IJP Assignment 2

Introduction to Java Programming: 2016-2017

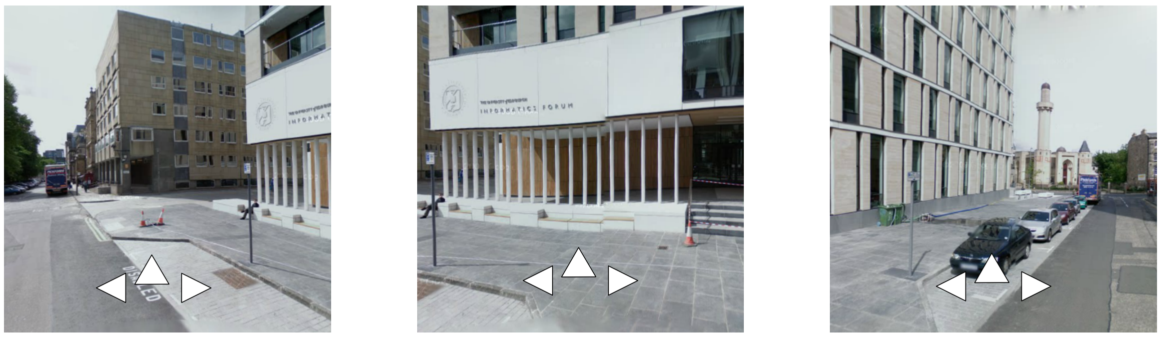
In the first assignment, you started from a “skeleton” application which we had provided. In this assignment, you will complete an entire application yourself - including the design of the class model, and the user interface. We would expect a good solution to be possible with about three days work (24 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 programming experience. However, it is important that you start thinking about the design in advance – you will need time to develop your design and discuss it with the demonstrators. You should read the relevant book chapters, watch the videos, and have a clear idea of your class structure before writing a lot of code.

- ❑ For a pass mark, you will need a good object-oriented design and you will need to describe this clearly on the worksheet. A working application alone will not be sufficient.
- ❑ The assignment includes some optional tasks for those of you with more experience. You should only attempt these if you are confident that you have a very good solution to the basic assignment.
- ❑ Don’t be intimidated by the more advanced tasks if you don’t have a lot of experience. It is possible to get a good mark without attempting sections 4 or 5, and it is better to concentrate on developing a good understanding of the basics.

As with the previous assignment, tasks that you will need to do are marked with a ✏, and you should read the marking scheme carefully to understand how these will be assessed.

1 The Application Design

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.



✏ [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:

[1]See: http://maps.google.com/help/maps/streetview/

- ❑ 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.
- ❑ Be imaginative! The `Assignments` menu on the course website shows some examples of student work from previous years.

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 location[2]. 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?

[4] Finally, think about the controller. What actions will it need to perform? Separating these from the GUI 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.

2 Creating the Model

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?

The design of the model is important – a good, well-reasoned choice of classes is necessary for a good mark. In practice, having a good design will make the coding much easier; a bad design will make it much harder. So, you will almost certainly find it useful to think about several possible alternatives - consider how easy it would be to implement each of the actions you identified in task [2] 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? I am hoping to produce some additional videos to help you with this, but these will not be available immediately.

STOP Before you start coding: Make sure that you are clear about your proposed design, and how you would use it to implement all of the necessary actions. A simple list of the proposed classes and their methods

[2] Note 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!)

will probably be sufficient. Before implementing the model, you will probably want to look ahead at the tasks in sections 4 and 5. If you are intending to attempt these tasks, then you may want to change your basic design to accommodate these, before you start coding.

✐ [5] Implement your model objects. You may use/modify code from the first assignment (including the supplied code), or the JavaFx examples if you find this a useful starting point. But make sure that you clearly state the source of any code which is not your own.

If you have high-resolution images, you will need to pre-scale them to make them smaller (See the FAQ). There is 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*. Large images may also cause you problems with memory allocation in your program – there are ways of dealing with this, but you probably want to avoid these extra complications.

ⓘ 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.

3 The Interface & Controller

✐ [6] Now implement the GUI that you designed in task [3]. You must use JavaFx for the interface, and create your design using SceneBuilder. You should then 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[3].

✐ [7] Implement the controller that you designed in task [4].

✐ [8] 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!

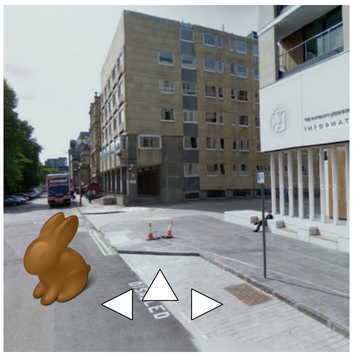
ⓘ For a real project, you would almost certainly want to use a *Version control* tool[4]. This would allow you to save the state of your code every time that you make a change. This means that you can easily return to previous versions, and you can clearly see what was changed, and when. These tools 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.

