Stanford University, CS 193A
Homework 4: Lists; Data files; Text-to-Speech; Fragments

The purpose of this assignment is to practice the material from the past few weeks, such as apps that display lists, read data from files, perform text-to-speech or speech-to-text operations, and/or use fragments. 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 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 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 **hw4-*sunetid-description*.zip**, such as, **hw4-jsmith12-TalkingMadLibs.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** or at least **2 fragments**.
- Your app should gracefully handle rotation from portrait to landscape **orientation**.
- 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 4.15 - 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**(light)*****: To-Do List**

Write a simple to-do list app that has a `ListView` of tasks that the user needs to complete. Initially the app is empty and has nothing in the to-do list. But if the user types text into a bottom `EditText` and clicks an Add button, the new item will be added to the top or bottom of the list.

It's also good to have a way to **remove items** from the list. You could achieve this by attaching a listener to the list that removes an item when that item is clicked on by the user. Or if you want to try something slightly different, try making it remove an item when the user performs a "long click" (pressing and holding the mouse on an item). You can do this by calling the `setOnItemLongClickListener` method of your list and passing an anonymous `AdapterView.OnItemLongClickListener` class. Android Studio can help you auto-generate the skeletons of these anonymous listener classes if you press Ctrl-Space in the editor at the right place in the code.

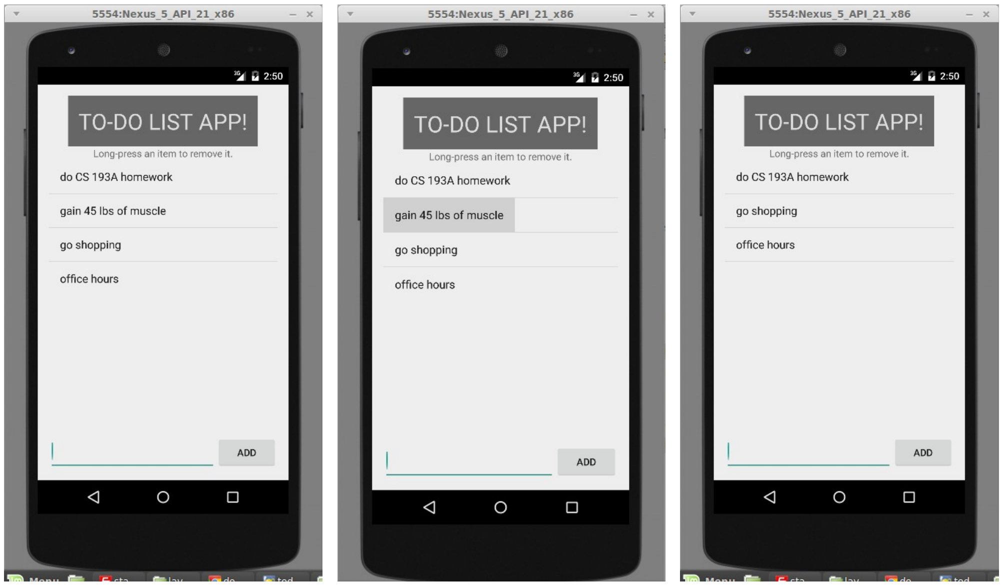


Figure: User long-clicking on second list item to delete it

If the items in your to-do list are stored into an `ArrayList`, by default the app's GUI won't notice when you add or remove an item from the list. That is, you'll modify the ArrayList state but the graphical list on the screen won't update to match. To fix this, you have to call the method `notifyDataSetChanged()` on your `ArrayAdapter` to tell it that the underlying array list has changed. To be able to do this, of course, you'll have to save your `ArrayList` and your `ArrayAdapter` as private fields inside your activity.

Make sure that the app doesn't forget your to-do items when you **rotate the screen** back and forth from portrait to landscape mode. To make a more robust app, you should also consider saving the state of the to-do list when the app is exited and re-entered later. Use a `Bundle` to save the list's data. If you want to make the app as robust as possible, you'd want to make it also save the to-do list to the device's storage so that it will survive a reboot. To do this, you'd need to write code that reads/writes files to the system's internal storage:

```
// write to a file
PrintStream out = new PrintStream(openFileOutput("filename.txt", MODE_PRIVATE));
out.println(text);
out.close();

// read a file
Scanner scan = new Scanner(openFileInput("filename.txt"));
while (scan.hasNextLine()) {
    String line = scan.nextLine();
    do something with line;
}
```

