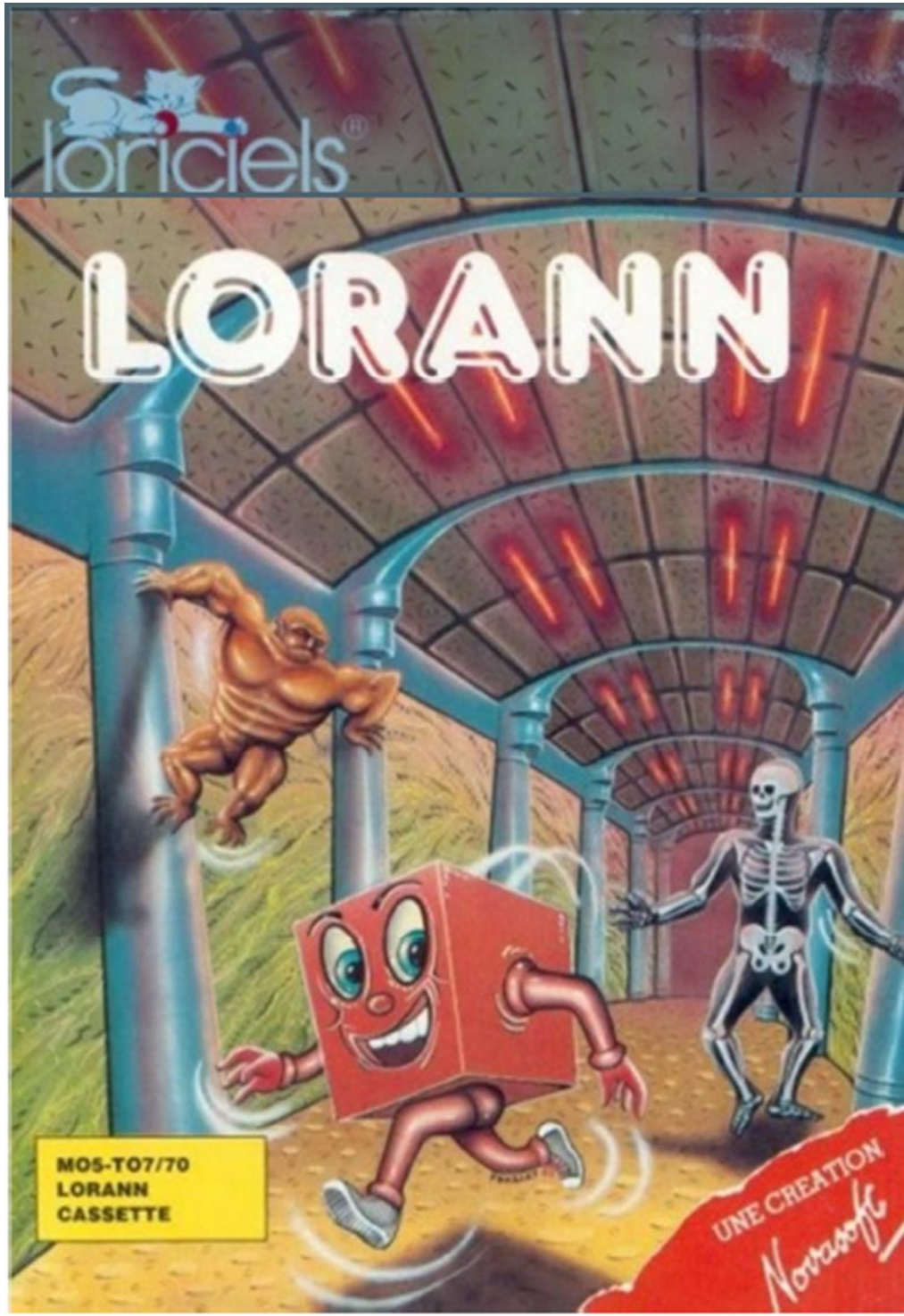


JAVA PROJECT



Project presentation:

LORANN is a pac-man like game released in 1985. In this game, we have to run away from ghosts trying to killing us and find a key in order to escape the level by a door. We had to re-code this abandonware in JAVA language. We code five levels more and more difficult with, for each one, a different map. Once you reach the door you passed to the next level. Finally you finished the game by ending the fifth level. When you play the game, make sure your sound system is on.

