Karlsruher Institut für Technologie

Programmieren – Wintersemester 2016/2017

Architecture-driven Requirements Engineering (ARE)
`https://sdqweb.ipd.kit.edu/wiki/Programmieren`
Jun.-Prof. Dr.-Ing. Anne Koziolek

Übungsblatt 5

Ausgabe: 10.01.2017 – 17:00
Abgabe: 26.01.2017 – 07:00

Bearbeitungshinweise

- Achten Sie darauf nicht zu lange Zeilen, Methoden und Dateien zu erstellen[1]
- Programmcode muss in englischer Sprache verfasst sein
- Kommentieren Sie Ihren Code angemessen: So viel wie nötig, so wenig wie möglich
- Wählen Sie geeignete Sichtbarkeiten für Ihre Klassen, Methoden und Attribute
- Verwenden Sie keine Klassen der Java-Bibliotheken ausgenommen Klassen der Pakete `java.lang`, `java.io` und `java.util`, es sei denn die Aufgabenstellung erlaubt ausdrücklich weitere Pakete[1]
- Achten Sie auf fehlerfrei kompilierenden Programmcode[1]
- Halten Sie alle Whitespace-Regeln ein[1]
- Halten Sie die Regeln zu Variablen-, Methoden und Paketbenennung ein und wählen Sie aussagekräftige Namen[1]
- Halten Sie die Regeln zu Javadoc-Dokumentation ein[1]
- Nutzen Sie nicht das default-Package[1]
- Halten Sie auch alle anderen Checkstyle-Regeln ein
- System.exit und Runtime.exit dürfen nicht verwendet werden[1]

Abgabemodalitäten

Die Praktomat-Abgabe wird am **Mittwoch, den 18. Januar 2017, um 13:00 Uhr**, freigeschaltet.

- Geben Sie die Java-Klassen als `.java`-Dateien ab. Laden Sie die Terminal-Klasse nicht mit hoch.

Bitte beachten Sie, dass das erfolgreiche Bestehen der öffentlichen Tests für eine erfolgreiche Abgabe nötig ist. Planen Sie entsprechend Zeit für Ihren ersten Abgabeversuch ein.

Präsenzübung

Bitte beachten Sie die weiteren Beispielaufgaben und Hinweise zum Aufbau der Präsenzübung[2], die auf der Vorlesungshomepage verfügbar sind.

[1] Der Praktomat wird die Abgabe zurückweisen, falls diese Regel verletzt ist.
[2] `https://sdqweb.ipd.kit.edu/lehre/WS1617-Programmieren/writtenTestExampleB.pdf`

Bewertungshinweise

Bei der auf diesem Übungsblatt vorgestellten Aufgabe handelt es sich um eine reduzierte und angepasste Version einer Abschlussaufgabe aus dem Vorjahr. Für die Bewertung Ihrer Lösung dieses Blattes fließt genau wie bei den Abschlussaufgaben neben der implementierten Funktionalität auch Ihre Methodik ein. **Für dieses Übungsblatt macht die Methodik 5 von 20 Punkten aus.** Diese wird **ähnlich** wie bei der Abschlussaufgabe in unterschiedliche Kategorien aufgeschlüsselt, die im Folgenden beschrieben werden. Die folgenden Kategorien und Kriterien orientieren sich an den Kategorien und Kriterien der Abschlussaufgaben, vereinfachen diese aber.

Bitte beachten Sie, dass die angegebenen Beispiele für Abzugsgründe ausdrücklich *Beispiele* und keine abschließende Auflistung darstellen. Ein Beispiel bedeutet weiterhin *nicht*, dass ein Vorkommen direkt zum Abzug aller oder eines Teils der Punkte führt, sondern es wird die Gesamtqualität Ihrer Lösung in den Kategorien bewertet.

Kommentarpraxis und JavaDoc

Abzüge bspw. für:

- fehlende Kommentierung von komplexen Codestellen,
- JavaDoc das ausschließlich die Methodensignatur enthält („Alibi“-JavaDoc).

