Stanford University, CS 193A
Homework 3: Multiple Activities; State; Data Files; Camera; etc.

The purpose of this assignment is to practice the material from the past few weeks, such as apps with multiple activities, apps that handle instance state and data, and interacting with the device's camera. As with past assignments, we'll suggest some ideas for programs you could write to practice layout. We will list several suggestions, but you **only need to write one app** to submit.

If you have a different idea for a program you want to make that will still allow you to explore these topics, please feel free to do something other than the suggestions provided. Another great option would be to **revisit** one of your past assignments and add features to them that use the new material, and re-submit that as your HW3. As long as you try to follow the constraints described in this document *(see the "make up your own" section later)*, you will receive credit even if your HW3 submission is just an enhanced version of a previous submission.

Turnin and Grading:

Instructions for turning in this program can be found on the class web site. After programs are turned in, you will be asked to **peer-evaluate** another student's submission.

Please turn in a **.ZIP file** containing the entire contents of your Android Studio project folder. Give this file a descriptive name of **hw3-*sunetid-description*.zip**, such as, **hw3-jsmith12-Friendsr.zip**.

Your submission will be graded quickly by simply running it and evaluating its functionality. It does not need to be perfect or bug-free to receive credit. Your code will not be graded on style, but we still encourage you to follow good overall coding style for your own sake. If you want to see some good examples of proper Java coding style, consult the **Style Guide** linked from the Homework web page.

Assignment Ideas and Suggestions:

Suggestion 0: Make Up Your Own

If you don't like our suggested assignment ideas listed on the following pages of this document, or if you prefer to do something unique of your own, please feel free to do so. Whether you do our suggestions or your own, we'd prefer to see an app that has the following qualities:

- Your app should be set up as an **Android Studio** project, so it can easily be opened/run/graded by others.
- Your app should use at least **2 different activities**.
- Your app should pass at least one piece of **"extra" data** between the activities.
- Your app should gracefully handle rotation from portrait to landscape **orientation**.
- Your app should save some kind of **state** so that it behaves well when the user exits and comes back later.
- Along with your app (inside the ZIP), turn in a file named **`README.txt`** that contains your name and email address along with the name of your app and a very brief description of it, along with any special instructions that the user might need to know in order to use it properly (if there are any). For example:

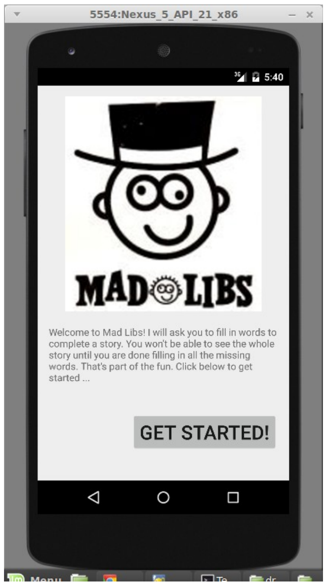
```
Joe Student <jstudent@stanford.edu>
NumberGame 2.05 - This app shows two numbers on the screen and asks
the user to pick the larger number.  Perfect for Berkeley students!
```

As always, these assignments, as well as this class in general, are meant to be **low-stress** and fun. If you want help, please feel free to show your code to others or ask for help in our online **Piazza forum**. Feel free to make an app as simple or as complex as you like, relative to your familiarity level and time constraints. If you work on your solution for roughly **2 hours** and are still not done, you can turn it in and we will award you credit.

Suggestion 1: Mad Libs (text processing app with 2-3 activities)

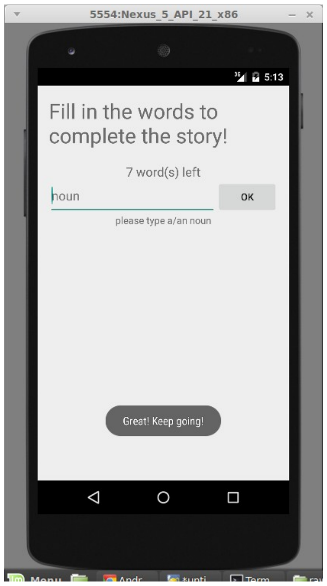
"Mad Libs" are short stories that have blanks called **placeholders** to be filled in. In the non-computerized version of this game, one person asks a second person to fill in each of the placeholders without the second person knowing the overall story. Once all placeholders are filled in, the second person is shown the resulting silly story.

