```
1  import static org.junit.Assert.*;
2  import org.junit.After;
3  import org.junit.Before;
4  import org.junit.Test;
5
6  /**
7   * test class ModelTest - geef hier een beschrijving van deze class
8   *
9   * @author Jorim Tielemans
10  * @version 08/01/2015
11  */
12 public class ModelTest
13 {
14     /**
15      * Constructor voor test class ModelTest
16      */
17     public ModelTest()
18     {
19     }
20
21     /**
22      * Opzetten van de test fixture.
23      *
24      * Aanroep voor elke test case method.
25      *
26      * Hier zou ik in principe de verschillende mogelijk gebruikte lev
   els kunnen aanroepen,
27      * ik heb echter besloten dit pas in de testcase zelf te doen om g
   een onnodige resources te gebruiken.
28      */
29     @Before
30     public void setUp()
31     {
32     }
33
34     /**
35      * Verwijderen van de test fixture.
36      *
37      * Aanroep na elke test case method.
38      */
39     @After
40     public void tearDown()
41     {
42     }
43
44     /**
45      * Met deze methode testen we of het Portaal 'gesloten' wordt als
   een speler deze betreedt;
46      * vervolgens weer geopend wordt als de speler eraf gaat;
47      * er enkel een speler op kan als ze vrij zijn;
48      * of een speler tegoei wordt geteleporteerd naar het andere porta
   al.
49      */
```

```
50      @Test
51      public void PortaalBetreden()
52      {
53          Model m = new Model(3); // als level 3 wordt ingeladen zijn er
 meteen portalen aanwezig
54          GroteView gv = new GroteView(m);
55          Animator a = new Animator(m,gv);
56          Thread t = new Thread(a);
57          t.start();
58          for(int aantalBewegingen = 0; aantalBewegingen < 10000; aantal
Bewegingen++)
59          {
60              m.beweegOnder(1);
61              m.beweegOnder(1);
62              m.beweegRechts(1); // speler1 staat voor portaal A en ze m
oeten nog betreedbaar zijn
63              assertTrue(m.portaal1A.isBetreedbaar());
64              assertTrue(m.portaal1B.isBetreedbaar());
65              m.beweegRechts(1); // speler1 is door portaal A gegaan en
staat nu op B, beide moeten dus gesloten zijn
66              assertFalse(m.portaal1A.isBetreedbaar());
67              assertFalse(m.portaal1B.isBetreedbaar());
68              m.beweegOnder(1); // speler1 is van portaal B afgestapt, b
eide moeten dus terug open zijn
69              assertTrue(m.portaal1A.isBetreedbaar());
70              assertTrue(m.portaal1B.isBetreedbaar());
71              m.beweegOnder(1);
72              m.beweegRechts(1);
73              m.beweegRechts(1);
74
75              m.beweegBoven(1);
76              m.beweegBoven(1);
77              m.beweegLinks(1); // ze moeten nog steeds open zijn want d
e speler staat nog voor portaal B
78              assertTrue(m.portaal1A.isBetreedbaar());
79              assertTrue(m.portaal1B.isBetreedbaar());
80              m.beweegLinks(1); // speler1 is terug door het portaal geg
aan, deze keer door B naar A
81              assertFalse(m.portaal1A.isBetreedbaar());
82              assertFalse(m.portaal1B.isBetreedbaar());
83
84              m.beweegBoven(2);
85              m.beweegBoven(2);
86              m.beweegLinks(2); // nu staat speler 2 voor portaal B, omd
at speler 1 er nog opstaat aan de andere kant zijn ze nog bezet
87              assertFalse(m.portaal1A.isBetreedbaar());
88              assertFalse(m.portaal1B.isBetreedbaar());
89              int xCo = m.speler2.getX(); // haal de xCo op van waar spe
ler 2 nu staat
90              int yCo = m.speler2.getY(); // haal de yCo op van waar spe
ler 2 nu staat
91              m.beweegLinks(2); // de speler kan niet verder naar links,
 het portaal is bezet, de coordinaten (zie hieronder) zijn dus niet ve
```

```
91  randert
92              assertEquals(xCo, m.speler2.getX());
93              assertEquals(yCo, m.speler2.getY());
94
95              m.beweegBoven(1); // de speler is er weer afgestapt en ze
    staan dus terug open
96              assertTrue(m.portaal1A.isBetreedbaar());
97              assertTrue(m.portaal1B.isBetreedbaar());
98              m.beweegBoven(1);
99              m.beweegLinks(1);
100             m.beweegLinks(1);
101
102             m.beweegLinks(2); // de speler kan nu wel verder naar link
    s, hij wordt verplaatst naar portaal B, de coordinaten (zie hieronder)
     van de speler zijn nu dus gelijk aan die van het portaal
103             assertEquals(m.portaal1A.getX(), m.speler2.getX());
104             assertEquals(m.portaal1A.getY(), m.speler2.getY());
105             assertFalse(m.portaal1A.isBetreedbaar());
106             assertFalse(m.portaal1B.isBetreedbaar());
107             m.beweegBoven(2);
108             assertTrue(m.portaal1A.isBetreedbaar());
109             assertTrue(m.portaal1B.isBetreedbaar());
110             m.beweegBoven(2);
111             m.beweegLinks(2);
112             m.beweegLinks(2);
113
114             m.beweegOnder(2);
115             m.beweegOnder(2);
116             m.beweegRechts(2);
117             assertTrue(m.portaal1A.isBetreedbaar());
118             assertTrue(m.portaal1B.isBetreedbaar());
119             m.beweegRechts(2);
120             assertFalse(m.portaal1A.isBetreedbaar());
121             assertFalse(m.portaal1B.isBetreedbaar());
122             m.beweegOnder(2);
123             assertTrue(m.portaal1A.isBetreedbaar());
124             assertTrue(m.portaal1B.isBetreedbaar());
125             m.beweegOnder(2);
126             m.beweegRechts(2);
127             m.beweegRechts(2);
128         }
129     }
130
131     /**
132      * Met deze methode laten we een random speler in random richtinge
    n bewegen
133      * vervolgens testen we steeds of deze speler zich nog wel in het
    veld bevind (wat in principe niet anders kan aangezien het volledige v
    eld is omringd door een muur).
