# CS478: Software Development for Mobile Platforms

*Project #4*

Due time: 11:59 pm on 11/15/2020

*Total points: 100*

Instructor: Ugo Buy

TAs: Ajith Nair, Vinay Chavan and David Shumway

In this project you will design and code a strategy game called *Guess Four* as an Android app. In this game, each of two players sets up a secret sequence of four decimal digits, where each digit is not repeated. For instance, 2018 is a legal sequence, but 2012 is not because the digit 2 is repeated. The two players take turns guessing the sequence of digits of their opponent. Each guess consists of a 4 digit number, without repeated digits. A player responds to an opponents guess by specifying the number of digits that were successfully guessed in the correct position and the number of digits that were successfully guessed but in the wrong positions. Thus, if a players chosen number is 2018 and the opponent guesses 1079, the opponent would be told that one digit was guessed correctly in the correct position (i.e., the 0), and another digit was guessed in the wrong position, (i.e., the 1).

In addition, the opponent would be told one of the missed digits. In the example above, the opponent might be told that the 7 is not one of the digits or that the 9 is not one of the digits. The missed digit to be revealed to the opponent must be chosen randomly. Evidently, no digit would be revealed if the opponent guess the four digits correctly, even though these digits might be in the wrong positions.

You are to write an application that has two background threads playing against each other while also updating the user interface with their moves. The first thread to correctly guess the opposing thread's number wins the game.

Your implementation will have two Java worker threads play against each other. The UI thread is responsible for creating and starting the two worker threads and for maintaining and updating the display. Each worker thread will take turns taking the following actions:

1. Waiting for a short time (1-2 seconds) in order for a human viewer to take note of the previous move on the display.
2. Figuring out the next guess of this thread.
3. Communicating this guess to the opponent thread.
4. Waiting for a response from the opponent thread.
5. Communicating both the guess and the response from the opposite thread to the UI thread.

While carrying out the above steps each worker thread must also be able to respond to guesses from the opponent thread. Whenever a guess from the opponent thread is received, a worker thread must respond by communicating the number of correctly positioned and incorrectly positioned digits contained in the guess to the opponent thread, as well as one of the bad guesses.

Furthermore, the game must proceed in lockstep between the two threads. A thread is not allowed to make two consecutive guesses without handling an intervening guess from the opponent thread.

The UI thread is specifically responsible for the following functionality:

1. Showing the two initial numbers chosen by the worker threads.

2. Receiving notifications of guesses and their outcomes by the worker threads.
3. Displaying the guesses and responses of each worker thread in an appropriate format as each worker thread communicates this information to the UI thead.
4. Displaying a button to start the game. Pressing this button while a game is in progress will void the current game and start a new game from scratch.
5. Checking on the status of the game, by determining whether one thread has won or the game needs to continue.
6. Halting the game after each thread has made 20 guesses without identifying the opponent's number.
7. Signaling the two worker threads that the game is over; the two threads should stop their execution as a result of this action.
8. Displaying the outcome of the game in the UI.

**Implementation constraints.** Your project must comply with the following requirements.

1. Use handlers to implement the communication between the three threads involved. Each thread must have a handler, a job queue and a looper.
2. You must include at least one runnable and one message in the job queue of the worker threads.
3. The two worker threads must use different strategies for winning the game.
4. Make sure that the game is played at such a speed that a human user can clearly see and understand the move of each thread.
5. The app's display should have two container views or two fragments, one for each worker thread, lying side-by-side and occupying the entire screen width. Each side (whether a view or a fragment) will display the secret number of the corresponding thread at the top and a scrollable list of the thread's guesses and the opponent thread's responses in the remainder of the vertical space.
6. You may assume that the app will be used in landscape mode only.
7. The two worker threads will use a random number generator to create their secret number. These numbers are communicated to the UI thread, which will immediately display both numbers.

*You must work alone on this project.* For this project use the usual Pixel 3 XL device running the usual Android platform (API 28—Pie). You are not required to provide backward compatibility with previous Android versions. The game should always be shown in landscape orientation even if the user rotates the device to portrait mode. 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 carrying out the functionality of the players will receive no credit. No late submissions will be accepted.

*Hint:* You should design the flow of information among threads *before* you start coding this project. Map out the messages being exchanged among threads, when each message is sent, and how each therads reacts to receiving a message (i.e., what actions the thread performs after receiving a message).