University of Zurich[UZH]

s.e.a.l. software evolution & architecture lab

Informatik I – Eprog HS11

Übung 4

1 Aufgabe: Information Hiding

1.1 Lernziele

1. Internes Abändern einer bestehenden Klasse.
2. Beibehalten der Schnittstellen

1.2 Aufgabenstellung

a) Student

Ein Student hat folgende Eigenschaften:

- *Name*
- *Alter* (um an der Universität zu Studieren muss man mindestens 18 sein)
- *Matrikelnummer* (besteht aus mindestens 8 Zeichen)

Implementieren Sie diese Ausgangslage in Java. Schreiben Sie Getter-Methoden und stellen Sie mit Setter-Methoden sicher, dass die Wertbeschränkungen eingehalten werden. Wird ein ungültiges Alter übergeben, soll eine Meldung ausgegeben und das Alter auf -1 gesetzt werden. Ist die Matrikelnummer zu kurz, soll eine Meldung ausgegeben und `"XXXXXXXX"` gesetzt werden.

Implementieren Sie die `toString()`-Methode. Sie soll entweder einen sinnvollen String mit Name, Alter und Matrikelnummer oder `"ungültige Studenteninformation"` zurückgeben.

Schreiben Sie einen `TestDriver1.java`, in welchem Sie alle Methoden testen.

b) Anpassen der Student-Klasse

Die Sekretärin beklagt sich bei den Informatikdiensten über dieses System, weil sie jedes Jahr das Alter aller Studenten anpassen muss. Passen Sie nun die Klasse so an, dass fortan anstelle des Alters das Geburtsjahr der Studenten gespeichert wird. Implementieren Sie die passenden Accessor-Methoden `getYearOfBirth()` und `setYearOfBirth(...)`. Da die Universität aber noch viele alte Programme am Laufen hat, die die Klasse `Student` verwenden, müssen die Zugänge über die alten Accessor-Methoden (`setAge(...)` und `getAge()`) unbedingt gewährleistet bleiben.

Fügen Sie Ihrer Klasse folgende Hilfsmethode hinzu. Danach erhalten Sie innerhalb der Klasse mit `getCurrentYear()` das jeweils aktuelle Jahr als Integer.

```
1   private int getCurrentYear() {
2       return (new java.util.GregorianCalendar()).get(java.util.
            GregorianCalendar.YEAR);
3   }
```

Schreiben Sie einen `TestDriver2.java`, in welchem Sie insbesondere Getter und Setter des Geburtsjahres testen.

Testen Sie Ihre abgeänderte Klasse nun mit dem `TestDriver1.java` aus der ersten Teilaufgabe und überprüfen Sie damit, ob die Schnittstellen gleich geblieben sind.

2 Aufgabe: Eindimensionales Array & Scanner

2.1 Lernziele

1. Array kennenlernen und Operationen auf Array korrekt ausführen können.
2. Java API anwenden können.

2.2 Aufgabenstellung

Anna betreut verschiedene Veranstaltungen und möchte Ende Semester für jede einzelne den Notenspiegel ihrer Studierenden zusammenstellen. Vor ihr liegen verschiedene, nach Veranstaltung getrennte Stapel von korrigierten und bewerteten Abschlussprüfungen. Allerdings sind die Stapel in keiner Weise sortiert. Sie wünscht sich, dass sie in einem Tool, welches ihre betreuten Fächer kennt, die Noten eingeben kann, indem sie eine Prüfung nach der anderen vom jeweiligen Stapel nimmt und deren Note eintippt. Das Tool soll zählen, wieviele Prüfungen mit einer bestimmten Note abgelegt wurden und am Schluss den Notenspiegel ausgeben.

1. Erstellen Sie dazu eine Klasse `Grades`, welche ganzzahlige Noten einlesen und zählen kann. Via Setter soll per `String` festgelegt werden können, von welchem Fach die Noten gespeichert werden. Die Noten selber, d. h. die Anzahl Prüfungen je Note, werden in einem `int`-Array abgespeichert. Standardmässig umfasst die Notenskala die Noten von 1 bis 6. Es soll aber auch möglich sein, eine andere Maximalnote zu definieren. Sie muss sinnvollerweise allerdings mindestens 2 betragen.
2. In der Methode `collectGrades()` sollen über die Kommandozeile solange Keyboard-Inputs eingelesen und verarbeitet werden, bis keine `int`-Werte mehr eingegeben werden. Verwenden Sie dazu einen `Scanner`.[1] Wird ein im Notenbereich liegender `int`-Wert eingelesen, wird das entsprechende Feld im Array, welches die Anzahl der betreffenden Note speichert, um 1 erhöht.
   Beispiel(Pseudocode mit alphabetischen Noten): Wird die Note A eingegeben, soll das Feld des Arrays, welches die Anzahl aller erreichten Noten A speichert, um 1 erhöht werden:
   ```
   noten[A] = noten[A] + 1;
   ```
3. Fügen Sie der Klasse die Methode `printGrades()` hinzu, welche je die Anzahl Prüfungen mit der entsprechenden Note ausgibt.
4. Testen Sie die Methoden in einem TestDriver.

[1] Falls nötig, konsultieren Sie das Buch, indem Sie im Index nach `Scanner` suchen. Zu den zur Verfügung stehenden Methoden der `Scanner`-Klasse gibt die Java-API Auskunft.

3 Aufgabe: Multidimensionales Array & Random

3.1 Lernziele

1. Multidimensionales Array korrekt benutzen können.
2. Java API anwenden können.
3. Sie können mit Hilfe der Klasse `Random` (Pseudo-)Zufallszahlen generieren.

