Benjamin Schürmann

Auswertung vom 10.05.2012

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.

```
Note: Some input files use unchecked or unsafe operations.
Note: Recompile with -Xlint:unchecked for details.
```

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 carsharing.Fahrzeugmanager	
✔						Klasse carsharing.FahrzeugmanagerTest	

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
✔					carsharing.Fahrzeugmanager	public carsharing.Fahrzeugmanager()	
✔					carsharing.Fahrzeugmanager	public ArrayList gibVerfuegbareFahrzeuge(String, String, String)	
✔					carsharing.Fahrzeugmanager	public boolean bucheFahrzeug(String, String, String)	
✔					carsharing.Fahrzeugmanager	public ArrayList gibFahrzeugnamen()	
✔					carsharing.Fahrzeugmanager	public void fuegeFahrzeugHinzu(String, String)	
				✔	carsharing.Fahrzeugmanager	public void bestimmeStandort(String, Fahrzeug)	
				✔	carsharing.FahrzeugmanagerTest	public void setUp()	
				✔	carsharing.FahrzeugmanagerTest	public void testGibFahrzeugnamen()	

				✔	carsharing.FahrzeugmanagerTest	public void testBucheFahrzeug()	
				✔	carsharing.FahrzeugmanagerTest	public void testGibVerfuegbareFahrzeuge()	

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.

Benjamin Schürmann

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 Fahrzeugmanager

3.1.1. Testsituation „Doppelter Fahrzeugname“

Die Testsituation wird in folgenden Schritten aufgebaut:

1.	Erzeuge Fahrzeugmanager.
2.	Erzeuge Fahrzeug Bahnhof 1 am Standort Bahnhof.
3.	Erzeuge Fahrzeug Bahnhof 2 am Standort Bahnhof.
4.	Erzeuge Fahrzeug Bahnhof 3 am Standort Bahnhof.
5.	Erzeuge Fahrzeug Rathaus 1 am Standort Rathaus.
6.	Buche Fahrzeug Bahnhof 1 von 2005/04/14 20:00 bis 2005/04/15 08:00.
7.	Buche Fahrzeug Bahnhof 1 von 2005/04/15 18:00 bis 2005/04/16 00:00.
8.	Buche Fahrzeug Bahnhof 2 von 2005/04/14 11:00 bis 2005/04/15 12:00.
9.	Buche Fahrzeug Bahnhof 3 von 2005/04/15 10:00 bis 2005/04/15 19:00.
10.	Versuch, erneut Fahrzeug Bahnhof 1 hinzuzufügen.

Tests der Methode gibFahrzeugnamen()

	Testfall	**Fehlerhinweis**	
✗	Prüfe Liste der Fahrzeugnamen.	Soll	[Bahnhof 1, Bahnhof 2, Bahnhof 3, Rathaus 1]
		Ist	[Bahnhof 1, Bahnhof 1, Bahnhof 2, Bahnhof 3, Rathaus 1]

3.1.2. Testsituation „Fahrzeuge“

Die Testsituation wird in folgenden Schritten aufgebaut:

1.	Erzeuge Fahrzeugmanager.
2.	Erzeuge Fahrzeug Bahnhof 1 am Standort Bahnhof.
3.	Erzeuge Fahrzeug Bahnhof 2 am Standort Bahnhof.
4.	Erzeuge Fahrzeug Bahnhof 3 am Standort Bahnhof.
5.	Erzeuge Fahrzeug Rathaus 1 am Standort Rathaus.
6.	Buche Fahrzeug Bahnhof 1 von 2005/04/14 20:00 bis 2005/04/15 08:00.
7.	Buche Fahrzeug Bahnhof 1 von 2005/04/15 18:00 bis 2005/04/16 00:00.
8.	Buche Fahrzeug Bahnhof 2 von 2005/04/14 11:00 bis 2005/04/15 12:00.
9.	Buche Fahrzeug Bahnhof 3 von 2005/04/15 10:00 bis 2005/04/15 19:00.

Benjamin Schürmann

Tests der Methode bucheFahrzeug(String, String, String)

	Testfall	Fehlerhinweis	
✔	Prüfe, ob das Fahrzeug Bahnhof 3 zwischen 2005/04/15 09:00 und 2005/04/15 10:00 buchbar ist.		
✘	Prüfe, ob das Fahrzeug Bahnhof 3 zwischen 2005/04/15 09:00 und 2005/04/15 11:00 buchbar ist.	Soll	false
		Ist	true
✔	Prüfe, ob das Fahrzeug Bahnhof 3 zwischen 2005/04/15 11:00 und 2005/04/15 18:00 buchbar ist.		
✘	Prüfe, ob das Fahrzeug Bahnhof 3 zwischen 2005/04/15 18:00 und 2005/04/15 20:00 buchbar ist.	Soll	false
		Ist	true
✔	Prüfe, ob das Fahrzeug Bahnhof 3 zwischen 2005/04/15 19:00 und 2005/04/15 20:00 buchbar ist.		
✘	Prüfe, ob das Fahrzeug Bahnhof 3 zwischen 2005/04/15 09:00 und 2005/04/15 20:00 buchbar ist.	Soll	false
		Ist	true

Tests der Methode gibFahrzeugnamen()

	Testfall	Fehlerhinweis
✔	Prüfe, ob der Fahrzeugmanager alle hinzugefügten Fahrzeuge enthält.	

