Software Engineering Large Practical

Stephen Gilmore (Stephen.Gilmore@ed.ac.uk)
School of Informatics

Document version 3.0.
Issued on: November 28, 2016

About

The Software Engineering Large Practical is a 20 point Level 9 course which is available for Year 3 undergraduate Informatics students including those on joint degrees. It is not available to visiting undergraduate students or students in Year 4 or Year 5 of their undergraduate studies. It is not available to postgraduate students. Year 4, Year 5 and postgraduate students have other practical courses which are provided for them.

Scope

The Software Engineering Large Practical consists of one large design and implementation project, done in three parts. The first part consists of a proposal document specifying functional and non-functional requirements on the project. The second part is a design document, presenting the plan of the implementation work which will realise the design. The third part is the implementation. This should be a well-engineered implementation of the previously-supplied design.

Part of the SELP	Deadline	Out of	Weight
Part 1 (Proposal document)	16:00 on Friday 14th October	0/1	0%
Part 2 (Design document)	~~16:00 on Friday 11th November~~ 16:00 on Monday 14th November	100	50%
Part 3 (Implementation)	16:00 on Wednesday 21st December	100	50%

Please note that Part 1 of this practical is for *feedback only*. Parts 2 and 3 are equally weighted and constitute the assessment for the Software Engineering Large Practical. There is no exam paper for this course.

Introduction

The requirement for the Software Engineering Large Practical is to use the *Android Studio* development environment to create an app implemented in Java and XML for an Android device. The app implements a mobile game which allows users to make words by collecting letters which are distributed around the University of Edinburgh's Central Area (see Figure 1 for an example). Inspired by the games *Pokémon GO* and *Scrabble*, the game is called *Grabble*.

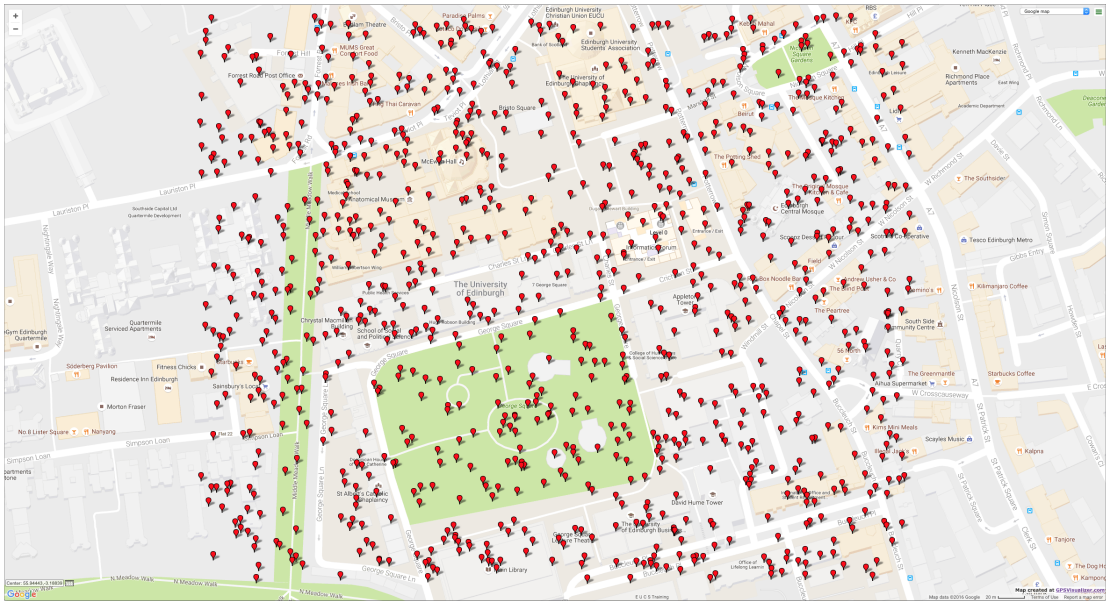


Figure 1: Letters indicated by red pins in the University's Central Area

Letters are collected by visiting their location. There is a different set of letters for each day of the week. Letters can only be collected once each day. (I.e. having visited a location to collect a letter it is not possible to move away from that location and then move back again to collect the letter a second time.)