Programmierstil, Verständlichkeit und Komplexität

Abzüge bspw. für:

- kryptische Wahl von Variablen-, Methoden- und Klassenbezeichnern,
- deutlich zu tiefe Verschachtelung von if-Anweisungen statt Aufteilung in sinnvolle Methoden,
- Implementierung von Programmlogik durch Exception-Handling,
- Pokémon-Exception-Handling[3] (à la `try { /* ... */ } catch (Throwable t) { /* ... */ }`).

Objektorientierte Modellierung und Kapselung

Abzüge bspw. für:

- sogenannte Gott-Klassen (d.h. eine Klasse die das komplette Programm enthält),
- unreflektierte Aufteilung von Logik in Klassen (beispielsweise weil Maximallänge der Klasse überschritten), statt nach Funktionalität oder sinnvollen Konzepten,
- nicht-sinnvoller oder falscher Einsatz von Vererbung,
- starke Verletzung von Kapselungsprinzipien (bspw. durch das unreflektierte Generieren aller Getter- und Setter-Methoden).

Funktionalität

Die Funktionalität Ihrer Lösung wird mit 15 von 20 Punkten bewertet.

[3] `http://stackoverflow.com/questions/2308979/exception-handling-question/2308988#2308988`

Fehlerbehandlung

Ihre Programme sollen auf ungültige Benutzereingaben mit einer aussagekräftigen Fehlermeldung reagieren. Aus technischen Gründen muss eine Fehlermeldung unbedingt mit `Error,` beginnen. Eine Fehlermeldung führt nicht dazu, dass das Programm beendet wird; es sei denn, die nachfolgende Aufgabenstellung verlangt dies ausdrücklich. Achten Sie insbesondere auch darauf, dass unbehandelte `RuntimeExceptions`, bzw. Subklassen davon—sogenannte *Unchecked Exceptions*—nicht zum Abbruch des Programms führen.

Terminal-Klasse

Laden Sie für **diese Aufgabe** die `Terminal`-Klasse[a] von unserer Homepage herunter und platzieren Sie diese unbedingt im Paket `edu.kit.informatik`. Die Methode `Terminal.readLine()` liest eine Benutzereingabe von der Konsole und ersetzt `System.in`. Die Methode `Terminal.printLine()` schreibt eine Ausgabe auf die Konsole und ersetzt `System.out`. Verwenden Sie für jegliche Konsoleneingabe oder Konsolenausgabe die `Terminal`-Klasse. Verwenden Sie in keinem Fall `System.in` oder `System.out`.
Fehlermeldungen werden ausschließlich über die Terminal-Klasse ausgegeben und müssen aus technischen Gründen unbedingt mit `Error,` beginnen.
Laden Sie die Terminal-Klasse niemals zusammen mit Ihrer Abgabe hoch.

[a] https://sdqweb.ipd.kit.edu/lehre/WS1617-Programmieren/Terminal.java

Studierendenportal (20 Punkte)

Die Grundfunktionalität eines Systems (ähnlich des Campus Management Portals) zur Verwaltung von Modulen mit zugehörigen Vorlesungen und Prüfungsleistungen von Studierenden soll implementiert werden.

Es gibt verschiedene Benutzerrollen: Professoren und Studierende. Professoren halten Vorlesungen und sind auch verantwortlich für die Klausurnote einer Vorlesung. Für Studierende können Prüfungsleistungen wie Noten für Klausuren eingetragen werden.

