Assignment

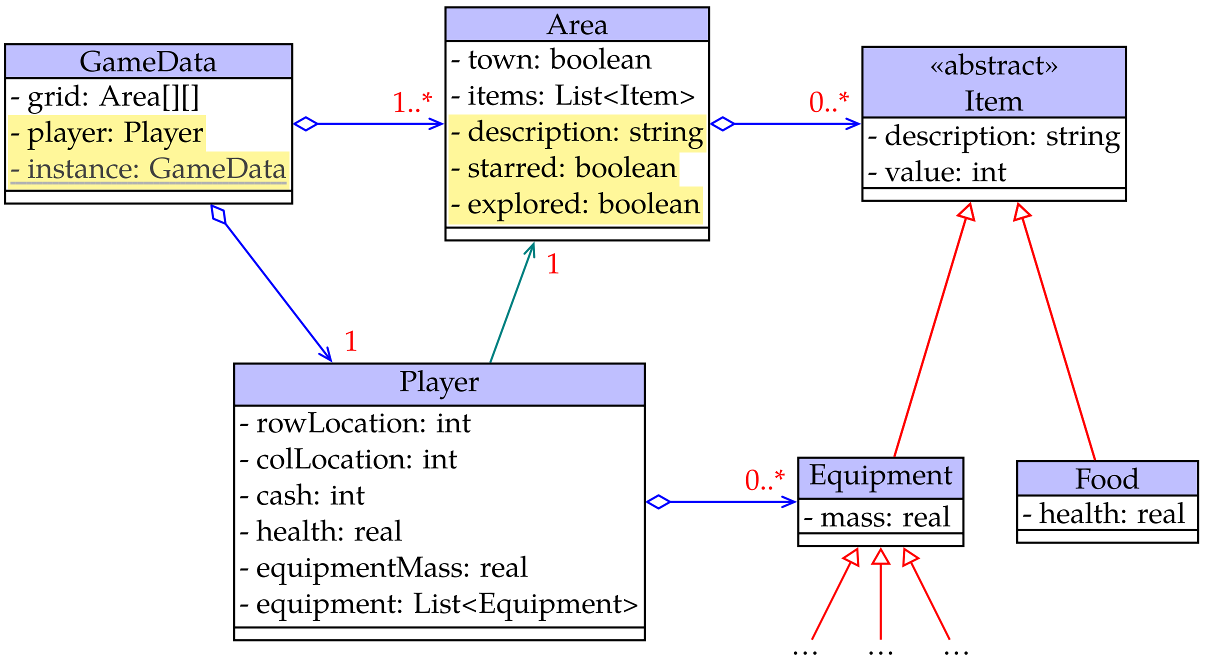
Date: Thursday 18 October, 23:59:59.

Weight: 25% of the unit mark.

In this assignment you will extend and significantly rewrite the game from worksheet 2, using concepts from worksheets 3 and 4.

1 Data Model

The data model is a slightly extended version of that from worksheet 2:



In other words:

(a) As in worksheet 2, `GameData`[1] contains a 2D array, where each element is an `Area`.

Now, it also owns the `Player` object. More importantly, it also now has a static field to keep track of an instance of itself, and should have a corresponding static method for retrieving and, if necessary, creating that instance. This is actually going to make your life somewhat easier!

[1]We actually called it `GameMap` before, but `GameData` seems more appropriate now.

(b) As in worksheet 2, each `Area` object either represents a town, or a wilderness area, and contains a set of zero or more `Items` (which can change, depending on what the player does).

Now, each area also has three additional properties:

- A player-editable description, initially blank.
- A "starred" boolean flag. By default, all areas of the grid are un-starred, and the player may choose to toggle the star on or off.
- An "explored" boolean flag. By default, all areas of the grid *except the starting location* are unexplored. Whenever the player moves to a new area, it must be set to explored.

Additionally, the map in this case must be randomly-generated. I'll leave the exact propabilities up to you, except that there must be a reasonable number of both towns and wilderness areas on the map, and the total set of items in the game should be reasonably spread out across the whole map.

