
IJP Assignment 2

Introduction to Java Programming: 2018-2019



Please read this entire document so that you have a clear idea of what is involved in the various stages and what you should do when, before starting work on any of the tasks.

In the first assignment, you started from a “skeleton” application which we had provided. In this assignment, you will complete an entire application yourself - including the design of the class model, and the user interface. We would expect a good solution to be possible with about three days work (24 hours), although you will probably need to spend longer if you are aiming for a particularly good mark, or don’t have a lot of previous experience.

Don’t be intimidated if you don’t have much previous programming experience – you do not have to complete all of the sections, and you may be surprised at how much you can achieve by working steadily and following the instructions carefully.

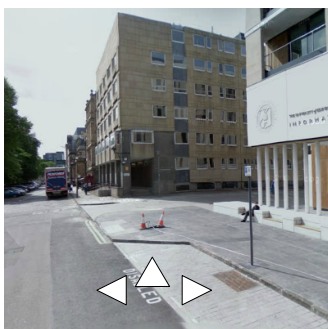
The design of the classes is the most important part of this assignment – you will need to have a good class model to pass, regardless of how well your application may appear to run. Of course, this is motivated by real life – large, reliable applications are impossible without a good design.

To help you with this, the assignment is split into stages: in the first stage (**part A**), you will need to submit an initial design. This will then be made available to the other students, and you will have a chance to see and discuss other people’s designs – and possibly change your mind as a result – before starting on your implementation (**part B**). The two parts have separate deadlines (see the schedule for **submission A** and **submission B**).

As with the previous assignment, tasks that you will need to do are marked with a , and harder, optional tasks are marked as  **harder**.

1 The Application

For this assignment, you will develop an application which allows the user to move around and manipulate objects in a very simple “virtual world”. The images will be two-dimensional, but the user should be able to move from one location to another and to look in different directions - similar to Google “Street View”¹, but with discrete images. In addition to the skills that you developed in the first assignment, this will give you some experience with designing object-oriented solutions, using graphical user interfaces, and working with large libraries of external code.



¹See: <http://maps.google.com/help/maps/streetview/>

1.1 Locations

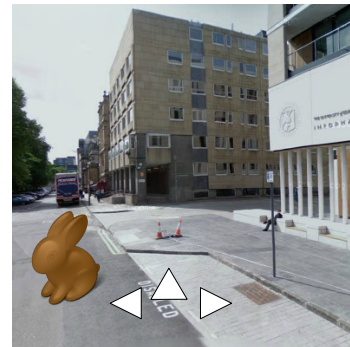
- [1] You will need to start by choosing a “theme” for your world. This should consist of small number of connected “locations” and a set of images for each location which represent views in different directions. For example:
- The locations might be the rooms in your flat. You would need to have a set of images for each room, looking in different directions, and a “logical map” of the flat which shows how the rooms are connected.
 - The locations might be an area around the University. Again you would need a set of photographs looking in different directions, and a logical map which shows how you can move from one location to another.
 - You might create a completely fictitious environment, perhaps with hand-drawn locations.
 - You won’t get extra marks for “imagination”, but creative scenarios are welcome!

If you decide to use images from elsewhere, make sure that you acknowledge the source of the images, and that you do not violate any copyright restrictions.

- [2] Now think about the basic actions that the user will be allowed to perform - as a minimum, the user probably needs to be able to turn around (left or right) and move forward into the next location². This requires some thought. What does “turn” mean? 90 degrees? What happens if there are two doors in the same wall - how do you distinguish between the directions? What happens if the user attempts to go forward when there is no exit? Is there an “up” and “down” as well?
- [3] Now, think about the interface. It is useful to sketch on paper what you think this might look like. How will the user specify the commands? Buttons? Typing commands? Menu items? How do you prevent the user from taking impossible actions? How should the program react if they do?

1.2 Portable Items

Now that you are able to move about in your virtual world, let’s add some items that you can carry from one place to another. For example, one location might contain a chocolate rabbit³. We should be able to pick up the rabbit, move to a different location and put it down again. Providing that the rabbit doesn’t melt or get eaten, it should stay in the new location and be visible every time we return there. Of course, we should be able to pick it up again and move it somewhere else.



- [4] Extend your interface design so that it displays any items which are present in the current location (whichever way we are facing). You might want to display your items overlaid on the main image, or in a separate section of the display, for example.
- [5] Think about the additional actions that the user will need to perform, such as picking up and putting down items.
- [6] Extend your interface design to support these actions. Use a different interface element for this interaction - for example, if you used buttons to move between the locations, you might use a

²Note that “turning” and always moving forward is much clearer than allowing the user to “move left” and “move right” (in which case it might not be clear whether they are looking in the same direction that they are moving!)