134      */
135     @Test
136     public void Binnenblijven()
137     {
```

```
138         Model m = new Model(4);
139         GroteView gv = new GroteView(m);
140         Animator a = new Animator(m,gv);
141         Thread t = new Thread(a);
142         t.start();
143         for(int aantalBewegingen = 0; aantalBewegingen < 10000; aantal
    Bewegingen++)
144         {
145             willekeurigeActie(m); // laat iemand iets willekeurigs doe
    n
146             assertTrue((m.speler1.getX() > 0) && (m.speler1.getX() < 2
    1)); // zit speler 1 nog binnen het x-bereik?
147             assertTrue((m.speler1.getY() > 0) && (m.speler1.getY() < 1
    5)); // zit speler 1 nog binnen het y-bereik?
148             assertTrue((m.speler2.getX() > 0) && (m.speler2.getX() < 2
    1)); // zit speler 2 nog binnen het x-bereik?
149             assertTrue((m.speler2.getY() > 0) && (m.speler2.getY() < 1
    5)); // zit speler 2 nog binnen het y-bereik?
150         }
151     }
152 
153     /**
154      * Met deze methode laten we een random speler in random richtinge
    n bewegen
155      * vervolgens testen we steeds of de spelers en portalen nog aanwe
    zig zijn (al dan niet onder elkaar).
156      */
157     @Test
158     public void Zelfdeblijven()
159     {
160         Model m = new Model(3);
161         GroteView gv = new GroteView(m);
162         Animator a = new Animator(m,gv);
163         Thread t = new Thread(a);
164         t.start();
165         for(int aantalBewegingen = 0; aantalBewegingen < 10000; aantal
    Bewegingen++)
166         {
167             willekeurigeActie(m); // laat weer iemand iets willekeurig
    s doen
168             if(m.getVoorwerp(m.speler1.getX(), m.speler1.getY()) == m.
    speler1)
169             {
170                 assertEquals(m.getVoorwerp(m.speler1.getX(), m.speler1
    .getY()), m.speler1); // is speler 1 aanwezig op zijn plaats?
171             }
172             else
173             {
174                 assertEquals(m.speler2.getVwOnderSpeler(), m.speler1);
     // anders zou hij onder speler 2 moeten zitten.
175             }
176             if(m.getVoorwerp(m.speler2.getX(), m.speler2.getY()) == m.
    speler2) // analoog aan vorige
```

```
177             {
178                 assertEquals(m.getVoorwerp(m.speler2.getX(), m.speler2
    .getY()), m.speler2); // is speler 2 aanwezig op zijn plaats?
179             }
180             else
181             {
182                 assertEquals(m.speler1.getVwOnderSpeler(), m.speler2);
     // anders zou hij onder speler 1 moeten zitten.
183             }
184             if(m.getVoorwerp(m.portaal1A.getX(), m.portaal1A.getY()) =
    = m.portaal1A)
185             {
186                 assertEquals(m.getVoorwerp(m.portaal1A.getX(), m.porta
    al1A.getY()), m.portaal1A); // ligt portaal A nog juist?
187             }
188             else
189             {
190                 if(m.speler1.getVwOnderSpeler() == m.portaal1A)
191                 {
192                     assertEquals(m.speler1.getVwOnderSpeler(), m.porta
    al1A); // misschien staat speler 1 er op
193                 }
194                 else
195                 {
196                     assertEquals(m.speler2.getVwOnderSpeler(), m.porta
    al1A); // of speler 2
197                 }
198             }
199             if(m.getVoorwerp(m.portaal1B.getX(), m.portaal1B.getY()) =
    = m.portaal1B) // analoog aan vorige
200             {
201                 assertEquals(m.getVoorwerp(m.portaal1B.getX(), m.porta
    al1B.getY()), m.portaal1B); // ligt portaal B nog juist?
202             }
203             else
204             {
205                 if(m.speler1.getVwOnderSpeler() == m.portaal1B)
206                 {
207                     assertEquals(m.speler1.getVwOnderSpeler(), m.porta
    al1B); // misschien staat speler 1 er op
208                 }
209                 else
210                 {
211                     assertEquals(m.speler2.getVwOnderSpeler(), m.porta
    al1B); // of speler 2
212                 }
213             }
214         }
215     }
216
217     /**
218      * Deze functie test of er een bom wordt gelegd en of deze nadien
    verdwijnd (dus terug grond is geworden).