Tests der Methode gibVerfuegbareFahrzeuge(String, String, String)

	Testfall	Fehlerhinweis	
✘	Prüfe, ob Fahrzeuge am Bahnhof zwischen 2005/04/15 11:30 und 2005/04/15 19:00 verfügbar sind.	Soll	[]
		Ist	[Fahrzeug{name=Bahnhof 1, standort=Bahnhof, startzeit=carsharing.Zeitraum@1bab50a, endzeit=carsharing.Zeitraum@c3c749}, Fahrzeug{name=Bahnhof 2, standort=Bahnhof, startzeit=carsharing.Zeitraum@150bd4d, endzeit=carsharing.Zeitraum@1bc4459}]
✘	Prüfe, ob Fahrzeuge am Bahnhof zwischen 2005/04/15 12:00 und 2005/04/15 18:00 verfügbar sind.	Soll	[Bahnhof 1, Bahnhof 2]
		Ist	[Fahrzeug{name=Bahnhof 1, standort=Bahnhof, startzeit=carsharing.Zeitraum@6e1408, endzeit=carsharing.Zeitraum@e53108}, Fahrzeug{name=Bahnhof 2, standort=Bahnhof, startzeit=carsharing.Zeitraum@f62373, endzeit=carsharing.Zeitraum@19189e1}]

✗	Prüfe, ob Fahrzeuge am Bahnhof zwischen 2005/04/15 19:15 und 2005/04/15 23:00 verfügbar sind.	Soll	[Bahnhof 2, Bahnhof 3]
		Ist	[Fahrzeug{name=Bahnhof 2, standort=Bahnhof, startzeit=carsharing.Zeitraum@1690726, endzeit=carsharing.Zeitraum@5483cd}, Fahrzeug{name=Bahnhof 3, standort=Bahnhof, startzeit=carsharing.Zeitraum@9931f5, endzeit=carsharing.Zeitraum@19ee1ac}]

3.1.3. Testsituation „Neues Fahrzeug“

Die Testsituation wird in folgenden Schritten aufgebaut:

1.	Erzeuge Fahrzeugmanager.
2.	Erzeuge Fahrzeug Bahnhof 1 am Standort Bahnhof.
3.	Erzeuge Fahrzeug Bahnhof 2 am Standort Bahnhof.
4.	Erzeuge Fahrzeug Bahnhof 3 am Standort Bahnhof.
5.	Erzeuge Fahrzeug Rathaus 1 am Standort Rathaus.
6.	Buche Fahrzeug Bahnhof 1 von 2005/04/14 20:00 bis 2005/04/15 08:00.
7.	Buche Fahrzeug Bahnhof 1 von 2005/04/15 18:00 bis 2005/04/16 00:00.
8.	Buche Fahrzeug Bahnhof 2 von 2005/04/14 11:00 bis 2005/04/15 12:00.
9.	Buche Fahrzeug Bahnhof 3 von 2005/04/15 10:00 bis 2005/04/15 19:00.
10.	Erzeuge Fahrzeug Fachhochschule 1 am Standort FH Gelsenkirchen.

Tests der Methode gibFahrzeugnamen()

	Testfall	**Fehlerhinweis**
✔	Prüfe Liste der Fahrzeugnamen.	

