LU09.A01 - Schulverwaltung

Sie können in einer komplexen Anwendung selbständig

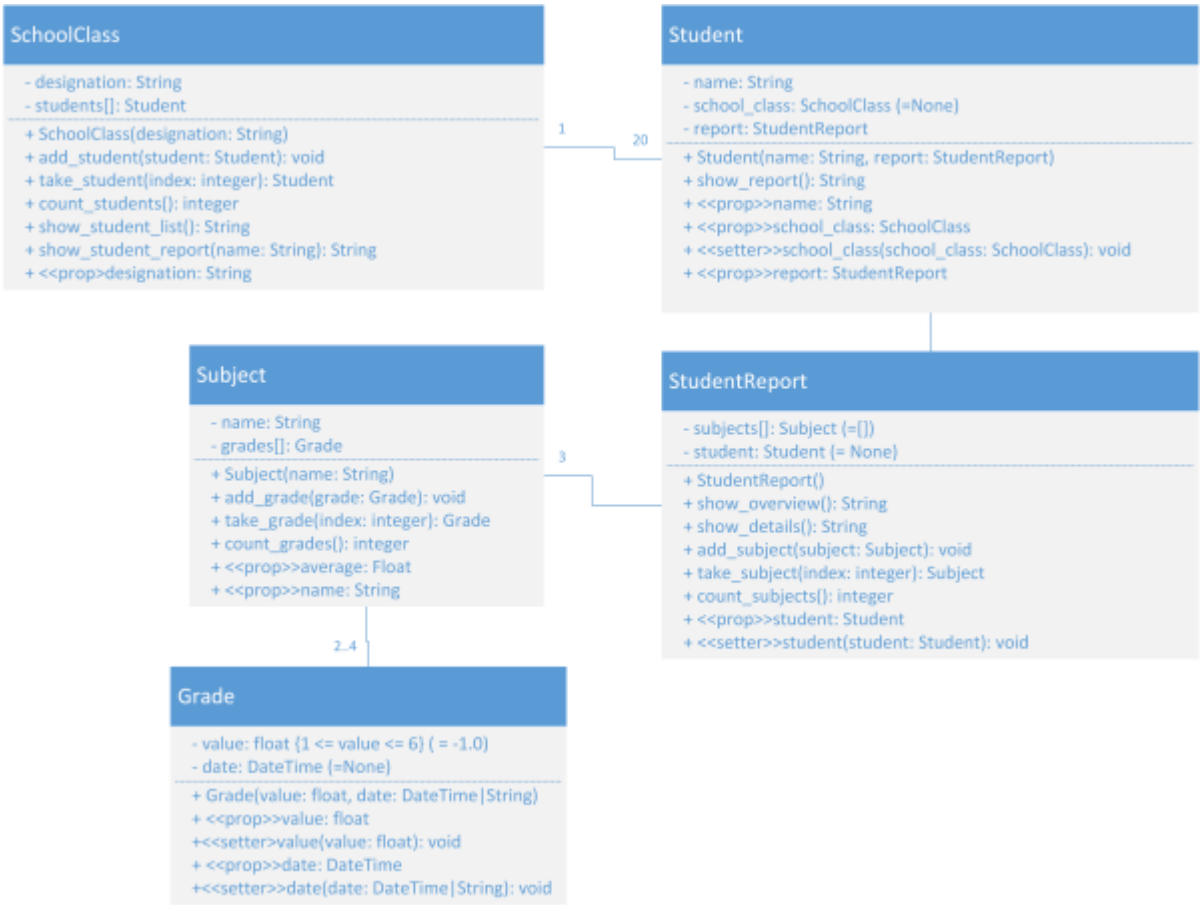
- die Klassen erstellen
- die Beziehungen einpflegen (einseitig, zweiseitig, mehrfache)
- den nötigen Ablauf selbst festlegen
- die gefordeten Ausgaben erzeugen

Vorgehen

- Studieren Sie jeweils das UML-Diagramm sowie die Erklärungen zu den einzelnen Methoden.
- 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.



