufen:
50 – 44 Punkte: Sehr gut, 43 – 38 Punkte: Gut, 37 – 32 Punkte: Befriedigend, 31 – 26 Punkte: Genügend

Aufgabe 1 (jew. 1 Punkt)	Erf.	Nicht erf.
Die Klasse Department bildet alle in der Angabe beschriebenen Daten ab.		
Die Klasse Department besitzt den notwendigen Aufbau (Annotations, Konstruktoren, ...) für die Persistierung mit dem OR Mapper.		
Ein Test AddDepartmentSuccessTest() beweist, dass ein Datensatz der Klasse Department in die Datenbank geschrieben werden kann.		
Die Klasse Task bildet alle in der Angabe beschriebenen Daten ab.		
Die Klasse Task besitzt den notwendigen Aufbau (Annotations, Konstruktoren, ...) für die Persistierung mit dem OR Mapper.		
Ein Test AddTaskSuccessTest() beweist, dass ein Datensatz der Klasse Task in die Datenbank geschrieben werden kann.		
Die Klasse Applicant bildet alle in der Angabe beschriebenen Daten ab.		
Die Klasse Applicant besitzt den notwendigen Aufbau (Annotations, Konstruktoren, ...) für die Persistierung mit dem OR Mapper.		
Ein Test AddApplicantSuccessTest() beweist, dass ein Datensatz der Klasse Applicant in die Datenbank geschrieben werden kann.		
Die Klasse Upload bildet alle in der Angabe beschriebenen Daten ab.		
Die Klasse Upload besitzt den notwendigen Aufbau (Annotations, Konstruktoren, ...) für die Persistierung mit dem OR Mapper.		
Ein Test AddUploadSuccessTest() beweist, dass ein Datensatz der Klasse Upload in die Datenbank geschrieben werden kann.		
Die Klasse ApplicantStatus sowie ihre Subklassen bilden alle in der Angabe beschriebenen Daten zum entsprechenden Bewerberstatus ab.		
Die Klasse ApplicantStatus besitzt den notwendigen Aufbau (Annotations, Konstruktoren, ...) für die Persistierung mit dem OR Mapper.		
Ein Test AddApplicantWithApplicantStatusSuccessTest() beweist, dass ein Datensatz der Klasse Applicant mit dem Status Rated in die Datenbank geschrieben werden kann.		

Aufgabe 2 (jew. 1 Punkt)	Erf.	Nicht erf.
GetClassStatistics (6 P)		
Die Klasse ClassStatistics wurde gemäß der Vorgabe korrekt erstellt.		
Die Methode ermittelt korrekt die positiven SchülerInnen der übergebenen Klasse.		
Die Methode ermittelt korrekt die negativen SchülerInnen der übergebenen Klasse.		
Die Methode ermittelt korrekt die positiven SchülerInnen jedes Gegenstandes der Klasse.		
Die Methode ermittelt korrekt die negativen SchülerInnen jedes Gegenstandes der Klasse.		
Die Methode ermittelt korrekt den Mittelwert der Note jedes Gegenstandes der Klasse.		
TryAddRegistration (10 P)		
Die Methode prüft, ob der Gegenstand in der Klasse des Schülers vorhanden ist.		

Die Methode prüft, ob der Gegenstand auch negativ beurteilt wurde.		
Die Methode prüft, ob der Gegenstand nicht schon als Exam eingetragen wurde.		
Die Methode prüft, ob der Tag nicht schon vergeben wurde.		
Die Methode schreibt den Datensatz korrekt in die Datenbank.		
Der Unittest TryAddRegistrationReturnsFalseIfSubjectDoesNotExist ist richtig gestaltet (Arrange Act Assert).		
Der Unittest TryAddRegistrationReturnsFalseIfSubjectIsNotNegative ist richtig gestaltet (Arrange Act Assert).		
Der Unittest TryAddRegistrationReturnsFalseIfExamExists ist richtig gestaltet (Arrange Act Assert).		
Der Unittest TryAddRegistrationReturnsFalseOnDateConflict ist richtig gestaltet (Arrange Act Assert).		
Der Unittest TryAddRegistrationReturnsSuccessTest ist richtig gestaltet (Arrange Act Assert).		

Aufgabe 3 (jew. 1 Punkt)	Erf.	Nicht erf.
/Konferenz/Index (2 P)		
Die Seite zeigt eine Liste aller Klassen im System an.		
Die Links verweisen auf die Detailseite der entsprechenden Klasse.		
/Konferenz/Klasse/(klasse) (6 P)		
Die Seite zeigt eine Liste aller SchülerInnen der Klasse an.		
Die Seite zeigt den Konferenzbeschluss mit JA oder NEIN an.		
Die Seite zeigt den Punkt *Prüfung anmelden* nur an, wenn negative Noten vorliegen.		
Die Seite zeigt den Punkt *Beschluss eintragen* nur an, wenn mindestens 2^ negative Noten vorliegen.		
Der Link *Prüfung anmelden* verweist auf die korrekte Anmeldeseite des Schülers.		
Der Link *Beschluss eintragen* verweist auf die korrekte Eintragungsseite des Schülers.		
/Konferenz/Beschluss/(guid) (4 P)		
Die Seite zeigt den SchülerInnennamen und die negativen Gegenstände an.		
Der Button JA speichert den Beschluss in der Datenbank.		
Der Button JA verweist nach dem Speichern auf die Klassenseite zurück.		
Der Button NEIN verweist nach dem Speichern auf die Klassenseite zurück.		
/Pruefung/Anmeldung/(guid) (7 P)		
Die Seite zeigt den SchülerInnennamen und die negativen Gegenstände an.		
Das Dropdownfeld der Gegenstände ist mit den negativen Gegenständen gefüllt.		
Das Dropdownfeld der Gegenstände schließt angemeldete Gegenstände aus.		
Das Datumsfeld ist vom korrekten Typ (*date*).		
Der Button *Speichern* speichert die Anmeldung in der Datenbank.		
Der Button *Speichern* verweist nach dem Speichern auf die Klassenseite zurück.		
Der Button *Abbruch* verweist ohne Speichern auf die Klassenseite zurück.		