Bao Doan, Rachel Settle, Grant Besner, Natalie Le

TESTING

Most of our tests were done using Log.d, but some are JUnit tests.

- **Weather API**
    - Tested whether a connection was successfully being made with the OpenWeatherMap API using log.d with tag “WEATHER”
    - Tested whether the OpenWeatherMap API was correctly reading Durham’s temperature using log.d with tag “WEATHER”
- **Camera functionality**
    - Tested whether the Bitmap was compressing correctly using log.d with tag “FILES”
    - Tested whether the customizable tent option was making the camera intent and leading to photo capabilities using log.d with tag “CHARACTER”
    - Affirmed that the correct path was being used to find the file which would store the photo using log.d with tag “PHOTO”
- **Adjustable Volume and Difficulty**
    - Tested whether the continuous volume seekbar was registering changes using log.d with tag “Vol”
    - Tested whether the discrete difficulty seekbar was registering changes using log.d with tag “Diff”
- **Victory Screech**
    - The first set of Logs with the tag “TAG” were used to test whether the mediaRecorder methods were properly functioning. Additionally, we used the “mediaNullMade” tag to see whether or not the recorder was null after certain lines of code because it was originally giving us NullPointerExceptions.
- **Game Over Screen**
    - We used a Log statement here to see what row was inserted into the local database. The tag for this Log statement is “rowInserted”
- **Character Select Adapter**
    - Log statements were used here to test whether the character names and positions on the screen matched what we thought they would be in this adapter.
- **For a few specific situations, we used JUnit tests:**
    - The first test we wrote was to see if we were actually retrieving only characters from the drawable file in the CharacterSelectionScreen activity. The method `R.drawable.class.getFields()` raised some concerns for us, so we just wanted to see if it did what we thought it was supposed to do
    - The second test we wrote was to see if there were equal number of write and read methods in our SharedPref class, which we used as a central location to handle SharedPreference. Since these two methods were overloaded in SharedPref, one pair for each data type, it would have been easy to create either

a write or read method for one data type without making a corresponding complementary read or write method. SharedPreferences is not very useful if data cannot both be read and written, so we felt like this was an appropriate JUnit test.