Jeder Benutzer hat einen festen Vor- und Nachnamen, welche sich der Einfachheit halber nur aus Kleinbuchstaben zusammensetzen. Es kann mehrere Benutzer geben, welche dieselbe Kombination aus Vor- und Nachnamen haben. Studierende haben zusätzlich noch eine feste, eindeutige Matrikelnummer zur Identifizierung. Eine Matrikelnummer ist immer eine positive, sechsstellige, ganze Zahl. Eine Matrikelnummer muss sich immer eindeutig einem Studierenden zuordnen lassen, jedoch muss sie nicht fortlaufend durchnummeriert werden. Professoren sind zusätzlich noch eindeutig einem Lehrstuhl zugeordnet, über welchen sie sich mit ihrem Vornamen und Nachnamen eindeutig identifizieren lassen. Demzufolge gibt es innerhalb eines Lehrstuhls nie zwei oder mehrere Mitarbeiter mit identischen Vor- und Nachnamen. Allerdings können zwei Mitarbeiter an einem Lehrstuhl identische Vornamen mit unterschiedlichen Nachnamen oder identischen Nachnamen mit unterschiedlichen Vornamen haben. Jeder Lehrstuhl hat einen eindeutigen Namen. Der Name für einen Lehrstuhl setzt sich erneut nur aus Kleinbuchstaben zusammen. Jeder Mitarbeiter hat genau einen Vornamen und genau einen Nachnamen. Jeder Lehrstuhl hat einen oder mehrere Mitarbeiter. Eine Person kann jeweils nur Professor oder Student sein (aber keine Kombination daraus).

Es gibt in einem solchen System verschiedene Arten von Lehrveranstaltungen: Module und Vorlesungen. Jede Vorlesung ist immer eindeutig genau einem Modul zugeordnet und besitzt einen eindeutigen Professor. Ein Professor kann mehrere Vorlesungen halten. Zu einem Modul kann es keine, eine oder mehrere Vorlesungen geben. Die Vorlesungen in einem Modul können jeweils von unterschiedlichen Professoren an unterschiedlichen Lehrstühlen gehalten werden.

Module und Vorlesungen haben unter anderem einen festen Namen, Leistungspunkte und eine eindeutig durchnummerierte Nummer als Identifikator. Der Name setzt sich hierbei wieder rein aus Kleinbuchstaben zusammen. Beim Anlegen einer neuen Lehrveranstaltung (Modul oder Vorlesung) im Studierendenportal, wird dieser automatisch ein eindeutiger Identifikator zugewiesen. Dieser Identifikator ist eine positive natürliche Zahl

größer Null, beginnend bei Eins und wird je angelegter Veranstaltung (Modul oder Vorlesung) um eine Stelle erhöht. Die erste Veranstaltung bekommt somit die Nummer 1.

Leistungspunkte werden durch natürliche Zahlen größer Null repräsentiert und werden im Rahmen der Berechnung der Durchschnittsnote verwendet. So wird die erreichte Note mit den Leistungspunkten der jeweiligen Prüfungsleitung multipliziert. Diese wird anschließend über alle Leistungen für einen Studierenden kumuliert und durch die Gesamtanzahl der Leistungspunkte dividiert. Hierbei haben Vorlesungen höchstens neun Leistungspunkte. Die Anzahl der Leistungspunkte eine Moduls setzt sich aus der Summe der Leistungspunkte seiner Vorlesungen zusammen, jedoch nie höher als 45 Leistungspunkte. Das heißt ein Modul darf in der Summe nie mehr als 45 Leistungspunkte beinhalten.

Studierende müssen sich für Prüfungsleistungen nicht anmelden, sondern bekommen ihre Noten direkt eingetragen. Hierbei ist eine Note eine ~~natürliche~~ reelle Zahl aus dem abgeschlossenen Intervall von Eins bis Fünf. Vorlesungen werden benotet. Wenn einmal für einen Studierenden in einer Vorlesung eine Note eingetragen wurde, kann diese nicht mehr geändert werden. Ein Student kann beliebig viele Prüfungsleistungen ablegen. Wurden eine oder mehrere Prüfungen nicht bestanden (Note 5) führt dies nicht zur Beendigung des Studiums.

