Mobile App Design Project:
Text to Speech Exercise

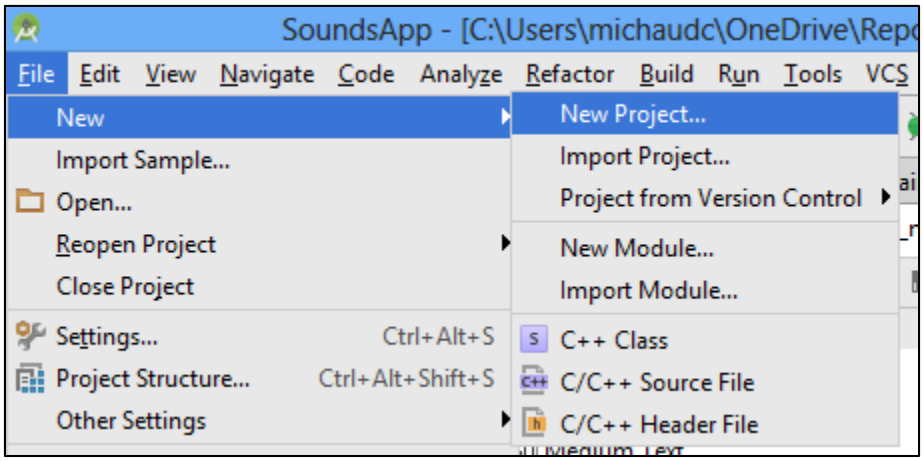
The Android API includes a class that will render text ("strings") as speech. The text to speech API also includes functions to change timbre and rate of the voice. This exercise will have the following Objects:

An instance of EditText named 'enterText' that will allow the user to type in a word or phrase.
An instance of a Button named 'speechButton' that will call a function to speak the string within the 'enterText" EditText object.

Process:

Phase 1: Start a new Android Project

1. Start Android Studio and 'File -> New -> New Project'



2. Fill out the New Android Application form as shown below:
    a. Application Name: SpeechApp
    b. Minimum Required SDK: Android 4.0 (IceCreamSandwich)
    c. Blank Activity
    d. Main Activity is called 'MainActivity'

Phase 2: Write the XML Code to Define the User Interface

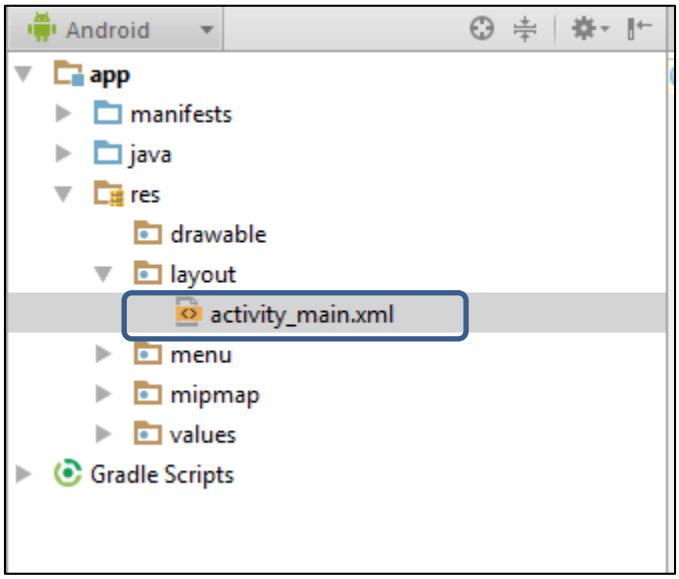
We need to define the User Interface elements with the XML Code. For this project, we will type in the User Interface setup in XML. While Android and Eclipse do provide a GUI interface builder – using the XML will give you more control in the layout and look of the User Interface.