³Rabbit icon from: http://www.iconfinder.com/icondetails/67005/128/_icon

menu, or a text command to select the item, or you may simply allow the user to click on the item. Make sure that your design is capable of supporting an arbitrary number of items.

2 Designing the Model (Part A)



The design of the class model is the most important part of this assignment – submission of a design document describing a good, well-reasoned choice of classes and methods is essential in order to pass. Your design will be assessed both on your initial design (part A), and on any changes that you decide to make for the final implementation (part B).

To model your world, you need to think about the design of the classes and how they would be used. For example, you may decide to have a class which models a `Location`. This would include a collection of images for the different views. What methods would be required? What are the different possibilities for storing the collection of images? Similarly, you may want to have a class which models the `World`. What methods would be required? How would this relate to the locations?

In practice, having a good design will make the coding much easier; a bad design will make it much harder. So, you will want to think carefully about the possible alternatives - consider how easy it would be to implement each of the actions you identified in tasks [2] and [5] using various different models. *Read the appropriate book chapters.* Think about *nouns* and *verbs*. Consider properties such as *cohesion* and *coupling* – how easy would it be for other people to use your classes in their own application?

In addition to the more concrete entities, such as locations and portable items, you will need to think carefully about the controller and the GUI. What methods should they support? Separating these clearly allows you to easily change the way in which the interface works - for example to change from a menu-based action to a button action. It also makes the system easier to test, and for multiple people to work on the same application. You may want to look carefully at the classes from the first assignment and how they are structured – you should notice some useful similarities.

Thinking about the following questions may help you to evaluate your own design - *you do not actually have to do any of these tasks* - just think about how easy, or hard they may be:

- ☐ How easy would it be to replace the JavaFx with some other graphical interface? What proportion of your classes depend on JavaFx? Could you replace the graphical interface with a text-based one, without changing most of your classes?
- ☐ How easy would it be to test your classes? Would it be possible to write a test to show that all of the locations are accessible from the starting point (possibly via other locations)?
- ☐ How easy would it be for someone else to add a different kind of “item” to your application (perhaps one which included sound). How many of your classes would be affected by this change?
- ☐ Rather than retrieving the images from the local filesystem, could you fetch them from the web? Could you make use of any of the services from the first assignment to do this? You probably don’t use the same `Picture` class as the services, but could you create an “adaptor” class which translated the `Picture` objects into the format that you use?
- ☐ If you were fetching the pictures from the web, would any of the proxy classes (`CacheProxy`, `RetryProxy`?) from the first assignment be helpful?
- ☐ How easy would it be for someone to create a game out of your application – for example, displaying a score to show how many items had been collected?


Such extensions should not require big changes to your code, and should only involve a small percentage


of your classes.

3 Submitting the Design (Part A)

For Part A of the assignment, you should submit a short document describing your design. There are various ways of doing this formally⁴, but the main aim is to show what classes you have, what their responsibilities are, and how they relate. So a simple list of classes and methods, with brief descriptions, and perhaps an informal diagram, is sufficient in this case. You should also *briefly* describe one or two significant choices that you had to make in the model design and explain why you chose your particular design over some plausible alternative. You should aim to be extremely clear: it is probably best to avoid lengthy text descriptions, and take special care if English is not your first language. You may create the design document using any application that you like, but please make sure that it is in PDF format – the [FAQ](#) pages on the course web site explain [how to generate PDF](#) – and make sure that it has no more than about two pages.

After submission, these documents will be made available to the other students on the course so that everyone can discuss and compare designs. If you include your name on the design document, it will be visible to other students - this is obviously helpful and other students will then be able to acknowledge your design, but you may choose to remain anonymous if you wish by not including your name in the document.


 [7] Create your design document as `design.pdf`.


 [8] Create a ZIP archive containing this single document. The [FAQ](#) pages on the course web site explain [how to create a zip archive](#). You may want to unpack the archive again yourself to verify that it contains the expected file.

 [9] Use the [online submission system](#) to submit your ZIP file.

As for the first assignment, it is a good idea to attempt your submission well before the deadline to allow for any problems that you may have with the submission system. If you need to update your submission (anytime before the deadline), you can simply make a new submission which will replace the previous one. Similarly, *there will no extensions* because the designs will be made available to the other students once they have been submitted.

