MMAD S2016
JSt vers February 3, 2016

Tingle App

During the semester, we will create an app called “Tingle”. The purpose of Tingle is to register and find physical things such as books, clothing, keys etc. Many of us spend a lot of time looking for things that we cannot find and could use a Google-like search engine for things in the physical world. Something like:

During the semester you will gradually improve the user interface and add functionality to your app. A big challenge with this app is to find convenient ways of registering things. We will therefore experiment with using camera, location, bar-code reading and storing things in databases.

This week you will do the very first version of Tingle.

Tingle App: version 1

The first version of the app (called TingleV1) looks like this:

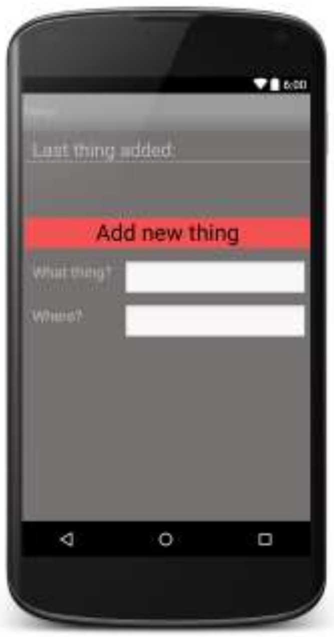
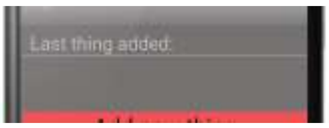


Figure 1 Layout of first version of Tingle app

The main part of the user interface allows you to describe a thing with two texts: "what" (the thing is) and "where" (it is located). When you have filled in these two text fields, you may add the "thing" by pushing the red button labelled "Add new thing". The two texts describing the thing will then appear in the text field just below "Last thing added":



This week you should create an Android app, called "Tingle" with a user interface like the one shown above.

Development of version 1 of the Tingle app

Follow the steps explained on p.2 – p.8 in the textbook to create version 1 of the Tingle app in Android Studio:

1. on p4. Use "Tingle" as the application name
2. on p6. Select Empty Activity (and not Blank Activity)

3 on p.7 use “TingleActivity” as the Activity Name and “activity_tingle” as the Layout Name
You will not be asked for a Title and Menu Resource Name (because you selected Empty Activity in step 2).

You should now have an Android project with all the necessary files. The next step is to change these files to become the Android code for version 1 of the Tingle app.

The layout

The layout consists of three LinearLayouts: two horizontal for each of the texts describing a new thing embedded in a vertical with elements describing the last thing, the add button etc.

```
<LinearLayout …
    android:orientation="vertical"
    … ">

    <!--  Last thing added -->
    <TextView
        …
        android:text="@string/headline"
        …"/>
    <TextView
        android:id="@+id/last_thing"
        …/>

    <!-- Button to add new thing-->
    <Button
        android:id="@+id/add_button"
        …
        android:text="@string/heading_create"/>
    <!—thin horizontal line →
    <View
        android:layout_width="match_parent"
        android:layout_height="10dp"/>

    <!-- What new thing -->
    <LinearLayout …
        android:orientation="horizontal">
        <TextView
            …
            android:text="@string/what_thing"
            …"/>
        <EditText
```

```
                android:id="@+id/what_text"
                …/>
        </LinearLayout>

        <!-- Where -->
        <LinearLayout …
            android:orientation="horizontal">
            <TextView
                …
                android:text="@string/where"
                …/>
            <EditText
                android:id="@+id/where_text"
                …/>
        </LinearLayout>
    </LinearLayout>
```

You should fill in all the missing parts indicated by … It is ok if your final layout does not look exactly like the one shown in Figure 1. As long as it has at least the elements shown in the xml file shown above

The Java code

The Java code consists of two files: TingleActivity.java and Things.Java. The last one is a class for a single Thing. We will elaborate a lot on this in the coming weeks, but here is the first version.

```
public class Thing {
   private String mWhat = null;
   private String mWhere = null;

   public Thing(String what, String where) {
      mWhat = what;
      mWhere = where;
   }
   @Override
   public String toString() { return oneLine("Item: ","is here: "); }

   public String getWhat() {  return mWhat;   }
   public void setWhat(String what) { mWhat = what; }
   public String getWhere() { return mWhere;   }
   public void setWhere(String where) { mWhere = where; }
   public String oneLine(String pre, String post) {
      return pre+mWhat + " "+post + mWhere;
   }
}
```

Here is a skeleton for the ThingsActivity.java:

```
public class TingleActivity extends Activity {

  // GUI variables
  private Button addThing;
  private TextView lastAdded;
  private TextView newWhat, newWhere;

  //fake database
  private List<Thing> thingsDB;

  @Override
  protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_tingle);
    thingsDB= new ArrayList<Thing>();
    fillThingsDB();

    //Accessing GUI element
    lastAdded= (TextView) findViewById(R.id.last_thing);
    updateUI();

    // Button
    addThing = …
    // Textfields for describing a thing

    newWhat= (TextView) findViewById(R.id.what_text);
    newWhere= …

    // view products click event
    addThing.setOnClickListener(new View.OnClickListener() {
      @Override
      public void onClick(View view) {
        if ((newWhat.getText().length() >0) && (newWhere.getText().length() >0
)){
          thingsDB.add(
               new Thing( newWhat.getText().toString(),
                          newWhere.getText().toString()));
          newWhat.setText(""); newWhere.setText("");
          updateUI();
        }
      }
    });
   }
  // This method fill a few things into ThingsDB for testing
  private void fillThingsDB() {
    thingsDB.add(new Thing("Android Pnone", "Desk"));
    … add as many as you like
    thingsDB.add(new Thing("Big Nerd book", "Desk"));
  }

  private void updateUI(){
    int s= thingsDB.size();
```

```
      if (s>0) {
        lastAdded.setText(thingsDB.get(s-1).toString());
      }
    }
}
```

Fill in code where it is missing indicated by …

Complete and run TingleV1

Fill in the missing code and try running the app from Android Studio.