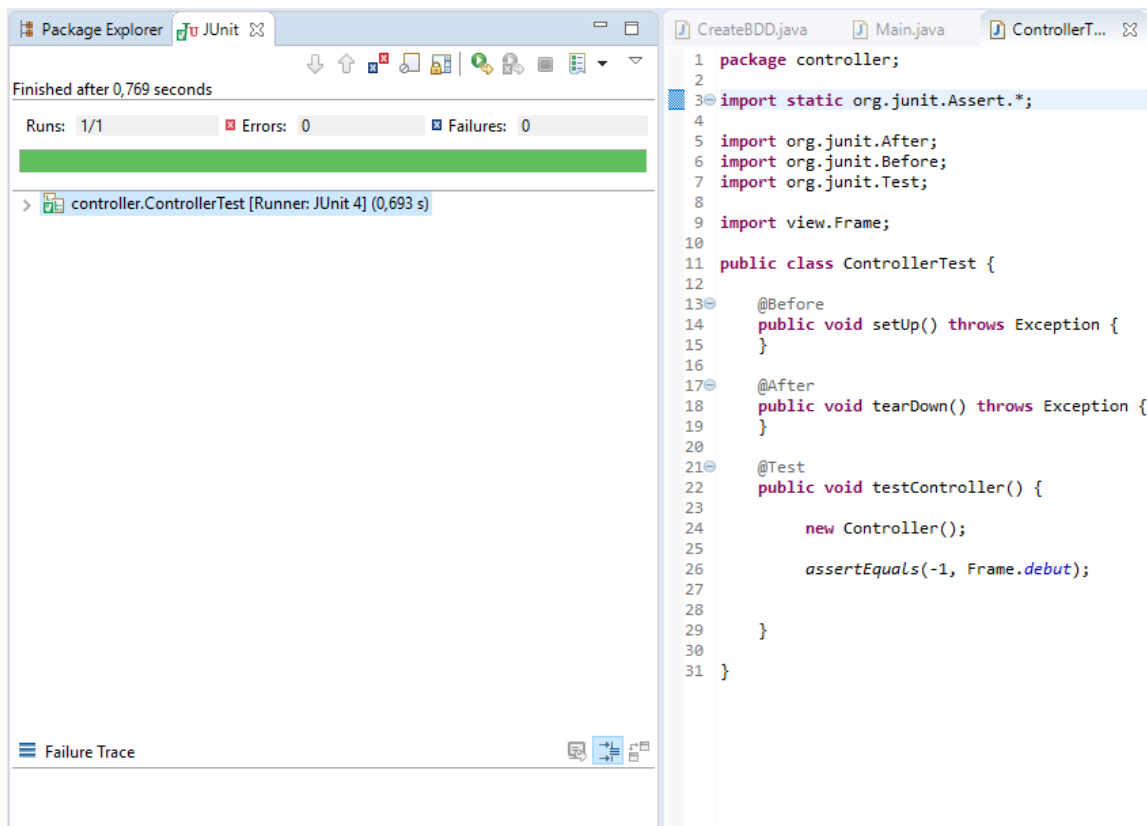
Deliverables

You can find the Javadoc in the component Contract.

All the diagrams are in the Java Project.vpp except the sequence diagram which is in Java Project2.vpp.

When we try to build a SureFire report, we had a lot of problems with Maeven's dependencies. So, we decided to had screenshot of every Junit Tests in order to show the well-working of them.

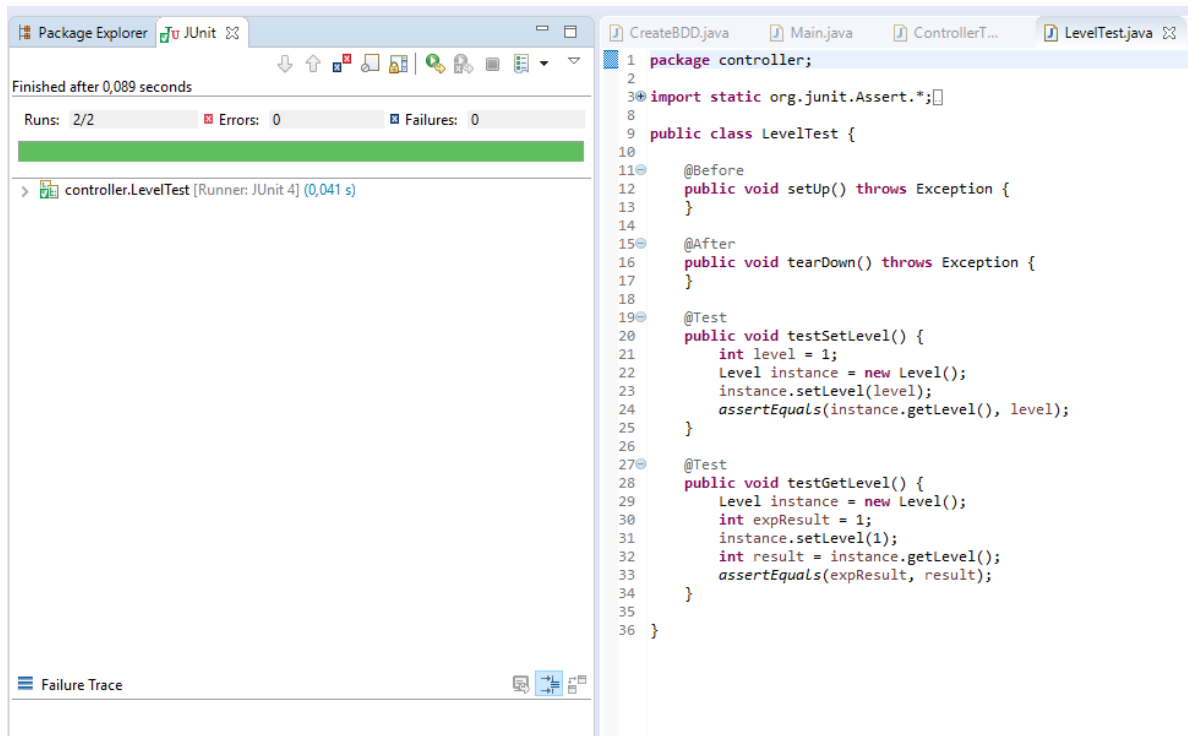
Controlleur test :