XML Objects in Text To Speech App:

EditText named enterText
Button named speechButton

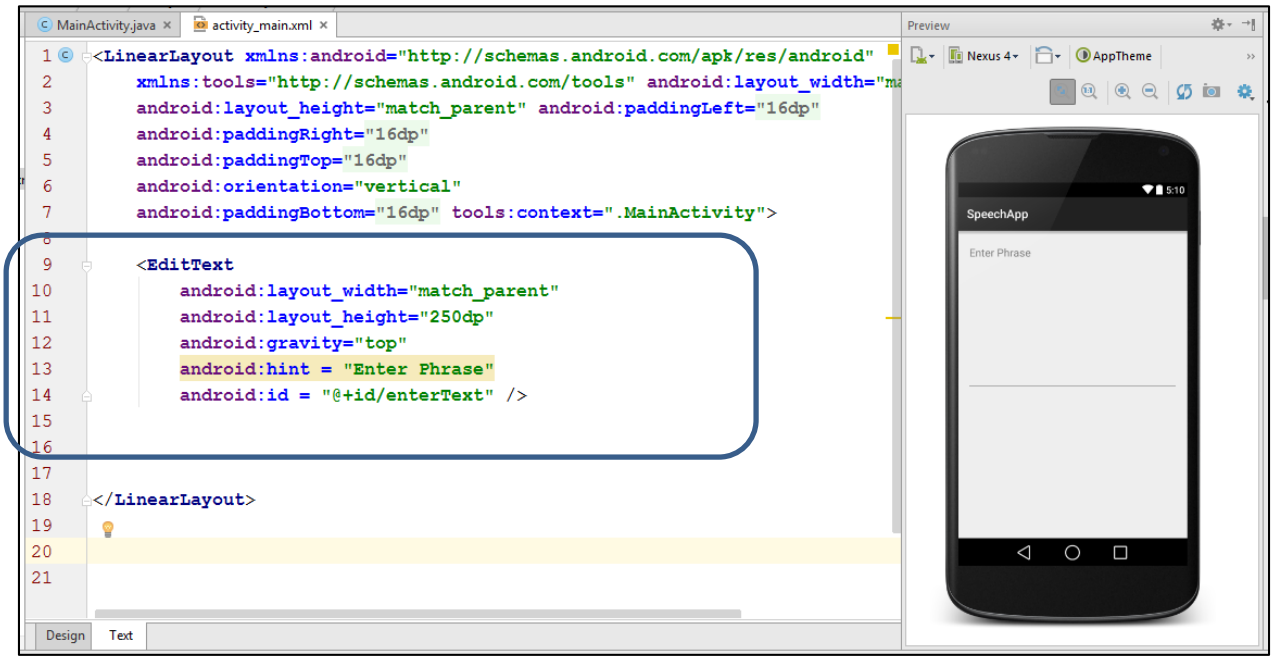
Process:

1. Go to res->layout->activity_main.xml by left clicking on 'activity_main.xml.'

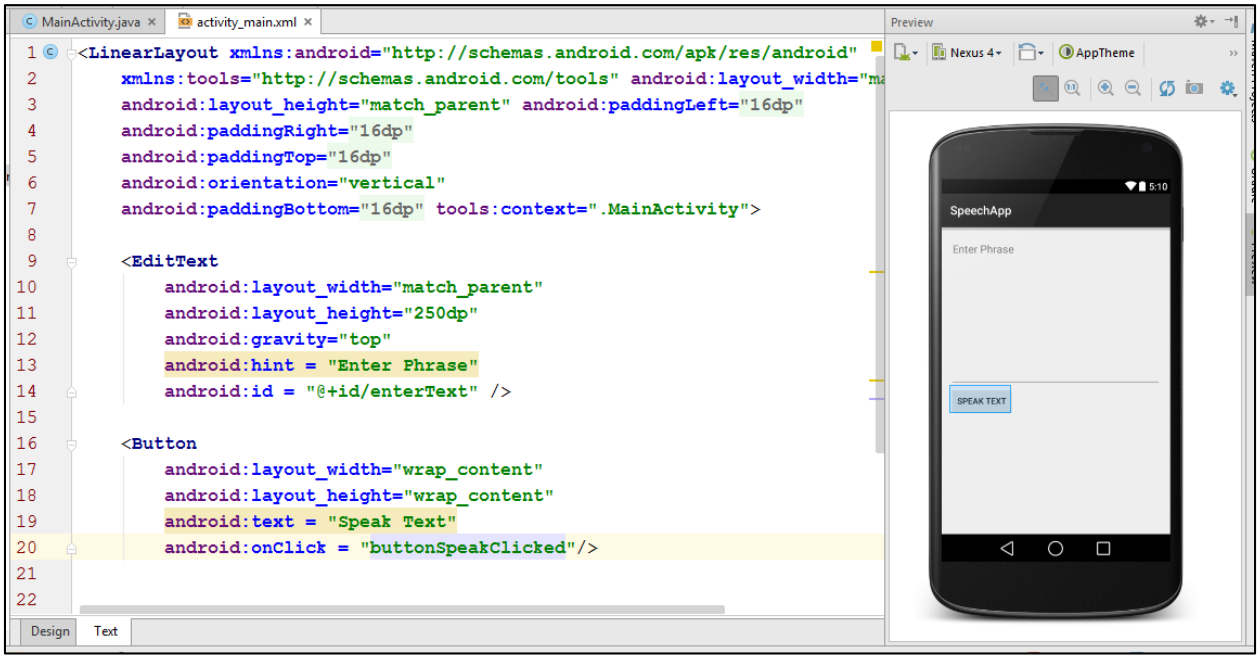


2. Go to activity_main.xml file and change the layout to LinearLayout. Add an attribute ‘android:orientation = “vertical”

3. On lines 9 through 14, code the EditText Object. This will hold the text the user types into the App. Note that we set the height to 250 pixels using the dp value.

