Phase 1 Specification

[Last updated Tuesday 8 October]

Overview

You and your team will design and create an Adroid app. We provide the specification below, but **if your team has an idea for a different app**, you must do the following:

1. Aim for your app to be as complex as the following specification.

2. Send an email to **csc207-2019-09@cs.toronto.edu (mailto:csc207-2019-09@cs.toronto.edu)** with the subject "CSC207 Project Proposal for group_XXXX". Replace the XXXX with the number that you see on MarkUs, when you click on "ProjectTeams". In that email, describe the app that you want to create. If it is too complex or not complex enough, we will let you know.

Your team repositories will be set up on Thursday 10 October.

Game Specification

You will create a game with multiple levels and a scoring system. It must allow users to create accounts, sign in, and also resume a previous game. The sign-in process can be entirely local to the project app, unconnected to the phone owner's account.

There should be at least three "levels" in your game. Each level must be a game or challenge that is structurally different from the other levels. Structures include, but are not limited to, navigating a maze, turning over pairs of tiles in order to match them to each other, and controlling a moving shape that bounces off of surfaces and/or into targets.

At least three statistics must be tracked for each user.

Example 1: The user plays a character who can collect points, has a number of "lives", and keeps track of magical objects that they have collected.

Example 2: The user faces a series of timed challenges. Your app can keep track of total points earned, average time to complete a challenge, and total time to complete all challenges thus far.

You are welcome to reuse or slightly modify features from other levels. For example, a card from a card-matching game can also be used as an obstacle for a pinball-like game.

Here are some examples of simple games: **Pong (https://en.wikipedia.org/wiki/Pong)**, **Nibbles (https://en.wikipedia.org/wiki/Nibbles_(video_game))**, **Frogger (https://en.wikipedia.org/wiki/Frogger)**, **Sliding Tiles (https://en.wikipedia.org/wiki/Sliding_puzzle)**, and **Gorilla**

(https://en.wikipedia.org/wiki/Gorillas_(video_game)) , and Hangman (https://en.wikipedia.org/wiki/Hangman_(game)) .

Don't copy any games

There are lots of Android games on the web. Feel free to browse for inspiration. You can even borrow code if you learn from it and the license lets you. For example, you might learn how to pass information from one game level to the next, or to display a grid for checkers, or save information when the user quits the app.

Any code from https://developer.android.com (https://developer.android.com) is fine to use, unless it's a game. You may find the developer samples (https://developer.android.com/samples) to be helpful.

Cite your sources by including a URL. Let us know where you're learning stuff!

Remember to organize your code and follow the SOLID principles.

Customization

The user should be able to customize their game experience in at least three different ways. Some examples of customizations include: selecting a colour scheme, choosing a character to play, selecting a second language to accompany any English* words that show up in your program, and so on. You do not have to include these particular customizations. Feel free to invent your own.

*Please note that the graders will expect all comments, Javadoc, and in-game words to be written in English, so that you can receive credit for your hard work. If you allow users to select a second language, have it be an easily-accessibly option so that anyone can switch without knowing the currently-selected language.

Storing Information Outside of your Program

If a user exits your app while in the middle of a game, they should be able to return to the beginning of the last level they played, with the same statistics that they had at the end of their previous session.

Information about how users customized their game should be stored as well, so that the same customized choices are present the next time they log in.

Game Design Considerations

Is it possible to lose your game? Some games are about scoring points only, so the idea of winning or losing does not apply. Other games are character-based, where a main goal is to keep

your character alive. In that case, running out of lives would mean that your game has ended and the user can choose to start again.

How do you know when a level is over? Some levels or games may be timed, so that the level ends when there are zero seconds left on the timer. Other levels or games may require the user to complete a challenge. Such a level can end when the challenge is completed, when time runs out, or when a certain number of points has been reached. Before creating your game, consider how you want each level to end.

