Guided Project - Iteration #3

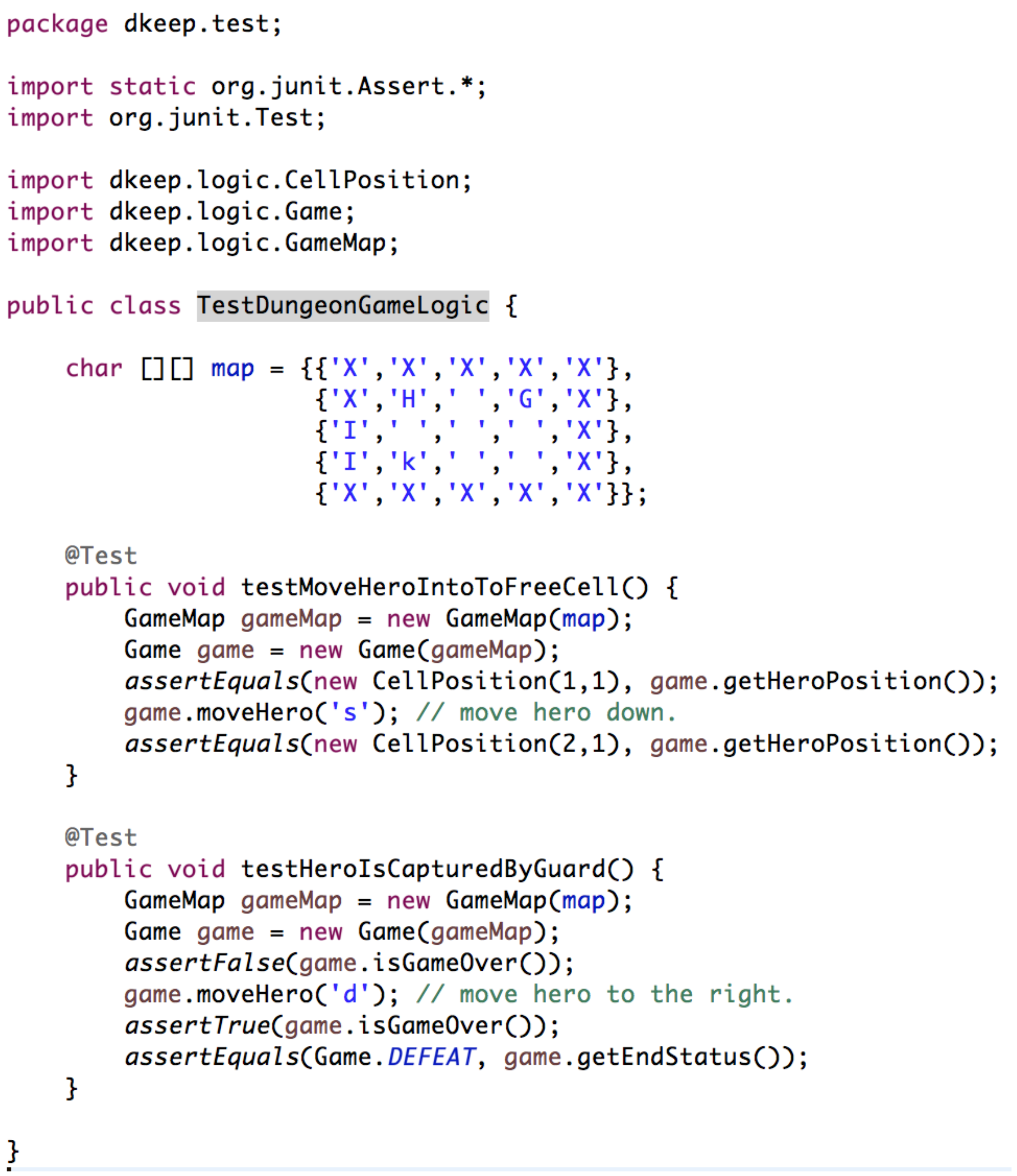
Iteration #3

A. Unit testing

- **Task #1 [at class, or at home]**. You are now going to create an automated unit test class, using JUnit 4 (as demonstrated at the lecture class) to test the game logic, at the Dungeon Level, with the Guard not moving. This class should contain several test methods to verify the correct behaviour of the game in the following situations (one method per situation):