Write an Android app that reads in a Mad Lib from a text file in a specific format. The text file represents placeholders as tokens that start and end with `< >` brackets, like `<adjective>` or `<proper-noun>`. Your app reads the file, looks for any such placeholders, and prompts the user to replace them with specific words. Once the user has typed in replacements for all placeholders, the completed story is shown on the screen. The screenshots below indicate a possible flow of the UI for such an app. Our flow has **three activities**: An initial "welcome" screen explaining the app, then a screen that repeatedly prompts the user to fill in placeholders, then a third activity to display the completed story. Of course you don't need to exactly match our sample's UI, but it may give you ideas.



Welcome to Mad Libs! I will ask you to fill in words to complete a story. You won't be able to see the whole story until you are done filling in all the missing words. That's part of the fun. Click below to get started ...

→



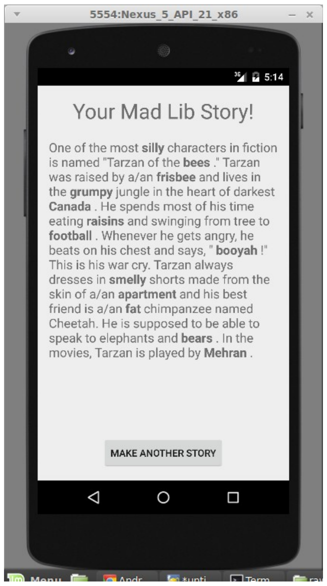
Fill in the words to complete the story!

7 word(s) left

noun

please type a/an noun

→



Your Mad Lib Story!

One of the most **silly** characters in fiction is named "Tarzan of the **bees** ." Tarzan was raised by a/an **frisbee** and lives in the **grumpy** jungle in the heart of darkest **Canada** . He spends most of his time eating **raisins** and swinging from tree to **football** . Whenever he gets angry, he beats on his chest and says, " **booyah** !" This is his war cry. Tarzan always dresses in **smelly** shorts made from the skin of a/an **apartment** and his best friend is a/an **fat** chimpanzee named Cheetah. He is supposed to be able to speak to elephants and **bears** . In the movies, Tarzan is played by **Mehran** .

On the course web site we provide several Mad Lib story text files you can use, such as `madlib1_tarzan.txt`. Here is the text of that file, to give you an idea of the Mad Lib format:

```
One of the most <adjective> characters in fiction is named
"Tarzan of the <plural-noun> ." Tarzan was raised by a/an
<noun> and lives in the <adjective> jungle in the
heart of darkest <place> . He spends most of his time
eating <plural-noun> and swinging from tree to <noun> .
Whenever he gets angry, he beats on his chest and says,
" <funny-noise> !" This is his war cry. Tarzan always dresses in
<adjective> shorts made from the skin of a/an <noun>
and his best friend is a/an <adjective> chimpanzee named
Cheetah. He is supposed to be able to speak to elephants and
<plural-noun> . In the movies, Tarzan is played by <person's-name> .
```

The code for reading the story text file, breaking it apart, looking for the placeholders, etc. is not particular to Android, so perhaps it is less relevant to this course. To make the assignment more manageable, if you want a head start toward implementing this particular option, we'll give you a file on the course web site called `Story.java` that you can optionally use as a building block. If you put `Story.java` into your project, you can construct a `Story` object and pass it an input stream or `Scanner` and it will read the text data from that source, break the text apart, and find the placeholders for you, etc. The Story object has other methods for filling in the placeholders later. If you use this helper object, you can focus more on the "Androidy" parts of this assignment and less on the string / text processing parts. Or if you want to try to write the story parsing logic yourself, that is fine.

