Aufgabe 3 - Schulverwaltung

Ziel

Sie können in einer komplexen Anwendung selbständig

- die Klassen erstellen
- die Beziehungen einpflegen (einseitig, zweiseitig, mehrfache)
- den nötigen Ablauf selbst festlegen
- die gefordeten Ausgaben erzeugen

Vorgehen

- Studieren Sie jeweils das UML-Diagramm sowie die Anweisungen im Code der zu bearbeitenden Klasse.
- Halten Sie sich an die Reihenfolge bei der Erstellung der Klassen.
- Testen Sie die jeweilige Klasse, bis alle Test erfolgreich ablaufen.

Auftrag

Es ist eine einfache Schulverwaltung gemäss folgendem Klassendiagramm zu implementieren.

Version für IMS	Version für API

Dabei nutzen Sie Ihr Wissen zu ein- und zweiseitiger Beziehung sowie den 4 gezeigten Fällen der Referenzzuweisung. Ebenso verwenden Sie Mehrfachbeziehungen.

Ablauf

1. Akzeptieren Sie das Assignment in GitHub Classroom und klonen Sie das Repository.

2. Erstellen Sie die Klasse `Grade` als `@dataclass`.
   // Hinweise:
   · Achten Sie auf die Zusicherung für den Wert von `value`. Diese nehmen Sie im Konstruktor vor, d.h. dass Sie keine setter-Methode schreiben (ist gemäss Klassendiagramm nicht vorgesehen).
   · Initialisieren Sie die Werte `value` und `date` gemäss Klassendiagramm. //

3. Testen Sie die Klasse `Grade`. (test_grade.py)

4. Implementieren Sie die Klasse `Subject`.
   Hinweise:
   · Beachten Sie, dass gemäss Klassendiagramm max. 4 Noten möglich sind. Das müssen Sie beim Zufügen von Noten (`Grade`-Objekte) umsetzen.
   · Die untere Grenze von 2 Noten müssen Sie (noch) nicht beachten.
   · Die Schreibweise `grades[] : Grade` im Klassendiagramm deutet auf die Nutzung eines

Array hin.
· *Bei den beiden get-Methoden //(IMS)* bzw. take-Methoden *(API)* muss sichergestellt sein, dass ein ungültiger Index zu keinem Laufzeitfehler führt. Für die Note (`get_value(idx)` bzw. `take_value(idx)`) soll im Fehlerfall der Wert 0 zurückgegeben werden. Beim Datum (`get_date(idx)` bzw. `take_date(idx)`) wird `None` geliefert.
· Die Methode `get_average` liefert bei fehlendem Noteneintrag (size = 0) den Wert 0 zurück. Dieser Test ist zwingend nötig, da sonst eine Division durch 0 zu einem Laufzeitfehler führt.//

5. Testen Sie die Klasse `Subject`. (test_subject.py)

6. Implementieren Sie die Klasse `ClassReport`.
   // Hinweise:
   · Lassen Sie die Methoden `set_student` und `get_student` vorerst weg. Um diese Methode zu testen, benötigen Sie zuerst ein Student-Objekt. Die entsprechende Klasse `Student` existiert aber noch nicht.
   · Beachten Sie, dass gemäss Klassendiagramm max. 3 Fächer möglich sind. Das müssen Sie beim Zufügen umsetzen
   · Stellen Sie immer sicher, dass geprüft wird, ob Subject-Objekte verfügbar sind. Ansonsten geben Sie den Wert `None` zurück.
   · Die Methode `to_string()` liefert ein Zeugnis mit allen Fächern und dem entsprechenden Notenschnitt. Eine mögliche Ausgabe kann wie folgt aussehen:
   · Die Methode `print_details()` liefert alle Fächern mit den einzelnen Noten. Eine mögliche Ausgabe kann wie folgt aussehen:
   //

7. Testen Sie die Klasse `ClassReport`. (test_classreport.py)

8. Erstellen Sie eine `main`-Methode in `classreport.py` und geben Sie die `to_string`-Methode sowie `print_details` auf dem Bildschirm aus.
   Die Ausgabe soll den oben gezeigten Screens entsprechen.
   Überlegen Sie sich, welche Objekte Sie benötigen und in welcher Reihenfolge diese zu erzeugen sind, damit die Ausagbe auf den Bildschirm erfolgreich ist.

9. Implementieren Sie die Klasse `Student`. Jetzt ist der Moment gekommen, um in der Klasse `ClassReport` die `set_student` und `get_student`-Methode zu ergänzen. (mit der Annotation `@property` bzw. `@student.setter`)
   // Hinweise:
   · Im Konstruktor wird die (eigene) Referenz dem `ClassReport`-Objekt mitgeteilt.
   · Die set-/get-Methoden implementieren Sie auch hier mit den oben erwähnten Annotationen. //

10. Testen Sie die Klasse `Student`. (test_student.py)

11. Erstellen Sie eine `main`-Methode in `student.py` und führen Sie die `to_string`-Methode des `ClassReport`-Objektes aus. Dazu müssen natürlich auch 3 Subject-Objekte und dazu ein paar Grade-Objekte erzeugt werden. Ebenso ein Student-Objekt, das den print dann auslöst.

12. Implementieren Sie die Klasse `SchoolClass`.
    // Hinweise:
    · Stellen Sie sicher, dass maximal 20 Student-Objekte zugefügt werden können. //

13. Testen Sie die Klasse `SchoolClass`. (test_school_class.py)

14. Erstellen Sie nun die `main`-Methode inder Datei `main.py`.
    // Hinweise :
    · Erzeugen Sie die Objekte in der Reihenfolge, wie sie auch für die Zuweisung in den Konstruktoren nötig sind. Wenn Sie unsicher sind, skizzieren Sie sich den Ablauf des Programms als Sequenzdiagramm auf.
    · Erzeugen Sie 3 Student-Objekte. Jedes Student-Objekt verweist auf einen Report mit den Fächern (Subject) und den Noten (Grade).//

Ausgabe

Das Programm liefert

- eine Liste der Studenten
- pro Student das Zeugnis (Report) mit dem Notenschnitt
- für einen Studenten alle Einzelnoten. Sie können hier frei wählen, für wen die Noten ausgegeben werden.

Die Ausgabe soll in etwa wie folgt aussehen:

Dauer

4 - 6 Stunden

Abgabe

Mittels Push ins GitHub Repository

© René Probst

From:
https://wiki.bzz.ch/ - **BZZ - Modulwiki**

Permanent link:
https://wiki.bzz.ch/modul/archiv/m320/learningunits/lu05/aufgaben/lu3-aufgabe_8?rev=1788091601

Last update: **2026/08/30 14:06**