> Eine saubere objektorientierte Modellierung zu planen und umzusetzen ist ein wichtiges Ziel dieser Aufgabe. Überlegen Sie sich, welche Klassen hierzu nötig sind und welche Referenzen zwischen ihnen bestehen müssen. Achten Sie dabei auf die Trennung von Ein-/Ausgabe und Funktionalität, sowie die anderen aus der Vorlesung bekannten Kriterien für gutes Design. Beachten Sie, dass Ihre Abgabe sowohl in Bezug auf Funktionalität, als auch im Hinblick auf objektorientierte Modellierung bewertet wird.

Student

Die Durchschnittsnote für einen Studierenden setzt sich aus den durch die Leistungspunkte gewichteten Durchschnittsnoten der Module zusammen, in welchen für den Studierenden bereits mindestens eine Vorlesung mit einer Note eingetragen wurde.

Die Durchschnittsnote eines Moduls für einen Studierenden setzt sich aus den durch die Leistungspunkte gewichteten Noten der zugehörigen Vorlesungen zusammen. Hierbei muss für einen Studierenden nicht für alle Vorlesungen eines Moduls eine Note eingetragen sein. Ist für eine Vorlesung noch keine Note eingetragen, so wird diese Vorlesung nicht bei der Berechnung der Durchschnittsnote eines Moduls beachtet.

Vorlesung

Die Durchschnittsnote für eine einzelne Vorlesung ist der Durchschnitt aller Noten von Studierenden, für die eine Note für diese Vorlesung im Studierendenportal eingetragen wurde.

Modul

Die Durchschnittsnote für ein einzelnes Modul beträgt der gewichtete Durchschnitt aller Noten der Vorlesungen, die dem Modul zugeordnet sind und für die bereits eine Note eingetragen wurde. Wurde für eine Vorlesung eines Moduls noch keine Note eingetragenen, so wird diese Vorlesung nicht bei der Berechnung der Durchschnittsnote eines Moduls beachtet.

Professor

Die Durchschnittsnote für einen Professor ist der Durchschnitt der gewichteten Durchschnittsnoten für alle Vorlesungen, die von diesem Professor gehalten werden. Hierbei wird die Durchschnittsnote für eine einzelne Vorlesung mit den Leistungspunkten der jeweiligen Vorlesung multipliziert und dann durch die Gesamtanzahl der Leistungspunkte dividiert.

Sortierung

Im Folge der Aufgabe müssen Sie mehrmals die Ausgabe für mehrere verschiedenen Benutzern oder Lehrveranstaltungen implementieren. Hierzu müssen diese Zeilen der Ausgaben jeweils nach den folgenden Vorgaben sortiert werden.

Professoren

Listen von Professoren werden lexikographisch, basierend auf `String#compareTo(String)`, aufsteigend nach dem Vornamen des Professors sortiert. Bei gleichen Vornamen sortieren Sie Ihre Ausgabe lexikographisch aufsteigend nach dem Nachnamen des Professors. Falls diese identisch sein sollten, werden diese Professoren lexikographisch aufsteigend nach dem Namen ihres zugehörigen Lehrstuhls sortiert.

Studierende

Listen von Studierenden werden numerisch aufsteigend nach ihrer eindeutigen Matrikelnummer sortiert.

Module und Vorlesungen

Listen von Lehrveranstaltungen werden numerisch aufsteigend nach ihrem automatisch zugewiesenen eindeutigen Identifikator sortiert.

Interaktive Benutzerschnittstelle

Ihr Programm arbeitet ausschließlich mit Befehlen, die nach dem Programmstart über die Konsole mittels `Terminal.readLine()` eingelesen werden. Nach Abarbeitung eines Befehls wartet das Programm auf weitere Befehle, bis das Programm durch die Eingabe des `quit`-Befehls beendet wird. Bei Programmstart enthält Ihr Studierendenportal weder Module noch Benutzer.

Falls eine Gleitkommazahl auf die Konsole ausgegeben werden soll, muss zuvor eine auf zwei Nachkommastellen formatierte textuelle Repräsentation erzeugt werden. Alle Ausgaben erfolgen mit 2 Nachkommastellen. Ausgaben von nicht ganzzahligen Werten (Gleitkommazahlen) müssen zuvor immer auf zwei Nachkommastellen aufgerundet werden.