— ◇ —

The object of the game is to make seven-letter words out of the letters which have been collected. Each letter has a point value associated with it and a score is assigned to a word by summing the scores of the letters in the word. The point value of each letter is given below: more commonly-occurring letters have lower values and less commonly-occurring letters have higher values.

A	B	C	D	E	F	G	H	I	J	K	L	M
3	20	13	10	1	15	18	9	5	25	22	11	14

N	O	P	Q	R	S	T	U	V	W	X	Y	Z
6	4	19	24	8	7	2	12	21	17	23	16	26

For the purposes of the game, a seven-letter sequence of characters is considered to be a word if it appears in the *Official Grabble Dictionary 2016*, available on-line at http://www.inf.ed.ac.uk/teaching/courses/selp/coursework/grabble.txt

— ◇ —

The *Official Grabble Dictionary 2016* will not be updated during this practical exercise so it is fine to download it and install it directly in your app. However, the dictionary will remain available at the above address throughout so you can access the online version from your app if you wish to do this instead. The dictionary has 23,869 entries.

— ◇ —

There is a *Grabble Letter Map* for each day of the week, made available in the Keyhole Markup Language (KML) format used by Google Earth and other geographic visualisation software. The letter maps are available at the following locations:

— http://www.inf.ed.ac.uk/teaching/courses/selp/coursework/sunday.kml
— http://www.inf.ed.ac.uk/teaching/courses/selp/coursework/monday.kml
— http://www.inf.ed.ac.uk/teaching/courses/selp/coursework/tuesday.kml
— http://www.inf.ed.ac.uk/teaching/courses/selp/coursework/wednesday.kml
— http://www.inf.ed.ac.uk/teaching/courses/selp/coursework/thursday.kml
— http://www.inf.ed.ac.uk/teaching/courses/selp/coursework/friday.kml
— http://www.inf.ed.ac.uk/teaching/courses/selp/coursework/saturday.kml

The day of the week when the app is started determines the map which is loaded. This map remains in use until play ends. It is not necessary to replace one map with another at midnight, if the game is being played then.

— ◇ —

Unlike the *Official Grabble Dictionary 2016*, any *Grabble Letter Map* may be updated at any time so it is important to use the on-line version to ensure that you are looking at the correct version of the map. Downloading and bundling these maps with your application would not achieve the desired result.

— ◇ —

Each letter map contains 1,000 points numbered from 1 to 1,000, each with an uppercase letter attached. The letters have been chosen at random and distributed at random. No letter occurs significantly more often than the others.

— ◇ —

The format of the KML files for the letter maps is outlined in Figure 2. A KML document is a list of Placemarks. Each Placemark contains a name giving the unique numerical identifier of the place, a description giving the letter which is available here, and a Point. A Point has coordinates in the format ⟨longitude, latitude, height⟩ where the height is always 0.

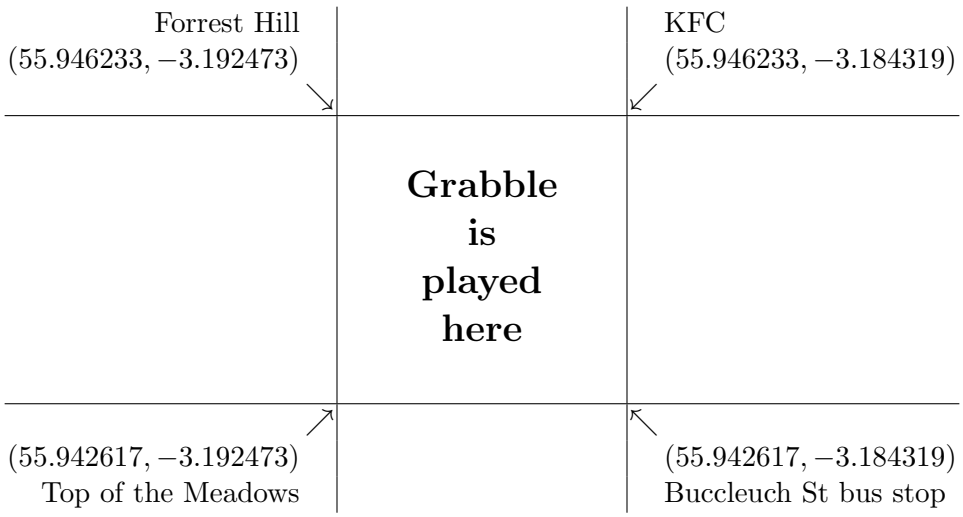
```
<?xml version="1.0" encoding="UTF-8"?>
<kml xmlns="http://www.opengis.net/kml/2.2">
  <Placemark>
    <name>Point 1</name>
    <description>N</description>
    <Point>
      <coordinates>-3.19144566846689,55.94300165613873,0</coordinates>
    </Point>
  </Placemark>
 ...
  <Placemark>
    <name>Point 1000</name>
    <description>Z</description>
    <Point>
      <coordinates>-3.1862219117766553,55.94453310098754,0</coordinates>
    </Point>
  </Placemark>
</kml>
```