The screenshot shows an IDE with the Package Explorer on the left and the JUnit console below it. The console indicates that the test finished after 0,769 seconds, with 1/1 runs, 0 errors, and 0 failures. The test class is controller.ControllerTest, run by JUnit 4, taking 0,693 seconds. The right pane displays the source code of ControllerTest.java:

```
1 package controller;
2
3 import static org.junit.Assert.*;
4
5 import org.junit.After;
6 import org.junit.Before;
7 import org.junit.Test;
8
9 import view.Frame;
10
11 public class ControllerTest {
12
13     @Before
14     public void setUp() throws Exception {
15     }
16
17     @After
18     public void tearDown() throws Exception {
19     }
20
21     @Test
22     public void testController() {
23
24         new Controller();
25
26         assertEquals(-1, Frame.debut);
27
28     }
29
30 }
31 }
```

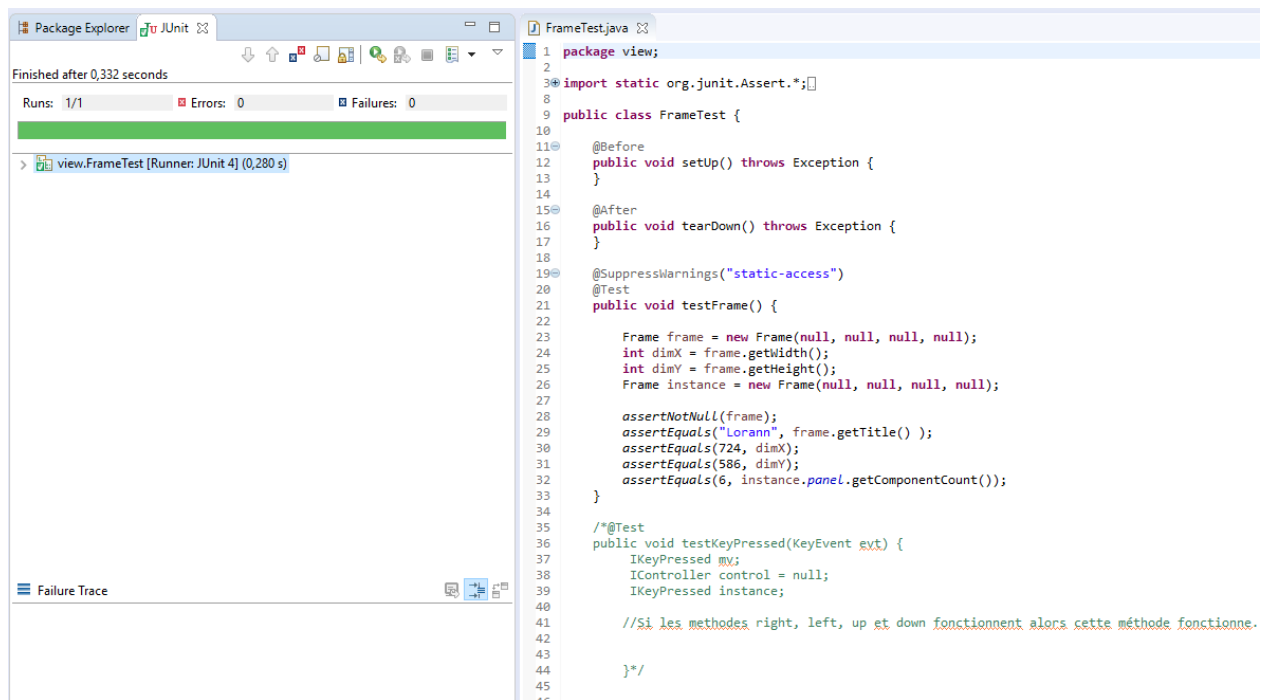
Level test :



The screenshot shows an IDE with the Package Explorer on the left and the JUnit console below it. The console indicates that the test finished after 0,089 seconds, with 2/2 runs, 0 errors, and 0 failures. The test class is controller.LevelTest, run by JUnit 4, taking 0,041 seconds. The right pane displays the source code of LevelTest.java:

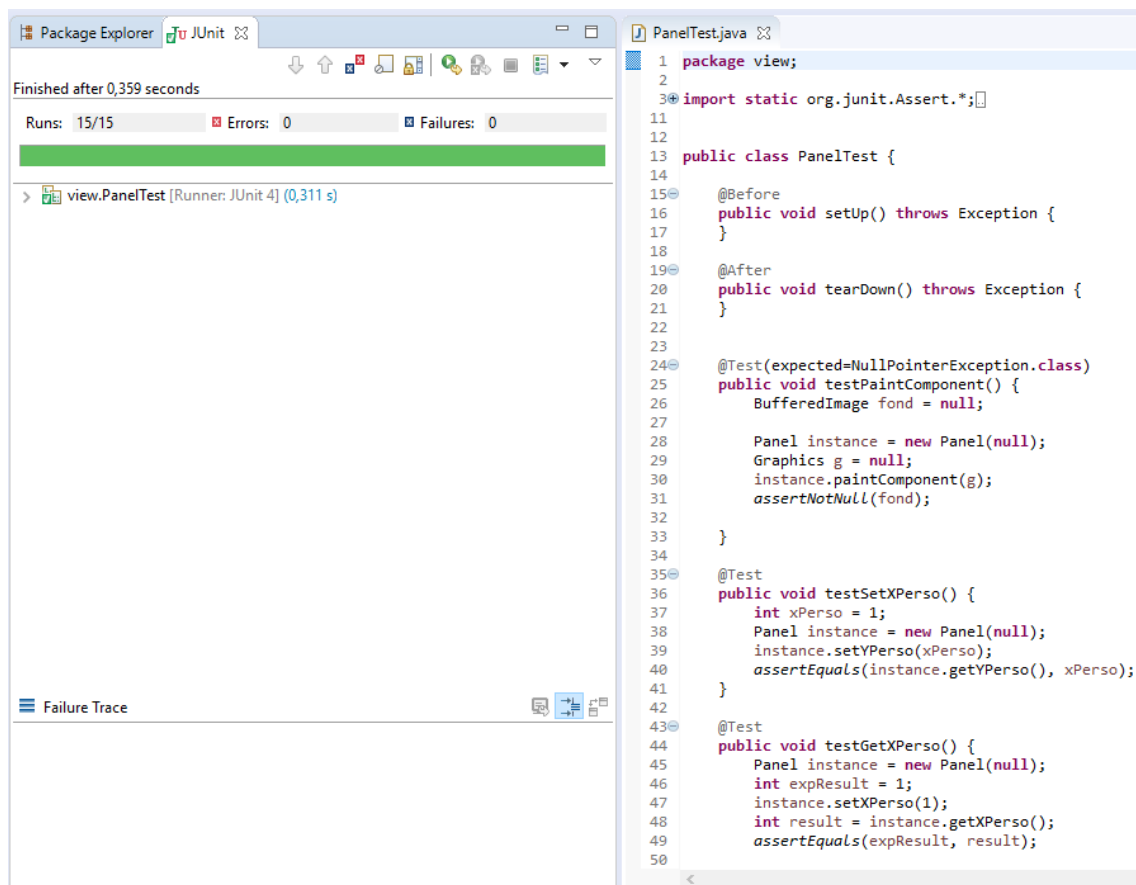
```
1 package controller;
2
3 import static org.junit.Assert.*;
4
5 public class LevelTest {
6
7     @Before
8     public void setUp() throws Exception {
9     }
10
11     @After
12     public void tearDown() throws Exception {
13     }
14
15     @Test
16     public void testSetLevel() {
17         int level = 1;
18         Level instance = new Level();
19         instance.setLevel(level);
20         assertEquals(instance.getLevel(), level);
21     }
22
23     @Test
24     public void testGetLevel() {
25         Level instance = new Level();
26         int expectedResult = 1;
27         instance.setLevel(1);
28         int result = instance.getLevel();
29         assertEquals(expectedResult, result);
30     }
31
32 }
33 }
```

Frame test :



```
1 package view;
2
3 import static org.junit.Assert.*;
4
5 public class FrameTest {
6
7     @Before
8     public void setUp() throws Exception {
9     }
10
11     @After
12     public void tearDown() throws Exception {
13     }
14
15     @SuppressWarnings("static-access")
16     @Test
17     public void testFrame() {
18
19         Frame frame = new Frame(null, null, null, null);
20         int dimX = frame.getWidth();
21         int dimY = frame.getHeight();
22         Frame instance = new Frame(null, null, null, null);
23
24         assertNotNull(frame);
25         assertEquals("Lorann", frame.getTitle() );
26         assertEquals(724, dimX);
27         assertEquals(586, dimY);
28         assertEquals(6, instance.panel.getComponentCount());
29     }
30
31     /*@Test
32     public void testKeyPressed(KeyEvent evt) {
33         IKeyPressed mv;
34         IController control = null;
35         IKeyPressed instance;
36
37         //Si les methodes right, left, up et down fonctionnent alors cette methode fonctionne.
38     }
39 */
40 }
```

Panel test :