Achten Sie darauf, dass durch Ausführung der folgenden Befehle die Spezifikationen nicht verletzt werden dürfen. Geben Sie im Fall der Eingabe eines Befehls, der direkt zu einer Verletzung der Spezifikation führt, eine aussagekräftige Fehlermeldung aus. Geben sie auch eine Fehlermeldung aus, falls kein, ein oder mehrere falsche oder nicht korrekt zu interpretierende Parameter eingegeben wurden. Beispielweise wäre ein solcher nicht korrekt zu interpretierende Parameter, eine Matrikelnummer für die kein Student im Studierendenportal hinterlegt ist oder wenn eine Vorlesung mit mehr als 45 Leistungspunkten hinzugefügt werden soll. Das heißt, wenn eine Eingabe nicht der hier vorgegebenen Spezifikation entspricht, wird immer nur eine Fehlermeldung auszugeben und führt zu keiner Änderung des Datensatzes. Nach der Ausgabe einer Fehlermeldung soll das Programm regulär auf die nächste Eingabe warten.

Befehle

Ihre interaktive Benutzerschnittstelle muss die folgenden 15 Befehle gemäß der Spezifikationen umsetzen. Beachten Sie, dass im Fehlerfall eine Fehlermeldung ausgegeben wird, ansonsten erfolgt die Ausgabe gemäß der Spezifikation des Ausgabeformats.

Im Folgenden wird die Ausgabe für einen Erfolgsfall spezifiziert. Im Fehlerfall (z.B. bei falsch spezifizierten Eingaben) müssen Sie eine aussagekräftige Fehlermeldung beginnend mit `Error,` ausgegeben.

Beispiele

Beachten Sie, dass bei den folgenden Beispielen die Eingabezeilen mit dem `>`-Zeichen eingeleitet werden, gefolgt von einem Leerzeichen. Diese beiden Zeichen sind ausdrücklich kein Bestandteil des eingegebenen Befehls, sondern dienen nur der Unterscheidung zwischen Ein- und Ausgabe. Auch sind die einzelnen Beispiele für die Befehle in der Regel unabhängig voneinander.

add-professor

Fügt einen neuen Professor hinzu.

Eingabeformat

`add-professor <firstname>;<lastname>;<chair>`

`<firstname>`
Der Vorname des Professors.

`<lastname>`
Der Nachname des Professors.

`<chair>`
Der Lehrstuhl an dem der Professor beschäftigt ist.

Ausgabeformat

`Ok`

Beispiel

```
> add-professor roman;forst;ddf
Ok
> add-professor dominik;saller;xxc
Ok
```

list-professor

Listet alle Professoren auf.

Eingabeformat

`list-professor`

Ausgabeformat

`<chair> <firstname> <lastname> <average>`

`<firstname>`
Der Vorname des Professors.

`<lastname>`
Der Nachname des Professors.

`<chair>`
Der Lehrstuhl an dem der zugehörige Professor beschäftigt ist.

`<average>`
Die Durchschnittsnote für einen einzelnen Professor.

Beispiel

```
> list-professor
sww andreas kappel 2.00
itc angelika huffman 3.00
xxc dominik saller 2.00
hgs maximiliane frei 1.00
ifd robert amsel 1.00
ddf roman forst 1.00
```

(Hinweis: Sortierung korrigiert.)

summary-professor

Gibt eine Liste der Vorlesungen eines Professors aus.

Eingabeformat

`summary-professor <firstname>;<lastname>;<chair>`

<firstname>
Der Vorname des Professors.

<lastname>
Der Nachname des Professors.

<chair>
Der Lehrstuhl an dem der Professor beschäftigt ist.

