# CMSC 436 Lab 5

# Content Providers and Saving Data

# Overview

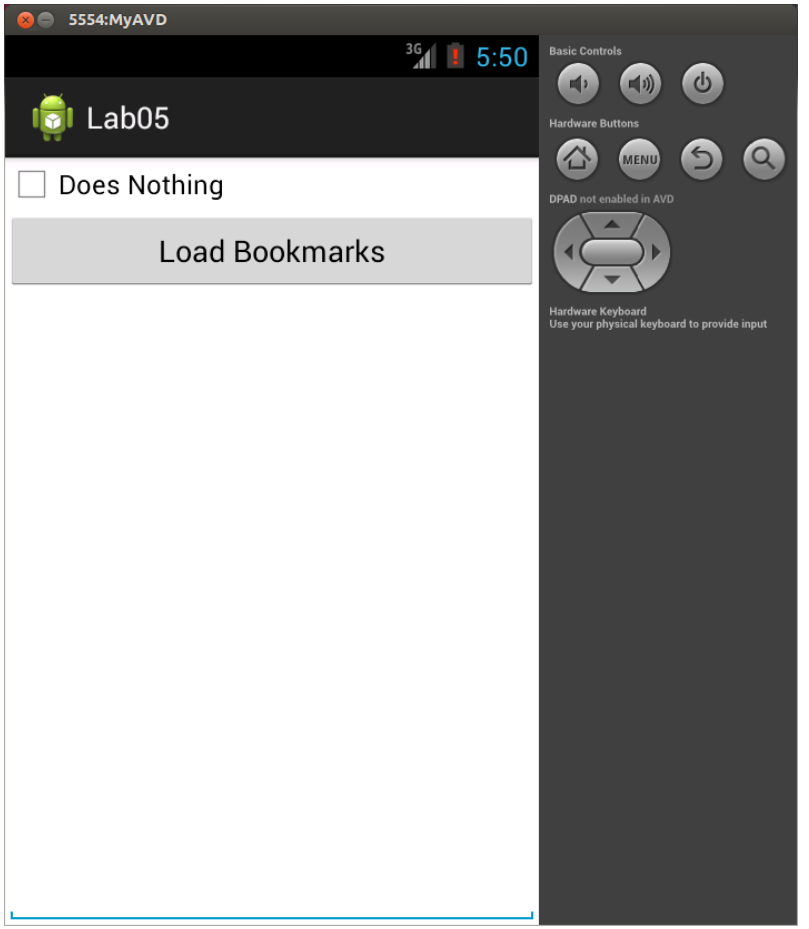
- For this lab you will create an app that saves the state of a check box and a text box when the app is destroyed, and restores them when the app is recreated
- The check box state will be saved as a key-value pair using a SharedPreferences object, and the text box contents will be stored in a file in the device's Internal Storage

# Overview

- You will also use a Content Provider to read the bookmark titles and URLs stored on the device and write them into the text box
- See the following links for more information
  http://developer.android.com/training/basics/data-storage/shared-preferences.html
  http://developer.android.com/training/basics/data-storage/files.html
  http://developer.android.com/guide/topics/providers/content-provider-basics.html