Figure 2: Sample of the KML format used in the letter maps. Point 1 is the letter N, point 1,000 is the letter Z.

In designing your game you should decide how near a Placemark the player physically needs to be before they can be considered to have "grabbed" that letter. GPS-based devices cannot determine your true location perfectly but the Android LocationManager API at least attempts to determine the accuracy of its estimated location.

— ◇ —

All points on every map have a latitude which lies between 55.942617 and 55.946233. All points on every map have a longitude which lies between −3.184319 and −3.192473.



Bonus features

In addition to the game features described above you should design and implement some *Bonus Features*, which set your app apart from others. These may be enhancements which are intended to make the game more interesting to play, or more rewarding, causing the user to play more frequently, or for longer sessions. What the bonus features are is up to you but you could consider enhancements in areas such as:

- scoreboards and statistics on play;
- setting goals such as word targets or distance targets;
- autocompletion or spelling correction of words;
- play modes (beginner, advanced, expert); or
- user interface modes (night mode, battery-saver mode).

You are not limited to the items above; this list is only to prompt you to think about your own bonus features.

Software engineering aspects of the practical

This practical helps you to develop three useful Software Engineering skills:

- **using version control systems:** you are to use the Git version control system to manage the source code of your application—learning how much and when to commit code is a useful skill;
- **writing automated tests:** you are to write automated tests for your code and submit these together with the source code of your application; and
- **writing readable source code:** the Java source code which you submit will be inspected for clarity and readability (as well as correctness) so you should try to write clear, easy-to-read code.

Frequently asked questions

- *I don't have an Android device. I've never written an app before. How can I do this practical?*
  - You don't need to have an Android device to do this practical exercise. The software which you develop will run on an emulator which is freely available for Windows, Mac OS X, and GNU/Linux platforms. There is no expectation that you have written an app before: you will learn how to do this in the course of this practical. You may also need to learn more about Java programming.

- *Can I develop my app on my laptop?*
  - Yes. You are strongly encouraged to do this because it will encourage you to investigate the Android SDK and related libraries. Of course, we recommend taking regular, well-organised backups.
- *Can I implement my app in Ruby/Python/Scala/C# instead?*
  - No, not for this practical. We need all students to be working in the same programming language in order to make a fair assessment.

    However, Java is not an arbitrary choice. Java is the most widely used programming language for the Android platform and there are many more Java language resources available online to learn Android development from than for any other language.