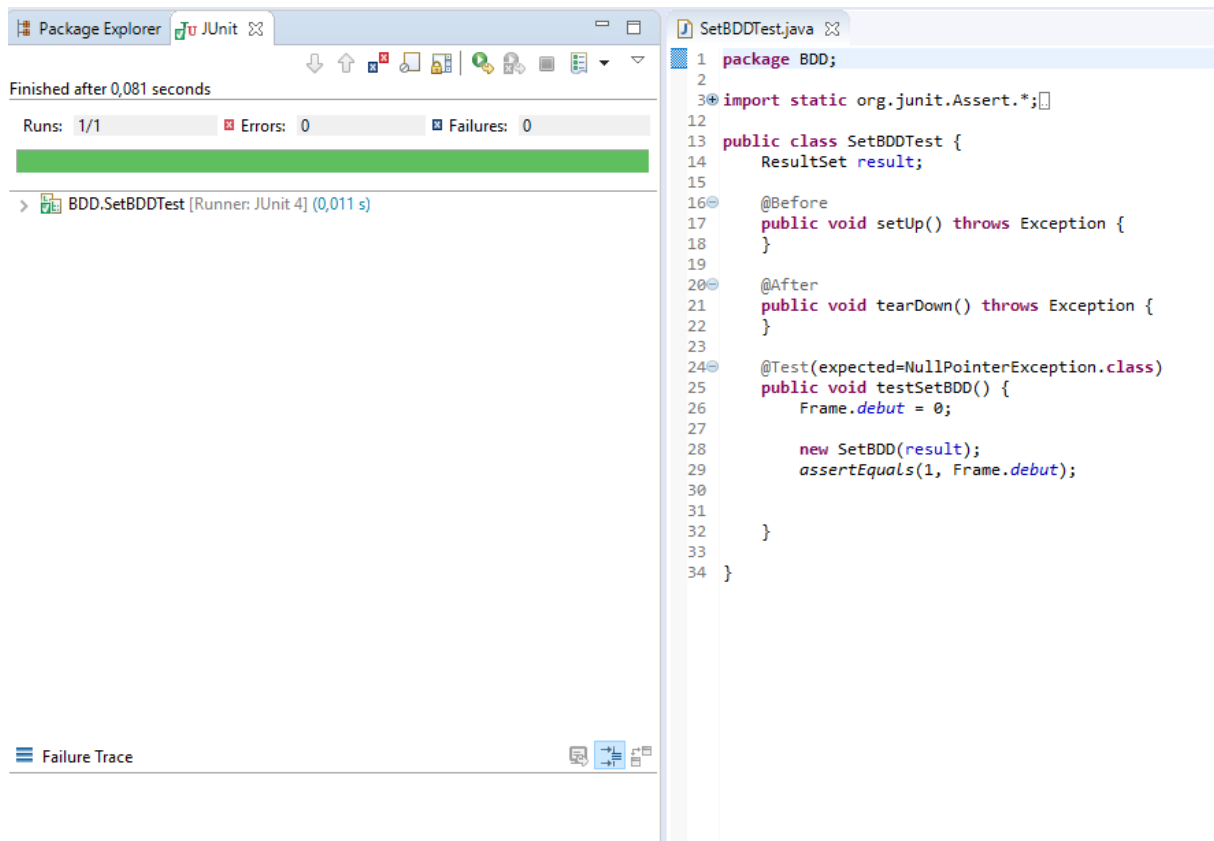


```
1 package view;
2
3 import static org.junit.Assert.*;
4
5 public class PanelTest {
6
7     @Before
8     public void setUp() throws Exception {
9     }
10
11     @After
12     public void tearDown() throws Exception {
13     }
14
15     @Test(expected=NullPointerException.class)
16     public void testPaintComponent() {
17         BufferedImage fond = null;
18
19         Panel instance = new Panel(null);
20         Graphics g = null;
21         instance.paintComponent(g);
22         assertNotNull(fond);
23     }
24
25     @Test
26     public void testSetXPerso() {
27         int xPerso = 1;
28         Panel instance = new Panel(null);
29         instance.setYPerso(xPerso);
30         assertEquals(instance.getYPerso(), xPerso);
31     }
32
33     @Test
34     public void testGetXPerso() {
35         Panel instance = new Panel(null);
36         int expResult = 1;
37         instance.setXPerso(1);
38         int result = instance.getXPerso();
39         assertEquals(expResult, result);
40     }
41 }
```

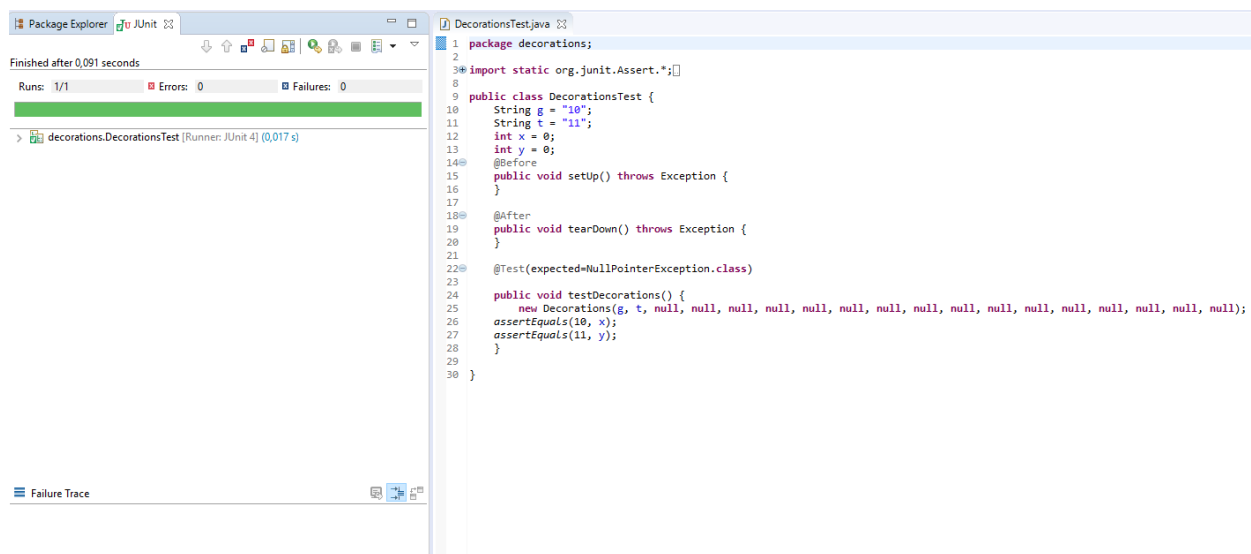
HaveBDD test :



SetBDD :