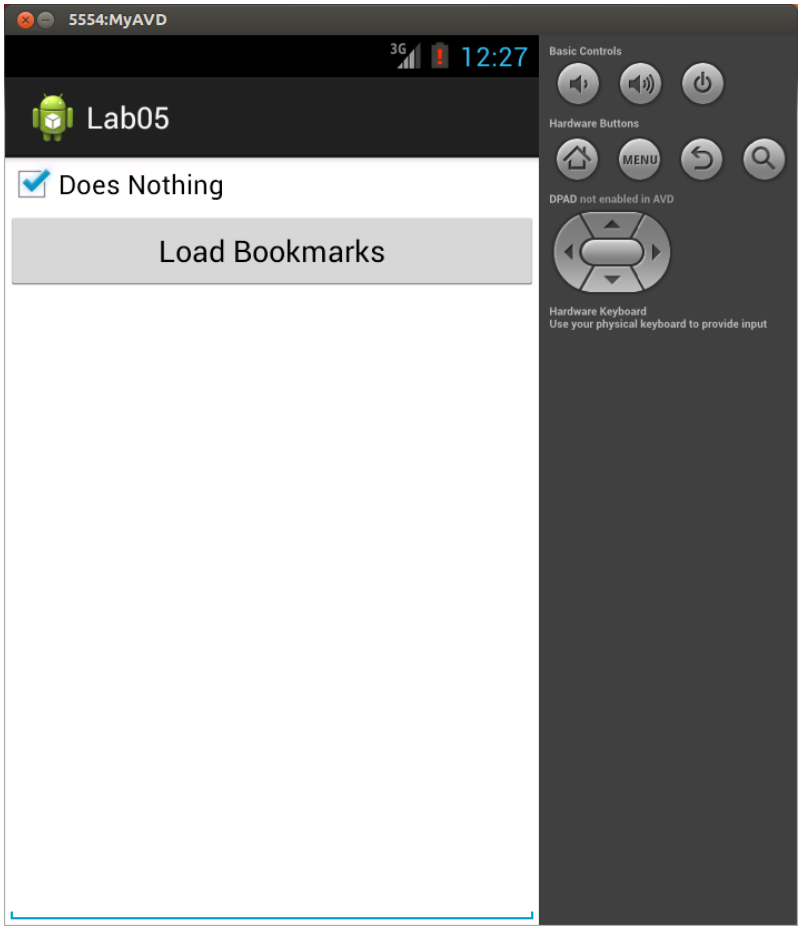
# Overview

- The app should initially contain a check box, a button, and a large text box (the EditText widget)



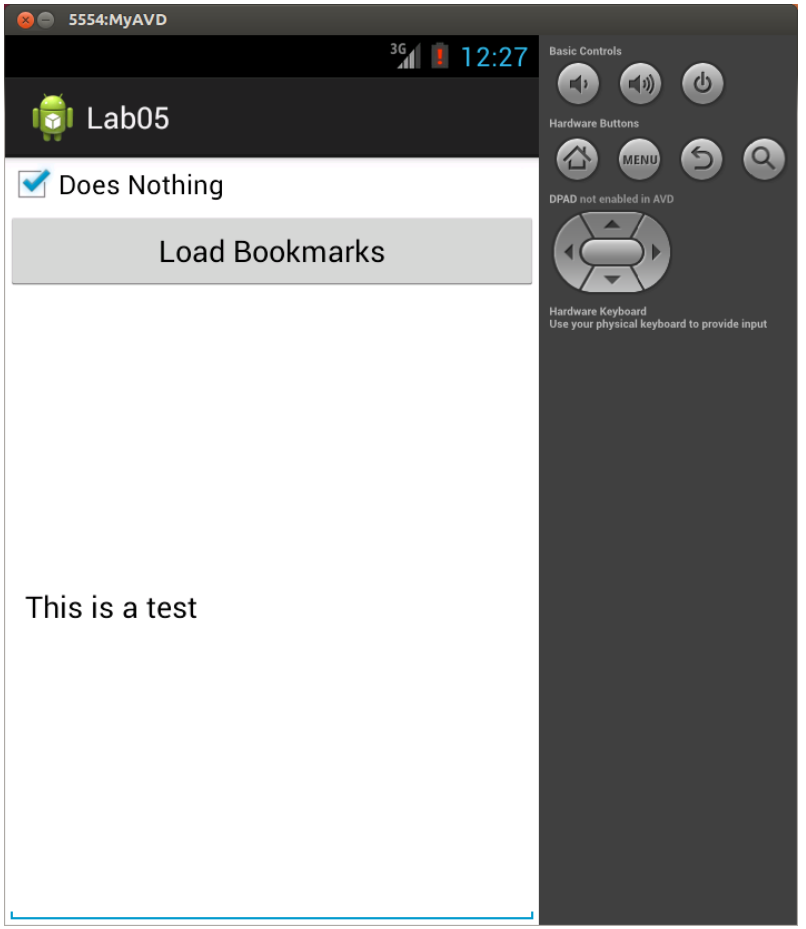
# Overview

- The state of the check box should persist when the activity is backed out of and restarted



