NAME: SAKSHI VINAYAK KITTURKAR
CSU ID: 2860273
HOMEWORK 9

Modify the JSON app: instead of displaying the list of verses in a sequence of toasts, store the verses in an array and display them in a ListView nicely as part of the layout for the main activity.

= In this homework we will display the list of verses in a sequence of toasts, and store the verses in an array and display them in a ListView nicely as part of the layout for the main activity.

In layout: MainActivity.java This code is an Android application written in Java using the Android SDK. It fetches Bible verses in Portuguese from a specific API endpoint, processes the JSON response, and displays the verses in a ListView.
ReadJSONFeedTask is an inner class that extends AsyncTask to perform background network operations.
doInBackground method fetches the JSON data from the specified URL using the readJSONFeed method.
onPostExecute method is called on the main thread after the background task is complete. It parses the JSON data, extracts verse text, adds it to the versesList, and notifies the adapter of the data change.

```
private class ReadJSONFeedTask extends AsyncTask<String, Void, String> {
   protected String doInBackground(String... urls) {
      return readJSONFeed(urls[0]);
   }
 protected void onPostExecute(String result) {
      try {
         // Parse JSON data
         JSONObject jsonObj = new JSONObject(result);
         JSONArray verses = jsonObj.getJSONArray("verses");
         for (int i = 0; i < verses.length(); i++) {
            JSONObject tmp = verses.getJSONObject(i);
            versesList.add(tmp.getString("text"));
         }
      } catch (Exception e) {
         e.printStackTrace();
      }
      adapter.notifyDataSetChanged();
   }
}
```

readJSONFeed method takes a URL as input, makes an HTTP request, reads the JSON response, and returns it as a string.

```
public String readJSONFeed(String address) {
    // ...
    // Code for making an HTTP request and reading the JSON response
    // ...
    return stringBuilder.toString();
}
```

In layout: ExampleInstrumentedTest this instrumented test verifies a basic aspect of the Android app's context, specifically checking the package name. It is a simple starting point for testing the setup of the Android application within the testing framework.

The @Test annotation marks the method as a test case.
The method name, useAppContext(), suggests that it is likely checking something related to the application's context.

```
@Test
public void useAppContext() {
```

Retrieves the context of the app under test using InstrumentationRegistry.
The assertEquals method is used to assert that the package name of the app's context is equal to the expected package name ("com.example.hw9").

```
Context appContext = InstrumentationRegistry.getInstrumentation().getTargetContext();
assertEquals("com.example.hw9", appContext.getPackageName());
```

In layout: ExampleUnitTest This code represents a local unit test for a simple mathematical operation in Java.

This is a standard JUnit test class for unit testing Java code.

```
public class ExampleUnitTest {
```

The @Test annotation marks the method as a test case.
The method name, addition_isCorrect(), suggests that it is likely testing the correctness of addition.

```
@Test
public void addition_isCorrect() {
```

The assertEquals method is used to assert that the result of the addition operation (2 + 2) is equal to the expected value (4).

```
assertEquals(4, 2 + 2);
```

In layout: AndroidMainFest.xml This XML code represents the AndroidManifest.xml file for an Android application. The AndroidManifest.xml file is a crucial configuration file for an Android app, as it contains essential information about the app's components, permissions, and other settings.

Declares the main activity of the app as MainActivity. exported: Indicates whether this activity can be launched by components of other applications. It's set to true.
<intent-filter>: Specifies the types of intents that the activity can respond to.
<action>: Specifies the main action for the activity (MAIN).
<category>: Specifies that the activity is a launcher activity.

```
<activity
    android:name=".MainActivity"
    android:exported="true">
    <intent-filter>
        <action android:name="android.intent.action.MAIN" />
        <category android:name="android.intent.category.LAUNCHER" />
    </intent-filter>
</activity>
```

The Screen looks like this after storing  the verses in an array.