(c) As in worksheet 2, each `Item` has a description (i.e. what it is), a value (price). It can either be `Food`, which increases the player's health, or a piece of `Equipment` that the player carries.

(d) As in worksheet 2, at any given point in time, the `Player` is at a particular area, has a certain amount of cash, has a health rating (0–100), and is carrying a certain amount of equipment.

Note that the diagram hints at subclasses of `Equipment`. You don't necessarily need subclasses, but this may be the most obvious thing to do. What for? Unlike in worksheet 2, some of the equipment items will be *usable*. When the player uses an item, it disappears from the player's posession, and also has a specific effect depending on what it is:

Portable Smell-O-Scope. (5kg.) When used, this displays (in a separate, newly-started activity) a list of all items up to two grid squares away from the player's current location.

Improbability Drive. ($-\pi$kg.[2]) When used, causes the entire map (all areas and the items in each area) to be randomly re-generated, except that the player keeps their current items, and their current health, cash and grid location.

Ben Kenobi. (0kg.[3]) When used in a town, you immediately acquire all items in the market without paying for them. The same happens in wilderness areas, even though this isn't terribly useful.

As in worksheet 2, distributed around the map must also be the **Jade Monkey**, the **Roadmap** and the **Ice Scraper**, which in combination will cause the player to win (but which don't otherwise do anything). There can also be any number of other food and equipment items that have no other special purpose.

Food and equipment items are not necessarily unique in the game; e.g. there could be more than one Ben Kenobi, even in the same area or in the player's possession.

[2] Improbably.
[3] He does not need to be carried by the player, as he can walk of course.

2 Database

The model, including all areas, items and player data, must be saved to a database, whenever any part of it is modified, and re-loaded when the player next launches the game. In effect, the player should be able to restart their whole device and continue playing from where they left off.

It is likely you will need *three* database tables for this: one for the areas, one for items, and a single-row table for the player.

However, this three-table arrangement is not a strict requirement, and there are different ways to approach the database design. You won't be assessed on adherance to "normal forms" as taught in Database Systems. However, be aware that there is *some* problem solving required here; e.g. how will you identify each area and item? How will you rebuild the connections between the objects in memory when the data is re-loaded?

3 Fragments and Activities

The game must consist of the original three activities (slightly modified) as well as two more activities and two fragments. You may create additional fragments at your discretion, as long as the ones here are faithfully implemented.

3.1 Status Bar Fragment

The status bar (originally simply a part of each of the worksheet 2 activities) must be converted into a fragment and re-used across all activities.

As before, it must show the player's cash, health, and equipment mass, as well as a message indicating whether the player has won or lost. It must also contain a "Restart" button that restarts the game, resetting all data to initial (and if necessary, randomly-generated) values.

3.2 Area Info Fragment

The centre display in the navigation activity must be converted into a fragment, to be used in the navigation activity itself, and also in the new "overview" activity, described below. It should now contain:

- The string "Town" or "Wilderness" as appropriate.
- A UI element to display and allow the user to edit the area's description.
- A UI element to toggle the area's starred status.

(Note that it *should not* contain a button for launching the market or wilderness activities. That should continue to be part of the navigation activity itself.)

3.3 Navigation Activity

In the main "navigation" activity (based on worksheet 2), we show where the player is on the grid; i.e. which `Area` object we're at. We only show the current area / grid square, not the whole grid at once here.

The game must show four buttons around the edge of the screen: "North" (at the top), "South" (at the bottom), "East" (on the right), and "West" (on the left). The player can use these buttons to move from one area to an adjacent another. However, the player cannot move off the edge of the map.

Travelling one square reduces the player's health, as follows (the same as in worksheet 2):

```
health = Math.max(0.0, health - 5.0 - (equipmentMass / 2.0))
```

If the player's health reaches zero, the game ends.