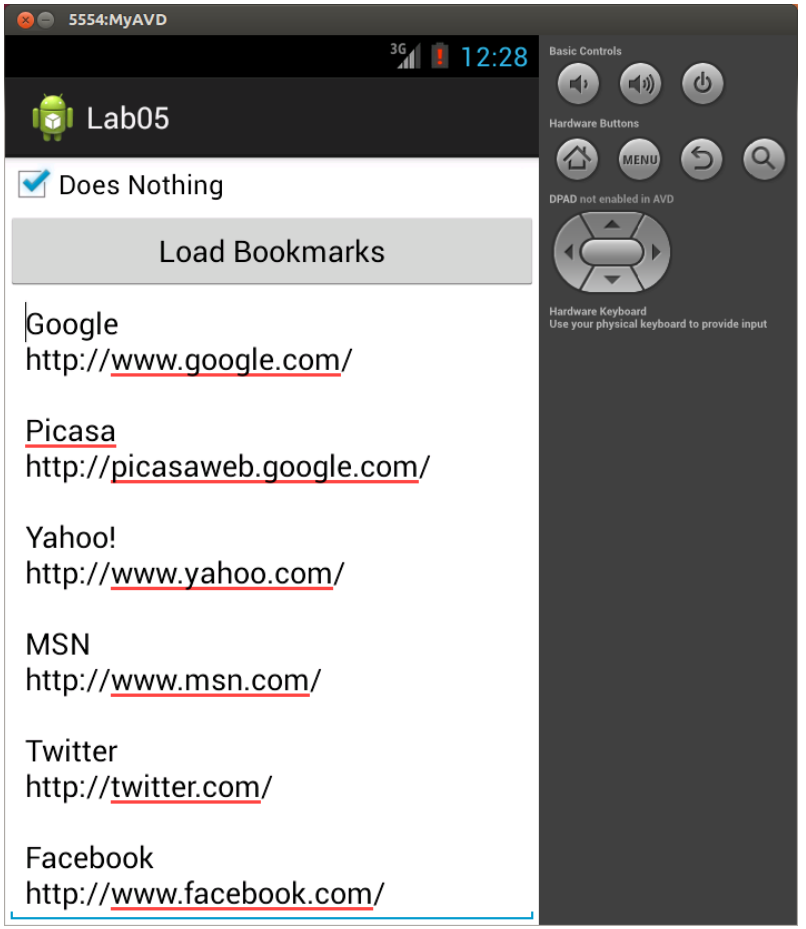
# Overview

- The user should be able to enter text in the text box, which should also persist when the app is backed out of and restarted



