The University of Queenland, School of ITEE
CSSE2002/7023 — 2018
Assignment 3 (15%)
Due: 1 Jun 18 17:00
Revision: 1.0.

Abstract

In this assignment, you will take supplied code and create graphical user interfaces to use it. *You should not modify the supplied code at all.* Copying the supplied code and modifying the copy is not allowed either.

Language requirements: Java version 1.8, JavaFX 8
Do not use anything outside of the `java` or `javafx` package hierarchies.

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[1].

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

For this assignment you are expected to produce a graphical user interface which has the same appearance as shown in this task sheet (there are a number of ways to do this). However, there are some aspects which are determined by your computer's graphical environment rather than your Java program (the window title bar is the main example). See Figure 1. The only thing you need to fix in your title bar is the title text.

Supplied material

- This task sheet
- A `.zip` file containing supplied source code for you to build on. Do not modify the supplied files except to add required methods.

Required classes

CrawlGui

CrawlGui is to extend `javafx.application.Application`. When run, it expects a single command line argument which is the name of a map file to load. If this argument is missing the message "Usage: java CrawlGui mapname" is to be printed to standard error and the program will exit with status 1. If the argument is present but the map can not be loaded for some reason, "Unable to load file" is to be printed to standard error and the program will exit with status 2. (Any output to standard error should be followed by a newline).

There are three main parts at the top level.

- *Button area*: The area to the right, it should keep a fixed width when the window is resized.
- *Message area*: The area at the bottom displays text messages (It should occupy the full width of the window and resize with the window).

[1] No attempt will be made to repair any code breakage caused by doing this.

Figure 1: Window title bar.

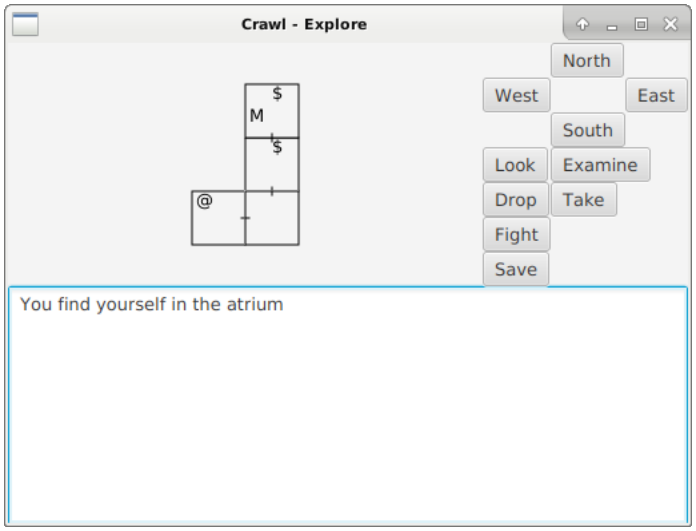

Figure 2: CrawlGUI window immediately after startup.

- *Map area*: displays the map. It should resize as the window is resized.

Cartographer

Cartographer is to extend `javafx.scene.canvas.Canvas`. This class will be responsible for rendering the "map" in the GUI. It will draw all reachable rooms starting from a "start room". (Pay particular attention to the `update()` method).

Each room is rendered as a square with small strokes into the interior at the midpoints of edges where there is an exit to an adjacent room (ignore exits which are not labelled North, South, East or West). For this assignment, you may assume that all exits are two way.

The player is to be indicated with a @ in the top left quadrant (ie the top left corner) of the room. If any treasure is present in the room, a $ is drawn in the top right quadrant. If any live critters are present in the room, a `M` is drawn in the bottom left quadrant. Any dead critters are indicated with a `m` in the bottom right quadrant.

Your cartographer must be large enough to show the entire loaded map. That is, your initial window size will vary depending on the map being loaded.

Functionality

In all of the following "displayed" means add to the message area. Whenever a list of things is to be displayed, each item is to be on a separate line. Where dialog boxes are required they will all use the same layout. Suggestion: Look at the `TextInputDialog` class and its `showAndWait` method.

Once the map has loaded successfully, it will display "You find yourself in " followed by the description of the start room.

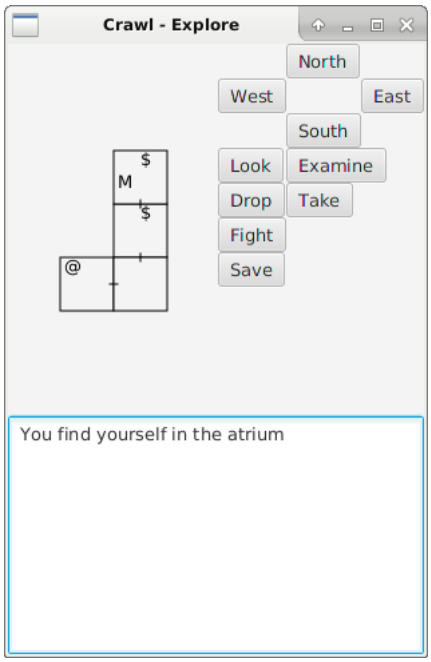

Figure 3: CrawlGUI window after being resized (to illustrate resize behaviour).

North/South/East/West buttons

If there is no exit in the specified direction, display "No door that way". If there is an exit in that direction but you can't leave, display "Something prevents you from leaving". Otherwise, move to the specified room and display "you enter " followed by the description of the room.

Look button

Display the following information:
description_of_room - you see:
followed by the short descriptions of each `Thing` in the room (add a leading space for each item). Then
`You are carrying:`
followed by the short descriptions of each `Thing` (add a leading space for each item) you are carrying. Then
"worth *total_worth_of_carried_items* in total"
(without the quotes and with the italics text replaced). Formatted for one decimal place. See Figure 4 for an example.