Ausgabeformat

`<id> <name> <average>`

<id>
Der automatisch zugewiesene eindeutige Identifikator einer Vorlesung des zugehörigen Professors.

<name>
Der Name dieser Vorlesung.

<average>
Die Durchschnittsnote für diese Vorlesung.

Beispiel

```
> summary-professor roman;forst;ddf
1 programmieren 1.50
7 robotik 2.05
33 theoretischeinformatik 2.4 2.40
```

add-student

Fügt einen neuen Studierenden hinzu.

Eingabeformat

`add-student <firstname>;<lastname>;<matrnr>`

<firstname>
Der Vorname des Studierenden.

<lastname>
Der Nachname des Studierenden.

<matrnr>
Die zugehörige Matrikelnummer des Studierenden.

Ausgabeformat
`Ok`

Beispiel

```
> add-student ada;smith;101215
Ok
> add-student grace;smith;912906
Ok
> add-student jean;doe;230328
Ok
> add-student frances;nguyen;543210
Ok
```

list-student

Listet alle Studierende auf.

Eingabeformat
`list-student`

Ausgabeformat
`<matrnr> <firstname> <lastname> <average>`

<matrnr>
Die zugehörige Matrikelnummer des Studierenden.

<firstname>
Der Vorname des Studierenden.

<lastname>
Der Nachname des Studierenden.

<average>
Die Durchschnittsnote dieses einzelnen Studierenden.

Beispiel

```
> list-student
101215 ada smith 1.23
230328 jean doe 4.37
481932 frances nguyen 3.29
912906 grace smith 2.31
```

summary-student

Gibt eine Liste der Noten samt zugehöriger Vorlesungen eines Studierenden aus.

Eingabeformat
`summary-student <firstname>;<lastname>;<matrnr>`

`<firstname>`
: Der Vorname des Studierenden.

`<lastname>`
: Der Nachname des Studierenden.

`<matrnr>`
: Die zugehörige Matrikelnummer des Studierenden.

Ausgabeformat

`<id> <lecture> <mark>`

`<id>`
: Der automatisch zugewiesene eindeutige Identifikator einer Vorlesung.

`<name>`
: Der Name dieser Vorlesung.

`<mark>`
: Die Note des Studierenden für diese Vorlesung.

Beispiel

```
> summary-student ada;smith;101215
1 programmieren 1.34
7 robotik 1.34
33 theoretischeinformatik 2.17
42 requirementsengineering 3.52
```

add-module

Fügt ein neues Modul hinzu.

Eingabeformat

`add-module <name>`

`<name>`
: Der Name des Moduls.

Ausgabeformat

`Ok`

Beispiel

```
> add-module algorithmentechnik
Ok
> add-module parallelverarbeitung
Ok
> add-module betriebssysteme
Ok
```

list-module

Listet alle Module auf.

Eingabeformat

`list-module`

Ausgabeformat

`<id> <name> <credits> <average>`

<id>
Der automatisch zugewiesene eindeutige Identifikator des zugehörigen Moduls.

<name>
Der Name dieses Moduls.

<credits>
Die Summe der Leistungspunkte aller Vorlesungen dieses Moduls.

<average>
Die Durchschnittsnote für dieses Moduls.

Beispiel

```
> list-module
1 algorithmentechnik 12 2.21
2 parallelverarbeitung 6 1.13
3 betriebssysteme 21 2.28
```

summary-module

Gibt für ein Modul die Liste mit allen zugehörigen Vorlesungen aus.

Eingabeformat

`summary-module <id>`

<id>
Der automatisch zugewiesene eindeutige Identifikator eines Moduls.

Ausgabeformat

`<id> <name> <credits> <average>`

<id>
Der automatisch zugewiesene eindeutige Identifikator einer Vorlesung des zugehörigen Moduls.

<name>
Der Name dieser Vorlesung.

<credits>
Die Leistungspunkte dieser Vorlesungen.

<id>
Die Durchschnittsnote für diese Vorlesung.