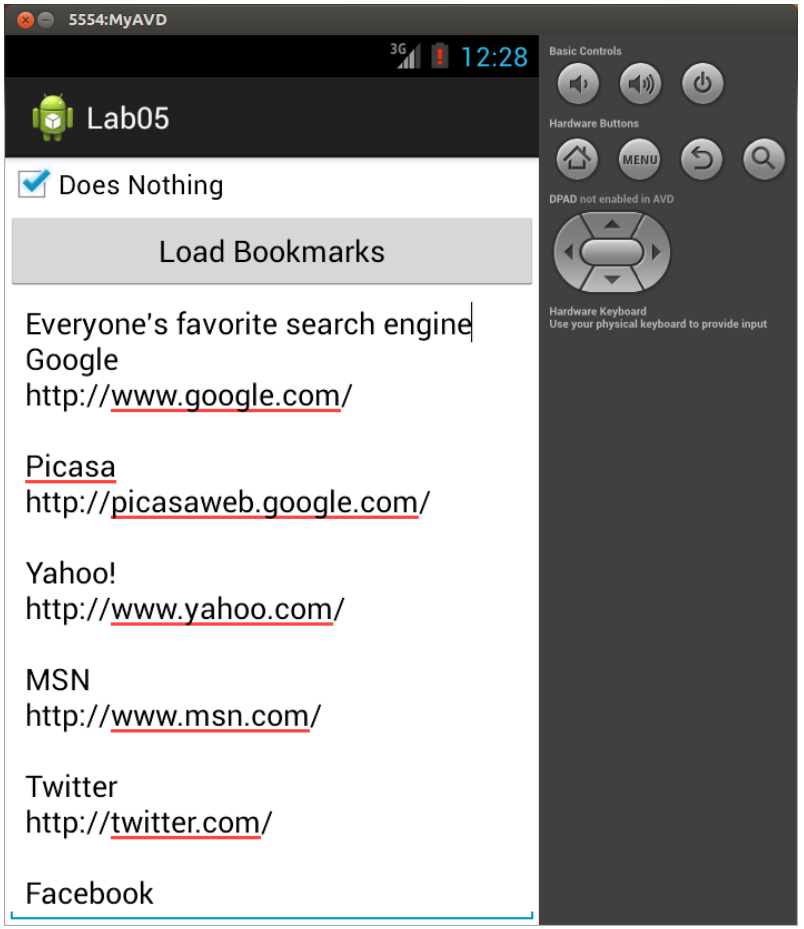
# Overview

- Clicking the button should cause the bookmark titles and URLs on the device to appear in the text box



# Overview

- The user should be able to edit the contents of the text box
- As before, this text should persist across restarts



# Shared Preferences

- The check box state should be stored during onPause(), since this is the only callback you are guaranteed to receive before the app is destroyed
- The state can be restored in onResume()
- The code for this is very straightforward- see the Saving Key-Value Sets example in the Saving Data training on the Android developer site

# Internal Storage

- Each app has its own private Internal Storage area that is automatically removed when the app is uninstalled
- The contents of the text box should be written to a file in the app's Internal Storage during onPause(), and read back during onResume()
- For information on using Internal Storage see the Saving Files example in the Saving Data training on the Android developer site

# Internal Storage

- Android can provide the Internal Storage area directory as a File object or can provide a FileInputStream or FileOutputStream
- The FileOutputStream can be written to directly or wrapped in a BufferedOutputStream
- The FileInputStream can be wrapped in an InputStreamReader which can be wrapped in a BufferedReader

# Internal Storage

- To view files stored in Internal Storage you can use the DDMS perspective in Eclipse (open the perspective with the button at the top right corner)
- Select the device in the left panel, and then select the File Explorer tab on the right
- The Internal Storage for the app will be located in, for example, /data/data/com.example.lab5/files

# Content Provider

- When the button is clicked, the titles and URLs of the bookmarks stored on the device should be written to the text box
- This information can be read from a Content Provider
- Information on Content Providers can be found in Content Provider Basics in the Content Providers API Guide on the Android developer site, but these slides should contain most of the information you need for this part

# Content Provider

- First note that to access the bookmark information you must include the appropriate uses-permission tag in your AndroidManifest.xml

```
<uses-permission
android:name="com.android.browser.permission.READ_HISTORY_BOOKMARKS"/>
```

# Content Provider

- To read the data you will need to call getContentResolver() and on the result of this call query() to get a Cursor that will iterate over the bookmarks
- The URI parameter for the query() specifies which data table you want to access and should be Browser.BOOKMARKS_URI

# Content Provider

- The projection parameter for the query() specifies which data columns you want to retrieve- in this case you want the title and URL, so you should pass in an array of strings like

```
String columns[] = new String[] {
        Browser.BookmarkColumns.TITLE,
        Browser.BookmarkColumns.URL
};
```

# Content Provider

- The remaining parameters for the query() specify which rows and what sort order you want- since you want all rows and the sort order doesn't matter, these parameters can all be null

# Content Provider

- Once you have the Cursor from the query() you can call moveToFirst() to begin the iteration and continue iterating while moveToNext() returns true
- At each iteration you can read the title and URL columns of the current row from Cursor c like

```
String title = c.getString(c.getColumnIndex(Browser.BookmarkColumns.TITLE));
String url = c.getString(c.getColumnIndex(Browser.BookmarkColumns.URL));
```