Benjamin Schürmann

4. Checkstyle-Prüfung

```
Starting audit...
Audit done.
```

5. Quellcode der Java-Klassen

5.1. Fahrzeugmanager.java

```
 1: //TODO class comment
 2: /*
 3:  * To change this template, choose Tools | Templates
 4:  * and open the template in the editor.
 5:  */
 6: package carsharing;
 7:
 8: import java.util.ArrayList;
 9: import java.util.Collections;
10:
11: /**
12:  *
13:  * @author apex
14:  */
15: public class Fahrzeugmanager {
16:
17:     ArrayList<Fahrzeug> fahrzeuge = new ArrayList();
18:     ArrayList<Standort> standorte = new ArrayList();
19:
20:     /**
21:      * Erzeugt Objekte dieser Klasse
22:      */
23:     public Fahrzeugmanager() {
24:     }
25:
26:     /**
27:      * Eine Instanzmethode, durch die dem Fahrzeugmanager ein Fahrzeug mit einem
28:      * bestimmten Namen und Standort hinzugefügt wird. Falls der Fahrzeugmanager
29:      * bereits ein Fahrzeug mit diesem Namen verwaltet, soll kein Fahrzeug
30:      * hinzugefügt werden.
31:      *
32:      * @param fahrzeugname Name des Fahrzeugs
33:      * @param standort Standort des Fahrzeugs
34:      */
35:     public void fuegeFahrzeugHinzu(String fahrzeugname, String standort) {
36:         /*
37:          * Wenn Fahrzeug noch nicht in der Liste vorhandener Fahrzeuge ist,
38:          * erstelle es
39:          */
40:         if (!fahrzeuge.contains(new Fahrzeug(fahrzeugname, standort))) {
41:             /*
42:              * Neues Fahrzeug mit einem Namen und Standort
43:              */
44:             Fahrzeug fahrzeug = new Fahrzeug(fahrzeugname, standort);
```

```
45:             /*
46:              * Hinzufügen des neuen Fahrzeugs zur Liste fahrzeuge
47:              */
48:             fahrzeuge.add(fahrzeug);
49:
50:             bestimmeStandort(standort, fahrzeug);
51:         }
52:     }
53:
54:     /**
55:      * Eine Instanzmethode, durch die ein neuer Standort und zugehörige
56:      * Fahrzeuge bestimmt wird.
57:      *
58:      * @param standort Name des neuen Standorts
59:      * @param fahrzeug Zugehöriges Fahrzeug
60:      */
61:     public void bestimmeStandort(String standort, Fahrzeug fahrzeug) {
62:         /*
63:          * Iteration über alle Standorte
64:          */
65:             if (standorte.contains(new Standort(standort))) {
66:                 standorte.get(standorte.indexOf(new
Standort(standort))).fuegeFahrzeugHinzu(fahrzeug);
67:             } else {
68:                 /*
69:                  * Erstellt einen neuen Standort und fügt diesen zur Liste aller
70:                  * Standorte hinzu
71:                  */
72:                 Standort neuerStandort = new Standort(standort);
73:                 standorte.add(neuerStandort);
74:                 /*
75:                  * Fügt dem Standort das neue Fahrzeug hinzu
76:                  */
77:                 neuerStandort.fuegeFahrzeugHinzu(fahrzeug);
78:             }
79:       //  }
80:     }
81:
82:     /**
83:      * Eine Instanzmethode, die die Namen aller Fahrzeuge alphabetisch sortiert
84:      * zurückgibt. Die Methode liefert also eine Liste mit Zeichenketten.
85:      *
86:      * @return Alphabetisch sortierte Liste mit den Namen aller Fahrzeuge
87:      */
88:     public ArrayList gibFahrzeugnamen() {
89:         /*
90:          * Neue Liste für alle Fahrzeugnamen
91:          */
92:         ArrayList fahrzeugnamen = new ArrayList();
```

```
 93:         /*
 94:          * Kopiert jeden Fahrzeugnamen aus fahrzeuge in die neue Liste
 95:          */
 96:         for (int i = 0; i < fahrzeuge.size(); i++) {
 97:             fahrzeugnamen.add(fahrzeuge.get(i).gibName());
 98:         }
 99:         /*
100:          * Sortiert die Fahrzeugnamensliste
101:          */
102:         Collections.sort(fahrzeugnamen);
103:         /*
104:          * Rückgabe des Fahrzeugnamens
105:          */
106:         return fahrzeugnamen;
107:     }
108:
109:     /**
110:      * Eine Instanzmethode, mit der das Fahrzeug mit dem angegebenen Namen für
111:      * einen bestimmten Zeitraum gebucht wird. Die Angabe der Zeitpunkte erfolgt
112:      * im Format JJJJ/MM/TT HH:MM.
113:      *
114:      * @param fahrzeugname Name des Fahrzeugs
115:      * @param startzeit Startdatum und Startzeitpunkt
116:      * @param endzeit Enddatum und Endzeitpunkt
117:      * @return true, wenn das Fahrzeug in dem gewünschten Zeitrahmen verfügbar
118:      * ist
119:      */
120:     public boolean bucheFahrzeug(String fahrzeugname, String startzeit, String
endzeit) {
121:         /*
122:          * Verfügbarkeit des Fahrzeugs
123:          */
124:         boolean istVerfuegbar = false;
125:         /*
126:          * Für alle Fahrzeuge prüfe...
127:          */
128:         for (int i = 0; i < fahrzeuge.size(); i++) {
129:             /*
130:              * ...ob Fahrzeug mit dem zu buchenden Fahrzeug übereinstimmt und
131:              * schon gebucht ist
132:              */
133:             if (fahrzeuge.get(i).gibName().equals(fahrzeugname)
134:                     && fahrzeuge.get(i).istFrei(startzeit, endzeit)) {
135:                 /*
136:                  * Fahrzeug ist verfügbar
137:                  */
138:                 istVerfuegbar = true;
139:                 /*
140:                  * Setze neue Buchung
```

```
141:                  */
142:                 fahrzeuge.get(i).setzeBuchung(startzeit, endzeit);
143:             }
144:         }
145:         /*
146:          * Rückgabe, ob Fahrzeug verfügbar
147:          */
148:         return istVerfuegbar;
149:     }
150:
151:     /**
152:      * Eine Instanzmethode, die die Namen aller Fahrzeuge des angegebenen
153:      * Standorts alphabetisch sortiert zurückgibt, die in einem bestimmten
154:      * Zeitraum verfügbar sind. Ein Fahrzeug ist in einem Zeitraum verfügbar,
155:      * wenn es für diesen Zeitraum gebucht werden kann. Die Methode liefert also
156:      * eine Liste von Zeichenketten.
157:      *
158:      * @param standort Standort der Fahrzeuge
159:      * @param startzeit Startdatum und Startzeitpunkt
160:      * @param endzeit Enddatum und Endzeitpunkt
161:      * @return true, wenn das Fahrzeug in dem gewünschten Zeitrahmen verfügbar
162:      * ist
163:      */
164:     public ArrayList gibVerfuegbareFahrzeuge(String standort, String startzeit,
String endzeit) {
165:
166:         /*
167:          * Eine neue Liste für alle verfügbaren Fahrzeuge
168:          */
169:         ArrayList verfuegbareFahrzeuge = new ArrayList();
170:
171:         /*
172:          * Iteration über alle Standorte
173:          */
174:         for (int i = 0; i < standorte.size(); i++) {
175:             /*
176:              * Ist Standort der gesuchte Standort?
177:              */
178:            if (standorte.get(i).gibName().equals(standort)) {
179:               // if (standorte.contains(new Standort(standort))) {
180:
181:                 /*
182:                  * Eine neue Liste mit allen Fahrzeugen des Standortes
183:                  */
184:                 ArrayList fahrzeugeDesStandorts =
standorte.get(i).getFahrzeugListe();
185:                 /*
186:                  * Iteration über alle Fahrzeuge des Standorts
187:                  */
```

```
188:                 for (int j = 0; j < fahrzeugeDesStandorts.size(); j++) {
189:                     /*
190:                      * Wenn Fahrzeug ein frei-verfügbares Fahrzeug ist, dann...
191:                      */
192:                     if (((Fahrzeug)
fahrzeugeDesStandorts.get(j)).istFrei(startzeit, endzeit)) {
193:                         /*
194:                          * füge Fahrzeug zu der Liste verfügbarer Fahrzeuge
195:                          * hinzu
196:                          */
197:                         verfuegbareFahrzeuge.add(fahrzeugeDesStandorts.get(j));
198:
199:                     }
200:
201:                 }
202:
203:             }
204:
205:         }
206:         /*
207:          * Gibt Liste aller verfügbaren Fahrzeuge zurück
208:          */
209:         return verfuegbareFahrzeuge;
210:     }
211: }
```

