Assignment 3 (110 points)
Fall 2014
Shlomo Hershkop
CIS 542
Out: Oct 22, 2014
Due: Midnight Nov 9

Andriod I

Skills we hope to cover:

- Java Programming
- Using the Android emulator
- Android GUI elements
- Handling user interaction, e.g. button clicks
- debugging andriod programs

In this Lab, you will build a simple Android typing game in which the user is shown a sentence and has to type it in word-for-word. When the user is done, the program indicates how long it took to type the sentence, checking it was entered correctly and scores the difficulty of the sentence and the individual words.

The application also keeps track of the best (lowest) time to type the sentence correctly for each category.

In addition you will write a generator which will generate random sentences based on user configuration. (example easy, medium, hard generates short, medium, longer sentences from different word mixes).

This document takes you step-by-step through most of what you need to do in order to implement this game.

Before you begin

This assignment assumes you have correctly set up your Android development environment as we did in class. Please see the uploaded java-andriod text file for pointers. Follow the simple set-up tutorial to make sure you can create the "Hello, Android" application before proceeding.

for the first couple of steps, please hard code a target sentence until you fold in the sentence generator.

A word about debugging in Android

When you're debugging your Android app, you'll probably be tempted to use standard java System.out.println
You'll notice immediately that the messages you print don't appear in the Android app, nor in the studio console. Android uses a logging mechanism called LogCat, in which messages are written to a file, along with the time they were written, any associated tags, etc.

Step 0. Create a new Android project

Give the project a meaningful name (something like "TypingGame") and choose a meaningful Java package name (e.g. “edu.upenn.cis542.lab3.typinggame").

Step 1. Arrange the Views in a Layout

We want the main playing screen (where the user types in the sentence that she is shown) to look something like this:

That is, there is a TextView along the top, an EditText taking up the space in the middle, and two Buttons at the bottom. in **addition** an image somewhere to display user progress showing happy, medium, sad face as time to type goes on. for example on first game, leave it happy and take the time it takes to solve (example 15 seconds). on the second game, after half time (7 second) show medium expression, after 15 seconds, show sad face etc.

When you launch your project in the emulator, be sure to save your work and then re-run the project in the emulator. **Don't close the emulator!** We're going to upload new versions of the app as we go along.

Step 2. Handle button clicks

When the user clicks the "Submit" button, we want to check whether she typed the sentence correctly in the EditText. We will do this by handling the button click and then comparing the String they typed to the target string.

See "Input Events" in the developer guide if you need help.

In your method, get the text from the EditText (where the user typed the sentence) and compare it to the String that the user was supposed to type, which is in the TextView at the top of the screen. To do this, you need to use the findViewById method to get each View object using its ID, which you can find in the activity_main.xml file: the TextView's ID is "target" and the EditText's is "userinput". Once you get the object, cast it to the appropriate type, and then use the getText method to see what is there, we did it in class. Something like:

```
EditText inputView = (EditText)findViewById(R.id.userinput);
String input = inputView.getText().toString();
```

Once you have both Strings, compare them and then display a Toast indicating whether the user typed the sentence correctly:

```
Toast.makeText(this, "That's right!", Toast.LENGTH_LONG).show();
```

See "Toast Notifications" in the developer guide for more information.

Handling the "Quit" button is easy. Implement the method that should be called when the "Quit" button is clicked (you can see that it is specified as "onQuitButtonClick" in activity_main.xml) in the same Java class as your "onSubmitButtonClick" method. In the method to handle quitting the game, simply call finish(); to exit the application.

Run your application in the emulator again and test that the button clicks are handled correctly. Check that you get the right message when you correctly enter the sentence, and also when you do it incorrectly. Also check that the user can quit the application.