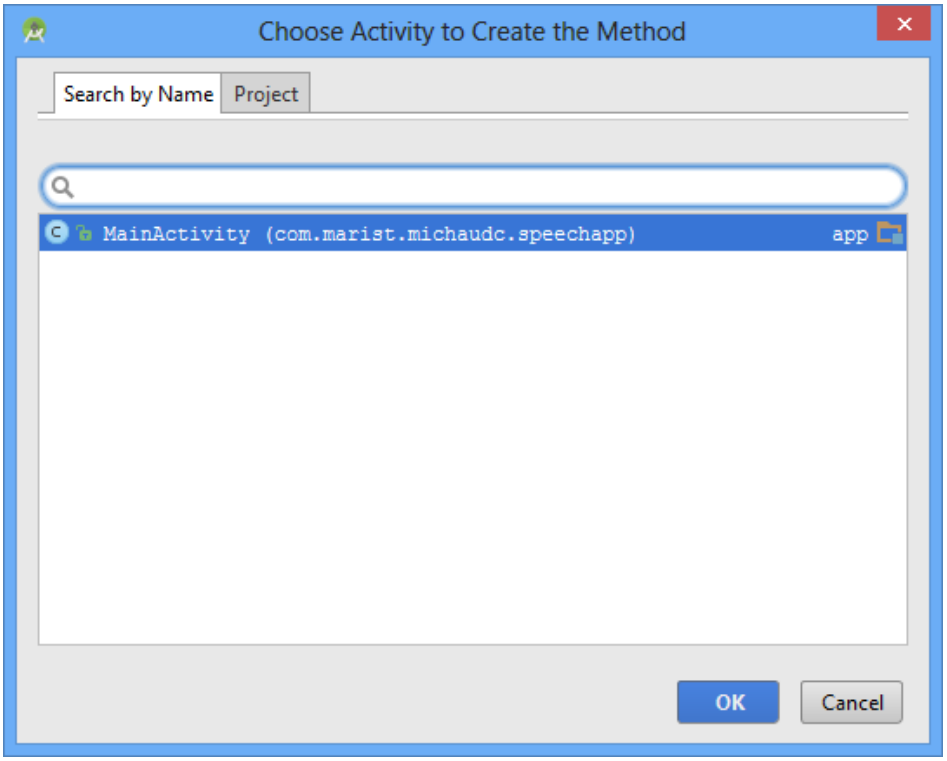


4. On lines 16 through 20, code the speechButton Object. This will call the speak function to start the App speaking:

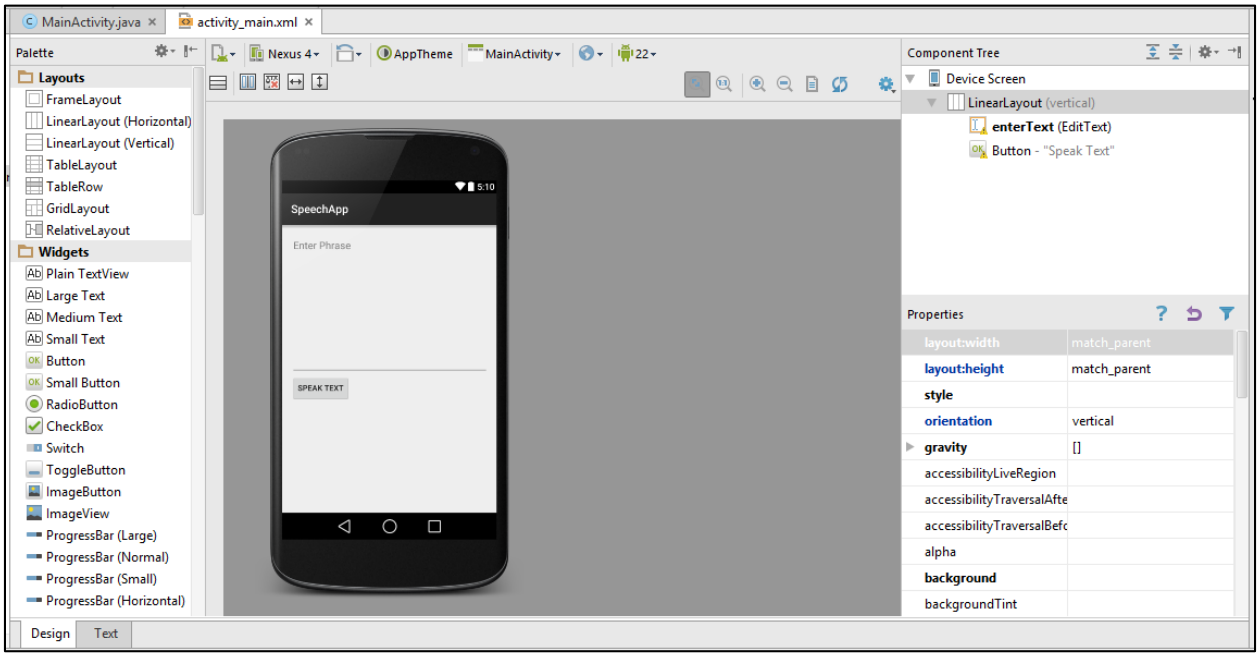



```
1  <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
2      xmlns:tools="http://schemas.android.com/tools" android:layout_width="ma
3      android:layout_height="match_parent" android:paddingLeft="16dp"
4      android:paddingRight="16dp"
5      android:paddingTop="16dp"
6      android:orientation="vertical"
7      android:paddingBottom="16dp" tools:context=".MainActivity">
8
9      <EditText
10         android:layout_width="match_parent"
11         android:layout_height="250dp"
12         android:gravity="top"
13         android:hint = "Enter Phrase"
14         android:id = "@+id/enterText" />
15
16     <Button
17         android:layout_width="wrap_content"
18         android:layout_height="wrap_content"
19         android:text = "Speak Text"
20         android:onClick = "buttonSpeakClicked"/>
21
22
```

Click the 'light bulb' when prompted on the onClick method to add the function to the MainActivity.java.



