Title: WhackAMole

Overview: The goal of this project is to create a whack-a-mole game app. The theme of the game is up to you, but the gameplay must be essentially whack-a-mole, as follows:

- The player scores points by tapping objects ("moles") that appear randomly on screen at a predetermined set of locations.
- After a mole appears, it remains on screen until either it is tapped by the player (a *hit*, worth a point), or a short amount of time passes (a *miss*).
- As the game progresses, the spawn rate of the moles gradually *increases*, and the time a mole remains on screen gradually *decreases*.
- When the player has missed a certain number of moles, the game is over and the total score is reported.

Academic dishonesty reminder: a reminder that you may not base your project on similar projects found elsewhere, including but not limited to tutorials, YouTube videos, code uploads, etc. *You should be starting from scratch*. If you need help getting started, talk to your TA!

General requirements (20 pts.):

1. The project's source code is organized following a pattern, e.g. MVC or similar. For example, model classes are defined in separate files, and are decoupled from the Activities/Fragments by use of public methods, getters/setters, etc. (10 pts.)

2. The project's source code is clean and well-formatted, and it is documented with Javadoc or similar. Expectations for Javadoc are provided in a separate file. (10 pts.)

Required functionality (80 pts.):

1. The main game UI shows a screen with a set of fixed locations ("holes") where objects ("moles") can spawn. (10 pts.)

2. During gameplay, moles appear and disappear at random holes over time. Moles do not spawn at a hole that is occupied by another mole. (10 pts.)

3. The rate at which moles appear and disappear increases over time as the game is played. (10 pts.)

4. Tapping on a mole causes it to disappear and increases the player's score by some amount. Tapping on a hole when no mole is visible has no effect. (10 pts.)

5. The game ends when the player has missed (i.e. failed to tap on) a certain number of moles. (10 pts.)

6. The player's score is shown in the main game UI and updated as it changes. After the game is over, the player is shown their final score. (10 pts.)

7. The highest score achieved in a game is stored and persists between opening/closing the app (i.e. persists on disk). (10 pts.)

8. The current high score is displayed somewhere in the main game UI. (10 pts.)

Optional bonus functionality:

- The game plays sound effects when a mole appears and when a mole is tapped. (5 pts. extra credit)

Hints:

- Your implementation does not have to be complex. You may need some tools not specifically covered in class, but consider how you can use what we *have* covered to your advantage.
- Consider making your moles ImageViews, with an appropriate onClickListener set.
- Consider using a Handler, e.g. the postAtTime() function, to drive your timing.
- The project is intended to be somewhat open-ended on the details. If it's not specifically called out here, then it's up to you!