5.2. FahrzeugmanagerTest.java

```
  1: /*
  2:  * To change this template, choose Tools | Templates
  3:  * and open the template in the editor.
  4:  */
  5: package carsharing;
  6:
  7: import java.util.ArrayList;
  8: import org.junit.Assert;
  9: import org.junit.Before;
 10: import org.junit.Test;
 11:
 12: /**
 13:  *
 14:  * @author apex
 15:  */
 16: public class FahrzeugmanagerTest {
 17:
 18:     public Fahrzeugmanager fahrzeugmanager = new Fahrzeugmanager();
 19:
 20:     public FahrzeugmanagerTest() {
 21:         fahrzeugmanager.fuegeFahrzeugHinzu("Rathaus 1", "Rathaus");
 22:         fahrzeugmanager.fuegeFahrzeugHinzu("Bahnhof 1", "Bahnhof");
```

```
 23:         fahrzeugmanager.fuegeFahrzeugHinzu("Bahnhof 2", "Bahnhof");
 24:         fahrzeugmanager.fuegeFahrzeugHinzu("Bahnhof 3", "Bahnhof");
 25:
 26:         fahrzeugmanager.bucheFahrzeug("Bahnhof 1", "2005/04/14 20:00", "2005/04/15
08:00");
 27:         fahrzeugmanager.bucheFahrzeug("Bahnhof 1", "2005/04/15 18:00", "2005/04/16
00:00");
 28:         fahrzeugmanager.bucheFahrzeug("Bahnhof 2", "2005/04/14 11:00", "2005/04/15
12:00");
 29:         fahrzeugmanager.bucheFahrzeug("Bahnhof 3", "2005/04/15 10:00", "2005/04/15
19:00");
 30:     }
 31:
 32:     @Before
 33:     public void setUp() {
 34:
 35:     }
 36:
 37:     /**
 38:      * Test of gibFahrzeugnamen method, of class Fahrzeugmanager.
 39:      */
 40:     @Test
 41:     public void testGibFahrzeugnamen() {
 42:
 43:         ArrayList sollErgebnis = new ArrayList();
 44:         sollErgebnis.add(new Fahrzeug("Bahnhof 1", "Bahnhof").gibName());
 45:         sollErgebnis.add(new Fahrzeug("Bahnhof 2", "Bahnhof").gibName());
 46:         sollErgebnis.add(new Fahrzeug("Bahnhof 3", "Bahnhof").gibName());
 47:         sollErgebnis.add(new Fahrzeug("Rathaus 1", "Rathaus").gibName());
 48:
 49:         Assert.assertEquals(sollErgebnis, fahrzeugmanager.gibFahrzeugnamen());
 50:     }
 51:
 52:     /**
 53:      * Test of gibVerfuegbareFahrzeuge method, of class Fahrzeugmanager.
 54:      */
 55:     @Test
 56:     public void testGibVerfuegbareFahrzeuge() {
 57:         ArrayList sollErgebnis1 = new ArrayList();
 58:
 59:         ArrayList sollErgebnis2 = new ArrayList();
 60:         sollErgebnis2.add(new Fahrzeug("Bahnhof 1", "Bahnhof"));
 61:         sollErgebnis2.add(new Fahrzeug("Bahnhof 2", "Bahnhof"));
 62:
 63:         ArrayList sollErgebnis3 = new ArrayList();
 64:         sollErgebnis2.add(new Fahrzeug("Bahnhof 2", "Bahnhof"));
 65:         sollErgebnis2.add(new Fahrzeug("Bahnhof 3", "Bahnhof"));
 66:
 67:         Assert.assertEquals(sollErgebnis1, fahrzeugmanager.gibVerfuegbareFahrzeuge
```

```
 68:                 ("Bahnhof", "2005/04/15 11:30", "2005/04/15 19:00"));
 69:         Assert.assertEquals(sollErgebnis2, fahrzeugmanager.gibVerfuegbareFahrzeuge
 70:                 ("Bahnhof", "2005/04/15 12:00", "2005/04/15 18:00"));
 71:         Assert.assertEquals(sollErgebnis3, fahrzeugmanager.gibVerfuegbareFahrzeuge
 72:                 ("Bahnhof", "2005/04/15 19:15", "2005/04/15 23:00"));
 73:     }
 74:
 75:     /**
 76:      * Test of bucheFahrzeug method, of class Fahrzeugmanager.
 77:      */
 78:     @Test
 79:     public void testBucheFahrzeug() {
 80:          Assert.assertEquals(true, fahrzeugmanager.bucheFahrzeug
 81:                 ("Bahnhof 3", "2005/04/15 09:00", "2005/04/15 10:00"));
 82:         Assert.assertEquals(false, fahrzeugmanager.bucheFahrzeug
 83:                 ("Bahnhof 3", "2005/04/15 19:00", "2005/04/15 11:00"));
 84:         Assert.assertEquals(false, fahrzeugmanager.bucheFahrzeug
 85:                 ("Bahnhof 3", "2005/04/15 11:00", "2005/04/15 18:00"));
 86:          Assert.assertEquals(false, fahrzeugmanager.bucheFahrzeug
 87:                 ("Bahnhof 3", "2005/04/15 18:00", "2005/04/15 20:00"));
 88:         Assert.assertEquals(true, fahrzeugmanager.bucheFahrzeug
 89:                 ("Bahnhof 3", "2005/04/15 19:00", "2005/04/15 20:00"));
 90:         Assert.assertEquals(false, fahrzeugmanager.bucheFahrzeug
 91:                 ("Bahnhof 3", "2005/04/15 09:00", "2005/04/15 20:00"));
 92:     }
 93:
 94: //    /**
 95: //     * Test of fuegeFahrzeugHinzu method, of class Fahrzeugmanager.
 96: //     */
 97: //    @Test
 98: //    public void testFuegeFahrzeugHinzu() {
 99: //        System.out.println("fuegeFahrzeugHinzu");
100: //        String fahrzeugname = "";
101: //        String standort = "";
102: //        Fahrzeugmanager instance = new Fahrzeugmanager();
103: //        instance.fuegeFahrzeugHinzu(fahrzeugname, standort);
104: //        // TODO review the generated test code and remove the default call to
fail.
105: //        fail("The test case is a prototype.");
106: //    }
107: //
108: //    /**
109: //     * Test of bestimmeStandort method, of class Fahrzeugmanager.
110: //     */
111: //    @Test
112: //    public void testBestimmeStandort() {
113: //        System.out.println("bestimmeStandort");
114: //        String standort = "";
115: //        Fahrzeug fahrzeug = null;
```

```
116: //        Fahrzeugmanager instance = new Fahrzeugmanager();
117: //        instance.bestimmeStandort(standort, fahrzeug);
118: //        // TODO review the generated test code and remove the default call to
fail.
119: //        fail("The test case is a prototype.");
120: //    }
121: }
```

5.3. CarsharingSuite.java

```
  1: /*
  2:  * To change this template, choose Tools | Templates
  3:  * and open the template in the editor.
  4:  */
  5: package carsharing;
  6:
  7: import org.junit.After;
  8: import org.junit.AfterClass;
  9: import org.junit.Before;
 10: import org.junit.BeforeClass;
 11: import org.junit.runner.RunWith;
 12: import org.junit.runners.Suite;
 13:
 14: /**
 15:  *
 16:  * @author apex
 17:  */
 18: @RunWith(Suite.class)
 19: @Suite.SuiteClasses({carsharing.FahrzeugTest.class,
carsharing.FahrzeugmanagerTest.class, carsharing.StandortTest.class,
carsharing.ZeitraumTest.class})
 20: public class CarsharingSuite {
 21:
 22:     @BeforeClass
 23:     public static void setUpClass() throws Exception {
 24:     }
 25:
 26:     @AfterClass
 27:     public static void tearDownClass() throws Exception {
 28:     }
 29:
 30:     @Before
 31:     public void setUp() throws Exception {
 32:     }
 33:
 34:     @After
 35:     public void tearDown() throws Exception {
 36:     }
 37:
 38: }
```