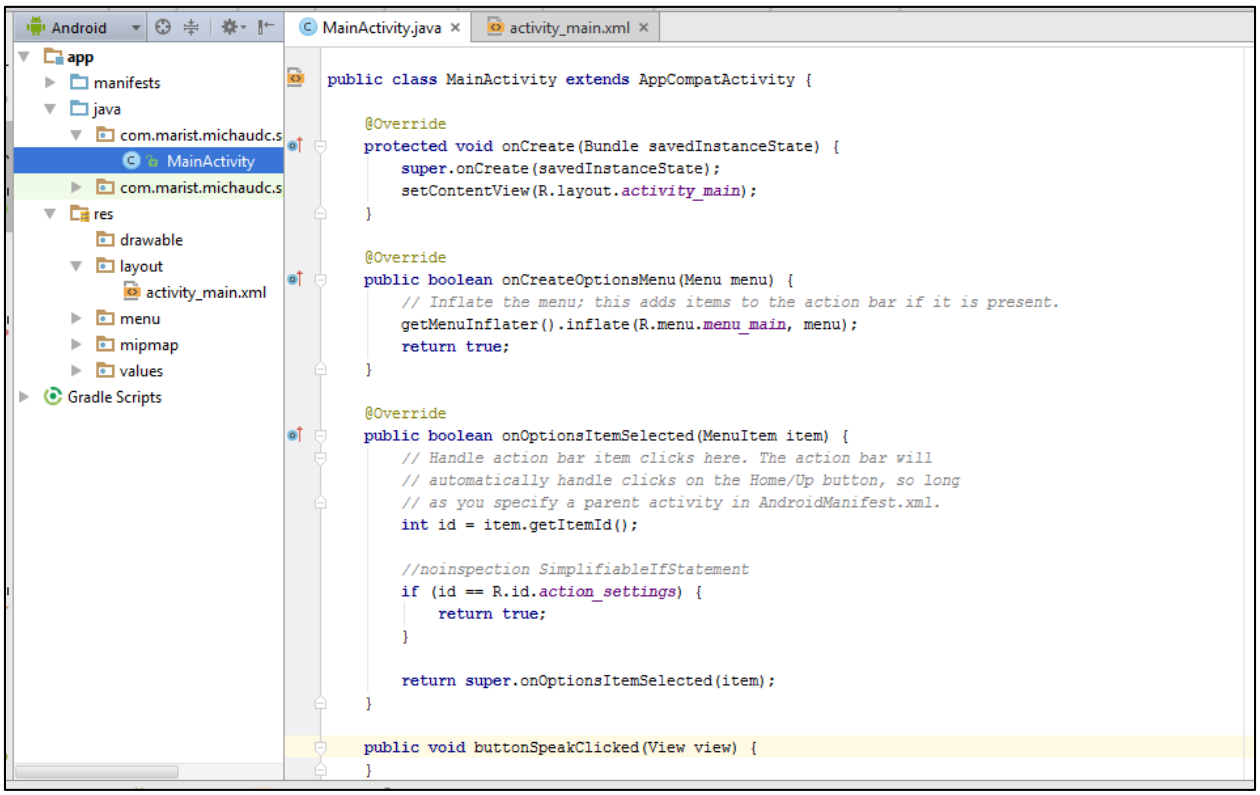
When you are Finished the layout XML and phone should look like this:



Phase 3: Write the Java for the MainActivity.java

Process:

1. Go to the src and find the MainActivity.java file:

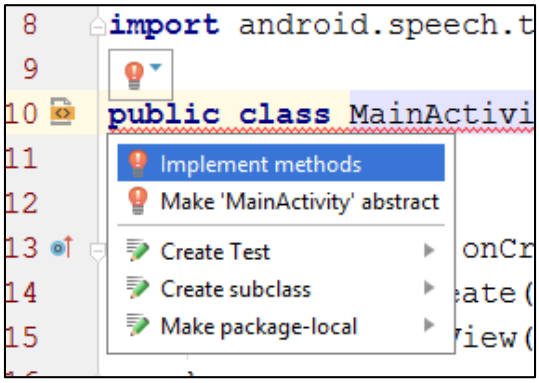


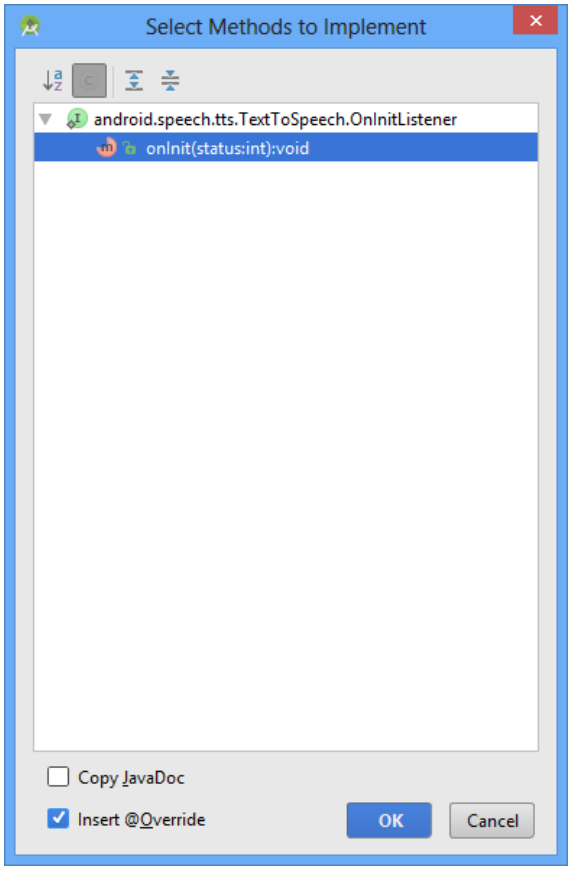
2. Modify the import statements to include the android.speech.tts.TextToSpeech.

