Benjamin Schürmann

Auswertung vom 14.04.2012

Benjamin Schürmann

1. Übersetzen der Java-Klassen

Dieser Abschnitt enthält etwaige Fehlermeldungen oder Warnungen des Übersetzers während des Übersetzungsvorgangs. Falls Sie hier keine Meldungen finden, kann dies bedeuten, dass Ihre hochgeladene Lösung keine Java-Dateien enthält oder alle Klassen sich fehlerfrei übersetzen ließen.

Benjamin Schürmann

2. Formale Prüfung

Dieser Abschnitt enthält das Ergebnis der formalen Prüfung. Es wird geprüft, ob alle in der Aufgabe geforderten Klassen mit allen geforderten Methoden vorhanden sind.

2.1. Übersicht der Klassen und Schnittstellen

❶	❷	❸	❹	❺	❻	Klasse oder Schnittstelle	Fehlerhinweis
✔						Klasse Mitarbeiter	
		✘				Klasse Vorgesetzter	

Legende: Die Klasse oder Schnittstelle ...

❶	ist vorhanden und ohne formale Fehler.
❷	ist nicht vorhanden oder nicht compilierbar.
❸	ist vorhanden, hat aber formale Fehler. Sofern es sich um formale Fehler in Methoden handelt, finden Sie Details dazu in der Methodenübersicht.
❹	enthält Fehler bzgl. der darin definierten symbolischen Konstanten.
❺	enthält Fehler bzgl. ihrer Oberklasse.
❻	enthält Fehler bzgl. der implementierten Schnittstellen.

2.2. Übersicht der Methoden

❶	❷	❸	❹	❺	Klasse	Methode	Fehlerhinweis
✔					Mitarbeiter	public Mitarbeiter(String)	
✔					Mitarbeiter	public String gibHierarchie()	
✔					Mitarbeiter	public String gibInfo()	
✔					Mitarbeiter	public static void setzeAllgemeinesLimit(int)	
✔					Mitarbeiter	public boolean darfBestellen(int)	
✔					Mitarbeiter	public void setzeVorgesetzten(Vorgesetzter)	
✔					Vorgesetzter	public Vorgesetzter(String)	
✔					Vorgesetzter	public String gibHierarchie()	
✔					Vorgesetzter	public String gibInfo()	
	✘				Vorgesetzter	public void setzeBestelllimit(int)	
✔					Vorgesetzter	public boolean darfBestellen(int)	
✔					Vorgesetzter	public void setzeVorgesetzten(Vorgesetzter)	
				✔	Vorgesetzter	public void setzteBestelllimit(int)	

Legende: Die Methode ist ...

❶	vorhanden und ohne formale Fehler.
❷	nicht vorhanden.

❸	vorhanden, hat aber nicht die geforderten Modifikatoren. Die von Ihnen deklarierten Modifikatoren stehen im Fehlerhinweis.
❹	vorhanden, hat aber nicht den geforderten Ergebnistyp. Der von Ihnen deklarierte Ergebnistyp steht im Fehlerhinweis.
❺	zusätzlich von Ihnen definiert.

3. Funktionale Prüfung

Dieser Abschnitt enthält das Ergebnis der funktionalen Prüfung. Es wird geprüft, ob alle in der Aufgabe geforderten Methoden für bestimmte Testdaten die erwarteten Ergebnisse liefern. Fehlermeldungen finden Sie in diesem Abschnitt auch, wenn geforderte Klassen oder Methoden fehlen. In diesem Fall scheitert bereits der Aufruf der Methoden.

3.1. Test der Klasse Mitarbeiter

3.1.1. Testsituation „AllgemeinesLimit30“

Die Testsituation wird in folgenden Schritten aufgebaut:

1.	Setze allgemeines Mitarbeiter-Limit auf 20 EUR.
2.	Erzeuge Mitarbeiter Walter Winkelmann.
3.	Erzeuge Mitarbeiter Willi Winzig.
4.	Erzeuge Vorgesetzten Waltraud Wichtig.
5.	Erzeuge Vorgesetzten Hermann Wichtiger.
6.	Mache Waltraud Wichtig zum Vorgesetzten von Walter Winkelmann.
7.	Mache Hermann Wichtiger zum Vorgesetzten von Waltraud Wichtig.
8.	Setze allgemeines Mitarbeiter-Limit auf 30 EUR.

Tests der Methode darfBestellen(int)

	Testfall	**Fehlerhinweis**
✔	Prüfe, ob Walter Winkelmann für 21 EUR bestellen darf.	
✔	Prüfe, ob Hermann Wichtiger für 25 EUR bestellen darf.	

