Ü 5a.1: Worttrainer-Model - Schritt 2

Done: View | To do: Make a submission | Done: Receive a grade

Due: Saturday, 15 October 2022, 9:00 PM

Überarbeite dein Modell und deinen Code aus Ü 5a.0 Worttrainer Model - Schritt 1 mit deinem gewonnenen Wissen aus den UML Unterlagen (insbesondere 5a.1: UML Klassendiagramm und 5a.1: UML Klassendiagramm in Java Datei)!

Funktionalität

- Der Trainer soll auch die Daten über die Statistik (Anzahl der bereits abgefragten Worte, wie viele davon sind richtig) verwalten können. Überlege dir dazu eine sinnvolle Erweiterung deines Entwurfs.

Klassendiagramm

- Sofern noch nicht geschehen: Modelliere alle Beziehungen zwischen Klassen als **Assoziationen mit Linien** (**nicht nur als Attribute**)
- Überprüfe nochmals alle Attribute und Operationen hinsichtlich Sichtbarkeiten, Datentypen, Parameter, Rückgabewert, static
- Überlege dir **sinnvolle** Multiplizitäten für alle Attribute und Assoziationen (verwende zumindest einmal [1] und einmal [1..*] oder [0..*])
- Definiere für jede Assoziation entweder einen **Assoziationsnamen** oder zumindest einen **Rollennamen**
- Wenn möglich, definiere sinnvolle **Startwerte**
- Betrachte das Verhältnis zwischen WortListe und WortEintrag. Handelt es sich hierbei um eine **spezielle Assoziation** mit besonderer Bedeutung?
  - WortListe ist für die Verwaltung der Einträge verantwortlich
  - Jeder Eintrag muss genau einer WortListe zugeordnet sein
- Überlege dir für alle Assoziationen sinnvolle **Navigierbarkeiten** (Pfeilrichtungen)
- Zeichne **Abhängigkeiten** zwischen deinen Klassen (nicht zu externen Klassen) korrekt ein
- Füge in das Diagramm auch die GUI-Komponenten ein! Achte dabei auf eine saubere Abbildung des MVC-Musters
- Erweitert: Gibt es **Zusicherungen**, die eingehalten werden müssen? Füge sie dem Klassendiagramm hinzu und argumentiere (im Abgabegespräch)!
- Erweitert: Überlege dir zumindest ein sinnvolles **abgeleitetes Attribut**!

Code

Aktualisiere deinen Code, sodass er wieder zum Model-Teil des Klassendiagramms passt! View und Controller, also die eigentliche GUI-Oberfläche, müssen noch nicht geschrieben werden.

- Überprüfe nochmals die Datentypen, Sichtbarkeiten, etc., sofern Änderungen nötig waren
- Überarbeite deinen Code, sodass alle **Multiplizitäten von Attributen und Assoziationen** eingehalten werden:
  - [*]: Stell sicher, dass mehrere Objekte / Werte gespeichert werden können (-> Array)
  - [1..*]: Stell sicher, dass immer zumindest ein Objekt / Wert vorhanden ist
  - [1]: Stell sicher, dass immer ein gültiges Objekt vorhanden ist
- Stell sicher, dass Attributnamen zum **Assoziationsnamen** bzw. **Rollennamen** passen
- Nimm alle nötigen Anpassungen vor, damit die Regeln **spezieller Assoziationen** eingehalten werden
  - Ist eine Klasse für die Verwaltung von Einträgen zuständig, sollte auch nur diese ein neues Objekt davon erstellen
- Stell sicher, dass alle **Pfeilrichtungen** auch im Code entsprechend abgebildet werden
- Erweitert: Erzwinge die Einhaltung aller **Zusicherungen**
- Erweitert: Bilde das **abgeleitete Attribut** entweder als eigenes Attribut oder berechnenden Getter ab!

Testen mit main

Teste nochmal alle Methoden der einzelnen Klassen separat mit einer main-Methode. Diese muss keine Ein- und Ausgaben enthalten, sondern nur die Methoden mit für einen Test sinnvollen Parameter-Werten aufrufen. Die Rückgabewerte müssen jedenfalls so verarbeitet werden, dass ersichtlich ist, ob der Methodenaufruf Erfolg hatte (im Zweifelsfall ausgeben).

Abgabe:

- Dokumentierte Java-Dateien der Klassen: WortEintrag, WortListe, WortTrainer und main in einem gezippten Package, dessen Name deinem Login-Namen entspricht.
- Datei des UML-Diagramms als PNG, JPG oder PDF

Submission status

Submission status	No submissions have been made yet
Grading status	Graded
Time remaining	Assignment is overdue by: 1 year 116 days
Last modified	-
Submission comments	Comments (0)

Grading criteria

Programmierstil	Programmierstil wenig berücksichtigt *0 points*	Einrückungen und Kommentare vorhanden aber verbesserungswürdig *1 points*	Einrückungen und Kommentare gut gemacht. *2 points*
Funktion	Funktion mangelhaft *0 points*	Funktion mit leichten Fehlern ok *2 points*	Funktion vollständig *4 points*
Abgabegespräch	Abgabegespräch mangelhaft *0 points*	Fragen und Änderungen im Programm konnten mit leichten Unsicherheiten beantwortet werden. *1 points*	Fragen und Änderungen im Programm konnten gut beantwortet werden. *2 points*
UML	UML nicht vorhanden *0 points*	UML nur grob umgesetzt. *3 points*	UML unter Einhaltung aller Regeln (siehe Teil2) umgesetzt *6 points*
Zeit	Nachfrist *0 points*	Knapp verspätet abgegeben. *1 points*	Im zeitlichen Rahmen abgegeben. *2 points*

Feedback

Grade	16.00 / 16.00
Graded on	Thursday, 10 November 2022, 9:35 AM
Graded by	HD Haselberger David
Feedback comments	Passt.

Grade breakdown

Programmierstil	Programmierstil wenig berücksichtigt *0 points*	Einrückungen und Kommentare vorhanden aber verbesserungswürdig *1 points*	Einrückungen und Kommentare gut gemacht. *2 points*
Funktion	Funktion mangelhaft *0 points*	Funktion mit leichten Fehlern ok *2 points*	Funktion vollständig *4 points*
Abgabegespräch	Abgabegespräch mangelhaft *0 points*	Fragen und Änderungen im Programm konnten mit leichten Unsicherheiten beantwortet werden. *1 points*	Fragen und Änderungen im Programm konnten gut beantwortet werden. *2 points*
UML	UML nicht vorhanden *0 points*	UML nur grob umgesetzt. *3 points*	UML unter Einhaltung aller Regeln (siehe Teil2) umgesetzt *6 points*
Zeit	Nachfrist *0 points*	Knapp verspätet abgegeben. *1 points*	Im zeitlichen Rahmen abgegeben. *2 points*