The University of Queenland, School of ITEE
CSSE2002/7023 — 2018
Assignment 1 (10%)
Due: 29 Mar 18 18:00
Revision: 1.2 (identical to 1.0.1).

Abstract

The goal of this assignment is to implement a set of classes and interfaces[1] to be used in later assignments. You will implement precisely the public and protected items described in the supplied documentation (no extra members or classes). Private members will be for you to decide.
Language requirements: Java version 1.8, JUnit 4

Preamble

All work on this assignment is to be your own individual work. As detailed in Lecture 1, code supplied by course staff is acceptable but there are no other exceptions.

You are expected to be familiar with "What not to do" from Lecture 1 and `http://www.itee.uq.edu.au/itee-student-misconduct-including-plagiarism`.

If material is found to be "lacking academic merit", that material may be removed from your submission prior to marking[2].

If you have questions about what is acceptable, please ask.

Supplied material

- This task sheet
- A `.zip` file containing html documentation (javadoc) for the classes you are to write. Unzip the bundle somewhere and start with `doc/index.html`.

Tasks

1. Implement each of the classes described in the javadoc. [Note: some of them are not very complicated]
2. Write JUnit4 tests for the methods in the following classes:
   - `Treasure` as `TreasureTest`
   - `Room` as `RoomTest`

[1] From now on, I'll shorten this to "classes"
[2] No attempt will be made to repair any code breakage caused by doing this.

Marking

The 100 marks available for the assignment will be divided as follows:

Symbol	*Marks*	*Marked*	*Description*
F	55	Electronically	Implementation and functionality: Does the submission conform to the documentation?
S	25	By "humans"	Style and clarity.
J	20	Electronically	Student supplied JUnit tests: Do the tests correctly distinguish between correct and incorrect implementations?

The overall assignment mark will be $A_1 = F + S + J$ with the following adjustments:

1. If $F < 5$, then $S = 0$ and $J = 0$ and "style" will not be marked.
2. If $S > F$, then $S = F$.
3. If $J > F$, then $J = F$.

For example: $F = 22, S = 25, J = 17 \Rightarrow A_1 = 22 + 22 + 17$.

The reasoning here is not to give marks to cleanly laid out classes which do not follow the specification.

Functionality marking

The number of functionality marks given will be

$$F = \frac{\text{Tests passed}}{\text{Total number of tests}} \cdot 55$$

Each of your classes will be tested independently of the rest of your submission. Other required classes for the tests will be copied from a working version.

Functionality testing does not apply to your JUnit tests.

Style marking

As a style guide, we are adopting[3] the Google Java Style Guide
`https://google.github.io/styleguide/javaguide.html` with some modifications:

4.2 Indenting is to be **+4** chars not +2.

4.4 Column limit for us will be 80 columns.

4.5.2 First continuation is to be `+8` chars.

There is quite a lot in the guide and not all of it applies to this course (eg no copyright notices). The marks are broadly divided as follows:

Naming	5
Commenting	6
Structure and layout	10
Good OO implementation practices	4

Note that this category does involve some aesthetic judgement (and the marker's aesthetic judgement is final).

[3] There is no guarantee that code from lectures complies.

Test marking

Marks will be awarded for test sets which distinguish between correct and incorrect implementations[4]. A test class which passes everything (or fails everything) will likely get a low mark.

There will be some limitations on your tests:

1. If your tests take more than 20 seconds to run, they will be stopped and a mark of zero given.
2. Each of your test classes must be less than 300 (non-empty) lines. If not, that test will not be used.

These limits are very generous, (eg your tests shouldn't take anywhere near 20 seconds to run).

Electronic Marking

The electronic aspects of the marking will be carried out in a virtual machine or linux box. The VM will not be running windows and neither IntelliJ nor Eclipse will be involved. For this reason, it is important that you name your files correctly.

It is also critical that your code compiles. If one of your classes does not compile, you will recieve zero for any electronically derived marks for that class.

Submission

Submission is via the course blackboard area `Assessment/Ass1/Ass1 Submission`.

Your submission is to consist of a single `.zip` file with the following internal structure:

`src/`	`.java` files for classes described in the javadoc
`test/`	`.java` files for the test classes

A complete submission would look like:

```
src/CrawlException.java
src/Critter.java
src/ExitExistsException.java
src/Explorer.java
src/Lootable.java
src/Mob.java
src/NullRoomException.java
src/Room.java
src/Thing.java
src/Treasure.java
test/RoomTest.java
test/TreasureTest.java
```

Your classes must not declare themselves to be members of any package. Do not submit any other files (eg no `.class` files). Any files or directories which do not match the above structure will be deleted before marking. Remember that java filenames are case sensitive when your filesystem isn't.

Late submission

See the ECP for rules and penalties regarding late submission.

[4] And get them the right way around

Revisions

If it becomes necessary to correct or clarify the task sheet or javadoc, a new version will be issued and a course announcement will be made on blackboard. No changes will be made on or after Monday of Week 6 (ie 2018 03 26).