```

```
219      */
220     @Test
221     public void wordtDeBomGelegd()
222     {
223         Model m = new Model(4);
224         GroteView gv = new GroteView(m);
225         Animator a = new Animator(m,gv);
226         Thread t = new Thread(a);
227         t.start();
228         for(int aantalBommen = 0; aantalBommen < 3; aantalBommen++)
229         {
230             long runTime = 0;
231             m.beweegOnder(1);
232             m.beweegOnder(1);
233             m.beweegRechts(1);
234             m.beweegRechts(1);
235             m.legBom(1);
236             long startTime=System.currentTimeMillis();
237             assertEquals(m.getVoorwerp(m.speler1.getX(), m.speler1.get
    Y()), m.speler1);
238             int speler1X = m.speler1.getX();
239             int speler1Y = m.speler1.getY();
240             m.beweegOnder(1);
241             m.beweegOnder(1);
242             m.beweegRechts(1);
243             m.beweegRechts(1);
244             assertEquals(m.getVoorwerp(m.speler1.getX(), m.speler1.get
    Y()), m.speler1);
245             assertTrue(m.getVoorwerp(speler1X, speler1Y) instanceof Bo
    m);
246             while (runTime < 4000)
247             {
248                 runTime = System.currentTimeMillis() - startTime;
249             }
250             assertEquals(m.getVoorwerp(m.speler1.getX(), m.speler1.get
    Y()), m.speler1);
251             assertTrue(m.getVoorwerp(speler1X, speler1Y) instanceof Gr
    ond);
252         }
253     }
254
255     /**
256      * Deze functie test of ik een leven kwijtspeel als ik een bom nee
    rleg zonder ver genoeg weg te lopen (of het hoekje niet omga, letterli
    jk dan).
257      * Wordt de variabele of er een dode is ook correct uitgevoerd, di
    t zou het spel moeten stilleggen
258      * --> in latere versies (ja, ik heb nog verscheidene verbeteringe
    n&vernieuwingen aangebracht) wordt dit overal beter gecontroleerd en v
    erdere acties of geluiden voorkomen.
259      */
260     @Test
261     public void speelIkEenLevenKwijt()
```

```
262     {
263         Model m = new Model(4);
264         GroteView gv = new GroteView(m);
265         Animator a = new Animator(m,gv);
266         Thread t = new Thread(a);
267         t.start();
268         int levens = 3;
269         assertFalse(m.getIsErEenDode());
270         for(int aantalBommen = 0; aantalBommen < 3; aantalBommen++)
271         {
272             long runTime = 0;
273             m.beweegOnder(1);
274             m.beweegOnder(1);
275             m.beweegRechts(1);
276             m.beweegRechts(1);
277             assertEquals(levens, m.speler1.getLevens());
278             m.legBom(1);
279             levens--;
280             long startTime=System.currentTimeMillis();
281             int speler1X = m.speler1.getX();
282             int speler1Y = m.speler1.getY();
283             m.beweegOnder(1);
284             while (runTime < 4000)
285             {
286                 runTime = System.currentTimeMillis() - startTime;
287             }
288             assertEquals(levens, m.speler1.getLevens());
289             m.beweegOnder(1);
290             m.beweegRechts(1);
291             m.beweegRechts(1);
292         }
293         assertTrue(m.getIsErEenDode());
294     }
295 
296     /**
297      * Deze functie gaat testen of level 2 (met het kruitspoor) correc
    t wordt ingeladen.
298      *
299      * Bij het inladen kunnen bepaalde locaties gewoon overschreven, o
    m die reden kon niet gewoon dezelfde lussen worden gebruikt;
300      * van zodra hij een waarde tegenkomt die niet klopt (ook al kan d
    ie in een latere lus wel kloppen) stopt de testing.
301      *
302      * Net toen ik klaarwas besefte ik dar er nog andere manier was om
     dit te controleren:
303      *     - een variabele int aantalJuiste gebruiken, telkens als hij
     iets tegenkomt dat wel juist is (ook al was deze eerder fout), wordt
    deze variabele verhoogt met 1.
304      *          Aangezien de 2D array een rechthoek is van 21x15 zou d
    e variabele op het einde overeen moeten komen met 315
305      *          --> deze zou waarschijnlijk niet werken door het vorko
    men van dubbele juiste waardes
306      *     - een nieuwe 2D array aanmaken van 21x15, deze wordt opgevu
```

```
306 ld met booleans (allemaal false) als hij deze keer een juiste overeenk
    omst in het level-rooster tegenkomt,
307      *          gaat hij diezelfde locatie in het nieuwe 'rooster' op
    true zetten, was deze eerder nog niet herkend als juist of was hij al
    wel herkend als juist
308      *          wordt deze op juist gezet of blijft deze juist --> dub
    bele waardes worden niet extra geteld en de laatste controle blijft ge
    ldig net zoals bij het echt inladen van een level
309      *          gaan we nadien deze nieuwe rooster controleren en blij
    ken alle locaties true te zijn, dan is de test geslaagd.
310      *
311      * Hoewel onderstaande gebruikte manier minder handig is waren er
    toch manieren om vlotter tot de onderstaande regels te komen.
312      * Ik heb in plaats van direct assertTrue(...); te controleren dez
    e met een System.out.println(...); kunnen uitprinten, de code heb ik t
    ussen "" laten staan en de variabelen niet,
313      * na die welbepaalde functie uit te voeren kon ik alle regels kop
    ieren, sorteren in wordt, en de dubbele er uit filteren (rekening houd
    end met de juiste volgorde).