Tests der Methode gibInfo()

	Testfall	**Fehlerhinweis**
✔	Prüfe Info von Hermann Wichtiger.	
✔	Prüfe Info von Walter Winkelmann.	
✔	Prüfe Info von Willi Winzig.	

3.1.2. Testsituation „Personal“

Die Testsituation wird in folgenden Schritten aufgebaut:

1.	Setze allgemeines Mitarbeiter-Limit auf 20 EUR.
2.	Erzeuge Mitarbeiter Walter Winkelmann.
3.	Erzeuge Mitarbeiter Willi Winzig.
4.	Erzeuge Vorgesetzten Waltraud Wichtig.

5.	Erzeuge Vorgesetzten Hermann Wichtiger.
6.	Mache Waltraud Wichtig zum Vorgesetzten von Walter Winkelmann.
7.	Mache Hermann Wichtiger zum Vorgesetzten von Waltraud Wichtig.

Tests der Methode darfBestellen(int)

	Testfall	**Fehlerhinweis**
✔	Prüfe, ob Walter Winkelmann für 15 EUR bestellen darf.	
✔	Prüfe, ob Walter Winkelmann für 20 EUR bestellen darf.	
✔	Prüfe, ob Walter Winkelmann für 21 EUR bestellen darf.	
✔	Prüfe, ob Hermann Wichtiger für 15 EUR bestellen darf.	
✔	Prüfe, ob Hermann Wichtiger für 25 EUR bestellen darf.	

Tests der Methode gibHierarchie()

	Testfall	**Fehlerhinweis**
✔	Prüfe Hierarchie von Waltraud Wichtig.	
✔	Prüfe Hierarchie von Hermann Wichtiger.	
✔	Prüfe Hierarchie von Walter Winkelmann.	
✔	Prüfe Hierarchie von Willi Winzig.	

3.1.3. Testsituation „Wichtig10“

Beim Aufbau der Testsituation ist es zu einem Fehler gekommen.

1.	Setze allgemeines Mitarbeiter-Limit auf 20 EUR.	
2.	Erzeuge Mitarbeiter Walter Winkelmann.	
3.	Erzeuge Mitarbeiter Willi Winzig.	
4.	Erzeuge Vorgesetzten Waltraud Wichtig.	
5.	Erzeuge Vorgesetzten Hermann Wichtiger.	
6.	Mache Waltraud Wichtig zum Vorgesetzten von Walter Winkelmann.	
7.	Mache Hermann Wichtiger zum Vorgesetzten von Waltraud Wichtig.	
8.	Setze allgemeines Mitarbeiter-Limit auf 30 EUR.	

9.	Setze Limit von Waltraud Wichtig auf 10 EUR.	java.lang.NoSuchMethodError: Vorgesetzter.setzeBestelllimit(I)V		
		FunktionstestMitarbeiter$3	setUp	65
		org.testsupport.junit.TestFixtureBuilder	build	78
		FunktionstestMitarbeiter	testDarfBestellen9	160

3.1.4. Testsituation „Wichtig5000“

Beim Aufbau der Testsituation ist es zu einem Fehler gekommen.

1.	Setze allgemeines Mitarbeiter-Limit auf 20 EUR.			
2.	Erzeuge Mitarbeiter Walter Winkelmann.			
3.	Erzeuge Mitarbeiter Willi Winzig.			
4.	Erzeuge Vorgesetzten Waltraud Wichtig.			
5.	Erzeuge Vorgesetzten Hermann Wichtiger.			
6.	Mache Waltraud Wichtig zum Vorgesetzten von Walter Winkelmann.			
7.	Mache Hermann Wichtiger zum Vorgesetzten von Waltraud Wichtig.			
8.	Setze allgemeines Mitarbeiter-Limit auf 30 EUR.			
9.	Setze Limit von Waltraud Wichtig auf 5000 EUR.	java.lang.NoSuchMethodError: Vorgesetzter.setzeBestelllimit(I)V		
		FunktionstestMitarbeiter$4	setUp	72
		org.testsupport.junit.TestFixtureBuilder	build	78
		FunktionstestMitarbeiter	testGibInfo1	187

3.1.5. Testsituation „WichtigOhneChef“

Die Testsituation wird in folgenden Schritten aufgebaut:

1.	Setze allgemeines Mitarbeiter-Limit auf 20 EUR.
2.	Erzeuge Mitarbeiter Walter Winkelmann.
3.	Erzeuge Mitarbeiter Willi Winzig.
4.	Erzeuge Vorgesetzten Waltraud Wichtig.
5.	Erzeuge Vorgesetzten Hermann Wichtiger.
6.	Mache Waltraud Wichtig zum Vorgesetzten von Walter Winkelmann.
7.	Mache Hermann Wichtiger zum Vorgesetzten von Waltraud Wichtig.
8.	Entziehe Waltraud Wichtig den Vorgesetzten.

