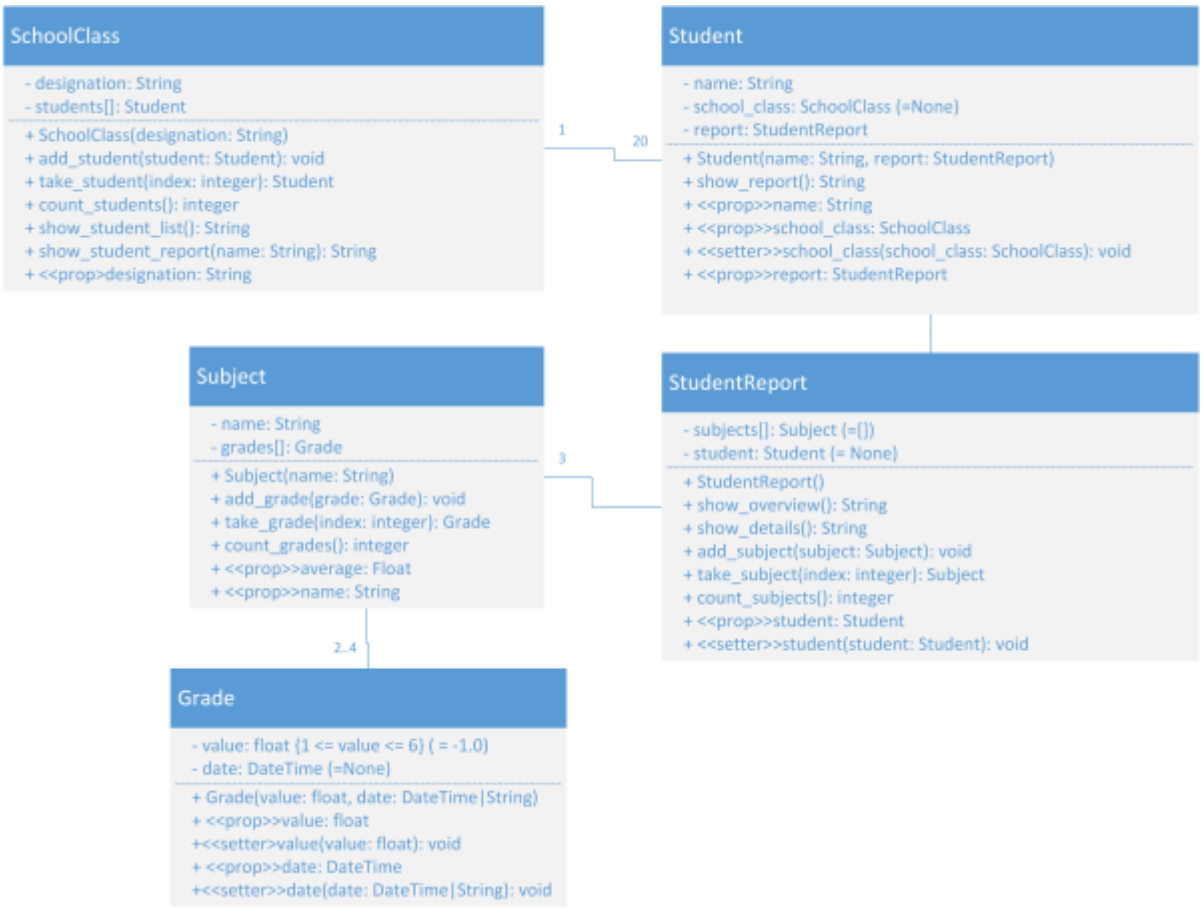
LU11.A02 - Schulverwaltung

Entwickeln Sie die Schulverwaltung als Gruppe testgetrieben. Jede Person entwickelt eine Klasse und ersetzt die Klassen der anderen in den Tests durch Mocks.
Sozialform: 4er-Gruppe · **Abgabe:** Push ins gemeinsame GitHub-Repository

Theorie: LU11a - Grundlagen, LU11b - Mock

UML Klassendiagramm



Hinweise

- Die Klasse `StudentReport` aus dem Diagramm heisst im Repository `ClassReport` (Datei `classreport.py`). Verwenden Sie den Namen aus dem Repository.
- Die Klasse `Grade` wird als `@dataclass` realisiert. Eine Note liegt zwischen 1.0 und 6.0.
- Beachten Sie die Multiplizitäten: Eine Klasse hat höchstens 20 Lernende, ein Fach höchstens 4 Noten. Wird die Grenze überschritten, löst die Methode einen `IndexError` aus.
- Die Methoden `show_…` **liefern einen String** zurück und geben nichts mit `print` aus.
- Benötigt eine Methode ein Objekt einer anderen Klasse, verwenden Sie im Test eine Mock-Klasse. Ihre Tests dürfen nicht davon abhängen, ob die Klassen der anderen schon fertig sind.