Once the designs are submitted, they will be made available online. You are then free to browse these, and to discuss them with other students. It is expected that you will want to make changes after seeing the other designs, so you are not committed to your initial design for your final application, although you will be expected to explain and justify any changes which you make.

 [10] Browse and discuss the designs with other students and demonstrators.

 [11] Consider modifying your original design in response to this.

⁴For example, UML: https://en.wikipedia.org/wiki/Unified_Modeling_Language

4 Implementation (Part B)



Especially if you have programmed before in some other language, *do not underestimate the importance of having a good, clear model before starting your implementation* – if the model is good, the implementation should be straightforward; if not, the implementation is likely to be awkward and difficult. So make sure that you think hard about the design, and beware of committing to an implementation before you have had a chance to compare some other designs.

We suggest that you start by implementing (only) those classes which are necessary to create the basic application (moving between locations), and leave the portable items until the basics are working. If you don't have a lot of experience, you may find it difficult to implement the portable items, and it is possible to pass without doing this, providing that you have a sound design, and a working implementation of the basic locations.


In your implementation, you may find it useful to use or modify code from the first assignment (including the supplied code), or the JavaFx examples, or code from elsewhere. Of course, you will only be credited for code which you have written yourself, and you must make sure that you clearly state the source of any code which is not your own - you should do this both in the code (comments), and on the worksheet [6:20]. If you do use external libraries, make sure that these are included in your submitted jar file.

You will find it helpful to adopt a standard size for the main images (say 512 x 384) and the item images (say 64 x 64). This will make it much easier to share your model with others (see section 5). There is also a limit on the size of the assignment file which you can submit, and *you will not be able to submit your assignment if it contains unnecessarily large image files*. In addition, large images may also cause you problems with memory allocation in your program – there are ways of dealing with this, but keeping the images sizes small will avoid these problems. The (FAQ) explains how to pre-scale and compress your images.

 [12] Implement the basic model classes.



In a real project, this would be a good time to write some unit tests - this would allow you to convince yourself that the model objects were working properly before you get involved in the complications of connecting them to the interface. Unless you are finding the assignment easy though, you probably want to move on and concentrate first on the following sections.

 [13] Implement the GUI. *You must use JavaFx for the interface, and create your design using SceneBuilder.*

You should now be able to check the appearance and some basic behaviour of the interface without writing any additional code. The [Video page](#) on the website has some videos and sample code to help you get started with JavaFx. This is probably new to most of you. This is intentional. It is deliberately intended to give you a realistic experience of working with an unfamiliar library using only the public documentation and support available on the web.

 [14] Implement the controller.

You should now have a basic working application. You should save a copy of the code at this stage – this will give you a working copy to demonstrate for marking if you break things while attempting the following sections!



For a real project, you would almost certainly want to use a *Version control* tool⁵. This would allow you to save the state of your code every time that you make a change. This means that you can easily return to previous versions, and you can clearly see what was changed, and when. These tools are also particularly valuable in tracking and merging changes when multiple people are working on the same project. They do take some time to learn but if you intend to continue programming, you will almost certainly find this worthwhile at some stage.

- [15] Once you have a working application which allows you to move between locations, you should extend this to include the portable items.

5 An Advanced (Optional) Task: Loading the Model

harder This section is intended for those of you with more programming experience who would like to learn a little more about using Java in practical applications - specifically locating and using external libraries, and reading structured data in JSON format.



It is possible to obtain a very good mark without attempting this section, and if it is implemented badly, it can spoil a good design of the basic program which may actually result in a *lower* mark! So, only attempt this if you are confident that you have a solid solution to all of the other parts of the assignment, and you are willing to spend additional time. You should save a copy of the code at this stage, so that you have a working copy to submit if you later decide that the extensions have resulted in a poorer class design, for example.

You will hopefully have realised from the first assignment, that it is useful to separate the “content” from the code: rather than “hard-wiring” the Munro names into the controller code, they could be stored in a separate property file. This allowed other people to easily change the data without understanding, or re-compiling your code. It also allowed the same data to be used with different implementations, simply by copying the property file.

In this case, our model is a little more complicated. It *would* be possible to represent this in a way which could be stored in a property file, but this would be rather awkward to deal with, and difficult for other applications to read. JSON⁶ is a standard way of representing structured data in plain text files. JSON libraries are available for most languages, so this makes it a convenient format to represent our model.

Figure 1 shows how a simple model might be represented in JSON⁷. This contains a number of rooms (bedroom, hall, ...). Each room specifies a collection of image files representing the view in different directions, from inside that room. It also has a number of exits (in different directions) leading to other rooms. And it has a list of items initially present in that room. The portable item records just define the image file representing each item.