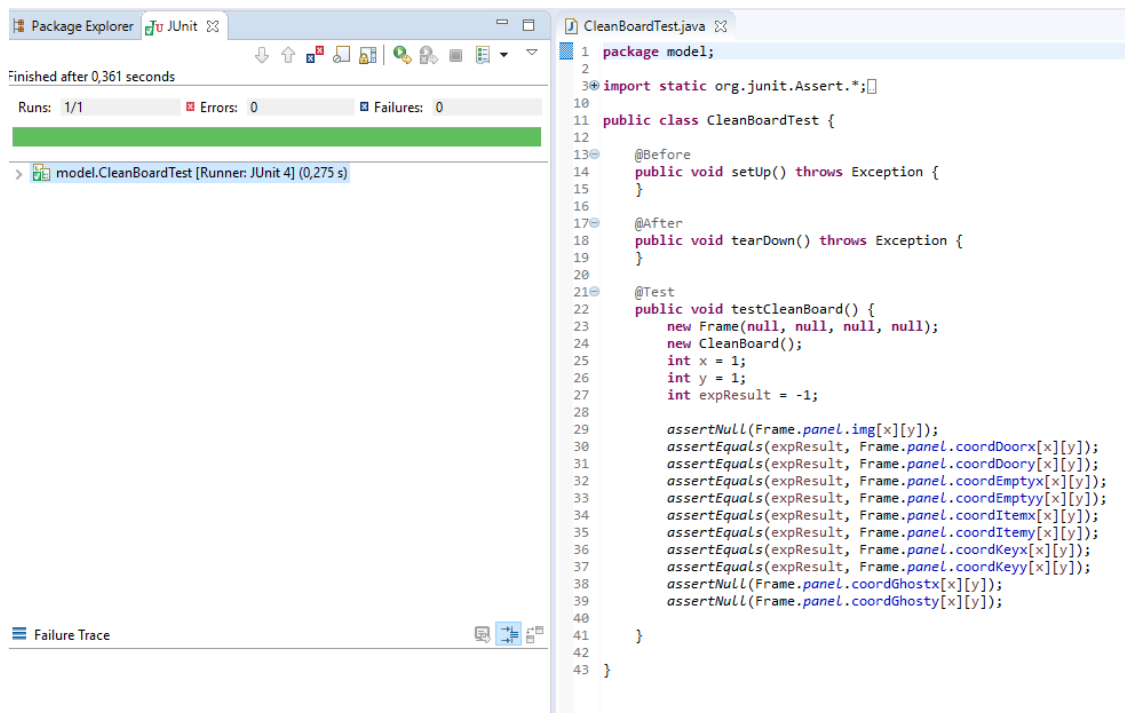
Decoration test :



The screenshot shows an IDE with the Package Explorer on the left, the JUnit runner output in the center, and the source code of DecorationsTest.java on the right. The test is successful, with 1/1 runs, 0 errors, and 0 failures. The source code includes package declarations, imports, and a test method that verifies the state of a Decorations object.

```
1 package decorations;
2
3 import static org.junit.Assert.*;
4
5 public class DecorationsTest {
6     String g = "10";
7     String t = "11";
8     int x = 0;
9     int y = 0;
10    @Before
11    public void setUp() throws Exception {
12    }
13    @After
14    public void tearDown() throws Exception {
15    }
16    @Test(expected=NullPointerException.class)
17    public void testDecorations() {
18        new Decorations(g, t, null, null, null, null, null, null, null, null, null, null, null, null, null, null);
19        assertEquals(10, x);
20        assertEquals(11, y);
21    }
22 }
```

CleanBoard test :



The screenshot shows an IDE with the Package Explorer on the left, the JUnit runner output in the center, and the source code of CleanBoardTest.java on the right. The test is successful, with 1/1 runs, 0 errors, and 0 failures. The source code includes package declarations, imports, and a test method that verifies the state of a CleanBoard object after a frame is created.

```
1 package model;
2
3 import static org.junit.Assert.*;
4
5 public class CleanBoardTest {
6
7     @Before
8     public void setUp() throws Exception {
9     }
10    @After
11    public void tearDown() throws Exception {
12    }
13    @Test
14    public void testCleanBoard() {
15        new Frame(null, null, null, null);
16        new CleanBoard();
17        int x = 1;
18        int y = 1;
19        int expResult = -1;
20
21        assertNull(Frame.panel.img[x][y]);
22        assertEquals(expResult, Frame.panel.coordDoorx[x][y]);
23        assertEquals(expResult, Frame.panel.coordDoory[x][y]);
24        assertEquals(expResult, Frame.panel.coordEmptyx[x][y]);
25        assertEquals(expResult, Frame.panel.coordEmptyy[x][y]);
26        assertEquals(expResult, Frame.panel.coordItemx[x][y]);
27        assertEquals(expResult, Frame.panel.coordItemy[x][y]);
28        assertEquals(expResult, Frame.panel.coordKeyx[x][y]);
29        assertEquals(expResult, Frame.panel.coordKeyy[x][y]);
30        assertNull(Frame.panel.coordGhostx[x][y]);
31        assertNull(Frame.panel.coordGhosty[x][y]);
32    }
33 }
```

MooveLorann test :

Package Explorer JUnit

Finished after 0,699 seconds

Runs: 4/4 Errors: 0 Failures: 0

> model.MooveLorannTest [Runner: JUnit 4] (0,292 s)

Failure Trace

```
1 package model;
2
3 import static org.junit.Assert.*;
10
11
12 public class MooveLorannTest {
13     int XPerso = 1;
14     int YPerso = 1;
15     Frame instance = new Frame(null, null, null, null);
16     MooveLorann mv = new MooveLorann();
17
18     @Before
19     public void setUp() throws Exception {
20     }
21
22     @After
23     public void tearDown() throws Exception {
24     }
25
26     @Test
27     public void testMooveRight() {
28         Frame.panel.setXPerso(XPerso);
29
30         mv.mooveRight();
31         int x1 = Frame.panel.getXPerso();
32         assertEquals(2, x1);
33     }
34
35     @Test
36     public void testMooveLeft() {
37         Frame.panel.setXPerso(XPerso);
38
39         mv.mooveLeft();
40         int x1 = Frame.panel.getXPerso();
41         assertEquals(0, x1);
42     }
43
44     @Test
45     public void testMooveUp() {
46         Frame.panel.setYPerso(YPerso);
47
48         mv.mooveUp();
49         int x1 = Frame.panel.getYPerso();
```