314      */
315     @Test
316     public void laadEenLevelIn()
317     {
318         Model m = new Model(2);
319         GroteView gv = new GroteView(m);
320         Animator a = new Animator(m,gv);
321         Thread t = new Thread(a);
322         t.start();
323 
324         assertEquals(2, m.getLevelNr());
325         assertFalse(m.getIsErEenDode());
326         assertFalse(m.getIsHetSpelGereset());
327 
328         assertTrue(m.getVoorwerp(0, 0) instanceof Muur);
329         assertTrue(m.getVoorwerp(0, 1) instanceof Muur);
330         assertTrue(m.getVoorwerp(0, 2) instanceof Muur);
331         assertTrue(m.getVoorwerp(0, 3) instanceof Muur);
332         assertTrue(m.getVoorwerp(0, 4) instanceof Muur);
333         assertTrue(m.getVoorwerp(0, 5) instanceof Muur);
334         assertTrue(m.getVoorwerp(0, 6) instanceof Muur);
335         assertTrue(m.getVoorwerp(0, 7) instanceof Muur);
336         assertTrue(m.getVoorwerp(0, 8) instanceof Muur);
337         assertTrue(m.getVoorwerp(0, 9) instanceof Muur);
338         assertTrue(m.getVoorwerp(0, 10) instanceof Muur);
339         assertTrue(m.getVoorwerp(0, 11) instanceof Muur);
340         assertTrue(m.getVoorwerp(0, 12) instanceof Muur);
341         assertTrue(m.getVoorwerp(0, 13) instanceof Muur);
342         assertTrue(m.getVoorwerp(0, 14) instanceof Muur);
343         assertTrue(m.getVoorwerp(1, 0) instanceof Muur);
344         assertTrue(m.getVoorwerp(1, 1) instanceof Krat);
345         assertTrue(m.getVoorwerp(1, 2) instanceof Krat);
346         assertTrue(m.getVoorwerp(1, 3) instanceof Krat);
347         assertTrue(m.getVoorwerp(1, 4) instanceof Grond);
```

```
348         assertTrue(m.getVoorwerp(1, 5) instanceof Grond);
349         assertTrue(m.getVoorwerp(1, 6) instanceof Grond);
350         assertEquals(m.getVoorwerp(1, 7), m.speler1);
351         assertTrue(m.getVoorwerp(1, 8) instanceof Krat);
352         assertTrue(m.getVoorwerp(1, 9) instanceof Grond);
353         assertTrue(m.getVoorwerp(1, 10) instanceof Grond);
354         assertTrue(m.getVoorwerp(1, 11) instanceof Krat);
355         assertTrue(m.getVoorwerp(1, 12) instanceof Krat);
356         assertTrue(m.getVoorwerp(1, 13) instanceof Krat);
357         assertTrue(m.getVoorwerp(1, 14) instanceof Muur);
358         assertTrue(m.getVoorwerp(2, 0) instanceof Muur);
359         assertTrue(m.getVoorwerp(2, 1) instanceof Krat);
360         assertTrue(m.getVoorwerp(2, 2) instanceof Muur);
361         assertTrue(m.getVoorwerp(2, 3) instanceof Grond);
362         assertTrue(m.getVoorwerp(2, 4) instanceof Muur);
363         assertTrue(m.getVoorwerp(2, 5) instanceof Grond);
364         assertTrue(m.getVoorwerp(2, 6) instanceof Muur);
365         assertTrue(m.getVoorwerp(2, 7) instanceof Grond);
366         assertTrue(m.getVoorwerp(2, 8) instanceof Muur);
367         assertTrue(m.getVoorwerp(2, 9) instanceof Grond);
368         assertTrue(m.getVoorwerp(2, 10) instanceof Muur);
369         assertTrue(m.getVoorwerp(2, 11) instanceof Grond);
370         assertTrue(m.getVoorwerp(2, 12) instanceof Muur);
371         assertTrue(m.getVoorwerp(2, 13) instanceof Krat);
372         assertTrue(m.getVoorwerp(2, 14) instanceof Muur);
373         assertTrue(m.getVoorwerp(3, 0) instanceof Muur);
374         assertTrue(m.getVoorwerp(3, 1) instanceof Krat);
375         assertTrue(m.getVoorwerp(3, 2) instanceof Grond);
376         assertTrue(m.getVoorwerp(3, 3) instanceof Grond);
377         assertTrue(m.getVoorwerp(3, 4) instanceof Grond);
378         assertTrue(m.getVoorwerp(3, 5) instanceof Grond);
379         assertTrue(m.getVoorwerp(3, 6) instanceof Grond);
380         assertTrue(m.getVoorwerp(3, 7) instanceof Grond);
381         assertTrue(m.getVoorwerp(3, 8) instanceof Grond);
382         assertTrue(m.getVoorwerp(3, 9) instanceof Grond);
383         assertTrue(m.getVoorwerp(3, 10) instanceof Grond);
384         assertTrue(m.getVoorwerp(3, 11) instanceof Grond);
385         assertTrue(m.getVoorwerp(3, 12) instanceof Grond);
386         assertTrue(m.getVoorwerp(3, 13) instanceof Krat);
387         assertTrue(m.getVoorwerp(3, 14) instanceof Muur);
388         assertTrue(m.getVoorwerp(4, 0) instanceof Muur);
389         assertTrue(m.getVoorwerp(4, 1) instanceof Grond);
390         assertTrue(m.getVoorwerp(4, 2) instanceof Muur);
391         assertTrue(m.getVoorwerp(4, 3) instanceof Grond);
392         assertTrue(m.getVoorwerp(4, 4) instanceof Muur);
393         assertTrue(m.getVoorwerp(4, 5) instanceof Grond);
394         assertTrue(m.getVoorwerp(4, 6) instanceof Muur);
395         assertTrue(m.getVoorwerp(4, 7) instanceof Grond);