NOTE! If your version of the emulator is having trouble with keyboard input (making it impossible... or very frustrating, at least... to play the game), then there is a better way. Go to Manymo.com and sign up for a free account. Then click "Launch Emulators" in the top right, click "Launch with: App" right underneath that link, and then click the "Upload an App" link on the next screen. When asked to upload a file, go into the bin/ directory of your Android project in the studio workspace, and choose the file with the .apk extension. Then choose an emulator to launch and you can play your typing game!

Step 3. Add "Are you ready?" Dialog

Now we're going to start adding functionality to measure how long it takes to type the sentence. We need some way for the user to indicate that she is ready (as opposed to just starting the clock as soon as the application starts), so we will show a Dialog asking "Are you ready?", like this:

Dialogs are covered in the "Dialogs" section of the developer guide, but the basics are described here:

In your Java class that extends Activity, create a field (member variable) like this:
private static final int READY_DIALOG = 1;
This will be the ID that we will use to show the Dialog.

Next, in the onCreate method, add the method call showDialog(READY_DIALOG); at the end. This will attempt to show the Dialog when the Activity starts.
Last, you need to write the method that will actually construct the Dialog object and display it. This can be a bit tricky, so here's the code that you need:

```
  protected Dialog onCreateDialog(int id) {
        if (id == READY_DIALOG) {
              AlertDialog.Builder builder = new AlertDialog.Builder(this);
           // this is the message to display
              builder.setMessage("Are you ready?");
           // this is the button to display
              builder.setPositiveButton("Yes!",
                    new DialogInterface.OnClickListener() {
                 // this is the method to call when the button is clicked
                    public void onClick(DialogInterface dialog, int id) {
                     // this will hide the dialog
                        dialog.cancel();
                    }
                   });
              return builder.create();
        }
        else return null;
   }
}
```

Run your application and make sure that the "Are you ready?" Dialog appears when the application first starts, and that the Dialog disappears when you click the "Yes!" button.

Step 4. Time how long it takes

Now we can time how long it takes for the user to type the sentence. We will do this by capturing the current time when the user clicks "Yes" in the "Are you ready?" Dialog, capturing the current time when the user clicks the "Submit" button, and calculating the difference.
To get the current time, use the System.currentTimeMillis(); method. This returns a long indicating the number of milliseconds since January 1, 1970. I

f you capture the start and end times, you can just subtract to find out how many milliseconds have elapsed.

Modify the Toast that is shown when the user correctly types the sentence so that the elapsed time is displayed in seconds (not milliseconds), to one decimal place. You'll have to do a bit of math here! It should look something like this:

Step 5. Add "Try again" Dialog

At this point, the application only figures out how long it takes to type the sentence the first time. If you try it again, the time elapsed is measured since the user exited the "Are you ready?" Dialog. So let's change it so that, instead of showing a Toast to the user after she clicks submit, we show a Dialog with a button on it, and allow her to reset the clock.

In your Java class, create two new fields (member variables) like this: private static final int CORRECT_DIALOG = 2; and private static final int INCORRECT_DIALOG = 3; These will be the IDs that we will use to show the "you got it right" and "you got it wrong" Dialogs, respectively.

Modify the "onSubmitButtonClick" method as follows: If the user correctly entered the sentence, rather than showing the Toast, call removeDialog(CORRECT_DIALOG); and then call showDialog(CORRECT_DIALOG); Be sure to call both methods, otherwise you may get some unexpected behavior.

Also modify that method such that, if the user typed the sentence incorrectly, you call removeDialog(INCORRECT_DIALOG); and then call showDialog(INCORRECT_DIALOG);

Again, be sure to call both methods.

Now you need to change onCreateDialog. Start with the code that is there and modify it by adding two new conditions to the "if" statement that is inspecting the value of the id parameter. If id is equal to CORRECT_DIALOG, then you want to use the AlertDialog.Builder class to create a Dialog that indicates that the user typed the sentence correctly, and displays the time it took in seconds; on the other hand, if id is equal to INCORRECT_DIALOG, then you want to use the AlertDialog.Builder class to create a Dialog that indicates that the user typed the sentence incorrectly.