ThreadIA test :

Package Explorer JUnit

Finished after 0,082 seconds

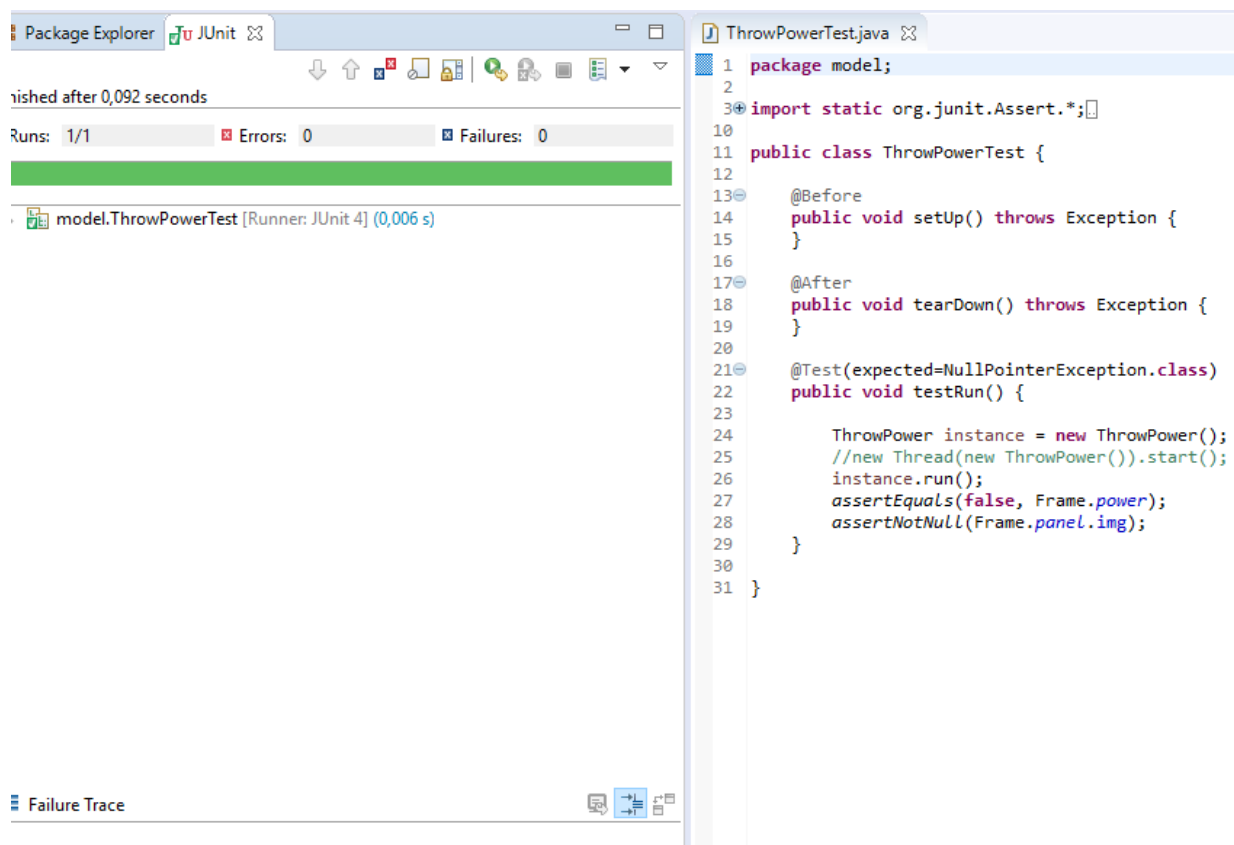
Runs: 1/1 Errors: 0 Failures: 0

> model.ThreadIATest [Runner: JUnit 4] (0,004 s)

Failure Trace

```
1 package model;
2
3 import static org.junit.Assert.*;
10
11 public class ThreadIATest {
12     public int coordEmptyx[][] = new int[22][17];
13     public int coordEmptyy[][] = new int[22][17];
14     public Integer coordGhostx[][] = new Integer[22][17];
15     public Integer coordGhosty[][] = new Integer[22][17];
16     int x = 1;
17     int y = 1;
18     @Before
19     public void setUp() throws Exception {
20     }
21
22     @After
23     public void tearDown() throws Exception {
24     }
25
26     @Test(expected=NullPointerException.class)
27     public void testRun() {
28         Frame.panel.coordGhostx = null;
29         new Thread(new ThreadIA()).start();
30         assertNotNull(Frame.panel.coordGhostx);
31         assertNotNull(Frame.panel.coordGhosty);
32     }
33
34
35 }
```

ThrowPower test :