396         assertTrue(m.getVoorwerp(4, 8) instanceof Muur);
397         assertTrue(m.getVoorwerp(4, 9) instanceof Grond);
398         assertTrue(m.getVoorwerp(4, 10) instanceof Muur);
399         assertTrue(m.getVoorwerp(4, 11) instanceof Grond);
400         assertTrue(m.getVoorwerp(4, 12) instanceof Muur);
```

```
401         assertTrue(m.getVoorwerp(4, 13) instanceof Grond);
402         assertTrue(m.getVoorwerp(4, 14) instanceof Muur);
403         assertTrue(m.getVoorwerp(5, 0) instanceof Muur);
404         assertTrue(m.getVoorwerp(5, 1) instanceof Grond);
405         assertTrue(m.getVoorwerp(5, 2) instanceof Grond);
406         assertTrue(m.getVoorwerp(5, 3) instanceof Kruit);
407         assertTrue(m.getVoorwerp(5, 4) instanceof Kruit);
408         assertTrue(m.getVoorwerp(5, 5) instanceof Kruit);
409         assertTrue(m.getVoorwerp(5, 6) instanceof Kruit);
410         assertTrue(m.getVoorwerp(5, 7) instanceof Kruit);
411         assertTrue(m.getVoorwerp(5, 8) instanceof Kruit);
412         assertTrue(m.getVoorwerp(5, 9) instanceof Kruit);
413         assertTrue(m.getVoorwerp(5, 10) instanceof Kruit);
414         assertTrue(m.getVoorwerp(5, 11) instanceof Kruit);
415         assertTrue(m.getVoorwerp(5, 12) instanceof Grond);
416         assertTrue(m.getVoorwerp(5, 13) instanceof Grond);
417         assertTrue(m.getVoorwerp(5, 14) instanceof Muur);
418         assertTrue(m.getVoorwerp(6, 0) instanceof Muur);
419         assertTrue(m.getVoorwerp(6, 1) instanceof Grond);
420         assertTrue(m.getVoorwerp(6, 2) instanceof Muur);
421         assertTrue(m.getVoorwerp(6, 3) instanceof Kruit);
422         assertTrue(m.getVoorwerp(6, 4) instanceof Muur);
423         assertTrue(m.getVoorwerp(6, 5) instanceof Grond);
424         assertTrue(m.getVoorwerp(6, 6) instanceof Muur);
425         assertTrue(m.getVoorwerp(6, 7) instanceof Grond);
426         assertTrue(m.getVoorwerp(6, 8) instanceof Muur);
427         assertTrue(m.getVoorwerp(6, 9) instanceof Grond);
428         assertTrue(m.getVoorwerp(6, 10) instanceof Muur);
429         assertTrue(m.getVoorwerp(6, 11) instanceof Kruit);
430         assertTrue(m.getVoorwerp(6, 12) instanceof Muur);
431         assertTrue(m.getVoorwerp(6, 13) instanceof Grond);
432         assertTrue(m.getVoorwerp(6, 14) instanceof Muur);
433         assertTrue(m.getVoorwerp(7, 0) instanceof Muur);
434         assertTrue(m.getVoorwerp(7, 1) instanceof Grond);
435         assertTrue(m.getVoorwerp(7, 2) instanceof Grond);
436         assertTrue(m.getVoorwerp(7, 3) instanceof Kruit);
437         assertTrue(m.getVoorwerp(7, 4) instanceof Grond);
438         assertTrue(m.getVoorwerp(7, 5) instanceof Grond);
439         assertTrue(m.getVoorwerp(7, 6) instanceof Grond);
440         assertTrue(m.getVoorwerp(7, 7) instanceof Grond);
441         assertTrue(m.getVoorwerp(7, 8) instanceof Grond);
442         assertTrue(m.getVoorwerp(7, 9) instanceof Grond);
443         assertTrue(m.getVoorwerp(7, 10) instanceof Grond);
444         assertTrue(m.getVoorwerp(7, 11) instanceof Kruit);
445         assertTrue(m.getVoorwerp(7, 12) instanceof Grond);
446         assertTrue(m.getVoorwerp(7, 13) instanceof Grond);
447         assertTrue(m.getVoorwerp(7, 14) instanceof Muur);
448         assertTrue(m.getVoorwerp(8, 0) instanceof Muur);
449         assertTrue(m.getVoorwerp(8, 1) instanceof Grond);
450         assertTrue(m.getVoorwerp(8, 2) instanceof Muur);
451         assertTrue(m.getVoorwerp(8, 3) instanceof Kruit);
452         assertTrue(m.getVoorwerp(8, 4) instanceof Muur);
453         assertTrue(m.getVoorwerp(8, 5) instanceof Grond);
```

```
454         assertTrue(m.getVoorwerp(8, 6) instanceof Muur);
455         assertTrue(m.getVoorwerp(8, 7) instanceof Grond);
456         assertTrue(m.getVoorwerp(8, 8) instanceof Muur);
457         assertTrue(m.getVoorwerp(8, 9) instanceof Grond);
458         assertTrue(m.getVoorwerp(8, 10) instanceof Muur);
459         assertTrue(m.getVoorwerp(8, 11) instanceof Kruit);
460         assertTrue(m.getVoorwerp(8, 12) instanceof Muur);
461         assertTrue(m.getVoorwerp(8, 13) instanceof Grond);
462         assertTrue(m.getVoorwerp(8, 14) instanceof Muur);
463         assertTrue(m.getVoorwerp(9, 0) instanceof Muur);
464         assertTrue(m.getVoorwerp(9, 1) instanceof Grond);
465         assertTrue(m.getVoorwerp(9, 2) instanceof Grond);
466         assertTrue(m.getVoorwerp(9, 3) instanceof Kruit);
467         assertTrue(m.getVoorwerp(9,  4) instanceof Krat);
468         assertTrue(m.getVoorwerp(9,  5) instanceof Krat);
469         assertTrue(m.getVoorwerp(9, 6) instanceof Grond);