5.4. Fahrzeug.java

```
  1: //TODO class comment
  2: /*
  3:  * To change this template, choose Tools | Templates
  4:  * and open the template in the editor.
  5:  */
  6: // Fahrzeug kann nicht zweimal gebucht werden
  7: package carsharing;
  8:
  9: /**
 10:  *
 11:  * @author apex
 12:  */
 13: public class Fahrzeug {
 14:
 15:     /*
 16:      * Name, Standort, Startzeit und Endzeit der Buchung des Fahrzeugs
 17:      */
 18:     public String name;
 19:     public String standort;
 20:     public Zeitraum startzeit;
 21:     public Zeitraum endzeit;
 22:
 23:     /**
 24:      * Erzeugt ein Objekt dieser Klasse
 25:      *
 26:      * @param name Name des Fahrzeugs
 27:      * @param standort Standort des Fahrzeugs
 28:      */
 29:     public Fahrzeug(String name, String standort) {
 30:         this.name = name;
 31:         this.standort = standort;
 32:     }
 33:
 34:     @Override
 35:     public boolean equals(Object obj) {
 36:         if (obj == null) {
 37:             return false;
 38:         }
 39:         if (getClass() != obj.getClass()) {
 40:             return false;
 41:         }
 42:         final Fahrzeug other = (Fahrzeug) obj;
 43:         if ((this.name == null) ? (other.name != null) :
!this.name.equals(other.name)) {
 44:             return false;
 45:         }
 46:         if ((this.standort == null) ? (other.standort != null) :
!this.standort.equals(other.standort)) {
```

```
 47:             return false;
 48:         }
 49:         if (this.startzeit != other.startzeit && (this.startzeit == null ||
!this.startzeit.equals(other.startzeit))) {
 50:             return false;
 51:         }
 52:         if (this.endzeit != other.endzeit && (this.endzeit == null ||
!this.endzeit.equals(other.endzeit))) {
 53:             return false;
 54:         }
 55:         return true;
 56:     }
 57:
 58:     @Override
 59:     public String toString() {
 60:         return "Fahrzeug{" + "name=" + name + ", standort=" + standort + ",
startzeit=" + startzeit + ", endzeit=" + endzeit + '}';
 61:     }
 62:
 63:     /**
 64:      * Gibt den Namen des Fahrzeugs zurück
 65:      *
 66:      * @return Fahrzeugname
 67:      */
 68:     public String gibName() {
 69:         return name;
 70:     }
 71:
 72:     /**
 73:      * Gibt zugeordneten Standort zurück
 74:      */
 75:     public String gibStandort() {
 76:         return standort;
 77:     }
 78:
 79:     /**
 80:      * Prüft, ob Fahrzeug frei zur Buchung ist
 81:      *
 82:      * @param startzeit Startdatum und Startzeitpunkt der Buchung
 83:      * @param endzeit Enddatum und Endzeitpunkt der Buchung
 84:      * @return true, wenn Fahrzeug gebucht werden kann
 85:      */
 86:     public boolean istFrei(String startzeit, String endzeit) {
 87:         /*
 88:          * Verfügbarkeit des Fahrzeugs
 89:          */
 90:         // 2012/11/24 23:12
 91:
 92:         boolean istFrei = false;
```

```
 93:
 94:         Zeitraum testeStartzeit = new Zeitraum(startzeit);
 95:         Zeitraum testeEndzeit = new Zeitraum(endzeit);
 96:
 97:         /*
 98:          * Das Fahrzeug besitzt bereits einen Buchungszeitraum
 99:          */
100:         if (this.startzeit != null && this.endzeit != null) {
101:             /*
102:              * Wenn zu testende Start- und Endzeiträume vor oder nach dem
103:              * vorhandenen Buchungszeitraum liegen und Startzeitraum kleiner als
104:              * Endzeitraum ist, dann...
105:              */
106:             if ((this.startzeit.pruefeRelation(testeStartzeit.getZeitraum())
107:                     == this.endzeit.pruefeRelation(testeStartzeit.getZeitraum()))
108:                     ||
109:                     (this.startzeit.pruefeRelation(testeEndzeit.getZeitraum())
110:                     == this.endzeit.pruefeRelation(testeEndzeit.getZeitraum()))
111:                     && testeStartzeit.getZeitraum() < testeEndzeit.getZeitraum())
{
112:                 /*
113:                  * ...setze istFrei auf true
114:                  */
115:                 istFrei = true;
116:             }
117:         } else if (this.startzeit == null && this.endzeit == null)  {
118:
119:             /*
120:              * Wenn noch kein Buchungszeitraum gesetzt wurde, dann...
121:              * ...setze istFrei auf true
122:              */
123:
124:             istFrei = true;
125:         }
126:
127:         return istFrei;
128:     }
129:
130:     /**
131:      * Führt eine neue Buchung auf das Fahrzeug aus
132:      *
133:      * @param startzeit Startdatum und Startzeitpunkt der Buchung
134:      * @param endzeit Enddatum und Endzeitpunkt der Buchung
135:      */
136:     public void setzeBuchung(String startzeit, String endzeit) {
137:         this.startzeit = new Zeitraum(startzeit);
138:         this.endzeit = new Zeitraum(endzeit);
139:     }
140: }
```