Tests der Methode gibHierarchie()

	Testfall	**Fehlerhinweis**

Benjamin Schürmann

✔	Prüfe Hierarchie von Walter Winkelmann.	

4. Checkstyle-Prüfung

```
Starting audit...
Mitarbeiter.java:12:5: Javadoc-Kommentar fehlt.
Mitarbeiter.java:12:23: Variable 'bestLimit' muss private sein.
Mitarbeiter.java:14:5: Javadoc-Kommentar fehlt.
Mitarbeiter.java:14:19: Variable 'name' muss private sein.
Mitarbeiter.java:15:5: Javadoc-Kommentar fehlt.
Mitarbeiter.java:15:25: Variable 'vorgesetzter' muss private sein.
Mitarbeiter.java:16:5: Javadoc-Kommentar fehlt.
Mitarbeiter.java:16:19: Variable 'position' muss private sein.
Mitarbeiter.java:18:5: Javadoc-Kommentar fehlt.
Mitarbeiter.java:18:31: 'Name' entspricht nicht dem Muster '^[a-z][a-zA-Z0-9]*$'.
Mitarbeiter.java:25:5: Javadoc-Kommentar fehlt.
Mitarbeiter.java:27:30: Vor '=' befindet sich kein Leerzeichen.
Mitarbeiter.java:31:5: Javadoc-Kommentar fehlt.
Mitarbeiter.java:37:5: Javadoc-Kommentar fehlt.
Mitarbeiter.java:43:5: Javadoc-Kommentar fehlt.
Mitarbeiter.java:48: Zeile länger als 80 Zeichen
Mitarbeiter.java:48:94: '+' sollte in einer neuen Zeile stehen.
Mitarbeiter.java:49: Zeile länger als 80 Zeichen
Mitarbeiter.java:50: Zeile länger als 80 Zeichen
Mitarbeiter.java:51: Zeile länger als 80 Zeichen
Mitarbeiter.java:54: Zeile länger als 80 Zeichen
Mitarbeiter.java:54:101: '+' sollte in einer neuen Zeile stehen.
Mitarbeiter.java:61:5: Javadoc-Kommentar fehlt.
Mitarbeiter.java:63: Zeile länger als 80 Zeichen
MitarbeiterTest.java:12: method def modifier bei Einrücktiefe 8 nicht an korrekter
Einrücktiefe 4
MitarbeiterTest.java:12:9: Javadoc-Kommentar fehlt.
MitarbeiterTest.java:13: Kind von method def bei Einrücktiefe 12 nicht an korrekter
Einrücktiefe 8
MitarbeiterTest.java:14: Kind von method def bei Einrücktiefe 12 nicht an korrekter
Einrücktiefe 8
MitarbeiterTest.java:15: Kind von method def bei Einrücktiefe 12 nicht an korrekter
Einrücktiefe 8
MitarbeiterTest.java:16: Kind von method def bei Einrücktiefe 12 nicht an korrekter
Einrücktiefe 8
MitarbeiterTest.java:17: Kind von method def bei Einrücktiefe 12 nicht an korrekter
Einrücktiefe 8
MitarbeiterTest.java:18: Kind von method def bei Einrücktiefe 12 nicht an korrekter
Einrücktiefe 8
MitarbeiterTest.java:19: Kind von method def bei Einrücktiefe 12 nicht an korrekter
Einrücktiefe 8
MitarbeiterTest.java:20: Kind von method def bei Einrücktiefe 12 nicht an korrekter
Einrücktiefe 8
MitarbeiterTest.java:21: Kind von method def bei Einrücktiefe 12 nicht an korrekter
Einrücktiefe 8
```

```
MitarbeiterTest.java:22: Kind von method def bei Einrücktiefe 12 nicht an korrekter
Einrücktiefe 8
MitarbeiterTest.java:23: Kind von method def bei Einrücktiefe 12 nicht an korrekter
Einrücktiefe 8
MitarbeiterTest.java:24: Kind von method def bei Einrücktiefe 12 nicht an korrekter
Einrücktiefe 8
MitarbeiterTest.java:25: Kind von method def bei Einrücktiefe 12 nicht an korrekter
Einrücktiefe 8
MitarbeiterTest.java:26: Kind von method def bei Einrücktiefe 12 nicht an korrekter
Einrücktiefe 8
MitarbeiterTest.java:27: Kind von method def bei Einrücktiefe 12 nicht an korrekter
Einrücktiefe 8
MitarbeiterTest.java:28: Kind von method def bei Einrücktiefe 12 nicht an korrekter
Einrücktiefe 8
MitarbeiterTest.java:29: Kind von method def bei Einrücktiefe 12 nicht an korrekter
Einrücktiefe 8
MitarbeiterTest.java:30: Kind von method def bei Einrücktiefe 12 nicht an korrekter
Einrücktiefe 8
MitarbeiterTest.java:31: Kind von method def bei Einrücktiefe 12 nicht an korrekter
Einrücktiefe 8
MitarbeiterTest.java:32: Kind von method def bei Einrücktiefe 12 nicht an korrekter
Einrücktiefe 8
MitarbeiterTest.java:33: Kind von method def bei Einrücktiefe 12 nicht an korrekter
Einrücktiefe 8
MitarbeiterTest.java:34: Kind von method def bei Einrücktiefe 12 nicht an korrekter
Einrücktiefe 8
MitarbeiterTest.java:35: Kind von method def bei Einrücktiefe 12 nicht an korrekter
Einrücktiefe 8
MitarbeiterTest.java:36: Kind von method def bei Einrücktiefe 12 nicht an korrekter
Einrücktiefe 8
MitarbeiterTest.java:37: Kind von method def bei Einrücktiefe 12 nicht an korrekter
Einrücktiefe 8
MitarbeiterTest.java:38: Kind von method def bei Einrücktiefe 12 nicht an korrekter
Einrücktiefe 8
MitarbeiterTest.java:39: Kind von method def bei Einrücktiefe 12 nicht an korrekter
Einrücktiefe 8
MitarbeiterTest.java:40: Kind von method def bei Einrücktiefe 12 nicht an korrekter
Einrücktiefe 8
MitarbeiterTest.java:41: Kind von method def bei Einrücktiefe 12 nicht an korrekter
Einrücktiefe 8
MitarbeiterTest.java:42: Kind von method def bei Einrücktiefe 12 nicht an korrekter
Einrücktiefe 8
Vorgesetzter.java:12:5: Javadoc-Kommentar fehlt.
Vorgesetzter.java:14:5: Javadoc-Kommentar fehlt.
Vorgesetzter.java:22:5: Javadoc-Kommentar fehlt.
Vorgesetzter.java:38: Zeile länger als 80 Zeichen
Vorgesetzter.java:38:83: '+' sollte in einer neuen Zeile stehen.
Vorgesetzter.java:39: Zeile länger als 80 Zeichen
Vorgesetzter.java:54: Zeile länger als 80 Zeichen
```

```
Vorgesetzter.java:54:101: '+' sollte in einer neuen Zeile stehen.
Audit done.
```