- [16] **harder** Explore some of the JSON libraries which are available for Java and choose an appropriate one for your application. For example, simplicity is probably a useful criteria in this case, but high performance is probably not.
- [17] **harder** Modify your application so that it reads the model from a JSON file in the same format as the one in figure 1.

If your application design is good, it should be possible to do this without major changes to the class

⁵For example GIT: <http://git-scm.com>

⁶<http://www.json.org/>, <https://en.wikipedia.org/wiki/JSON>

⁷There are, of course, many other possible representations

```

{
  "rooms": {
    "bedroom": {
      "views": {
        "north": "bedroom-north.jpg",
        "west": "bedroom-west.jpg",
        ...
      },
      "exits": {
        "north": "bathroom",
        "south": "hall"
      },
      "contents": [ "cat", "dog" ]
    },
    "hall": {
      ...
    },
    ...
  },
  "items": {
    "cat": { "image": "cat.jpg" },
    "dog": { "image": "dog.jpg" }
  }
}


```

Figure 1: Representing the model in JSON.

structure. You may find the following hints helpful:

- ❑ The JSON entries (for example the exits) reference the rooms by their names. You will need some way of looking up the names to find the corresponding objects. And you will need to think whether it is better to store the names of the rooms (and look up the objects when you need them), or store the references to the objects themselves.
- ❑ Delegate the responsibility for interpreting the various parts of the JSON to the appropriate classes.


Your application should provide some method for the user to select different JSON files (and hence different models) - perhaps from a menu, or from a prompt when the application starts. If this is to be useful for sharing models, you will need to be careful about the location of the image files - you should assume that these will be in the same directory as the JSON file.


-  [18] See if you can find someone else who is attempting this section (try the Piazza forum), and exchange your model with theirs (do not exchange any code). Your applications should be able to display each other's model.

You might find it fun to try and merge two models: you should easily be able to create a composite JSON file which allows you to move between different worlds (via specific exits) in the same application!

6 Submitting the Implementation (Part B)







As well as submitting your code, you will also need to submit a worksheet containing the following:

-  [19] A *brief* description of any changes you made to your design after looking at the other solutions. If you did not make any changes, then you should write a brief justification of this.


-  [20] Any extra comments that you would like to make about your solution, or the assignment in general. In particular, you should note any external resources from which you took code, and any significant collaborations with others.

As before, you should probably attempt your submission before the deadline to allow for any problems that you may have with the submission system, and *there will no extensions* because answers to assignment questions and feedback will be made available during the demonstration sessions in the following week.

Before you submit your assignment ...

-  [21] You may want to test your jar file on a different platform – for example, test it on DICE if you have developed on a Windows machine (or visa-versa). It should be possible to run your application simply by (double) clicking on the jar file. If your code is not portable and does not run on the markers machine, then we will rely on your demonstration for clarification..
-  [22] Look at the course web page on [code readability](#) and use this to review the readability of your code.
-  [23] Please make sure that you understand about good scholarly practice – see the [information on the course website](#) for further details. For this assignment, we will be using automated tools for checking both worksheets and code for potential plagiarism. If you are in any doubt about any part of your submission, please discuss this with the course lecturer.
-  [24] Generate a few screen dumps showing various screens from your application. These will show the marker what your application looks like if (s)he is unable to run your code.
-  [25] Create a ZIP archive containing the following (please use these exact filenames):
- ☐ `src` - a directory containing all of the source files necessary to compile your application.
 - ☐ `screendumps` - a directory containing images of a few screen dumps from your running application.
 - ☐ `worksheet.pdf` - your answers to the worksheet questions.
 - ☐ `assignment2.jar` - the jar file containing your runnable application.
-  [26] Use the [online submission system](#) to submit your ZIP file.

7 That's (almost) It

-  [27] In the final lab session, you will be asked to show your running application to a demonstrator and a small group of other students. This will give you an opportunity to explain any problems that you had with the code and to get some feedback on your solution. This is difficult to schedule, so please make sure that you attend this lab session on time, and that you have all of the necessary files to run and demonstrate the same version of your code as the one you submitted (we will check this).

The demonstrator will not be marking your assignment, but we will refer to the demonstrator's notes if, for example, we cannot run your submitted assignment code on our systems. If you do not attend the demonstration, code which we are unable to run (for any reason) will not be awarded any marks.

8 That's (really) It!

We hope that you found this a useful and realistic exercise. Please do let us know what you think. Ask the lab demonstrators if you would like any further feedback, or use the forum to discuss any issues

relating to this exercise.