Dabei nutzen Sie Ihr Wissen zu ein- und zweiseitiger Beziehung sowie den 4 gezeigten Fällen der

Referenzzuweisung. Ebenso verwenden Sie Mehrfachbeziehungen.

Vorgehen

Die Beschreibung der Klassen und Methoden erfolgt von links nach rechts und von oben nach unten. Erstellen Sie selbständig einen Plan, in welcher Reihenfolge Sie die Klassen erstellt wollen. Ein paar Tipps:

- Es lohnt sich zunächst das Gerüst einer Klasse mit allen Methoden zu bauen.
- Klassen die keine Referenzen zu anderen Klassen enthalten, sollten zuerst umgesetzt werden.
- Versuche danach Klassen umzusetzen, bei denen die referenzierte Klasse bereits erstellt ist.
- Der Konstruktor einer Klasse sollte in der Regel zuerst erstellt werden.
- Danach folgen Methoden ohne Logik, z.B. `property` und `setter`.
- Bei Methoden die eine Verarbeitungslogik enthalten, wird zunächst einfach ein hart codierter Returnwert zurück gegeben.

Hinweise

- Testen Sie jede Klasse/Methode mit den jeweiligen Unit Tests. Führen Sie die Tests einzeln aus, da vor allem zu Beginn sehr viele Tests scheitern werden.
- Die Methoden `show_…` liefern immer einen String als Returnwert. Der `print`-Befehl wird nur im `main()` genutzt.

SchoolClass

Konstruktor

- Die Schreibweise `students[] : Student` im Klassendiagramm zeigt an, dass es sich um eine Liste (Array) handelt.
  Initialisieren Sie das Attribut als leere Liste.

add_student

- Fügt einen Studenten in die Liste ein.
- Beachten Sie, dass gemäss Klassendiagramm max. 20 Studenten möglich sind. Das müssen Sie beim Zufügen von Studenten umsetzen.
- Beim Versuch mehr als 20 Studenten einzufügen, soll die Methode einen `OverflowError` werfen.

count_students

- Gibt die Anzahl Studenten zurück.

take_student(index)

- Liefert den Studenten beim angegebenen Index.
- Bei einem ungültigen Index soll ein `IndexError` ausgelöst werden.

show_student_list

- Diese Methode liefert eine Liste aller Studenten. Die Ausgabe könnte wie folgt aussehen:

```
Max
Pia
Cem
```

show_student_report(name)

- Diese Methode liefert das Zeugnis für einen Studenten mit allen Fächern und dem Notenschnitt. Die Ausgabe könnte wie folgt aussehen:

```
Zeugnis für: Cem
      Mathe     :  4.25
      Deutsch   :  5.50
      Turnen    :  5.50
```

Die Methode erzeugt selber keine Liste, sie ruft lediglich eine Methode in einem anderen Objekt auf.

Student

Konstruktor

- Beachten Sie die Parameter und Defaultwerte.

show_report

Gibt die Referenz auf das `report`-Objekt aus dem Attribut zurück.

StudentReport

Konstruktor

- Die Schreibweise `subjects[] : Subject` im Klassendiagramm zeigt an, dass es sich um eine Liste (Array) handelt.
  Initialisieren Sie das Attribut als leere Liste.

add_subject

- Beachten Sie, dass gemäss Klassendiagramm max. 3 Fächer möglich sind.
- Beim Versuch mehr als 3 Fächer (`Subject`) einzufügen, soll die Methode einen `OverflowError` werfen.

take_subject

- Liefert das Fach (`Subject`) beim angegebenen Index.
- Bei einem ungültigen Index soll ein `IndexError` ausgelöst werden.

show_overview

- Diese Methode liefert ein Zeugnis mit allen Fächern und dem entsprechenden Notenschnitt. Eine mögliche Ausgabe kann wie folgt aussehen:

```
Zeugnis für: Cem
      Mathe     :  4.25
      Deutsch   :  5.50
      Turnen    :  5.50
```

show_details

- Diese Methode liefert alle Fächern mit den einzelnen Noten. Eine mögliche Ausgabe kann wie folgt aussehen:

```
Fach: Mathe      mit 2 Noten
 - 1: 5.00 am 01.01.2011
 - 2: 3.50 am 02.02.2022
```

```
 Schnitt: 4.25
Fach: Deutsch    mit 3 Noten
 - 1: 5.50 am 03.03.2033
 - 2: 6.00 am 04.04.2044
 - 3: 5.00 am 05.05.2055
 Schnitt: 5.50
```