470         assertTrue(m.getVoorwerp(9, 7) instanceof Grond);
471         assertTrue(m.getVoorwerp(9, 8) instanceof Grond);
472         assertTrue(m.getVoorwerp(9, 9) instanceof Krat);
473         assertTrue(m.getVoorwerp(9, 10) instanceof Krat);
474         assertTrue(m.getVoorwerp(9, 11) instanceof Kruit);
475         assertTrue(m.getVoorwerp(9, 12) instanceof Grond);
476         assertTrue(m.getVoorwerp(9, 13) instanceof Grond);
477         assertTrue(m.getVoorwerp(9, 14) instanceof Muur);
478         assertTrue(m.getVoorwerp(10, 0) instanceof Muur);
479         assertTrue(m.getVoorwerp(10, 1) instanceof Grond);
480         assertTrue(m.getVoorwerp(10, 2) instanceof Muur);
481         assertTrue(m.getVoorwerp(10, 3) instanceof Kruit);
482         assertTrue(m.getVoorwerp(10, 4) instanceof Muur);
483         assertTrue(m.getVoorwerp(10, 5) instanceof Krat);
484         assertTrue(m.getVoorwerp(10, 6) instanceof Muur);
485         assertTrue(m.getVoorwerp(10, 7) instanceof Grond);
486         assertTrue(m.getVoorwerp(10, 8) instanceof Muur);
487         assertTrue(m.getVoorwerp(10, 9) instanceof Krat);
488         assertTrue(m.getVoorwerp(10, 10) instanceof Muur);
489         assertTrue(m.getVoorwerp(10, 11) instanceof Kruit);
490         assertTrue(m.getVoorwerp(10, 12) instanceof Muur);
491         assertTrue(m.getVoorwerp(10, 13) instanceof Grond);
492         assertTrue(m.getVoorwerp(10, 14) instanceof Muur);
493         assertTrue(m.getVoorwerp(11, 0) instanceof Muur);
494         assertTrue(m.getVoorwerp(11, 1) instanceof Grond);
495         assertTrue(m.getVoorwerp(11, 2) instanceof Grond);
496         assertTrue(m.getVoorwerp(11, 3) instanceof Kruit);
497         assertTrue(m.getVoorwerp(11, 4) instanceof Krat);
498         assertTrue(m.getVoorwerp(11, 5) instanceof Krat);
499         assertTrue(m.getVoorwerp(11, 6) instanceof Grond);
500         assertTrue(m.getVoorwerp(11, 7) instanceof Grond);
501         assertTrue(m.getVoorwerp(11, 8) instanceof Grond);
502         assertTrue(m.getVoorwerp(11, 9) instanceof Krat);
503         assertTrue(m.getVoorwerp(11, 10) instanceof Krat);
504         assertTrue(m.getVoorwerp(11, 11) instanceof Kruit);
505         assertTrue(m.getVoorwerp(11, 12) instanceof Grond);
506         assertTrue(m.getVoorwerp(11, 13) instanceof Grond);
```

```
507         assertTrue(m.getVoorwerp(11, 14) instanceof Muur);
508         assertTrue(m.getVoorwerp(12, 0) instanceof Muur);
509         assertTrue(m.getVoorwerp(12, 1) instanceof Grond);
510         assertTrue(m.getVoorwerp(12, 2) instanceof Muur);
511         assertTrue(m.getVoorwerp(12, 3) instanceof Kruit);
512         assertTrue(m.getVoorwerp(12, 4) instanceof Muur);
513         assertTrue(m.getVoorwerp(12, 5) instanceof Grond);
514         assertTrue(m.getVoorwerp(12, 6) instanceof Muur);
515         assertTrue(m.getVoorwerp(12, 7) instanceof Grond);
516         assertTrue(m.getVoorwerp(12, 8) instanceof Muur);
517         assertTrue(m.getVoorwerp(12, 9) instanceof Grond);
518         assertTrue(m.getVoorwerp(12, 10) instanceof Muur);
519         assertTrue(m.getVoorwerp(12, 11) instanceof Kruit);
520         assertTrue(m.getVoorwerp(12, 12) instanceof Muur);
521         assertTrue(m.getVoorwerp(12, 13) instanceof Grond);
522         assertTrue(m.getVoorwerp(12, 14) instanceof Muur);
523         assertTrue(m.getVoorwerp(13, 0) instanceof Muur);
524         assertTrue(m.getVoorwerp(13, 1) instanceof Grond);
525         assertTrue(m.getVoorwerp(13, 2) instanceof Grond);
526         assertTrue(m.getVoorwerp(13, 3) instanceof Kruit);
527         assertTrue(m.getVoorwerp(13, 4) instanceof Grond);
528         assertTrue(m.getVoorwerp(13, 5) instanceof Grond);
529         assertTrue(m.getVoorwerp(13, 6) instanceof Grond);
530         assertTrue(m.getVoorwerp(13, 7) instanceof Grond);
531         assertTrue(m.getVoorwerp(13, 8) instanceof Grond);
532         assertTrue(m.getVoorwerp(13, 9) instanceof Grond);
533         assertTrue(m.getVoorwerp(13, 10) instanceof Grond);
534         assertTrue(m.getVoorwerp(13, 11) instanceof Kruit);
535         assertTrue(m.getVoorwerp(13, 12) instanceof Grond);
536         assertTrue(m.getVoorwerp(13, 13) instanceof Grond);
537         assertTrue(m.getVoorwerp(13, 14) instanceof Muur);
538         assertTrue(m.getVoorwerp(14, 0) instanceof Muur);
539         assertTrue(m.getVoorwerp(14, 1) instanceof Grond);
540         assertTrue(m.getVoorwerp(14, 2) instanceof Muur);
541         assertTrue(m.getVoorwerp(14, 3) instanceof Kruit);
542         assertTrue(m.getVoorwerp(14, 4) instanceof Muur);