5. Quellcode der Java-Klassen

5.1. Mitarbeiter.java

```
 1: /*
 2:  * To change this template, choose Tools | Templates
 3:  * and open the template in the editor.
 4:  */
 5:
 6: /**
 7:  *
 8:  * @author apex
 9:  */
10: public class Mitarbeiter {
11:
12:     public static int bestLimit = 20;
13:
14:     public String name;
15:     public Vorgesetzter vorgesetzter;
16:     public String position;
17:
18:     public Mitarbeiter(String Name) {
19:
20:         this.name = Name;
21:         position = "Mitarbeiter";
22:
23:     }
24:
25:     public static void setzeAllgemeinesLimit(int limit) {
26:
27:         Mitarbeiter.bestLimit= limit;
28:
29:     }
30:
31:     public void setzeVorgesetzten(Vorgesetzter chef) {
32:
33:         this.vorgesetzter = chef;
34:
35:     }
36:
37:     public boolean darfBestellen(int limit) {
38:
39:         return limit <= bestLimit;
40:
41:     }
42:
43:     public String gibInfo() {
44:
```

```
 45:         StringBuffer infotext = new StringBuffer();
 46:
 47:         if (vorgesetzter != null) {
 48:             infotext.append("Ich bin Mitarbeiter, Name " + name + ". Mein
Vorgesetzter ist " +
 49:                     vorgesetzter.name + ". Mein Bestelllimit ist " + bestLimit + "
EUR.");
 50:         } else if (vorgesetzter != null && name.equals(vorgesetzter.name)) {
//TODO
 51:             infotext.append("Ich bin Vorgesetzter, Name " + name + ". Mein
Bestelllimit ist " + bestLimit
 52:                     + " EUR.");
 53:         } else if (vorgesetzter == null) {
 54:             infotext.append("Ich bin freier Mitarbeiter, Name " + name + ". Mein
Bestelllimit ist " +
 55:                     bestLimit + " EUR.");
 56:         }
 57:
 58:         return infotext.toString();
 59:     }
 60:
 61:     public String gibHierarchie() {
 62:
 63:         return ((vorgesetzter != null) ? vorgesetzter.gibHierarchie() +
"\nMitarbeiter " + this.name
 64:                 : "freier Mitarbeiter " + this.name);
 65:
 66:     }
 67:
 68: }
```