3.2 Aufgabenstellung

a) Worträtsel

In dieser Aufgabe werden Sie ein kleines Wortsuchspiel programmieren, ähnlich wie Sie es vielleicht aus Rätselheftchen kennen. Auf einem 2-dimensionalen Spielfeld werden Ihnen zufällige Buchstaben angezeigt, aus denen Sie ein zusammenhängendes Wort suchen können.

```
1  S K F D I M C B Z G L U
2  W P R O F G A L L W E W
3  Q P E S I X F R O E P V
4  P N E F C S P T K I F X
5  D V S V R E T L P U O D
6  Y H W X E P R O G M S R
```

1. Implementieren Sie zunächst eine Hilfsmethode `nextRandomLetter()`, die einen Zufallsbuchstaben zurückgibt. Verwenden Sie hierzu die Klasse `java.util.Random` um eine Zufallszahl zu erhalten (die Methoden der Klasse, sowie Beispiele zu deren Benutzung entnehmen Sie der Java API). Wie Sie eine Zahl in einen Buchstaben verwandeln können, entnehmen Sie dem Abschnitt 3.4. Testen Sie diese Methode in einem TestDriver.
2. Verwenden Sie ein mehrdimensionales Array um das Spielfeld zu speichern. Implementieren Sie eine Methode `initializeGame(int width, int height)`, welche die Masse des Spielfeldes setzt und das Array initialisiert. Implementieren Sie Ihr Spiel so, dass Sie beliebige Masse für die Spielfeldgrösse verwenden können.
3. Implementieren Sie eine Methode `fillBoard()` welche das Spielfeld zufälligerweise mit Buchstaben füllt.
4. Implementieren Sie eine Methode `drawBoard()`, welche das Spielfeld auf der Kommandozeile sinnvoll ausgibt.
5. Testen Sie Ihren gesamten Code mit einem TestDriver. Nun können Sie das Spiel ein erstes Mal spielen und versuchen einige Wörter horizontal, vertikal oder diagonal zu finden.[2]
6. Schreiben Sie nun in Ihrer Klasse eine weitere Methode `play(...)`, die (mittels der erstellten Methoden) zuerst das Spiel initialisiert, dann das Spielfeld auffüllt und es ausgibt. Ändern Sie nun alle Methoden und Instanzvariablen (bis auf `play(...)`) von `public` nach `private`. Was spricht für dieses Vorgehen?

b) Zusatzaufgabe

1. Um die Chancen zu erhöhen, dass Sie ein Wort finden, sollten häufiger Vokale erscheinen. Eine einfache Möglichkeit dies umzusetzen wäre, dass in $1/3$ aller Fälle ein Vokal und in den restlichen Fällen irgend ein Buchstabe eingefüllt wird.[3] Probieren Sie verschiedene Verhältnisse aus.

[2] Suchen können Sie vor dem Bildschirm mit Stift und Papier, dazu müssen Sie nichts implementieren.
[3] Dass in letzterem Falle sowohl Konsonanten wie auch Vokale eingefüllt werden, kann vernachlässigt werden.

3.3 Regelerweiterung zum Worträtsel: Boggle

Das Spiel "Boggle" wird mit 16 Buchstabenwürfeln gespielt, die auf einem 4×4-grossen Spielfeld aufgereiht werden. Aus dieser Ausgangslage gilt es Wörter zu finden, die mindestens drei Buchstaben lang sind. Die einzelnen Buchstaben müssen sich dabei vertikal, horizontal oder diagonal berühren. Die Wörter `GANS` und `MODE` wären mögliche Lösungen in der folgenden Situation:

```
1 G D S Z
2 A N R K
3 M I E T
4 T O D I
```

Passen Sie Ihr Spielfeld nun auf eine Grösse von 4×4 an und Sie können nun dieses Spiel mit diesen Regeln spielen. Wenn Sie all Ihre Schleifen dynamisch genug geschrieben haben, brauchen Sie dazu nur `<name_der_game_variablen>.play(4,4);` aufzurufen. Viel Spass!

Ausführliche Regeln finden Sie beispielsweise beim Spielehersteller Hasbro (als PDF) oder auf Wikipedia.

3.4 Anhang: Type Casting int zu char

Das folgende Beispiel ist aus dem Buch Savitch, Seite 89:

```
1 char symbol = '7';
2 System.out.println( (int)symbol );
```

Die Ausgabe dieses Snippets ist nicht etwa `7`, sondern `55`. Wie kommt das? Java (und alle anderen Programmiersprachen) verwenden eine (zufällige) Nummerierung aller darstellbaren Zeichen (Zahlen, Buchstaben, Satzzeichen, Sonderzeichen etc.), so dass diese einer bestimmten Zahl zugewiesen sind. Ziffern (wie die 7) werden genauso wie andere Zeichen betrachtet. Als diese Nummerierung aufgestellt wurde, hat man sich nicht die Mühe gemacht, den Ziffern 0-9 die Positionen 0-9 zuzuweisen. So hat die Ziffer '7' die Position 55 in dieser Nummerierung erhalten.

Diese Nummerierung ist das Unicode Character Set.[4] Die Zeichen 'A' bis 'Z' sind den Zahlen 65 bis 90 zugewiesen. Eine Tabelle mit den wichtigsten Zeichen finden Sie im Savitch-Buch als Appendix 7 auf Seite 901 oder auch auf dieser Webseite.

Um nun aus einer Zahl das entsprechende Zeichen zu erhalten, kehren wir das obige Code-Beispiel um. Wir führen nun ein Type-Casting von einem Integer zu einem Character durch:

```
1 int code = 71;
2 char myCharacter = (char)code;
3 System.out.println(myCharacter);
4
5 //Ausgabe:
6 G
```

[4] Falls Sie schon von ASCII gehört haben: Unicode umschliesst die ASCII-Nummerierung. Für die Buchstaben A-Z ist die Nummerierung identisch.