Suggestion 2: Friendsr (online dating app with 2 activities)

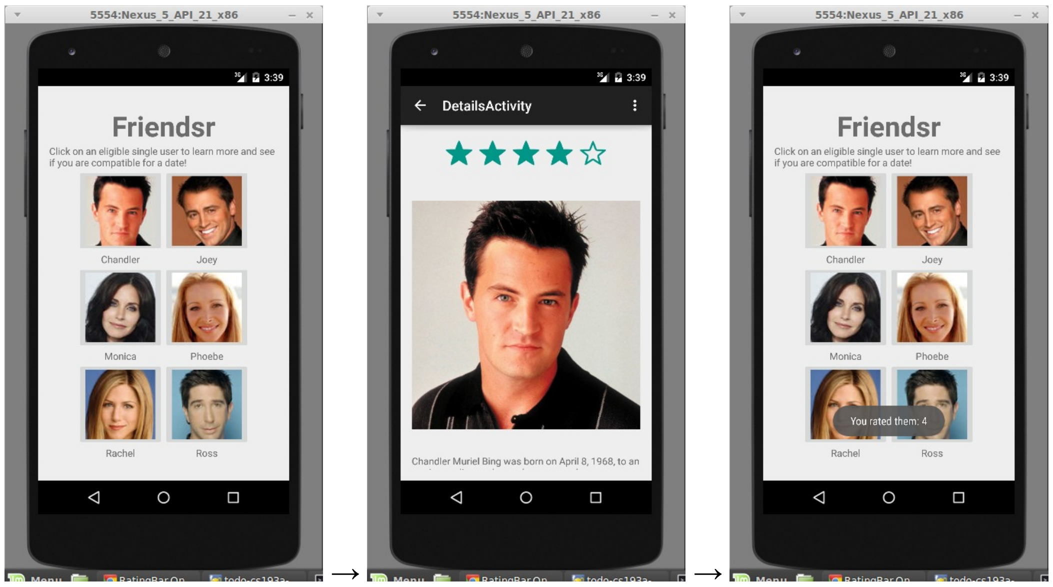
Write an "online dating" type of app, similar to sites/apps like Tinder and OKCupid, or Plenty of Fish. When the user first loads the app, it will show a list of available single users. The user can click on each one to see a bigger photo and get more information about that user. On the screen is a way to rate the user, either some kind of "star" rating system or a "yes/no" vote like what Tinder uses.

The app by nature will be an **incomplete** experience. You don't need to let the user make an actual account, and you can just show them a fixed set of profiles that never changes. (We haven't really learned enough to make a "real" experience for an app like this yet.) For the set of user profiles to show, you could gather your own data and make up your own profiles, or you can download our provided data that uses the six cast members of the 1990s TV show *Friends* as the users. (We'll ignore the fact that many of the *Friends* characters are not single by the show's end!)

Give your app at least **two activities**: One to display all eligible user profiles, and another to view the details of a given profile when it is clicked by the user, such as a larger photo and a description of that user. Such images and text are provided in our ZIP starter archive if you want them for the *Friends* characters. Practice **passing information** back and forth from one activity to another using `Intent`s, such as having the second activity pass back what kind of rating the user gave to the profile. A good way to use the information would be to display all profiles' ratings by the user in the main activity next to that profile's name, or something to that effect.

If you want to make a more complete and robust experience, have your app **remember its state** using `Bundle`s and `SharedPreference`, such as the ratings the user has given to each profile. That way if the user exits the app and revisits it later, the ratings are remembered and still shown on the screen.

If you want your app to play **background music** with the `MediaPlayer` class, we also provide an MP3 version of the *Friends* theme song. Try making the music pause, resume, etc. as the user switches between activities. Incorporate this into your `Bundle` so that the music state is remembered if the user exits and returns to the app.



Click on an eligible single user to learn more and see if you are compatible for a date!

Chandler Muriel Bing was born on April 8, 1968, to an

If you want to implement a **rating bar** of stars like the one shown in our screenshots, look into Android's `RatingBar` view class. You can place a `RatingBar` into a layout and interact with it in Java code by calling its `getRating` method. If you want to trigger an event when the user clicks the stars in the rating bar, it's a bit trickier than attaching an `onClick` handler on a `Button`. You'd need to attach the listener in the Java code instead of in the XML; look into the rating bar's method `setOnRatingBarChangeListener` for more details; or, for simpler coding, you can have a separate "Submit" button that sends the user back to the main activity.

If you want to practice using the **Android camera**, you could make an option for the user to make a profile for themselves and let them take a photo as part of the profile creation.

Copyright © Stanford University and Marty Stepp. Licensed under Creative Commons Attribution 2.5 License. All rights reserved.