Examine button

Show a dialog box (see Figure 5) to get the short description of the `Thing` to examine. The first matching item will have its long description displayed. Check the player's inventory first, then if no match is found, the contents of the current room. If no match is found, display "Nothing found with that name".

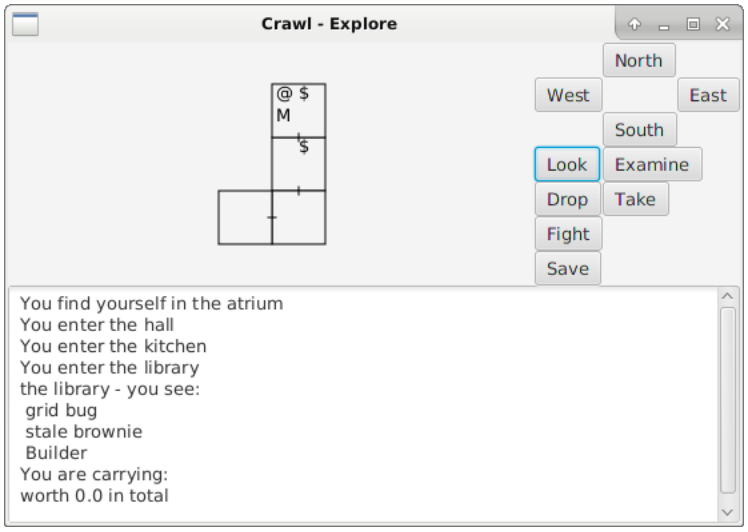

Figure 4: Output having just clicked Look

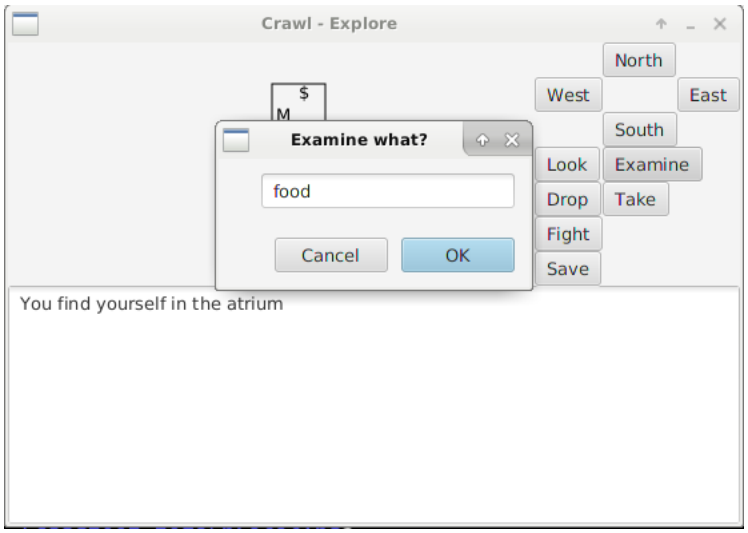

Figure 5: Dialog box to use when the Examine button is clicked (and main window)

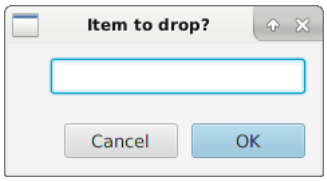

Figure 6: Dialog box to use when the Drop button is clicked

Drop button

Show a dialog box (See Figure 6) to get the short description of the item to remove from the player's inventory and add to the current room. If the player is not carrying a matching item, display "Nothing found with that name"

Take button

Similar to Drop but the dialog box is to be titled "Take what?". Objects of type `Player` should be skipped when looking for short description matches. There are additional cases where the operation will fail (silently):

- If an attempt is made to pick up a live `Mob`
- If the leave call to remove the item returns false. [Remember that, to remove an item from a room, you will need to call `leave` on the room].

Fight button

Show a dialog box (titled "Fight what?") to get the short description of a `Critter` in the current room to fight. If the fight occurs, display either "You won" or "Game over" as appropriate. (If you lose the fight, all of the buttons on the GUI should be disabled). Silent failures will occur if:

- There is no matching `Critter`.
- There is a matching `Critter` but it is not alive.

Save button

Show a dialog box (titled "Save filename?") for a file name to use with `MapIO.saveMap`. Display either "Saved" or "Unable to save" as appropriate.

Marking

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

Symbol	*Marks*	*Marked*	*Description*
F	65	Human	Implementation and functionality: Does the submission conform to this task sheet?
S	35	Human	Style and clarity.

The overall assignment mark will be $A_2 = F + S$ with the following adjustments:

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

Functionality marking

The number of functionality marks given will be awarded in three broad categories:

- Does the cartographer component render correctly? [20 Marks]
- Correctness of the visual aspects of the rest of the GUI (layout, titles etc). [15 marks]
- Do the buttons work as described in this document? [30 marks]

Style marking

As a style guide, we are adopting[2] 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.

- All private/"package private" members must be commented (you may use javadoc comments but are not required to)[3].
- Java source files must be encoded as either ASCII or UTF-8.

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	11
Readability and layout	8
Implementation practices (including OO)	11

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

[2] There is no guarantee that code from lectures, pracs and tutes complies.
[3] You must of course comment all public and protected members with Javadoc.

Submission

Submission is via the course blackboard area `Assessment/Ass3/Ass3 Submission`.
Your submission is to consist of a single `.zip` file containing a `src/` directory with all of your java files directly inside it.
Note: you must submit supplied classes as well.
Your classes must not declare themselves to be members of any package.

Remember that java filenames are case sensitive even when your filesystem isn't.

Late submission

See the ECP for rules and penalties regarding late submission.

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 2018-05-28.