Mögliche Ausgaben von `ClassReport`:

```
show_overview()            show_details()
Mathe      4.5             Mathe     4.0 5.0
Deutsch    5.8             Deutsch   5.5 6.0
```

Vorgehen

Gruppe bilden (Classroom 50)

Die Gruppen bilden Sie selbst, wenn Sie die Aufgabe annehmen. Die Gruppe teilt sich ein Repository. Classroom 50 legt es an, sobald das erste Mitglied angenommen hat, und nummeriert es, z.B. **…**-`m450-lu11-a01-school-group-3`.

Eine Person gründet die Gruppe:

1. Öffnen Sie den Link zur Aufgabe im Moodlekurs (LU11.A02 - Schulverwaltung).
2. Klicken Sie auf **Create group**. Das Feld **Group name (optional)** können Sie leer lassen.
3. Öffnen Sie danach **Manage my group** und fügen Sie die drei anderen mit ihrem GitHub-Username hinzu (**Add**). Das ist der schnellste Weg.

Die anderen treten der Gruppe bei:

- Wurden Sie von der Gründerin bzw. dem Gründer hinzugefügt, öffnen Sie den Link zur Aufgabe erneut. Sie haben jetzt Zugriff auf das Gruppen-Repository.
- Oder Sie stellen selbst eine Anfrage: Klicken Sie unter **Join a group** bei Ihrer Gruppe auf **Ask to join**. Es öffnet sich die Gruppenseite auf GitHub. Wählen Sie dort **Request to join**. Die Gründerin bzw. der Gründer bestätigt die Anfrage über **Manage my group** → **Review join requests on GitHub**. Klicken Sie danach auf der Accept-Seite auf **Check again**.

- Eine Gruppe hat höchstens 4 Mitglieder. Bilden Sie keine zweite Gruppe, wenn Ihre Gruppe schon existiert: Treten Sie der bestehenden bei.
- Nur die Person, die die Gruppe gegründet hat, kann Mitglieder hinzufügen und entfernen. Sie kann die Gruppe nicht verlassen. Bei Problemen hilft die Lehrperson.

Vorbereitung

1. Klonen Sie das gemeinsame Repository in PyCharm.
2. Verteilen Sie die Klassen in der Gruppe:

Person	Klassen	Titel des Issues
1	`Grade` und `Student`	Grade und Student
2	`Subject`	Subject
3	`ClassReport`	ClassReport
4	`SchoolClass`	SchoolClass

1. **Jede Person** erstellt im Gruppen-Repository auf GitHub (Reiter *Issues* → *New issue*) **ein** Issue

für ihren Teil, mit dem Titel aus der Tabelle, und trägt sich selbst als *Assignee* ein. So entstehen pro Gruppe 4 Issues, und alle sehen, wer woran arbeitet.

Realisierung

Entwickeln Sie Ihre Klasse nach dem Zyklus **Red → Green → Refactor**:

1. Beginnen Sie mit dem Test für den **Konstruktor**. Danach folgen die Methoden einzeln.
2. **Red:** Schreiben Sie einen Test, führen Sie ihn aus (rot) und committen Sie mit einer Message, die mit `red:` beginnt, z.B. `red: test add_grade`.
3. **Green:** Programmieren Sie die Methode, bis alle Tests grün sind. Commit mit `green: add_grade`.
4. **Refactor:** Räumen Sie auf, bis pylint keine Meldungen mehr hat. Commit mit `refactor: …`.
5. Holen Sie vor jedem Push die Änderungen der anderen: **Pull**, dann **Push**.

Ist Ihr Teil fertig, schliessen Sie Ihr Issue mit einem Commit, dessen Message `closes #n` enthält (n ist die Nummer Ihres Issues).

Ping-Pong als Variante: Person A schreibt einen Test, Person B lässt ihn grün werden und schreibt den nächsten Test. Stellen Sie fest, dass ein Test einer anderen Person unvollständig oder falsch ist, besprechen Sie das mit ihr, bevor Sie ihn ändern.

Abschluss

Sind alle Klassen fertig, ersetzen Sie in `main.py` die Aufrufe `print_…` durch `print(…show_…())` und führen Sie das Programm aus. Jetzt arbeiten die echten Klassen zusammen. Funktioniert etwas nicht, obwohl alle Unit Tests grün sind, haben Sie einen Fehler an einer Schnittstelle gefunden. Genau dafür gibt es Integrationstests (LU15).

Abgabe

Die Abgabe erfolgt durch den Push ins gemeinsame GitHub-Repository. Bewertet werden

- die Tests (Konstruktor, Normalfälle, Grenzwerte, Fehlerfälle, Mocks statt echter Klassen),
- die Commits (Red, Green, Refactor nachvollziehbar),
- pylint `10.00/10` und alle Tests grün.

Die Bewertung gilt für alle Mitglieder der Gruppe.

M450-LU11, M450-C2I, M450-C1A

Marcel Suter

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

Permanent link:
https://wiki.bzz.ch/modul/m450/learningunits/lu11/aufgaben/school

Last update: **2026/11/23 16:24**