3. Modify the public class declaration to extend the Activity type to implement TextToSpeech.OnInitListener

```
9
10 public class MainActivity extends AppCompatActivity implements TextToSpeech.OnInitListener {
11
```

You will see an error on the class declaration. This is because the implementation of TextToSpeech.OnInitListener needs some additional functions. Hover over the error and follow the prompts to fix the additional functions.





4. Define the field for the MainActivity class: (Lines 12 and 13)

```
9
10  public class MainActivity extends AppCompatActivity implements TextToSpeech.OnInitListener {
11
12      // Text To Speech Object in Field
13      private TextToSpeech speaker;
14
15
16      @Override
17      protected void onCreate(Bundle savedInstanceState) {
18          super.onCreate(savedInstanceState);
19          setContentView(R.layout.activity_main);
20      }
21
```

5. Modify the protected void onCreate() to connect and initialize the fields. Remember, the onCreate() acts like the constructor in Android Activities:

```
15
16      @Override
17      protected void onCreate(Bundle savedInstanceState) {
18          super.onCreate(savedInstanceState);
19          setContentView(R.layout.activity_main);
20
21          speaker = new TextToSpeech(this, this);
22
23      }
24
```

6. Leave the public boolean onCreateOptionsMenu() as it is in the default. This code exists to define the layout and fields for the Options Menu.

```
38
39      @Override
40      public boolean onCreateOptionsMenu(Menu menu) {
41          // Inflate the menu; this adds items to the action bar if it is present.
42          getMenuInflater().inflate(R.menu.activity_main, menu);
43          return true;
44      }
45
```

7. Start writing the code for the buttonSpeakClicked() function. The speak function sets the phrase string to the text within the Edit Text Box. First, bind an EditText instance to the XML object enterText.

```
47
48      public void buttonSpeakClicked(View view) {
49
50          // Get Text from editText
51          EditText editText = (EditText)findViewById(R.id.enterText);
52
53
54
55      } // end buttonSpeakClicked()
56
```

8. Create an instance of Editable named 'input' to gather the text from the enterText object:

```
48
49      public void buttonSpeakClicked(View view) {
50
51          // Get Text from editText
52          EditText editText = (EditText)findViewById(R.id.enterText);
53
54          Editable input = editText.getText();
55
56
57
58      } // end buttonSpeakClicked()
59
```

9. Finish the function by converting the input to a String and then having the speaker object speak the text. Do not worry about the crossed out speak. This method is 'deprecated' (meaning they do not use it anymore). However, it will still work.

```
48
49      public void buttonSpeakClicked(View view) {
50
51          // Get Text from editText
52          EditText editText = (EditText)findViewById(R.id.enterText);
53
54          Editable input = editText.getText();
55
56          // Convert to a string
57          String phrase = input.toString();
58
59          // Speak the phrase
60          speaker.speak(phrase, 1, null);
61
62      } // end buttonSpeakClicked()
63
```

10. Finish the MainActivity class code. Make sure to code the onDestroy() to release the speaker object when the App closes.

```
63
64      @Override
65      public void onInit(int status) {
66
67      }
68
69      public void onDestroy() {
70          if (speaker != null) {
71              speaker.stop();
72              speaker.shutdown();
73          }
74          super.onDestroy();
75      } // end onDestroy
76
77  } // end class Main Activity
78
```

Save the App and run on the Emulator or Your Phone.