Purpose

- to implement, test and deploy a guess a number Android app

Work Assignment

Please complete this assignment individually.

If you need help, that's OK! Just ask... I'm here-to-help!

Remember: share ideas, not code (!).

The Assignment

Using the Java programming language, you are to design, build, test and deploy a guess a number Android app.

Functional Requirements

The functional requirements for this assignment:

Initial State of App

Generate a random number between the range of 1 and 1000 (inclusive); call this number: theNumber

Game Play

- display a static text prompting the user to guess a number; call this number: userGuess
- if userGuess == theNumber then display a winning Toast message; else display a Toast message indicating whether theGuess was too high (or too low) when compared to theNumber. Please note: you must consider this bullet and remaining bullets together --- please ask me for clarification if required
- the user must guess theNumber within 10 guesses (see next bullet for the 2 win states and loss state)
- if the user takes 5 or less guesses, display a Toast message: "Superior win!"; if the user takes 6 to 10 guesses, display a Toast message: "Excellent win!"; if the user takes more than 10 guesses, display a Toast message: "Please Reset!"
- The guess button will only display the "Please Reset" message if the user takes more than 10 guesses. It will not recognize correct guesses after this point.

Reset Button :: onClick( )

Reset the game: randomly pick a new theNumber, and re-set the number of user guesses back to 0 (zero).

Reset Button :: onLongClick( )

Use an Intent to inflate another activity, called ResultsActivity. The ResultsActivity displays the following information: userGuess (0 if the user has not guessed yet),

theNumber, total number of guesses, number of high guesses, number of low guesses, number of perfect wins, number of wins, number of losses.

Guess Button :: onClick( ) - See Game Play section above

About Dialog

Provide an information icon to the app's Action Bar. When clicked, display a dialog (i.e. AlertDialog) that displays your full name (first & last) and username. For example: Gerry Hurdle (hurdleg). Dismiss the dialog when the OK button is clicked.

Android Project Requirements

Launch Android Studio, close all projects, and Start a new Android Studio project:

Configure your new project

- Application Name: **HiLo**
- Company Domain: ***yourCollegeUsername***.algonquincollege.com

Select the form factors your app will run on

- Select: Phone and Table
- Minimum SDK: API 28: Android 9.0 (Pie)

Add an activity to Mobile

- Select: Empty Activity

Customize the Activity

- Check Backwards Compatibility (AppCompat)

The checkbox Backwards Compatibility (AppCompat) determines whether or not your app is backwards compatible to older versions of Android.

When Material Design was released by Android (Google) back in Android 5.0 (Lollipop), they needed a way to provide backwards compatible implementations of the new features to older versions of Android. At that time, Android 4.x (KitKat) was still being used by many devices.

The solution? Android created a library, and named it Support Library (https://developer.android.com/topic/libraries/support-library/index.html)

Next, Android created a class, called AppCompatActivity, which is the base class for activities that use the Support Library features.

So when you check the Backwards Compatibility checkbox, class MainActivity will be declared as:

```
public class MainActivity extends AppCompatActivity { //java code }
```

If you don't need to support backwards compatibility to versions of Android prior to Lollipop (5.0), then un-check the Backwards Compatibility checkbox. Then class MainActivity will be declared as:

```
public class MainActivity extends Activity { //java code }
```

Data Input and Validation

The user is permitted to enter **any** input!

Validate the user's input:

- accept range: 1 2 3 4 ... 100 ... 998 999 1000
- reject: all other input; set an appropriate error message (setError( )) and request the focus to the edit text

Visual Requirements

You have the freedom to express yourself in designing your app :)

Your design is to have the following <View> elements:

- an icon image of the user's competitor; for example, the image could be Duke (Java's mascot)
- an edit text for the user to enter their guesses
- a guess button (call to action)
- a reset button (secondary call to action)

Colours

Set the colours to use: **primary**, **primaryDark** and **primaryLight** as mono-chromatic colour values in **res/values/colors.xml**. Also add a **colorTextDark** and a **colorTextLight** to the colors.xml file.

Java Code Structure

Use Android Studio to beautify your code :)

Open MainActivity.java in the editor: Code -> Reformat Code...

Documentation Requirements

DO

- To the top of MainActivity.java, put the following header comment:
- ```
  /**
   *  Purpose/Description of your app
   *  @author YourFirstname + Lastname (your AC email)
   */
  ```

DO NOT || Optional

- documentation for constructors, fields (class & instance), methods (static and instance)
- generate the JavaDoc documentation

Deliverables

There are 2 deliverables:

1. A demonstration of your lab during your scheduled lab on **Tuesday December 4th between 6:30pm and 8:00pm (last demo!)**
2. A zip-file of your Android Studio project. Upload your zip-file to Brightspace before the due date.

Assessment

Your assignment should reflect your best work.

In short, your assignment should:

- behave correctly (i.e. it must work)
- be aesthetically pleasing (i.e. it must look good)
  ```