    For sound educational reasons, we believe that the choice of Java as the development language should help most students to complete this practical on Android successfully.
- *Do I have to develop in Android Studio? I much prefer Eclipse/Emacs/vi etc.*
  - You are required to submit an Android Studio project so we strongly recommend developing in Android Studio for this practical exercise. An Android project developed in Eclipse uses a different build system from Android Studio and can require some significant effort to be made to work in Android Studio.
    - *[If it does not seem possible to use Android Studio on any platform which you have access to then please contact the course lecturer to discuss alternative arrangements.]*
- *Is there a specified device for this practical or a specified Android version?*
  - No. You can choose an Android device and an Android version. If you have an Android device then you could choose a suitable specification for that device, to allow you to test your app on a real device. If you do not have an Android device then choose the emulator for a relatively recent device and a relatively recent version of the Android platform. In particular, please note that backwards compatibility is *not* a requirement: we do not mind if your app does not run on older devices.
- *I have a server where I can host web services. Can I transfer some of the app's functionality to the server side?*
  - In principle, yes, but please consult the course lecturer with specifics, particularly with regard to the services made available and the programming language (or languages) used on the server side. You will also need to submit your server-side code for assessment, and it too should be readable and clear.

Part 1
Software Engineering Large Practical

Stephen Gilmore (Stephen.Gilmore@ed.ac.uk)
School of Informatics

1.1 Introduction

This part of the SELP is zero-weighted: all of the assessment is based on Part 2 and Part 3 of the practical. Nonetheless, you are strongly encouraged to complete this part. Completion of this part of the practical will provide useful feedback and guidance on how to progress with your work.

1.2 Description

This part of the SELP consists of a proposal document specifying functional and non-functional requirements on the project. This document is a *proposal*. It forms a basis for your design and should be modified as necessary in response to the feedback which you receive. The implementation which you deliver later will not be judged against this proposal.

— ◇ —

You should specify the functional requirements which your app is to fulfil, including your current ideas on the *bonus features* which you will add to the already-specified requirements. You should also give details of the non-functional requirements of your app, including decisions which you have made about the kind of Android device (or devices) that you will target, and the Android version which you will target. You should explain the factors which influenced your decision. Users of apps are also sensitive to slow or long-running operations in their apps. Identify any aspects of your application which you think potentially have long run-times and give a proposal on how you will deal with these.

— ◇ —

The expected length of this proposal document is between 2 and 4 pages. The choice of font, font size, and margins is up to you but please consider the readability of your submission. The submission format is PDF only.

1.3 How to submit

Please submit your proposal document from your DICE account using the command:

```
submit selp 1 proposal.pdf
```

(Assuming that your proposal is in a document called `proposal.pdf`.)

1.4 Things to consider

- It is better to promise less and deliver more than to promise more and deliver less, so don't make your list of bonus features too long.
- You will need to investigate Android concepts and programming in order to be able to make informed decisions about the bonus features which you will implement, and to estimate which aspects of the app may take a long time to execute.

Part 2
Software Engineering Large Practical

Stephen Gilmore (Stephen.Gilmore@ed.ac.uk)
School of Informatics

2.1 Introduction

This part and the next part of the SELP are for credit: all of the assessment is based on Part 2 and Part 3 of the practical, weighted equally.

— ◇ —

Good Scholarly Practice: Please remember the University requirement as regards all assessed work for credit. Details and advice about this can be found at:

http://web.inf.ed.ac.uk/infweb/admin/policies/academic-misconduct

and links from there. Note that, in particular, you are required to take reasonable measures to protect your assessed work from unauthorised access. For example, if you put any such work on a public repository then you must set access permissions appropriately (generally permitting access only to yourself, or your group in the case of group practicals).

— ◇ —

The Software Engineering Large Practical is not a group practical, so all work that you submit for assessment must be your own, or be acknowledged as coming from a publically-available source such as Android tutorial examples, Android sample projects, or open-source projects hosted on GitHub, BitBucket or elsewhere.

2.2 Using version control

From this point onwards you will be creating XML and Java resources which will form part of your implementation, to be submitted in Part 3 of this practical. These resources should be placed under version control using the Git version control system. You are free to use any private Git repository to keep copies of your work. A suitable hosting service is *BitBucket* (https://bitbucket.org/), which currently offers free, private Git repositories for small projects. Create a login on the BitBucket service and create a Git repository there referring to Figure 2.3 for instructions on how to set up such a repository and referring to the course lectures for details on how to commit files into the repository.

