Datenstrukturen PVL 2- SS2023

Professur Softwaretechnik

4| 2023

Administratives

Abgabe Termin

08.05.2023 - 23:59

Abgabe

Ihre Abgabe darf sowohl in Java oder Python geschehen. Nutzen Sie für die Abgabe das Code-Template aus dem OPAL-Bereich "Prüfungsvorleistung 2". Sie dürfen weitere Dateien zur Lösung der Aufgabe erstellen. Bitte laden Sie alle Dateien, die zum Ausführen des Codes notwendig sind, im OPAL-Bereich "Prüfungsvorleistung 2" hoch.

Erlaubte Klassen

Sie dürfen Klassen und Methoden aus der Standardbibliothek von Java bzw. Python nutzen. Bitte benutzen Sie keine Bibliotheken von Dritten.

Information zu Plagiaten

Es ist **erlaubt**, Lösungen aus dem Internet in Ihre Abgabe zu integrieren. Bitte **markieren** Sie jeglichen Code, der nicht von Ihnen selbst stammt, indem Sie einen Kommentar mit der Quelle des Codes (URL ist ausreichend) an die entsprechende Stelle setzen. Dies soll verhindern, dass Ihre Lösung fälschlicherweise als Plagiat erkannt wird.
Da es sich hierbei um eine Prüfungsvorleistung handelt, ist es notwendig, dass Sie die Aufgaben selbständig bearbeiten. Daher werden alle Abgaben untereinander auf Plagiate überprüft. Sollten Plagiate erkannt werden, gilt die Abgabe von **sämtlichen** beteiligten Parteien als ungültig.

Kompilierbarkeit und Clean Code

Es werden nur Abgaben gewertet, welche sich in einem ausführbaren Zustand befinden. Achten Sie außerdem auf die Leserlichkeit des Codes. Unverständliche oder unangemessene Bezeichner von Klassen, Variablen, etc. können zu Punkteabzügen führen.

Fragen

Bitte stellen Sie ihre Fragen im Forum im Thread PVL2, damit alle Studierenden Zugang zu allen notwendigen Informationen erhalten.

Aufgabe

Schreiben Sie eine Klasse **StudentAdministration** zur Verwaltung von Studenten einer Universität.
Ein Student wird dabei durch die Matrikelnummer (1-999) eindeutig gekennzeichnet, außerdem besitzt ein Student einen Vor- und Nachnamen. Weiterhin soll ein Student eine Sammlung von Kursen besitzen, in denen bereits eine Prüfung abgelegt wurde. Die Sammlung kann eine Datenstruktur Ihrer Wahl sein. Pro Kurs sollen die Note und die Anzahl der Versuche gespeichert werden. Die möglichen Noten liegen im Intervall 1.0 bis 5.0, eine 5.0 gilt als nicht bestanden.
Implementieren Sie für die Klasse **StudentAdministration** folgende Methoden:

```
enroll_student(String first_name, String surname)
```

Die Methode fügt einen Studenten mit dem gegebenen Vor- und Nachnamen der Verwaltung hinzu und weist diesem Studenten eine eindeutige Matrikelnummer zu. Dies soll die erste freie Matrikelnummer im Intervall [1,999] sein. Geben Sie die Matrikelnummer zurück, wenn der Student erfolgreich angelegt wurde, ansonsten geben Sie -1 zurück.

```
disenroll_student(int number)
```

Exmatrikulieren Sie den Studenten mit der gegebenen Matrikelnummer, indem Sie seine Daten aus der Studierendenverwaltung entfernen. Geben Sie ”True” zurück, sollte die Operation erfolgreich gewesen sein, ansonsten ”False”.

```
take_exam(int number, String courseID, float grade)
```

Die Methode fügt beim Studierenden mit der Matrikelnummer ”number” den jeweiligen Kurs hinzu, sofern dieser noch nicht vorhanden sein sollte. Sollte der Kurs bereits vorhanden sein und wurde er bereits bestanden, so soll nichts passieren. Wenn der Kurs bereits vorhanden ist, aber noch nicht bestanden wurde, soll er anhand der Note aktualisiert werden. Eine 5.0 steht dabei für Durchgefallen. Sollte die Prüfung als nicht bestanden zählen und ein 3. Versuch sein, exmatrikulieren Sie den Studenten.

```
get_student(int number)
```

Finden Sie den Studenten mit der gegebenen Matrikelnummer und geben Sie seine Daten als Liste von Strings zurück. Sollte kein Student mit der gegebenen Matrikelnummer existieren, geben Sie eine leere Liste zurück. Die Liste $\mathcal{L}$ soll das folgende Format haben:

- $\mathcal{L}[0]$ = der Vorname des Studenten
- $\mathcal{L}[1]$ = der Nachname des Studenten
- $\mathcal{L}[2]$ = die Matrikelnummer des Studenten
- Ab dem Index 3 sollen die Kurse gespeichert sein. Die Reihenfolge sei dabei egal, ein Kurs sei wie folgt formatiert: ”Kursname Note AnzahlDerVersuche”.

Beispiel:

```
enroll_student("Sam", "Ple") % gibt 1 zurück
take_exam(1, "Math", 5.0)
take_exam(1, "Math", 4.0)
take_exam(1, "English", 2.0)
getStudent(1) % gibt zurück ["Sam", "Ple","1","Math 4.0 2","English 2.0 1"]
```