CS 478 **Project #4** **Fall 2019**

Threads

Due Date: Sunday, November 24th, 2019 @11:59pm

Project Details:

You are to design and code a game of gopher hunting as an Android app. Imagine you are in a field occupied by a gopher. The gopher could be hiding in one of any number of holes in the field's ground. The goal of the game is to find the hole that contains the gopher. (The game does not say what to do with the gopher, once it is found.)

The game is played by two worker threads contained in your app, with the threads playing against each other. There are exactly 100 holes in the field; the holes are arranged as a 10×10 matrix and equally spaced with respect to each other. The threads take turns at guessing the hole containing the gopher; the first thread to find gopher wins the game.

Implementation Details:

The app supports both a guess-by-guess game play as well as a continuous mode whereby the two threads play without interruption until one thread wins the game. Either way, the app contains a display showing all the guesses that the two threads have made so far. (Use different colors or shapes to distinguish the guesses of the two playing threads.) Appropriate buttons will let a user decide when to let the threads make a guess (when in move-by-move mode), or to switch to continuous play mode. When in continuous play mode, make sure to delay the execution of the worker threads enough that a human viewer can see and understand the guesses of the two threads on the app's display.

Whenever a thread makes a guess, the game system provides one of five possible responses:

1. Success—The thread guesses the hole containing the gopher and wins the game.

2. Near miss—The thread guesses one of the 8 holes adjacent to the gopher's hole. The game continues with the other thread making a guess.

3. Close guess—The thread guesses a hole that's 2 holes away from the gopher's hole in any directions. In other words, it is possible to go from the guessed hole to the gopher's hole by making two moves to adjacent holes from the guessed hole. Each move could be in a horizontal, vertical or diagonal direction.

4. Complete miss—The thread gets this response in all other cases.

5. Disaster—A thread inadvertently guesses a hole that's already been guessed (either by the same thread or by the other thread.)

Here are some additional requirements on the app.

1. You are at liberty to design the app in the way that you find most appropriate, including the number and type of components in the app. (Hint: You should probably have at least two activities, one for starting a game and another to display the progress of the game.)

2. Your app must contain at least three threads, the two worker threads playing against each other as well as the main thread. Each thread must have a job queue, a looper and a handler. The two worker threads should work in parallel to compute the best strategy for winning the game. The main thread should manage the game, e.g., starting the game, determining when the game is over, notifying the worker threads to stop playing and displaying the winner on the app's display.

3. The threads must communicate with each other and with the main thread using handlers. You must include both runnables and messages (at least one of each) in the job queue of the worker thread. (When adding jobs to handlers, make sure to post at least a Runnable and send at least a Message somewhere in your project code.)

4. The two worker threads must use different strategies (e.g., heuristics) for both their initial guess and subsequent guesses.

5. Configure your app in portrait mode. Do not worry about the quality of the display if the device is rotated to landscape mode.

6. Make sure that the game is played at such a speed that a human user can clearly see and understand the guesses of each thread.

Submission Details:

You must work alone on this project. For this project use a Pixel 2 device running the usual Android platform (API 28—Pie). You are not required to provide backward compatibility with previous Android versions. Submit one Studio project as a zip archive using the submission link in the assignment's page on Blackboard. Code that does not include worker threads will receive no credit. No late submissions will be accepted.

Academic Integrity:

Unless stated otherwise, all work submitted for grading *must* be done individually. While we encourage you to talk to your peers and learn from them, this interaction must be superficial with regards to all work submitted for grading. This means you *cannot* work in teams, you cannot work side-by-side, you cannot submit someone else's work (partial or complete) as your own. The University's policy is available here:

https://dos.uic.edu/conductforstudents.shtml.

In particular, note that you are guilty of academic dishonesty if you extend or receive any kind of unauthorized assistance. Absolutely no transfer of program code between students is permitted (paper or electronic), and you may not solicit code from family, friends, or online forums. Other examples of academic dishonesty include emailing your program to another student, copying-pasting code from the internet, working in a group on a homework assignment, and allowing a tutor, TA, or another individual to write an answer for you. It is also considered academic dishonesty if you click someone else's iClicker with the intent of answering for that student, whether for a quiz, exam, or class participation. Academic dishonesty is unacceptable, and penalties range from a letter grade drop to expulsion from the university; cases are handled via the official student conduct process described at https://dos.uic.edu/conductforstudents.shtml.