5.2. Vorgesetzter.java

```
  1: /*
  2:  * To change this template, choose Tools | Templates
  3:  * and open the template in the editor.
  4:  */
  5:
  6: /**
  7:  *
  8:  * @author apex
  9:  */
 10: public class Vorgesetzter extends Mitarbeiter {
 11:
 12:     private int bestLimit;
 13:
 14:     public Vorgesetzter(String name) {
 15:
 16:         super(name);
 17:         super.position = "Vorgesetzter";
```

```
 18:         bestLimit = -1;
 19: 
 20:     }
 21: 
 22:     public void setzteBestelllimit(int limit) {
 23: 
 24:         this.bestLimit = limit;
 25: 
 26:     }
 27: 
 28:     @Override
 29:     public String gibInfo() {
 30: 
 31:         StringBuffer infotext = new StringBuffer();
 32:         String vorgesetzter = "";
 33: 
 34:         if (super.vorgesetzter != null) {
 35:             vorgesetzter = ". Mein Vorgesetzter ist " + super.vorgesetzter.name;
 36:         }
 37: 
 38:         infotext.append("Ich bin Vorgesetzter, Name " + super.name + vorgesetzter
+
 39:                 ". Mein Bestelllimit" + " ist " + ((bestLimit == -1) ?
super.bestLimit : this.bestLimit) + " EUR.");
 40: 
 41:         return infotext.toString();
 42:     }
 43: 
 44:     @Override
 45:     public boolean darfBestellen(int limit) {
 46: 
 47:         return limit <= ((bestLimit == -1) ? super.bestLimit : this.bestLimit);
 48: 
 49:     }
 50: 
 51:     @Override
 52:     public String gibHierarchie() {
 53: 
 54:         return ((this.vorgesetzter != null) ? this.vorgesetzter.gibHierarchie() +
"\nVorgesetzter " +
 55:                 this.name : "Vorgesetzter " + this.name);
 56: 
 57:     }
 58: 
 59: }
```

5.3. MitarbeiterTest.java

```
  1: /*
  2:  * To change this template, choose Tools | Templates
```

```
 3:  * and open the template in the editor.
 4:  */
 5:
 6: /**
 7:  *
 8:  * @author apex
 9:  */
10: public class MitarbeiterTest {
11:
12:         public static void main(String[] args) {
13:             Mitarbeiter walter = new Mitarbeiter("Walter Winkelmann");
14:             Vorgesetzter waltraud = new Vorgesetzter("Waltraud Wichtig");
15:             Vorgesetzter hermann = new Vorgesetzter("Hermann Wichtiger");
16:             walter.setzeVorgesetzten(waltraud);
17:             waltraud.setzeVorgesetzten(hermann);
18:             System.out.println(walter.darfBestellen(15));
19:             System.out.println(walter.darfBestellen(20));
20:             System.out.println(walter.darfBestellen(21));
21:             System.out.println(hermann.darfBestellen(15));
22:             System.out.println(hermann.darfBestellen(25));
23:             Mitarbeiter.setzeAllgemeinesLimit(30);
24:             Mitarbeiter willi = new Mitarbeiter("Willi Winzig");
25:             System.out.println(walter.darfBestellen(21));
26:             System.out.println(hermann.darfBestellen(25));
27:             waltraud.setzteBestelllimit(10);
28:             System.out.println(waltraud.darfBestellen(10));
29:             System.out.println(waltraud.darfBestellen(11));
30:             waltraud.setzteBestelllimit(5000);
31:             System.out.println(waltraud.darfBestellen(2000));
32:             System.out.println(waltraud.darfBestellen(7000));
33:             System.out.println(waltraud.gibInfo());
34:             System.out.println(waltraud.gibHierarchie());
35:             System.out.println(hermann.gibInfo());
36:             System.out.println(hermann.gibHierarchie());
37:             System.out.println(walter.gibInfo());
38:             System.out.println(walter.gibHierarchie());
39:             waltraud.setzeVorgesetzten(null);
40:             System.out.println(walter.gibHierarchie());
41:             System.out.println(willi.gibInfo());
42:             System.out.println(willi.gibHierarchie());
43:     }
44:
45: }
46:
```