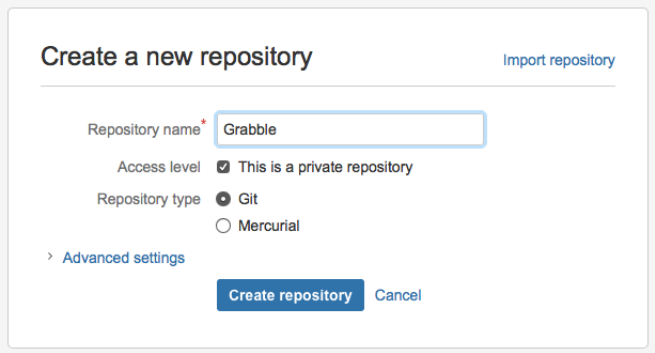


Figure 2.3: Creating a private Git repository (https://bitbucket.org/repo/create)

2.3 Description

The second part of the SELP involves the creation of a design document, presenting the plan of the implementation work which will realise the design. The design document commits you to certain decisions, which are to be realised in the implementation.

— ◇ —

Your design document should outline the **software architecture of your app** in terms of the libraries it will use and dependencies that it will have, together with a list of the activities which will make up your app.

— ◇ —

Your design document should be illustrated with **screenshots of the views which you have designed** for the activities in your app. These can be taken either from the Android emulator running a (partially complete) version of your app, or from the visual editor in Android Studio, rendering your XML layouts, with no Java code attached. Include as many as possible, to give the clearest picture of how your app will work in practice.

— ◇ —

Your design document should include a **definitive list of the bonus features** which are offered by your app. Your implementation will be expected to provide these bonus features, in addition to being a playable implementation of the Grabble game. The list which you provide here could be significantly different from the list which you provided in your proposal document. No penalty will be applied for this.

— ◇ —

The expected length of this design document is between 4 and 8 pages. The choice of font, font size, and margins is up to you but please consider the readability of your submission. The submission format is PDF only.

2.4 How to submit

Please submit your design document from your DICE account using the command:

```
submit selp 2 design.pdf
```

(Assuming that your design is in a document called `design.pdf`.)

2.5 Marking criteria

The following criteria will be used in determining a mark for your submitted design. The criteria listed below are in no particular order and none of them is significantly more important than the others.

- The coverage of the activities which your app will include. Have any major application functions been forgotten? is each activity a coherent, self-contained item of work which needs to be done?
- The completeness of the libraries and dependencies which you list. Have you included everything which you will need to implement the bonus features which you are promising?
- The added value provided by the *bonus features* which you offer. Are they just some additional decorations or do they actually make the game more interesting to play, or more rewarding?
- The coherency and consistency of the screens which you show from your application. Do they clearly belong to the same app, giving the impression of an app which has been designed thoughtfully? Do they represent aesthetically-pleasing design?

2.6 Things to consider

- If in doubt, leave it out. Don't promise to deliver bonus features which you have no idea how to implement. Think ahead to your implementation and try to be at least 80% sure that you know how to implement the features that you are promising to deliver.
- Although this is a design document, and you do not have to supply any code at this stage, you are *strongly encouraged to begin coding your app now*, to clarify your ideas about various details of Android application structure and to ensure that you have made a start on the implementation of the app comfortably in advance of the Part 3 deadline for the SELP.

Part 3
Software Engineering Large Practical

Stephen Gilmore (Stephen.Gilmore@ed.ac.uk)
School of Informatics

3.1 Introduction

As noted above, this part and the previous part of the SELP are for credit: all of the assessment is based on Part 2 and Part 3 of the practical, weighted equally.

— ◇ —

The third part is the implementation. This should be a well-engineered implementation of the previously-supplied design. The code which you submit for assessment should be readable, well-structured, and thoroughly tested. Any automated tests which you have written for your code should also be submitted. For your Android app, these should be in the `test` or `androidTest` folders of your project.

3.2 Documentation

Please submit documentation describing your implementation. You should use this to report information such as the following, and any other special features of your implementation which would be relevant for the marker to know.