Package Explorer JUnit

Finished after 0,092 seconds

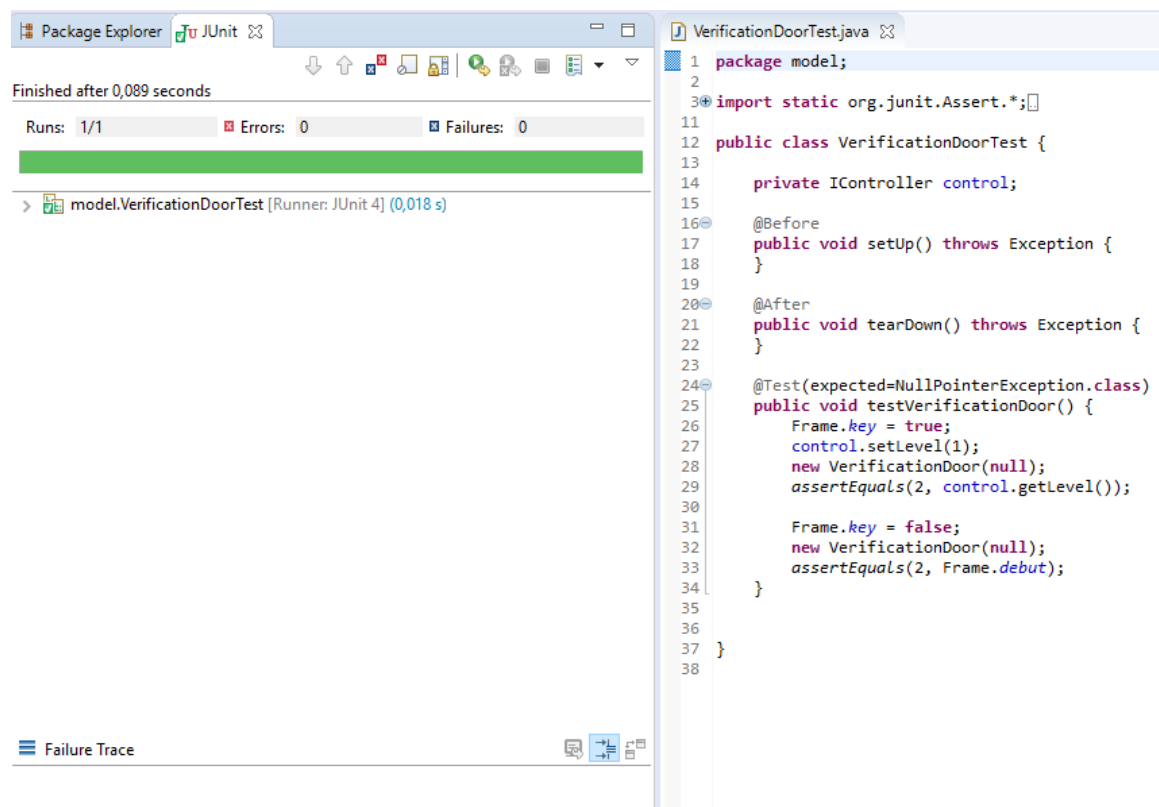
Runs: 1/1 Errors: 0 Failures: 0

model.ThrowPowerTest [Runner: JUnit 4] (0,006 s)

```
1 package model;
2
3 import static org.junit.Assert.*;
10
11 public class ThrowPowerTest {
12
13     @Before
14     public void setUp() throws Exception {
15     }
16
17     @After
18     public void tearDown() throws Exception {
19     }
20
21     @Test(expected=NullPointerException.class)
22     public void testRun() {
23
24         ThrowPower instance = new ThrowPower();
25         //new Thread(new ThrowPower()).start();
26         instance.run();
27         assertEquals(false, Frame.power);
28         assertNotNull(Frame.panel.img);
29     }
30 }
31 }
```

Failure Trace

VerifictaionDoor test :



Package Explorer JUnit

Finished after 0,089 seconds

Runs: 1/1 Errors: 0 Failures: 0

> model.VerificationDoorTest [Runner: JUnit 4] (0,018 s)

```
1 package model;
2
3 import static org.junit.Assert.*;
11
12 public class VerificationDoorTest {
13
14     private IController control;
15
16     @Before
17     public void setUp() throws Exception {
18     }
19
20     @After
21     public void tearDown() throws Exception {
22     }
23
24     @Test(expected=NullPointerException.class)
25     public void testVerificationDoor() {
26         Frame.key = true;
27         control.setLevel(1);
28         new VerificationDoor(null);
29         assertEquals(2, control.getLevel());
30
31         Frame.key = false;
32         new VerificationDoor(null);
33         assertEquals(2, Frame.debut);
34     }
35
36
37 }
38 }
```

Failure Trace

Personal review:

Rémi PAPIN: I miss half of the project due to a mononucleosis. Nevertheless, I follow the project's advancements and help to write some parts of the game. When I come back, I write all the Junit tests and help Corentin for the diagram sequence. I must learn how does every method work for writing the tests. I met many problems but with the help of my teammate we release all the deliverables asked.

This project was long nevertheless, it was interesting and gave us a huge sense of accomplishment.

Corentin BRION: During this project I worked on the UML and the program. I have some problems with Maven when we start but with the time I have learned to know what is Maven. I have learned a lot of method during this project.

To finish I think this project was cool but too long.

Corentin BOURGEY: During this project, I worked about the code. I think this project was good and funny.

To finish I think this project was cool but too long. I need much time to add other functionality.

Information :

There were problems during the project with maven, so only my Computer putted project on Github.

There is a guide for the game
(command...)

IMPORTANT : USE 1DK 10 (I don't know if
the project work in JDK 8).