The centre of the screen is somewhat modified since worksheet 2. It should contain:

- The "area info" fragment.
- A button labelled "Options", used to launch the market or wilderness activity as appropriate.
- A button labelled "Overview", used to launch a new activity described below.

3.4 Market Activity

As in worksheet 2, if the current area is a town, the "Options" button will start the market activity, showing all the items that might be bought or sold.

Now, this must incorporate the status bar fragment. It must also be implemented using `RecyclerView` (rather than, for instance, hard-coding the items or having "Next" or "Previous" buttons). There are several possible ways to use a `RecyclerView` list for this purpose. You could have a single unified list showing all items, each with either a "Buy" or "Sell" button. You could alternatively have two lists, one for buying and one for selling.

As in worksheet 2, you will also need a "Leave" button to return to the navigation activity. The buying and selling works the same way as in worksheet 2 as well:

Buying items: The items available to be bought are those contained in the relevant `Area`. As you would expect, purchasing an item can only be done if the player has enough `cash`, and will subtract the item's value from the `cash` amount, and remove it from the `Area`.

If the `Item` is an instance of `Food`, the player will eat it, and `health` will increase by the specified amount (up to a maximum of 100.0). Food could also be poisonous, and decrease the player's health. They could lose the game at this point if their health drops to zero.

If the `Item` is an instance of `Equipment`, the player will add it to their inventory, and `equipmentMass` will increase by the specified amount (with no specific upper limit). Also, if the item being purchased is the last of the three being sought, the player has won the game. The game should return to the navigation activity, which should display a victory message in the status bar.

Selling equipment: The player may sell any equipment in their inventory. Selling an equipment item is always possible. However, the player will only receive 75% of the item's value. (The player cannot sell food, as that never gets added to the player's inventory in the first place.)

Selling an item will also cause it to be added to the relevant `Area` object, and will reduce the equipment mass being carried by the player.

Using equipment: The player may alternatively *use* certain equipment in their inventory. Not all equipment items can necessarily be used, but for any that can, there should be a "Use" button alongside it in the list.

3.5 Wilderness Activity

If the current area is not a town, the options button will instead start the wilderness activity. This will actually work much like the market, except that there is no cash involved. That is, the player can simply pick up any items present, and/or drop any from their inventory. Usable equipment items work the same way here.

3.6 Overview Activity

The new "overview" activity will be launched from the navigation activity via the "overview" button.

It must incorporate both the status bar and area info fragments. But most importantly, it must display a graphical view of the whole map grid, as follows:

- Each unexplored area must be shown in black (or, if you like, a particular graphic indicating that it is unexplored).
- Each explored area must have a graphic indicating whether it is a town or wilderness area. (You can reuse the vector images from worksheet 3 if you like; e.g. a house representing a town, and open grass or trees representing a wilderness area. However, feel free to use any other appropriate graphics instead.)
- The player's current location must be indicated on the grid.
- All starred areas must be indicated on the grid. (You don't absolutely need to use an actual star, but it would be nice!)

You can use either a horizontally or vertically scrolling `RecyclerView` (as in worksheet 3). Alternatively, if you like, you can implement a two-way scrolling `GridView`.

Clicking/tapping on a grid cell must cause that area's information to be displayed in the area info fragment, where it can be edited.

There must also have a "Leave" button (like the market/wilderness activities) that cause it to end and revert back to the navigation activity.

