Portfolioprüfung
Teilaufgabe 2

Zimmermann, Hufenbach, Himmelspach, Leneweit

Nordakademie

15. August 2018




Prüfungen I

Erstellen Sie

- ein neues Projekt (exam)
- in diesem eine *Klasse* mit dem Namen `Exam`
- einen *Konstruktor* in der Klasse `Exam`, der die Nummer der Teilprüfung und die Anzahl der Prüflinge als Parameter übergeben bekommt und ersteren in einer Exemplarvariablen (`examNumber`) abspeichert.
- eine *verändernde Methode* und eine *sondierende Methode* für die *Exemplarvariable* `examNumber`
- Erweitern Sie den *Zustand* der *Klasse* `Exam` durch das Einfügen einer weiteren *Exemplarvariable*, die die Namen von teilnehmenden Studierenden enthalten soll (`students`)

Prüfungen II

- fügen Sie eine *Methode* ein (`addStudent`), die es erlaubt einen Studenten „hinzuzufügen“ (mit einem *Parameter* für den Namen (*Typ*: String)). Sollten keine weiteren Studenten hinzugefügt werden können geben Sie die Meldung „Keine weiteren Prüfungsplätze vorhanden“ aus. (*bedingte Anweisung, boolescher Ausdruck*)
- Fügen Sie eine *Methode* ein, die die Namen aller Studierenden ausgibt (`printStudents`). Integrieren Sie eine *bedingte Anweisung*, so dass `null`-Werte nicht ausgegeben werden (formulieren Sie dazu den entsprechenden *booleschen Ausdruck*).
- fügen Sie eine weitere *Exemplarvariable* ein. Diese verwaltet die Note der Prüfungsleistung pro Student (`grades`). Noten sollen im Gleitpunktformat gespeichert werden (siehe *primitive Datentypen*).

Prüfungen III

- fügen Sie eine *Methode* ein (`setStudentGrade`), die es erlaubt eine Note für einen Studenten festzulegen (mit einem *Parameter* für den Namen und einen für die Note). Noten dürfen nur für vorhandene Studenten gesetzt werden. Andernfalls geben Sie die Meldung aus „Kein Student mit diesem Namen gefunden“ (*bedingte Anweisung, boolescher Ausdruck*)
- fügen Sie eine (Geschäfts-)*Methode* hinzu (`meanGrade`), die den Durchschnitt aller Noten ausrechnet und diesen zurückgibt (*Ergebnis*). Verwenden Sie dazu eine for-*Schleife*. Gehen Sie davon aus, dass für jeden Studenten vorher eine Note erfasst wurde. Gehen Sie ebenfalls davon aus, dass mindestens ein Student existiert.
- fügen Sie eine (Geschäfts-)*Methode* hinzu, die die beste aller Noten findet (`bestGrade`) und diese zurückgibt (*Ergebnis, lokale Variable*). Verwenden Sie dazu eine for-each-*Schleife*. Gehen Sie davon aus, dass mindestens ein Student mit einer Note existiert.

Prüfungen IV

- fügen Sie eine (Geschäfts-)*Methode* hinzu `studentsWithGrade`, die die Namen der Studenten zurückgibt, die eine übergebene Note haben (*Ergebnis*). Verwenden Sie dazu while-*Schleifen*.
- fügen Sie eine (Geschäfts-)*Methode* hinzu, die Namen der Studenten mit der besten Note in die Konsole ausgibt (`bestStudents`) - denken Sie dabei an das Konzept: *interne Methodenaufrufe*. Gehen Sie davon aus, dass mindestens ein Student mit einer Note bereits erfasst wurde.
- fügen Sie eine (Geschäfts-)*Methode* hinzu (`printStudentsWithGrades`), die die Namen der Studierenden zusammen mit der Notenstufe (sehr gut, gut, befriedigend, ausreichend, mangelhaft) auf die Konsole ausgibt.
- Fügen Sie eine *Klasse* `Exams` in ihr Projekt ein.
- Fügen Sie der *Klasse* `Exams` einen *Zustand* hinzu, der es erlaubt bis zu 10 Instanzen der *Klasse* Exam zu verwalten (in Form einer *Exemplarvariable*).

Prüfungen V

- Fügen Sie im Konstruktor der *Klasse* `Exams` 5 Prüfungen hinzu (*Klassen definieren Typen*; *Eine Klasse, viele Instanzen/Exemplare*
- Fügen Sie eine *Methode* ein, die eine Prüfung anhand der Indexnummer (*Parameter*) an den Aufrufer als *Ergebnis* zurückgibt
- Fügen Sie eine *Methode* ein (`printExams`), die die vorhandenen Prüfungen in die Konsole ausgibt (Nummer der Prüfung und Durchschnittsnote) (*externer Methodenaufruf*).
- Ermöglichen Sie das Hinzufügen `addExam` - hier soll ein Examen übergeben werden. (*Methode, Parameter, Typ*) Wenn alle Examen hinzugefügt worden sind führt jeder weitere Aufruf der Methode zu der Ausgabe „Es können keine weiteren Prüfungsergebnisse erfasst werden!“ (*Methode, bedingte Anweisung, boolescher Ausdruck*)

Prüfungen VI

- Ermöglichen Sie das Löschen eines Examens `removeExam`. Das zu löschende Examen soll durch einen Parameter identifiziert werden. Sollte kein solches Examen existieren wird die Meldung „Das angegebenen Examen existiert nicht und kann somit nicht gelöscht werden“ ausgegeben werden. (*Methode, bedingte Anweisung, boolescher Ausdruck*)

Hinweise

- Zum Vergleich zweier Gleitkommazahlen (a und b) verwenden Sie bitte den Ausdruck
  `Double.compare(a, b)`.

	0	wenn $a = b$
Dieser liefert	-1	wenn $a < b$
	+1	wenn $a > b$

- Halten Sie sich an die (partiellen) *Signaturen* der *Methoden*, die gegebenen Bezeichner für *Exemplarvariablen* und *Klassen* und an die Reihenfolge der *Parameter*.
- Annahme: Namen von Studenten sind niemals gleich!
- Verhindern Sie Fehler durch Zugriffe auf nicht vorhandene Elemente (bedingte Anweisungen). Fehlermeldungen sollen nur ausgegeben werden, wenn diese spezifiziert wurden. Bei Methoden mit Ergebnis geben Sie bitte einen Fehlerwert zurück (z.B. null bei Objekten, -1 bei Zahlen)

Hochladen der Lösung

1. Speichern Sie Ihr Projekt als jar Datei <xxx>_exam.jar achten Sie darauf, dass der *Quelltext* in den .jar Dateien enthalten ist. <xxx> steht dabei für Ihren Nachnamen.
2. Laden Sie die Datei in moodle hoch.
3. Gehen Sie zum gemütlichen Teil des Abends über.