Fehler vermeiden (JUnit)

Lernziele

- Die Studierenden können JUnit-Tests schreiben.
- Die Studierenden kennen die `assertEquals`-Methode von JUnit und können diese verwenden.
- Die Studierenden kennen den Zugriffsmodifikator `package`.

Aufgabe 1

Sie haben in einem vorhergehenden Praktikum ein Programm zum Analysieren der Worthäufigkeit in Texten geschrieben. Dieses Programm soll nun getestet werden. Importieren Sie als Basis das Projekt 07_Praktikum_Worthaeufigkeitsanalyse, welches das fertige Programm beinhaltet, in Eclipse. Wenn Sie möchten, so können Sie auch Ihren Lösungsvorschlag aus jenem Praktikum testen, dazu müssen Sie die nachfolgenden Teilaufgaben entsprechend auf Ihre Lösung (z.B. bez. Methodennamen) anpassen.

a) Schreiben Sie einen Test für die Methode `entferneSatzzeichen`. Schreiben Sie einen Test, der alle Satzzeichen testet.

   Hinweis: Wenn Sie jetzt einen JUnit-Test schreiben wollen, um die Methode `entferneSatzzeichen` zu testen, dann können Sie dies nicht tun, da der Zugriffsmodifikator `private` ist. Diese Situation trifft oft auf, wenn nachträglich zu bestehendem Code JUnit-Tests geschrieben werden sollen. Welche Möglichkeiten haben wir, damit die Methode trotzdem getestet werden kann?

   - Die erste Möglichkeit ist, den Zugriffsmodifikator von `private` auf `public` zu ändern. Meistens ist dies aber nicht gewünscht, da dadurch die Schnittstelle der Klasse verändert wird.
   - Die zweite Möglichkeit arbeitet mit Vererbung und dem Zugriffsmodifikator `protected`. Da wir das Konzept der Vererbung noch nicht behandelt haben, verschieben wir die Diskussion dieser Möglichkeit auf später.
   - Eine dritte Möglichkeit, welche wir für diese Aufgabe benutzen wollen, ist im Anhang *Appendix Zugriffsmodifikator* beschrieben. Lesen Sie diesen Anhang, damit Sie die Aufgabe lösen können.

b) Schreiben Sie einen Test für die Methode `verarbeiteText`. Diese Methode hat keinen Rückgabewert. Um diese Methode trotzdem testen zu können, müssen Sie in der zu testenden Klasse Hilfsmethoden einfügen. Überlegen Sie sich, welche Fälle zu testen sind (insbesondere ob auch Wiederholungen eines Worts richtig erkannt werden) und welches Ergebnis Sie erwarten. Überlegen Sie sich auch einen negativen Test.

Aufgabe 2

Sie haben in einem vorhergehenden Praktikum ein Programm zum Verwalten von Prüfungsergebnissen geschrieben. Dieses Programm soll nun getestet werden. Importieren Sie als Basis das Projekt 07_Praktikum_Notenprogramm, welches das fertige Programm beinhaltet, in Eclipse. Auch hier können Sie alternativ Ihren eigenen Lösungsvorschlag aus jenem Praktikum testen.

a) In der Klasse `Pruefungsverwaltung` gibt es die Methode `rundeAufHalbeNote`. Testen Sie, ob diese Methode die Noten korrekt rundet. Überlegen Sie sich dazu zuerst die möglichen Äquivalenzklassen. Schreiben Sie anschliessend die dazugehörigen Unit-Tests.

b) In der Klasse `Pruefungsverwaltung` gibt es die Methode `generiereText`. Testen Sie, ob diese Methode den Text richtig generiert.

c) In der Klasse `ZufaelligeNotengebung` gibt es die Methode `generiereZufaelligePruefugsnote`. Diese Methode liefert (pseudo-)zufällig verteilte Noten von 1 bis 6 zurück. Solche Methoden sind schwierig zu testen, da der Rückgabewert nicht vorhersehbar ist. Überlegen Sie sich, wie Sie trotzdem einen Test schreiben können, welcher gewährleistet, dass die Methode die korrekten Prüfungsnoten von 1 bis 6 generiert.

Appendix Zugriffsmodifikator

Sie kennen bereits die Zugriffsmodifikatoren `private` und `public`. Der Zugriffsmodifikator `public` erlaubt einen globalen Zugriff auf Methoden und Variablen – es gibt also keinerlei Einschränkung um auf eine solche `public` Methode oder Variable zuzugreifen.

Der `private`-Zugriffsmodifikator hingegen ist der restriktivste, da er nur einen Zugriff innerhalb der gleichen Klasse erlaubt. Auch eine Subklasse hat keinen Zugriff.

Sie lernen jetzt einen dritten Zugriffsmodifikator kennen. In der Vorgabe für die Aufgabe 1 finden Sie in Worthaeufigkeitsanalyse.java die Methode `entferneSatzzeichen`. Diese Methode hat folgende Signatur:

```
private String entferneSatzzeichen (String wort)
```

Wenn Sie jetzt einen JUnit-Test schreiben wollen, um diese Methode zu testen, dann können Sie dies nicht tun, da der Zugriffsmodifikator `private` ist. Eine (schlechte) Möglichkeit wäre, den Zugriffsmodifikator von `private` auf `public` zu ändern. Der Nachteil dabei ist, dass dann diese Methode global aufgerufen werden kann. Eine elegantere Lösung ist, einen Zugriffsmodifikator zu wählen, welcher nicht so restriktiv wie `private` ist, aber auch nicht so global wie `public` – sozusagen ein `private`-Zugriffsmodifikator, welcher zusätzlich den Zugriff für den JUnit Test erlaubt. Dies ist möglich, wenn man *keinen* Zugriffsmodifkator definiert, dann ist der Zugriff innerhalb desselben Package erlaubt:

```
String entferneSatzzeichen (String wort)
```

Da in diesem Praktikum die Testklasse und die zu testenden Klassen im gleich Package liegen (dem sog. default-Package, da wir noch keine expliziten Packages verwenden), kann die Testklasse deshalb auf die Methode zugreifen.

Die untere Tabelle zeigt eine Übersicht, woher in Abhängigkeit des Zugriffmodifikators die Verwendung einer Metode (oder eines Datenfelds) möglich ist.

Zugriffsmodifikator	Klasse	Package	Sub-Klasse	Global
`public`	Ja	Ja	Ja	Ja
-	Ja	Ja	Nein	Nein
`private`	Ja	Nein	Nein	Nein

Hinweis: Es gibt verschiedene Namen für den neuen Zugriffsmodifikator. Teilweise wird er Package-, Kein- oder auch Friendly-Zugriffsmodifikator genannt.

Studieren Sie die folgende Website von Oracle, welche alle Zugriffsmodifikatoren und deren Eigenschaften im Detail beschreibt:
http://docs.oracle.com/javase/tutorial/java/javaOO/accesscontrol.html