3.7 Smell-O-Scope Activity

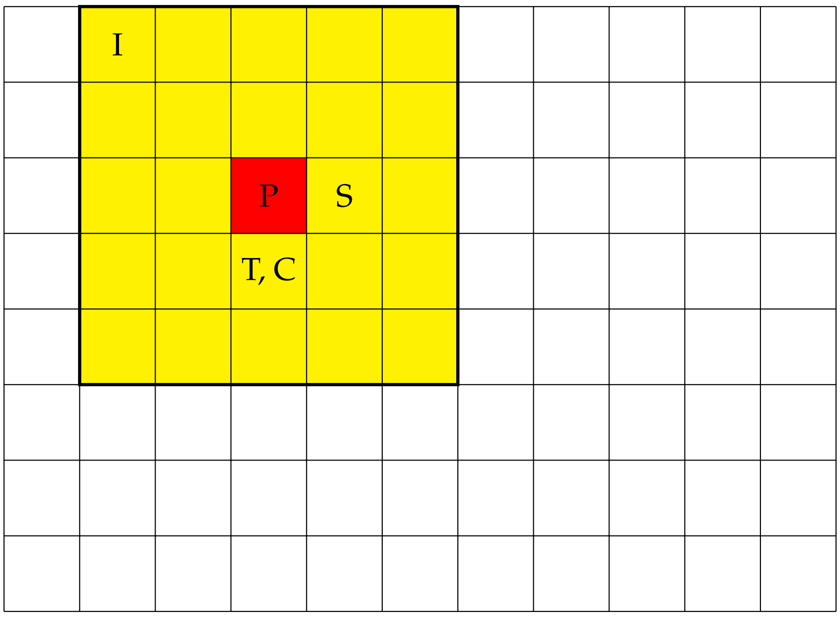
When the player uses the special Smell-O-Scope item, the game must start up an activity showing a list of all the items up to two grid squares away from the player, horizontally and/or vertically. It may or may not display the items at the current location, at your discretion.

The list should contain the name of each item, and its location relative to the player. For instance:

Sandwich	1 east	0 north
Ice Scraper	2 west	2 north
Tesseract	0 east	1 south
Curry Puff	0 east	1 south

This list is be purely to display information. It does not need updating, and should not provide any editing functionality. There must be a "Leave" button that causes the activity to close.

To help visualise what's going on here, imagine that the following is the whole map grid, showing the player's current location in red and several items within range (yellow) of the Smell-O-Scope:



P = Player location
S = Sandwich
I = Ice Scraper
T = Tesseract
C = Curry Puff

(Note: this *isn't* what the game itself should display. It's just to establish how the Smell-O-Scope works.)

4 Submission

Submit your assignment electronically, via Blackboard, before the deadline. To submit, do the following:

(a) Fill out and sign a declaration of originality. A photo, scan or electronically-filled out form is fine. Whatever you do, ensure the form is complete and readable! Place it (as a .pdf, .jpg or .png) inside your project directory.

(b) Zip up your entire project directory as-is (as a .zip or .tar.gz file). Leave nothing out.

(c) Submit your zip/tar.gz file to the assignment area on Blackboard.

(d) Re-download, open, and run your submitted work to ensure it has been submitted correctly.

You are responsible for ensuring that your submission is correct and not corrupted. You may make multiple submissions, but only your newest submission will be marked. The late submission policy (see the Unit Outline) will be strictly enforced.

Please note:

- DO NOT use WinRar.
- DO NOT have nested zip/tar.gz files. One is enough!
- DO NOT try to email your submission as an attachment. Curtin's email filters are configured to *silently discard* emails with potentially executable attachments.

In an emergency, if you cannot upload your work to Blackboard, please instead upload it to Google Drive, Github, or an equivalent online service that *preserves timestamps*.

5 Demonstration

You will be required to demonstrate and discuss your application with a marker in-person. This may include, for instance:

- Rebuilding and running your application.
- Answering questions about any aspect of your submission.

You must either be available during the practical sessions in week 14, or schedule an alternate time (in agreement with the marker/lecturer) *before* the start of the exam weeks.

6 Academic Integrity

This is an assessable task. If you use someone else's work, or obtain someone else's assistance, to help complete part of the assignment that is intended for you to complete yourself, you will have compromised the assessment. You will not receive marks for any parts of your submission that are not your own original work.

Further, if you do not *reference* any external sources that you use, you are committing plagiarism and/or collusion, and penalties for Academic Misconduct may apply.

The unit coordinator may require you to provide an oral justification of, or to answer questions about, any piece of written work submitted in this unit. Your response(s) may be referred to as evidence in an Academic Misconduct inquiry.

End of Assignment