Is there a story behind your game? Some games are based on a narrative (for example: rescue someone who is in trouble, find the treasure, unlock a secret, etc.). Other games can be a lose collection of challenges that are united by a scoring system. You can choose how you want to structure your game.

Which statistics do you want to track? You are required to track at least three pieces of statistical information about each user during their game. For example, "lives" are the number of times a user can restart the current level before the game ends. You do not have to implement a number of lives, but it is one possible piece of statistical information. You are welcome to get creative about which information you want to track throughout the game.

Do you want a scoreboard? There are many ways to indicate to the user their statistical data. One way is to keep a score board and display the top 3 (or 5 or 10) statistical results of all players and show the current user how they fit into that ranking. Another way is to display a summary of the game upon its completion. Or you can set up a user history, so that the user can compair their statistics from the current game to previous games of theirs. **You do not have to implement any of these options for Phase 1.** But this is the sort of extra feature that may be included in Phase 2.

Starting the project

Read 19 pages of the textbook

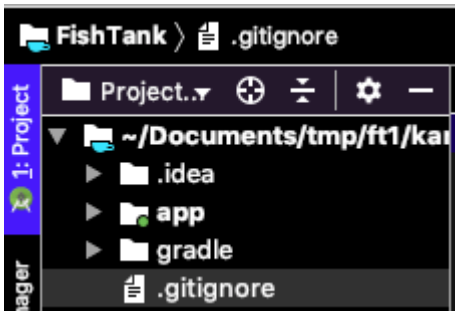
Read The Missing Chapter (pages 303–321) in **the course textbook (https://q.utoronto.ca/courses/111135/pages/readings-for-weeks-1-12?module_item_id=863424)** . Read it carefully and actively discuss your code organization with your group as you work on the project. Disciplined package use will one of the project marking criteria. **Be kind to each other while you're working together, especially when you're brainstorming.**

Create the project and add it to the MarkUs repo

One of you should do the following **with at least one other team member watching** to help prevent this from going badly. :-)

- Clone your team repo.

- Create a single Android Project in your team repo **in the phase1 directory**.
  - Name it something reasonable and use Android 9.0 (API level 28). This is Google Play's rule for new apps on the Play Store.
  - We recommend that you create an Empty Activity. Make sure it compiles and runs.
- Switch to the Project Files view. Edit .gitignore. This lists all the filename patterns that should **not** be added to the repo — all the automatically-generated files. Replace its contents with [this automatically-generated .gitignore (https://www.gitignore.io/?templates=android,intellij,androidstudio)](https://www.gitignore.io/?templates=android,intellij,androidstudio) file. Save.



- Commit and push your project.

Another team member should clone the project and make sure that it works before you do any more work.

You must use your team's MarkUs repository. Please do not use another code hosting website such as GitHub! if you like, you can migrate the code to GitHub, Bitbucket, or GitLab after the course is over.

You can create other subdirectories in the phase1 directory if you wish.

Google Play Store!

You can upload your game to the Google Play Store after the term is over if you like. Check out the resources on the main page of [https://developer.android.com (https://developer.android.com)](https://developer.android.com).

How to get a good mark

Your grade will be based on how **easy it is to read, extend, and maintain your code**. Is your code **encapsulated** so that a change to one part of the code does not require changes in multiple other places? Is your code **free of code smells**? You are welcome to look up relevent design patterns. We will be discussing (at least) the MVP, Strategy, Factory Method, Observer, and Dependency Injection design patterns during lecture.

In order to receive a passing mark for your Phase 1, the user must be able to play through all three levels of the game (although perhaps with simpler versions of the games than you would like) and view their statistics. Phase 2 will involve extending your

program in various directions which will be described on a subsequent Quercus page. Phase 2 will also include some new features that are specific to your team's app.

If your Phase 2 mark is higher than your Phase 1 mark, we will replace your Phase 1 mark with your Phase 2 mark. The goal of Phase 1 is to give you a chance to get far enough so that you can receive feedback on your work.