5.5. FahrzeugTest.java

```
  1: /*
  2:  * To change this template, choose Tools | Templates
  3:  * and open the template in the editor.
  4:  */
  5: package carsharing;
  6:
  7: import org.junit.Assert;
  8: import org.junit.Before;
  9: import org.junit.Test;
 10:
 11: /**
 12:  *
 13:  * @author apex
 14:  */
 15: public class FahrzeugTest {
 16:
 17:     Fahrzeug auto;
 18:
 19:     public FahrzeugTest() {
 20:     }
 21:
 22:     @Before
 23:     public void setUp() {
 24:         auto = new Fahrzeug("BMW", "Essen");
 25:     }
 26:
 27:     /**
 28:      * Test of toString method, of class Fahrzeug.
 29:      */
 30:     @Test
 31:     public void testToString() {
 32:         System.out.println("toString");
 33:         String descr = "Fahrzeug{name=BMW, standort=Essen, startzeit=null,
endzeit=null}";
 34:         Assert.assertEquals(descr, auto.toString());
 35:     }
 36:
 37:     /**
 38:      * Test of gibName method, of class Fahrzeug.
 39:      */
 40:     @Test
 41:     public void testGibName() {
 42:         System.out.println("gibName");
 43:         Assert.assertEquals("BMW", auto.gibName());
 44:     }
 45:
 46:     /**
```

```
 47:      * Test of gibStandort method, of class Fahrzeug.
 48:      */
 49:     @Test
 50:     public void testGibStandort() {
 51:         System.out.println("gibStandort");
 52:         Assert.assertEquals("Essen", auto.gibStandort());
 53:     }
 54:
 55:     /**
 56:      * Test of istFrei method, of class Fahrzeug.
 57:      */
 58:     @Test
 59:     public void testIstFrei() {
 60:         System.out.println("istFrei");
 61:         Assert.assertEquals(true, auto.istFrei("2005/04/15 10:00", "2005/04/16
08:00"));
 62:         auto.startzeit = new Zeitraum("2005/04/15 08:00");
 63:         auto.endzeit = new Zeitraum("2005/04/16 10:00");
 64:         Assert.assertEquals(false, auto.istFrei("2005/04/15 10:00", "2005/04/16
08:00"));
 65:     }
 66:
 67:     /**
 68:      * Test of setzeBuchung method, of class Fahrzeug.
 69:      */
 70:     @Test
 71:     public void testSetzeBuchung() {
 72:         System.out.println("setzeBuchung");
 73:         auto.setzeBuchung("2005/04/15 09:30", "2005/04/16 11:10");
 74:         Assert.assertEquals(200504150930L, auto.startzeit.getZeitraum());
 75:         Assert.assertEquals(200504161110L, auto.endzeit.getZeitraum());
 76:     }
 77:
 78: //    /**
 79: //     * Test of equals method, of class Fahrzeug.
 80: //     */
 81: //    @Test
 82: //    public void testEquals() {
 83: //        System.out.println("equals");
 84: //        Object obj = null;
 85: //        Fahrzeug instance = null;
 86: //        boolean expResult = false;
 87: //        boolean result = instance.equals(obj);
 88: //        assertEquals(expResult, result);
 89: //        // TODO review the generated test code and remove the default call to
fail.
 90: //        fail("The test case is a prototype.");
 91: //    }
 92:
```

```
93: //    @After
94: //    public void tearDown() {
95: //    }
96: }
```

5.6. Standort.java

```
 1: //TODO class comment
 2: /*
 3:  * To change this template, choose Tools | Templates
 4:  * and open the template in the editor.
 5:  */
 6: package carsharing;
 7:
 8: import java.util.ArrayList;
 9:
10: /**
11:  *
12:  * @author apex
13:  */
14: public class Standort {
15:
16:     /*
17:      * Name des Standorts
18:      */
19:     public String name;
20:     /*
21:      * Liste mit allen zugeordneten Fahzeugen
22:      */
23:     public ArrayList<Fahrzeug> fahrzeugListe = new ArrayList();
24:
25:     /**
26:      * Erzeugt ein Objekt dieser Klasse
27:      *
28:      * @param name Name des Standorts
29:      */
30:     public Standort(String name) {
31:         /*
32:          * Setzt den Namen des Standortes
33:          */
34:         this.name = name;
35:     }
36:
37:     @Override
38:     public boolean equals(Object obj) {
39:         if (obj == null) {
40:             return false;
41:         }
42:         if (getClass() != obj.getClass()) {
43:             return false;
```

```
 44:         }
 45:         final Standort other = (Standort) obj;
 46:         if ((this.name == null) ? (other.name != null) :
!this.name.equals(other.name)) {
 47:             return false;
 48:         }
 49:         return true;
 50:     }
 51:
 52:     @Override
 53:     public String toString() {
 54:         return "Standort{" + "name=" + name + ", fahrzeugListe=" + fahrzeugListe +
'}';
 55:     }
 56:
 57:     public String getIdent() {
 58:         return getClass().getName() + '@' + Integer.toHexString(hashCode());
 59:     }
 60:
 61:     /**
 62:      * Liefert den Namen des Standortes
 63:      *
 64:      * @return Name des Standortes
 65:      */
 66:     public String gibName() {
 67:         return this.name;
 68:     }
 69:
 70:     /**
 71:      * Fügt dem Standort ein neues Fahrzeug hinzu
 72:      *
 73:      * @param fahrzeug
 74:      */
 75:     public void fuegeFahrzeugHinzu(Fahrzeug fahrzeug) {
 76:         /*
 77:          * Wenn Fahrzeug noch nicht in Fahrzeugliste des Standorts, füge es
 78:          * hinzu
 79:          */
 80:         if (!fahrzeugListe.contains(fahrzeug)) {
 81:             this.fahrzeugListe.add(fahrzeug);
 82:         }
 83:     }
 84:
 85:     /**
 86:      * Gibt alle Fahrzeuge des Standorts zurück
 87:      *
 88:      * @return alle Fahrzeuge des Standorts
 89:      */
 90:     public ArrayList<Fahrzeug> getFahrzeugListe() {
```

```
91:         return fahrzeugListe;
92:     }
93:
94: }
```