[3]JavaFx is probably new to most of you. This is deliberately intended to give you a realistic experience of working with an unfamiliar library using only the documentation and support available on the web.

[4]For example GIT: `http://git-scm.com`

4 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 rabbit[5]. 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.



[9] Extend your model to support some "portable" items. Make sure that your model is capable of supporting more than one item.

[10] Extend your interface 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.

[11] Extend your interface so that the user can pick up and put down the items. Use a different interface element for this interaction - for example, if you used buttons to move between the locations, you might use a menu, or a text command to select the item.

[12] You should now have a working application which allows the user to move items between locations.

5 Optional Extensions

If you have significant previous programming experience, and are finding this assignment fairly easy, you may like to try implementing one, or more of these advanced features. These are designed to give you an introduction to handling some of the typical problems that you may come across in a real application: image handling, persistent data, and networking.

It is possible to obtain a distinction without attempting these, and if they are implemented badly, it can spoil a good design of the basic program which may actually result in a *lower* mark! So, only attempt these 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 this will give you a working copy to submit if you later decide that the extensions have resulted in a poorer class design.

[13] Rather than having a number of separate images for the views from each location, create a single panorama image and allow the user to scroll continuously around each location, in a similar way to Google StreetView.

[14] Modify your application so that the entire model of the world is read from a datafile, rather then being "hardwired" into the code[6]. You should be able to change the portable items, as well as the

[5]Rabbit icon from: http://www.iconfinder.com/icondetails/67005/128/_icon
[6]Think how the property files were used in the first assignment.

images and relationships of the locations without recompiling the code. Use some standard format such as XML or JSON for the datafile, and a third-pary library to manage the loading of the data. If you can find another student who is prepared to use the same data format, then you should be able to use each other's "worlds" in you own application.

✐ [15] Modify your code to load the images, and the datafile (if you are using one) from a remote web server. This should allow anyone to create a "world" which your application can use, simply by adding the images and datafile to their web server. Think carefully about how to handle latency and errors in the remote service.

If you choose to attempt any of these, make sure that you do not compromise the clear design of the basic application and the description on your worksheet.

6 Worksheet & Documentation

As well as submitting your code, you will also need to submit a worksheet with answers to a few questions. A pass mark requires a clear and well-written discussion of your model on the worksheet (see the marking scheme), so please pay particular attention to this, especially if English is not your native language. You may create the worksheet 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 800 words:

✐ [16] Make sure that your name and matriculation number are clearly visible at the top of the worksheet.

✐ [17] Answer the following questions on your worksheet. Please number the answers clearly:

1. *Briefly* describe your class design, including the function and relationships between the important classes. You may do this using text and/or diagrams, but the result must be clear and concise.
2. *Briefly* describe one or two significant choices that you had to make in the model design and explain why you chose your design over some plausible alternative.
3. Add 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.
   - ❑ It would also be useful if you could indicate here if you would be prepared to allow us to publish your screenshots – many students produce interesting interfaces and images and it is useful to be able to show these to others – for example, the course website shows some previous implementations.

7 Submission

You should 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. *No marks will be awarded for submissions received after the deadline*[7].

[7] See the School policy: `http://edin.ac/18DwiRZ`

Before you submit your assignment ...

✐ [18] 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 to verify the stage that your code has reached.

✐ [19] 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 suspected plagiarism. If you are in any doubt about any part of your submission, please discuss this with the course lecturer.

✐ [20] Generate a few screen dumps showing various screens from your application.

✐ [21] Create a ZIP archive (see the submission page on the website) containing the following files (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.

The FAQ pages on the course web site explain how to create a zip archive. Be careful: you will not be credited for missing files, so you may want to unpack the archive again yourself to verify that it contains all of the files that you expect.

✐ [22] Use the online submission system (described on the website) to submit your ZIP file.

8 That's It!

During one of the lab sessions, you will be asked to show your solution to the demonstrator and a small group of students. This is a valuable opportunity to get feedback and to see potentially different approaches to tackling the same problem. The demonstrator will also be asked to check that your code runs as expected, and give you chance to explain the code, and any problems that it may have.

✐ [23] You must attend your scheduled 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*[8].

✐ [24] You may be given a small additional task to complete during this session which will be part of the assessment.

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.

[8] If you cannot demonstrate the same version of your code as the one you submitted (we can check this), then your demonstration will be discounted for the assessment.