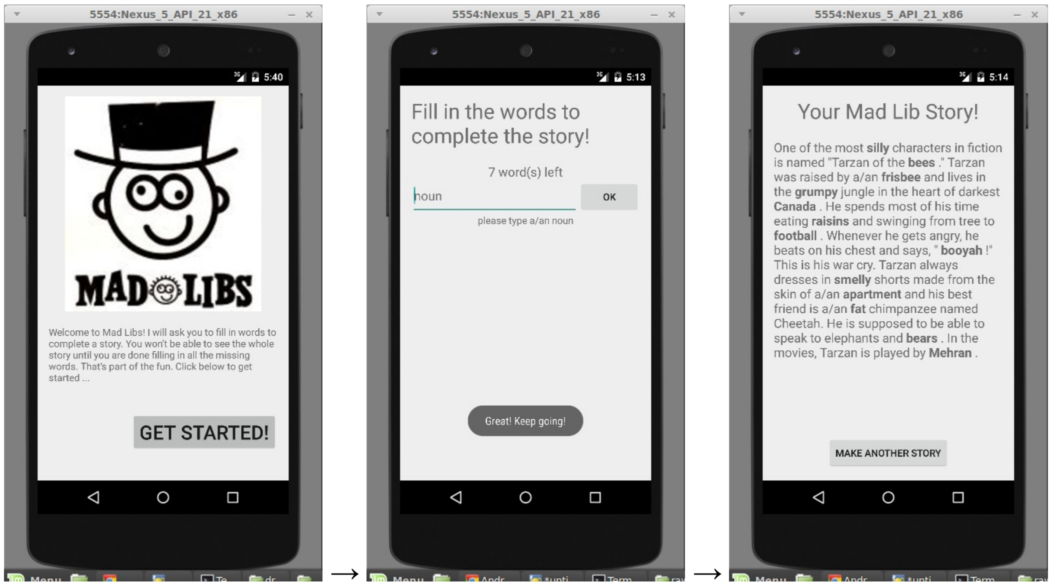
If you want to add **text-to-speech** to this app, such as making it read the to-do list aloud, please feel free to do so.

Suggestion 2 (light): **Talking Mad Libs**

This suggestion is only practical if you did the "Mad Libs" program for HW3. For your HW4, improve the program by adding features we've just learned. Here are the kinds of features to add:

- Provide an initial **list** as a `ListView` (or drop-down list, represented by a `Spinner`) that shows all the mad lib files (such as madlib1_tarzan.txt, madlib2_university.txt) and lets the user pick one by clicking on the list.
- Use **text-to-speech** so that in addition to showing prompts as text, like "Please type an adjective," the program will also speak those prompts aloud.
- Add a "Talk" button that uses **speech-to-text** so that you can speak your word rather than typing it manually using the Android keyboard.

If you didn't do Mad Libs for your HW3, you can still do this option, but you'd need to implement the Mad Libs assignment as described in HW3 first, which might take too much time. It's up to you.



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

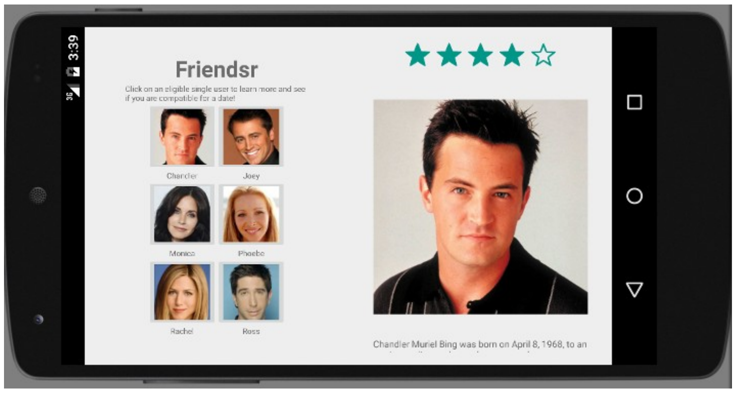
please type a/an noun

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** .

Suggestion 3 (medium): **Friendsr with Fragments**

If you did the Friendsr dating app for HW3, modify it to use **fragments** so that when the screen is rotated into landscape orientation, it shows the details about the user in the same activity, rather than jumping into a new activity. This would be very similar to what we did in class with our most recent version of the TMNT app. When the app is in portrait mode, it should follow the old behavior of having the user list and user details be separate activities.



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