- *Algorithms and data structures used for the core functions of the implementation.* Describe the algorithms and data structures which are used in your implementation for:
  - download and parsing of the daily Grabble letter map from the server;
  - efficient lookup of words in the Grabble dictionary; and
  - efficient detection of the letters which can be grabbed at the user's current location.
- *Parts of your design which have not been realised in the implementation.* It could be that you have not been able to implement something which you had planned in your design. If so, document that here.
- *Additional features of your implementation which were not described in your design.* It could be that you have implemented something which you had not planned in your design. If so, document that here.

- *Your use of version control systems to manage your project source code.* Which parts of your project were archived in a version control system? Give details of the repository which you used. If you have used BitBucket for your project then please give BitBucket user `stephengilmore` read access to your repository.
- *The types of testing which you applied to your implementation.* Provide details of the types of testing which you applied to your project. Did you test on the emulator only, or also on a physical device? Do you have instrumented tests for your application (in `androidTest`)? Do you have unit tests for your application (in `test`).
- (Only if relevant[1].) *Instructions for installing the server-side part of your project, if you have one.* If you have a server-side component to your project implemented in Python, PHP, Ruby on Rails, or similar, then provide details of how to install and run this component.

— ◇ —

The expected length of this document is between 2 and 4 pages, but longer submissions will also be accepted. The choice of font, font size, and margins is up to you but please consider the readability of your submission. The submission format is PDF only.

3.3 Preparing your submission

- Create a new folder to contain your submission. Assuming that your folder is called `grabble`, create two sub-folders `grabble/doc` and `grabble/android`.

  Place a copy of your documentation in the `grabble/doc` folder and place a copy of your Android Studio project in the `grabble/android` folder.

  (Only if relevant.) If your submission has a server-side component then create a third sub-folder called `grabble/server` and place a copy of your server-side code in there.
- (Optional, you can skip this step if you wish to.) You can make the submitted file smaller in size by removing any precompiled code from your Android Studio project. Specifically, you can remove the folders `grabble/android/build` and `grabble/android/app/build`. The contents of these folders will be regenerated when your project is loaded into Android Studio for testing.
- (Optional, you can skip this step if you wish to.) You can remove any keys which you have generated for Google Maps or other APIs. Replace these with the text `YOUR_KEY_HERE`.

[1] Depending on the bonus features which you decided to implement, a server-side component might have been required; if you do not have one then you need not include anything on this in your document.

- Make a compressed version of your folder using ZIP compression.
  - On Linux systems use the command `zip -r grabble.zip grabble`.
  - On Windows systems use Send to > Compressed (zipped) folder.
  - On Mac systems use File > Compress "grabble".

3.4 How to submit

Please submit your implementation work from your DICE account using the command:

```
submit selp 3 grabble.zip
```

(Assuming that your implementation is in a ZIP archive called `grabble.zip`.)

3.5 Marking criteria

- Your submission should be a playable implementation of the Grabble game, extended with bonus features of your choosing.
- Bonus features will be evaluated on whether they make the game more interesting to play, or more rewarding. More significant bonus features are more credit-worthy.
- Your submission should be consistent with your previously-submitted design.
- The game should be robust. Failing with a `NullPointerException` or other Java run-time error will be considered a serious fault.
- Your game should correctly assign points to words according to the Grabble table of letter values (see page 2) such that, for example

$$value(\text{LOOKING}) = 11 + 4 + 4 + 22 + 5 + 6 + 18 = 70.$$

- Your game should correctly differentiate dictionary words from non-dictionary words.
- Your game should load the correct Grabble letter map from the server, as determined by the day of the week when play starts.
- The game should be usably efficient, without significant stalls while playing.
- Your submitted code should be readable and clear.

3.6 Things to consider

- Your submitted Java (and other) code will be read and assessed by a person, not a script. It is not a waste of time to add comments to your code, documenting your intentions. It is not a waste of time to structure your code well, introducing private methods and encapsulating code and data structures.
- Logging statements (`Log.d` and friends) are useful debugging tools for Android apps. You do not need to remove them from your submitted code. It is fine for these to appear in your submission.
- All else being equal, a submission with automated tests should receive a higher mark than one without automated tests. More comprehensive tests are more creditworthy.
- All else being equal, a project where a version control system (such as Git) has been used to manage the project code and resources should receive a higher mark than one where version control has not been used.