In either case, when the user clicks the "OK" button in the Dialog, you should clear the text in the EditText (by calling its setText("") method, reset the timer, and let the user keep playing the game.

Here's a screen shot of how it should appear when the user correctly enters the sentence:

Run your application again. At this point, it should:

- show the "Are you ready?" Dialog when it starts
- indicate whether the user correctly or incorrectly typed the sentence (in a Dialog, not a Toast)
- if the user correctly typed the sentence, indicate the elapsed time in the Dialog
- clear the EditText, reset the timer, and let the user keep playing the game

---

Step 6. Keep track of best (lowest) time

The next feature you'll add is the ability for the application to keep track of the best time (i.e., the lowest time elapsed).

Add a variable to the Java class to track the best time recorded so far. If the user correctly types the sentence, compare the elapsed time to the best time. If the user beat the best time, indicate that in the Dialog that is displayed, and update the best time; if the user didn't beat the best time, just show what the best time so far is.

Note that you should not display the "top score" message if this is the first time the user has entered the particular sentence. Also, you don't need to record top scores across program executions, only for the current invocation of the program.

Step 7 Adding sentence generator

The next step is to add code to generate sentences. You can use any system to generate sentences of length 4, 7, 10 words. see for example:
http://codegolf.stackexchange.com/questions/21571/generate-an-understandable-sentence
or
http://udel.edu/~caviness/Classb/CISC367-99S/example-progs/randomSentences/RandomSentences.java

http://www.raywenderlich.com/56109/make-first-android-app-part-2:
pictures, share, welcome page

here also add 3 faces, 15s delay and half 2nd time

Step 8 Adding levels

Now add a dialog which allows the user to choose easy, medium, and hard levels. you now need to track high low scores for each category.

myActivity
intent

Add screen for scores and user settings

add a screen to show off fastest and slowest times for each level of difficulty. also add a method to generate some kind of scrore of the target sentence. for example, can score characters at edge of keyboard more than characters at center (or some other scheme so that “zebra” is scored different than “apple”. now fold in score + time to the game

Add user names

Now add a screen to allow the user to enter their name as a “login” so that scores can be associated with specific names. You will need to also allow the user to “logout”. You don’t need a password, just a way to collecting their names and keep track of high/low per name. Adopt the score screen to show this new information.

Extra Credit:

capture something from the phone after the game is done. either short microphone or snap a picture of the user. associate the information with the user’s score and allow them to replay the image/sound when they view the scores.

Just for fun....

If you develop an Android app in studio and want to install it on an actual device, you can avoid all the messiness of using drivers and cables and just email it to yourself.

In the workspace (on disk), you'll find a directory for your project, and in the "bin" directory you will find a file with the extension .apk. I

f you just email it to yourself, and then check your email from your Android device, you can install it directly.

Android will complain that you shouldn't go installing apps that don't come from Google Play, but if you trust yourself, you can go ahead and install it anyway.
If Android complains about your app being of a higher version than your device, you can modify the minSdkVersion setting in AndroidManifest.xml and set it to a lower version number.

Submission

For this particular assignment, please follow these specific instructions:

Please include a plain-text README file that lists: your name, the Android SDK and studio versions that you used for your assignment; your host operating system (Mac, Linux, Windows, etc.); and, any known issues with your program. This file should be in the root directory of your studio project (not its parent!).

You need to submit the entire studio project for your application. Go into the parent directory of the studio project on your file system (not the directory itself!), and tar or zip the project directory. If you don't know how to do this, ask someone for help!
Please be sure that you are only submitting one zip or tar file that includes your studio project and your README file.

Failure to properly follow the submission instructions could result in a delay of grading your assignment and/or a lateness penalty, so please be sure to do it correctly!

this should be your own work. if you find code online that you want to use (sentence generator etc) be sure to cite source in your README file.

please reach out for help if you get stuck.