Subject

Konstruktor

Die Schreibweise `grades[] : Grade` im Klassendiagramm zeigt an, dass es sich um eine Liste (Array) handelt. Initialisieren Sie das Attribut als leere Liste.

add_grade

- Beachten Sie, dass gemäss Klassendiagramm max. 4 Noten möglich sind. Das müssen Sie beim Zufügen von Noten (`Grade`-Objekte) umsetzen.
- Beim Versuch mehr als 4 Noten einzufügen, soll die Methode einen `OverflowError` werfen.
- Die untere Grenze von 2 Noten müssen Sie (noch) nicht beachten.

take_grade

- Liefert die Note (`Grade`) beim angegebenen Index.
- Bei einem ungültigen Index soll ein `IndexError` ausgelöst werden.

count_grades

- Gibt die Anzahl Noten (`Grade`) zurück.

average

- Die Methode berechnet den Notendurchschnitt aus allen `Grade`-Objekten in der Liste.
- Falls keine `Grade`-Objekte in der Liste vorhanden sind, wird der Wert `0.00` zurück gegeben.

Unit tests

- Testen Sie die Klasse `Subject` mit den Testfällen in `test_subject.py`

Grade

Die Klasse `Grade` wird als `@dataclass` realisiert.

Konstruktor

- Initialisieren Sie die Werte `value` und `date` gemäss Klassendiagramm.
- Achten Sie auf die Zusicherung für den Wert von `value`. Diese nehmen Sie im `post_init` vor. Falls der Wert ungültig ist (keine Zahl, zu gross, zu klein) lösen Sie einen `ValueError` aus.

date.setter

Diese Methode ist bereits vorgegeben.

- Diese Methode schreibt das Attribut `self._date`.
- Je nach Art des Inputs wird dieser unterschiedlich verarbeitet:
  - `DateTime` ⇒ direkt speichern
  - `String` ⇒ Umwandeln in `DateTime`
  - Alles andere ⇒ Der aktuelle Zeitstempel wird gespeichert.

main

In der `main`-Methode erzeugen Sie die verschiedenen Objekte und zeigen die Zeugnisse an.

Ausgabe

```
Moritz
Pia
Cem

----
Zeugnis für: Moritz
      Mathe     :  4.25
      Deutsch   :  5.00
      Turnen    :  5.00
----
Zeugnis für: Pia
      Mathe     :  5.50
      Deutsch   :  5.33
      Turnen    :  5.25
----
Zeugnis für: Cem
      Mathe     :  4.25
      Deutsch   :  5.50
      Turnen    :  5.50
----
Student Theo nicht gefunden
```

```
Fach: Mathe      mit 2 Noten
 - 1: 5.00 am 01.01.2011
 - 2: 3.50 am 02.02.2022
 Schnitt: 4.25
Fach: Deutsch    mit 3 Noten
 - 1: 5.50 am 03.03.2033
 - 2: 6.00 am 04.04.2044
 - 3: 5.00 am 05.05.2055
 Schnitt: 5.50
Fach: Turnen     mit 4 Noten
 - 1: 4.50 am 06.06.2066
 - 2: 6.00 am 07.07.1977
 - 3: 6.00 am 08.08.1988
 - 4: 5.50 am 09.09.1999
 Schnitt: 5.50
```

Dauer

4 - 6 Stunden

Abgabe

Mittels Push ins GitHub Repository

⇒ *GitHub Repo für externe Besucher*

GitHub Repository https://github.com/templates-python/m320-lu09-a01-school

Lernende am BZZ müssen den Link zum GitHub Classroom Assignment verwenden

M320-LU09

René Probst, bearbeitet durch Marcel Suter

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

Permanent link:
https://wiki.bzz.ch/modul/m320_2024/learningunits/lu09/aufgaben/schulverwaltung

Last update: **2024/09/26 10:04**