Beispiel

```
> summary-module 3
79 betriebssysteme 3 1.00
83 betriebssysteme 6 2.80
89 systementwurf 9 2.20
97 lowpower 6 1.42
```

add-lecture

Fügt eine neue Vorlesung hinzu.

Eingabeformat

`add-lecture <name_lecture>;<id_module>;<firstname_professor>;<lastname_professor>;<chair_professor>;<credits>`

<name_lecture>
Der Name des Vorlesung.

<id_module>
Der zugewiesene Identifikator des Moduls, welchem die Vorlesung hinzugefügt werden soll.

<firstname_professor>
Der Vorname des Professors, welcher für die Vorlesung verantwortlich ist.

<laststname_professor>
Der Nachname des Professors, welcher für die Vorlesung verantwortlich ist.

<chair_professor>
Der Lehrstuhl, an welchem der Professor beschäftigt ist.

<credits>
Die Leistungspunkte der Vorlesung.

Ausgabeformat

`Ok`

Beispiel

```
> add-lecture betriebssysteme;3;felix;martell;ittg;3
Ok
> add-lecture betriebssysteme;3;francis;bruhn;igf;5
Ok
> add-lecture graphenalgorithmen;55;joerg;hettel;idi;3
Ok
```

list-lecture

Listet alle Vorlesungen auf.

Eingabeformat

`list-lecture`

Ausgabeformat

`<id> <name> <exercise> <credits> <average>`

korrigiert: `<id> <name> <credits> <average>`

<id>
Der automatisch zugewiesene eindeutige Identifikator der zugehörigen Vorlesung.

<name>
Der Name dieser Vorlesung.

<credits>
Die Leistungspunkte dieser Vorlesung.

<average>
Die Durchschnittsnote für diese Vorlesung.

Beispiel

```
> list-lecture
7 robotik 8 1.34
33 theoretischeinformatik 7 2.17
42 requirementsengineering 5 3.52
79 betriebssysteme 3 1.00
83 betriebssysteme 6 2.80
89 systementwurf 9 4.20
97 lowpower 6 1.42
```

summary-lecture

Gibt für eine Vorlesung die Liste aller zugehörigen Noten der Studierenden aus.

Eingabeformat

`summary-lecture <id>`

<id>
Der automatisch zugewiesene eindeutige Identifikator der zugehörigen Vorlesung.

Ausgabeformat

`<matrnr> <firstname> <lastname> <mark>`

<matrnr>
Die zugehörige Matrikelnummer des Studierenden.

<firstname>
Der Vorname des Studierenden.

<lastname>
Der Nachname des Studierenden.

<mark>
Die Note des Studierenden für diese Vorlesung.

Beispiel

```
> summary-lecture 83
101215 ada smith 1.23
230328 jean doe 4.37
481932 frances nguyen 3.29
912906 grace smith 2.31
```

examination-marking

Trägt eine Note für einen Studierenden ein.

Eingabeformat

`examination-marking <id>;<matrnr>;<mark>`

<id>
Der automatisch zugewiesene eindeutige Identifikator einer Vorlesung.

<matrnr>
Die zugehörige Matrikelnummer des Studierenden.

`<mark>`
: Die entsprechende Note des Studierenden für diese Vorlesung.

Ausgabeformat
: `Ok`

Beispiel

```
> examination-marking 79;230328;5.00
Ok
> examination-marking 97;101215;4.50
Ok
> examination-marking 83;230328;4.05
Ok
```

reset

Initialisiert das Studierendenportal neu. Der Zustand des Studierendenportal entspricht nach dieser Operation dem Zustand bei einem frischen Start des Programms.

Eingabeformat
: `reset`

Ausgabeformat
: `Ok`

Beispiel

```
> reset
Ok
```

quit

Beendet das Programm.

Eingabeformat
: `quit`

Ausgabeformat
: Im Fehlerfall wird eine Fehlermeldung ausgegeben, ansonsten wird nichts ausgegeben.

Beispiel

```
> quit
```