5.7. StandortTest.java

```
 1: /*
 2:  * To change this template, choose Tools | Templates
 3:  * and open the template in the editor.
 4:  */
 5: package carsharing;
 6:
 7: import java.util.ArrayList;
 8: import junit.framework.Assert;
 9: import static org.junit.Assert.assertEquals;
10: import static org.junit.Assert.fail;
11: import org.junit.Before;
12: import org.junit.BeforeClass;
13: import org.junit.Test;
14:
15: /**
16:  *
17:  * @author apex
18:  */
19: public class StandortTest {
20:
21:     private static Standort duisburg;
22:     private static Fahrzeug auto;
23:
24:     public StandortTest() {
25:
26:     }
27:
28:     @BeforeClass
29:     public static void setUpClass() throws Exception {
30:         duisburg = new Standort("Duisburg");
31:         auto = new Fahrzeug("BMW", "Duisburg");
32:     }
33:
34: //    @Before
35: //    public void classSetUp() {
36: //    }
37:
38:     /**
39:      * Test of toString method, of class Standort.
40:      */
41:     @Test
42:     public void testToString() {
43:         System.out.println("toString");
```

```
 44:         Assert.assertEquals("Standort{name=Duisburg, fahrzeugListe=[]}",
duisburg.toString());
 45:     }
 46:
 47:     /**
 48:      * Test of gibName method, of class Standort.
 49:      */
 50:     @Test
 51:     public void testGibName() {
 52:         System.out.println("gibName");
 53:         Assert.assertEquals("Duisburg", duisburg.gibName());
 54:     }
 55:
 56:     /**
 57:      * Test of fuegeFahrzeugHinzu method, of class Standort.
 58:      */
 59:     @Test
 60:     public void testFuegeFahrzeugHinzu() {
 61:         System.out.println("fuegeFahrzeugHinzu");
 62:         duisburg.fuegeFahrzeugHinzu(auto);
 63:         Assert.assertEquals(auto, duisburg.fahrzeugListe.get(0));
 64:         System.out.println(duisburg.getIdent());
 65:     }
 66:
 67:     /**
 68:      * Test of getFahrzeugListe method, of class Standort.
 69:      */
 70:     @Test
 71:     public void testGetFahrzeugListe() {
 72: //        duisburg.fuegeFahrzeugHinzu(auto);
 73:         System.out.println(duisburg.getIdent());
 74:         System.out.println("getFahrzeugListe");
 75:         ArrayList<Fahrzeug> fahrzeuge = new ArrayList<Fahrzeug>();
 76:         fahrzeuge.add(auto);
 77:         Assert.assertEquals(fahrzeuge, duisburg.getFahrzeugListe());
 78:     }
 79: //    /**
 80: //     * Test of equals method, of class Standort.
 81: //     */
 82: //    @Test
 83: //    public void testEquals() {
 84: //        System.out.println("equals");
 85: //        Object obj = null;
 86: //        Standort instance = null;
 87: //        boolean expResult = false;
 88: //        boolean result = instance.equals(obj);
 89: //        assertEquals(expResult, result);
 90: //        // TODO review the generated test code and remove the default call to
fail.
```

```
91: //        fail("The test case is a prototype.");
92: //    }
93: //    @After
94: //    public void tearDown() {
95: //    }
96:
97: }
```

5.8. Zeitraum.java

```
 1: /*
 2:  * To change this template, choose Tools | Templates
 3:  * and open the template in the editor.
 4:  */
 5: package carsharing;
 6:
 7: /**
 8:  *
 9:  * @author apex
10:  */
11: public class Zeitraum {
12:
13:     /*
14:      * Bestandteile des übergebenen Textausdruckes für einen Zeitraum Jahr, Tag,
15:      * Monat, Stunde, Minute, Vollständiges Datum, Vollständige Zeit,
16:      * Vollständiger Zeitraum
17:      */
18:     public long jahr;
19:     public long tag;
20:     public long monat;
21:     public long stunde;
22:     public long minute;
23:     public long datum;
24:     public long uhrzeit;
25:     public long zeitraum;
26:
27:     /**
28:      * Erzeugt ein Objekt dieser Klasse Hierbei wird ein Textausdruck für einen
29:      * Zeitraum in dessen Bestandteile zerlegt und den Instanzvariablen
30:      * zugeordnet
31:      *
32:      * @param zeitraum Textausdruck im Format JJJJ/MM/TT SS:MM, z.B. 2012/05/21
33:      * 23:12
34:      * 201205212321
35:      * 2012/05/23 14:05
36:      * 2009/04/12 09:15
37:      */
38:     public Zeitraum(String zeitraum) {
39:         this.jahr = new Long(zeitraum.substring(0, 4));
40:         this.monat = new Long(zeitraum.substring(5, 7));
```

```
41:         this.tag = new Long(zeitraum.substring(8, 10));
42:         this.stunde = new Long(zeitraum.substring(11, 13));
43:         this.minute = new Long(zeitraum.substring(14, 16));
44:         this.datum =
45:                 new Long((zeitraum.substring(0, 4)
46:                 + zeitraum.substring(5, 7)
47:                 + zeitraum.substring(8, 10)));
48:         this.uhrzeit =
49:                 new Long((zeitraum.substring(11, 13)
50:                 + zeitraum.substring(14, 16)));
51:         this.zeitraum =
52:                 new Long((zeitraum.substring(0, 4)
53:                 + zeitraum.substring(5, 7)
54:                 + zeitraum.substring(8, 10)
55:                 + zeitraum.substring(11, 13)
56:                 + zeitraum.substring(14, 16)));
57:     }
58:
59:     /**
60:      * Liefert den vollständigen Zeitraum als Wert zurück
61:      *
62:      * @return vollständiger Zeitraum als Wert
63:      */
64:     public long getZeitraum() {
65:         return zeitraum;
66:     }
67:
68:     @Override
69:     public boolean equals(Object obj) {
70:         if (obj == null) {
71:             return false;
72:         }
73:         if (getClass() != obj.getClass()) {
74:             return false;
75:         }
76:         final Zeitraum other = (Zeitraum) obj;
77:         if (this.jahr != other.jahr) {
78:             return false;
79:         }
80:         if (this.tag != other.tag) {
81:             return false;
82:         }
83:         if (this.monat != other.monat) {
84:             return false;
85:         }
86:         if (this.stunde != other.stunde) {
87:             return false;
88:         }
89:         if (this.minute != other.minute) {
```

```
 90:             return false;
 91:         }
 92:         if (this.datum != other.datum) {
 93:             return false;
 94:         }
 95:         if (this.uhrzeit != other.uhrzeit) {
 96:             return false;
 97:         }
 98:         if (this.zeitraum != other.zeitraum) {
 99:             return false;
100:         }
101:         return true;
102:     }
103:
104:     /**
105:      * Liefert das vollständige Datum als Wert zurück
106:      *
107:      * @return vollständige Datum als Wert
108:      */
109:     public long getDatum() {
110:         return datum;
111:     }
112:
113:     /**
114:      * Liefert die vollständige Uhrzeit als Wert zurück
115:      *
116:      * @return vollständige Uhrzeit als Wert
117:      */
118:     public long getUhrzeit() {
119:         return uhrzeit;
120:     }
121:
122:     /**
123:      * Liefert das Jahr als Wert zurück
124:      *
125:      * @return Jahr als Wert
126:      */
127:     public long getJahr() {
128:         return jahr;
129:     }
130:
131:     /**
132:      * Liefert die Minuten als Wert zurück
133:      *
134:      * @return Minuten als Wert
135:      */
136:     public long getMinute() {
137:         return minute;
138:     }
```

```
139:
140:     /**
141:      * Liefert den Monat als Wert zurück
142:      *
143:      * @return Monat als Wert
144:      */
145:     public long getMonat() {
146:         return monat;
147:     }
148:
149:     /**
150:      * Liefert die Stunden als Wert zurück
151:      *
152:      * @return Stunden als Wert
153:      */
154:     public long getStunde() {
155:         return stunde;
156:     }
157:
158:     /**
159:      * Liefert den Tag als Wert zurück
160:      *
161:      * @return Tag als Wert
162:      */
163:     public long getTag() {
164:         return tag;
165:     }
166:
167:     /**
168:      * Prüft, ob der übergebene Zeitraum vor oder nach einem schon bestehenden
169:      * Zeitraum liegt
170:      *
171:      * @param testeZeitraum zu testender Zeitraum
172:      * @return Position des Zeitraums
173:      */
174:     public long pruefeRelation(long testeZeitraum) {
175:
176:         /*
177:          * -1: fehlerhafte Auswertung
178:          */
179:         long status = -1;
180:
181:         if (testeZeitraum < this.zeitraum) {
182:             /*
183:              * 0: Zeitraum liegt vor dem bestehenden Zeitraum
184:              */
185:             status = 0;
186:         }
187:
```

```
188:         if (testeZeitraum > this.zeitraum) {
189:             /*
190:              * 1: Zeitraum liegt nach dem bestehenden Zeitraum
191:              */
192:             status = 1;
193:         }
194: 
195:         return status;
196:     }
197: }
```