543         assertTrue(m.getVoorwerp(14, 5) instanceof Grond);
544         assertTrue(m.getVoorwerp(14, 6) instanceof Muur);
545         assertTrue(m.getVoorwerp(14, 7) instanceof Grond);
546         assertTrue(m.getVoorwerp(14, 8) instanceof Muur);
547         assertTrue(m.getVoorwerp(14, 9) instanceof Grond);
548         assertTrue(m.getVoorwerp(14, 10) instanceof Muur);
549         assertTrue(m.getVoorwerp(14, 11) instanceof Kruit);
550         assertTrue(m.getVoorwerp(14, 12) instanceof Muur);
551         assertTrue(m.getVoorwerp(14, 13) instanceof Grond);
552         assertTrue(m.getVoorwerp(14, 14) instanceof Muur);
553         assertTrue(m.getVoorwerp(15, 0) instanceof Muur);
554         assertTrue(m.getVoorwerp(15, 1) instanceof Grond);
555         assertTrue(m.getVoorwerp(15, 2) instanceof Grond);
556         assertTrue(m.getVoorwerp(15, 3) instanceof Kruit);
557         assertTrue(m.getVoorwerp(15, 4) instanceof Kruit);
558         assertTrue(m.getVoorwerp(15, 5) instanceof Kruit);
559         assertTrue(m.getVoorwerp(15, 6) instanceof Kruit);
```

```
560         assertTrue(m.getVoorwerp(15, 7) instanceof Kruit);
561         assertTrue(m.getVoorwerp(15, 8) instanceof Kruit);
562         assertTrue(m.getVoorwerp(15, 9) instanceof Kruit);
563         assertTrue(m.getVoorwerp(15, 10) instanceof Kruit);
564         assertTrue(m.getVoorwerp(15, 11) instanceof Kruit);
565         assertTrue(m.getVoorwerp(15, 12) instanceof Grond);
566         assertTrue(m.getVoorwerp(15, 13) instanceof Grond);
567         assertTrue(m.getVoorwerp(15, 14) instanceof Muur);
568         assertTrue(m.getVoorwerp(16, 0) instanceof Muur);
569         assertTrue(m.getVoorwerp(16, 1) instanceof Grond);
570         assertTrue(m.getVoorwerp(16, 2) instanceof Muur);
571         assertTrue(m.getVoorwerp(16, 3) instanceof Grond);
572         assertTrue(m.getVoorwerp(16, 4) instanceof Muur);
573         assertTrue(m.getVoorwerp(16, 5) instanceof Grond);
574         assertTrue(m.getVoorwerp(16, 6) instanceof Muur);
575         assertTrue(m.getVoorwerp(16, 7) instanceof Grond);
576         assertTrue(m.getVoorwerp(16, 8) instanceof Muur);
577         assertTrue(m.getVoorwerp(16, 9) instanceof Grond);
578         assertTrue(m.getVoorwerp(16, 10) instanceof Muur);
579         assertTrue(m.getVoorwerp(16, 11) instanceof Grond);
580         assertTrue(m.getVoorwerp(16, 12) instanceof Muur);
581         assertTrue(m.getVoorwerp(16, 13) instanceof Grond);
582         assertTrue(m.getVoorwerp(16, 14) instanceof Muur);
583         assertTrue(m.getVoorwerp(17, 0) instanceof Muur);
584         assertTrue(m.getVoorwerp(17, 1) instanceof Krat);
585         assertTrue(m.getVoorwerp(17, 2) instanceof Grond);
586         assertTrue(m.getVoorwerp(17, 3) instanceof Grond);
587         assertTrue(m.getVoorwerp(17, 4) instanceof Grond);
588         assertTrue(m.getVoorwerp(17, 5) instanceof Grond);
589         assertTrue(m.getVoorwerp(17, 6) instanceof Grond);
590         assertTrue(m.getVoorwerp(17, 7) instanceof Grond);
591         assertTrue(m.getVoorwerp(17, 8) instanceof Grond);
592         assertTrue(m.getVoorwerp(17, 9) instanceof Grond);
593         assertTrue(m.getVoorwerp(17, 10) instanceof Grond);
594         assertTrue(m.getVoorwerp(17, 11) instanceof Grond);
595         assertTrue(m.getVoorwerp(17, 12) instanceof Grond);
596         assertTrue(m.getVoorwerp(17, 13) instanceof Krat);
597         assertTrue(m.getVoorwerp(17, 14) instanceof Muur);
598         assertTrue(m.getVoorwerp(18, 0) instanceof Muur);
599         assertTrue(m.getVoorwerp(18, 1) instanceof Krat);
600         assertTrue(m.getVoorwerp(18, 2) instanceof Muur);
601         assertTrue(m.getVoorwerp(18, 3) instanceof Grond);
602         assertTrue(m.getVoorwerp(18, 4) instanceof Muur);
603         assertTrue(m.getVoorwerp(18, 5) instanceof Grond);
604         assertTrue(m.getVoorwerp(18, 6) instanceof Muur);
605         assertTrue(m.getVoorwerp(18, 7) instanceof Grond);
606         assertTrue(m.getVoorwerp(18, 8) instanceof Muur);
607         assertTrue(m.getVoorwerp(18, 9) instanceof Grond);
608         assertTrue(m.getVoorwerp(18, 10) instanceof Muur);
609         assertTrue(m.getVoorwerp(18, 11) instanceof Grond);
610         assertTrue(m.getVoorwerp(18, 12) instanceof Muur);
611         assertTrue(m.getVoorwerp(18, 13) instanceof Krat);
612         assertTrue(m.getVoorwerp(18, 14) instanceof Muur);
```

```
613         assertTrue(m.getVoorwerp(19, 0) instanceof Muur);
614         assertTrue(m.getVoorwerp(19, 1) instanceof Krat);
615         assertTrue(m.getVoorwerp(19, 2) instanceof Krat);