  1. Hero moves successfully into a free cell.
  2. Hero tries to, unsuccessfully, move into a wall.
  3. Hero moves into an adjacent position to the Guard and the game ends with defeat.
  4. Hero moves towards the closed exit doors and fails to leave.
  5. Hero moves into the lever cell and the Dungeon exit doors open.
  6. Hero moves into the open Dungeon exit doors and progresses into the Keep.

  In each of these methods, use a very simple game map that allows you to test each situation in one or few game turns (Hero's movements), as depicted below (just an example):



```
package dkeep.test;

import static org.junit.Assert.*;
import org.junit.Test;

import dkeep.logic.CellPosition;
import dkeep.logic.Game;
import dkeep.logic.GameMap;

public class TestDungeonGameLogic {

    char [][] map = {{'X','X','X','X','X'},
                     {'X','H',' ','G','X'},
                     {'I',' ',' ',' ','X'},
                     {'I','k',' ',' ','X'},
                     {'X','X','X','X','X'}};

    @Test
    public void testMoveHeroIntoToFreeCell() {
        GameMap gameMap = new GameMap(map);
        Game game = new Game(gameMap);
        assertEquals(new CellPosition(1,1), game.getHeroPosition());
        game.moveHero('s'); // move hero down.
        assertEquals(new CellPosition(2,1), game.getHeroPosition());
    }

    @Test
    public void testHeroIsCapturedByGuard() {
        GameMap gameMap = new GameMap(map);
        Game game = new Game(gameMap);
        assertFalse(game.isGameOver());
        game.moveHero('d'); // move hero to the right.
        assertTrue(game.isGameOver());
        assertEquals(Game.DEFEAT, game.getEndStatus());
    }

}
```

By the end of this task, your project should have three packages: **dkeep.cli** (command line interface), **dkeep.logic** (the game logic) and **dkeep.test** (the unit testing classes)**.**

- **Task #2 [at class, or at home].** As similar to the previous task, now create a test class to verify the game logic for the Keep level, with the Ogre not moving or swinging his club. It should test the following situations:
    1. Hero moves into an adjacent position to the Ogre and the game ends with defeat.
    2. Hero moves into the Keep's exit door key cell and changes its representation to "K".
    3. Hero moves into the closed Keep's exit door, without the key, and fails to open it.
    4. Hero moves into the closed Keep's exit door, with the key, and the door opens.
    5. Hero moves into the open Keep's exit door and the game ends with victory.
- **Task #3 [at class, or at home].** Testing the Ogre's random behaviour (movement, club swinging) requires a more complex testing procedure. Therefore, create an additional test class that verifies (at each game turn, that is, Hero's movement), the expected "random" behaviour of the Ogre by checking if the next state is one of the expected, failing otherwise. Below is just a suggestion how to organize your test code:

```
@Test(timeout=1000)
public void testSomeRandomBehaviour {
   boolean outcome1 = false, outcome2 = false, ...;
   while (! outcome1 || ! outcome2 ...) {
      someAction;
      if (condition1)
         outcome1 = true;
      else if (condition2)
         outcome2 = true;
      ...
      else
         fail("Some error message");
   }
}
```

- **Task #4 [at class, or at home].** Using the **EclEmma** tool, analyse the code coverage and add more test cases, so that the code coverage of the game logic package is >= 75%.
- **Task #5 [at class, or at home].** Using the **PIT** tool, analyse the mutant's coverage and add more test cases, so that the mutant's coverage of the game logic package is >= 50%.

Última alteração: Quinta, 2 Março 2017, 13:41