5.9. ZeitraumTest.java

```
  1: /*
  2:  * To change this template, choose Tools | Templates
  3:  * and open the template in the editor.
  4:  */
  5: package carsharing;
  6: 
  7: import org.junit.After;
  8: import org.junit.Assert;
  9: import org.junit.Before;
 10: import org.junit.Test;
 11: 
 12: /**
 13:  *
 14:  * @author apex
 15:  */
 16: public class ZeitraumTest {
 17: 
 18:     Zeitraum testZeitraum;
 19: 
 20:     public ZeitraumTest() {
 21:     }
 22: 
 23:     @Before
 24:     public void setUp() {
 25:         testZeitraum = new Zeitraum("2005/04/15 10:00");
 26:     }
 27: 
 28:     /**
 29:      * Test of getZeitraum method, of class Zeitraum.
 30:      */
 31:     @Test
 32:     public void testGetZeitraum() {
 33:         System.out.println("getZeitraum");
 34:         Assert.assertEquals(200504151000L, testZeitraum.getZeitraum());
 35: 
 36:     }
 37: 
```

```
38:     /**
39:      * Test of getDatum method, of class Zeitraum.
40:      */
41:     @Test
42:     public void testGetDatum() {
43:         System.out.println("getDatum");
44:         Assert.assertEquals(20050415L, testZeitraum.getDatum());
45:     }
46:
47:     /**
48:      * Test of getUhrzeit method, of class Zeitraum.
49:      */
50:     @Test
51:     public void testGetUhrzeit() {
52:         System.out.println("getUhrzeit");
53:         Assert.assertEquals(1000L, testZeitraum.getUhrzeit());
54:     }
55:
56:     /**
57:      * Test of getJahr method, of class Zeitraum.
58:      */
59:     @Test
60:     public void testGetJahr() {
61:         System.out.println("getJahr");
62:         Assert.assertEquals(2005L, testZeitraum.getJahr());
63:     }
64:
65:     /**
66:      * Test of getMinute method, of class Zeitraum.
67:      */
68:     @Test
69:     public void testGetMinute() {
70:         System.out.println("getMinute");
71:         Assert.assertEquals(0L, testZeitraum.getMinute());
72:     }
73:
74:     /**
75:      * Test of getMonat method, of class Zeitraum.
76:      */
77:     @Test
78:     public void testGetMonat() {
79:         System.out.println("getMonat");
80:         Assert.assertEquals(4L, testZeitraum.getMonat());
81:     }
82:
83:     /**
84:      * Test of getStunde method, of class Zeitraum.
85:      */
86:     @Test
```

```
 87:     public void testGetStunde() {
 88:         System.out.println("getStunde");
 89:         Assert.assertEquals(10L, testZeitraum.getStunde());
 90:     }
 91:
 92:     /**
 93:      * Test of getTag method, of class Zeitraum.
 94:      */
 95:     @Test
 96:     public void testGetTag() {
 97:         System.out.println("getTag");
 98:         Assert.assertEquals(15L, testZeitraum.getTag());
 99:     }
100:
101:     /**
102:      * Test of pruefeRelation method, of class Zeitraum.
103:      */
104:     @Test
105:     public void testPruefeRelation() {
106:         System.out.println("pruefeRelation");
107:         Assert.assertEquals(0, testZeitraum.pruefeRelation(200504140900L));
108:         Assert.assertEquals(1, testZeitraum.pruefeRelation(200504161000L));
109:     }
110:
111: //    /**
112: //     * Test of equals method, of class Zeitraum.
113: //     */
114: //    @Test
115: //    public void testEquals() {
116: //        System.out.println("equals");
117: //        Object obj = null;
118: //        Zeitraum instance = null;
119: //        boolean expResult = false;
120: //        boolean result = instance.equals(obj);
121: //        assertEquals(expResult, result);
122: //        // TODO review the generated test code and remove the default call to
fail.
123: //        fail("The test case is a prototype.");
124: //    }
125:
126: //    @After
127: //    public void tearDown() {
128: //    }
129: }
```