616         assertTrue(m.getVoorwerp(19, 3) instanceof Krat);
617         assertTrue(m.getVoorwerp(19, 4) instanceof Grond);
618         assertTrue(m.getVoorwerp(19, 5) instanceof Grond);
619         assertTrue(m.getVoorwerp(19, 6) instanceof Krat);
620         assertEquals(m.getVoorwerp(19, 7), m.speler2);
621         assertTrue(m.getVoorwerp(19, 8) instanceof Grond);
622         assertTrue(m.getVoorwerp(19, 9) instanceof Grond);
623         assertTrue(m.getVoorwerp(19, 10) instanceof Grond);
624         assertTrue(m.getVoorwerp(19, 11) instanceof Krat);
625         assertTrue(m.getVoorwerp(19, 12) instanceof Krat);
626         assertTrue(m.getVoorwerp(19, 13) instanceof Krat);
627         assertTrue(m.getVoorwerp(19, 14) instanceof Muur);
628         assertTrue(m.getVoorwerp(20, 0) instanceof Muur);
629         assertTrue(m.getVoorwerp(20, 1) instanceof Muur);
630         assertTrue(m.getVoorwerp(20, 2) instanceof Muur);
631         assertTrue(m.getVoorwerp(20, 3) instanceof Muur);
632         assertTrue(m.getVoorwerp(20, 4) instanceof Muur);
633         assertTrue(m.getVoorwerp(20, 5) instanceof Muur);
634         assertTrue(m.getVoorwerp(20, 6) instanceof Muur);
635         assertTrue(m.getVoorwerp(20, 7) instanceof Muur);
636         assertTrue(m.getVoorwerp(20, 8) instanceof Muur);
637         assertTrue(m.getVoorwerp(20, 9) instanceof Muur);
638         assertTrue(m.getVoorwerp(20, 10) instanceof Muur);
639         assertTrue(m.getVoorwerp(20, 11) instanceof Muur);
640         assertTrue(m.getVoorwerp(20, 12) instanceof Muur);
641         assertTrue(m.getVoorwerp(20, 13) instanceof Muur);
642         assertTrue(m.getVoorwerp(20, 14) instanceof Muur);
643     }
644
645     /**
646      * Deze functie laat een bepaalde speler een willekeurige actie do
    en. Deze kan ook worden aangesproken door de volgende functie, deze ge
    eft dan echter een willekeurige speler mee.
647      * Om te voorkomen dat de speler niet te veel bommen legt, gebeurt
     dit enkel als het willekeurige nummertje 5 wordt getrokken.
648      * In alle ander gevallen gaat de fucntie willekeurigBewegen() wor
    den opgeroepen om een willekeurige richting te kiezen en deze laten ui
    t te voeren door de eerder vernoemde speler.
649      *
650      * 1,2,3,4 = bewegen
651      * 5 = bom leggen
652      */
653     public void willekeurigeActie(Model testM, int rndspeler)
654     {
655         if (!testM.getIsErEenDode())
656         {
657             int rndactie = (int)(1+5*Math.random());
658             if (rndactie == 7)
659             {
660                 testM.legBom(rndspeler);
```

```
661             }
662             else
663             {
664                 willekeurigBewegen(testM, rndspeler);
665             }
666         }
667     }
668 
669     /**
670      * Deze functie laat een willekeurige speler een willekeurige acti
    e doen, met het willekeurige nummer dat wordt getrokken wordt de vorig
    e functie functie uitgevoerd.
671      */
672     public void willekeurigeActie(Model testM)
673     {
674         if (!testM.getIsErEenDode())
675         {
676             int rndspeler = (int)(1+2*Math.random());
677             willekeurigeActie(testM, rndspeler);
678         }
679     }
680 
681     /**
682      * Deze functie laat een bepaalde speler een willekeurige beweging
     doen. Deze kan ook worden aangesproken door de volgende functie, deze
     geeft dan echter een willekeurige speler mee.
683      *
684      * 1=beweeglinks
685      * 2=beweegrecht
686      * 3=beweegonder
687      * 4=beweegboven
688      */
689     public void willekeurigBewegen(Model testM, int rndspeler)
690     {
691         if (!testM.getIsErEenDode())
692         {
693             int rndrichting = (int)(1+4*Math.random());
694             switch (rndrichting)
695             {
696                 case 1: testM.beweegLinks(rndspeler); break;
697                 case 2: testM.beweegRechts(rndspeler); break;
698                 case 3: testM.beweegOnder(rndspeler); break;
699                 case 4: testM.beweegBoven(rndspeler); break;
700             }
701         }
702     }
703 
704     /**
705      * Deze functie laat een willekeurige speler een willekeurige bewe
    ging doen, met het willekeurige nummer dat wordt getrokken wordt de vo
    rige functie uitgevoerd.
706      */
707     public void willekeurigBewegen(Model testM)
```

```
708     {
709         if (!testM.getIsErEenDode())
710         {
711             int rndspeler = (int)(1+2*Math.random());
712             willekeurigBewegen(testM, rndspeler);
713         }
714     }
715 }
```