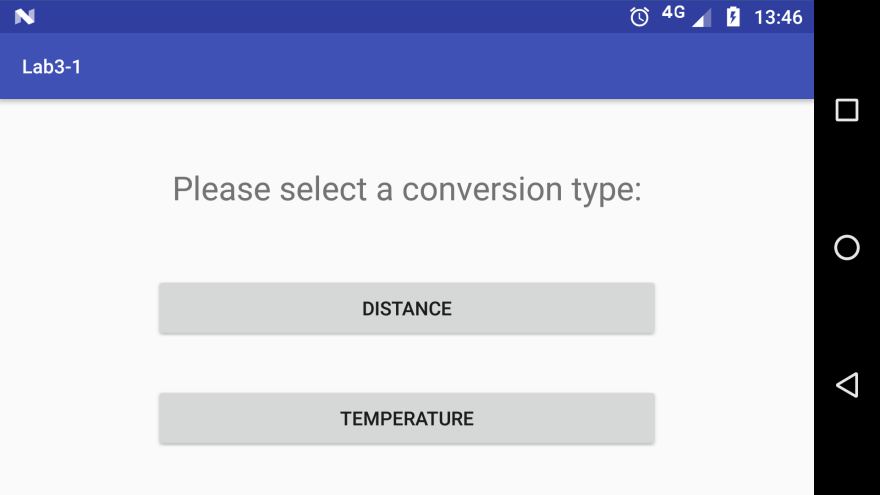
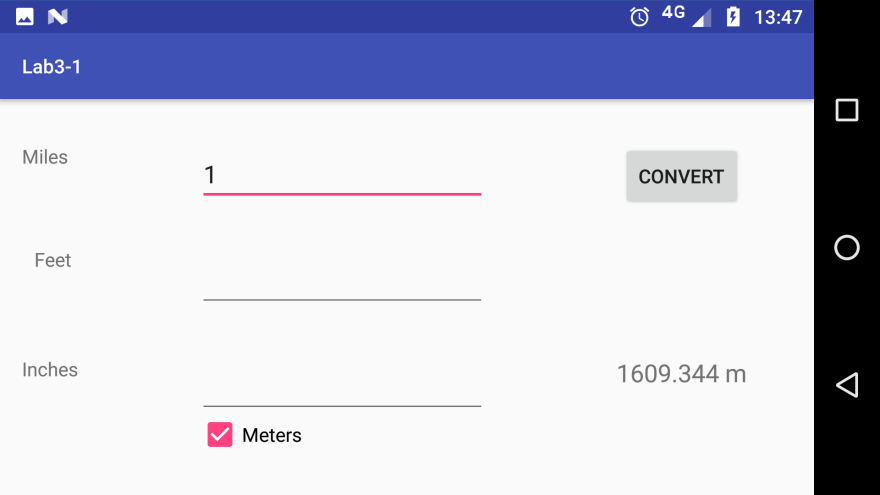
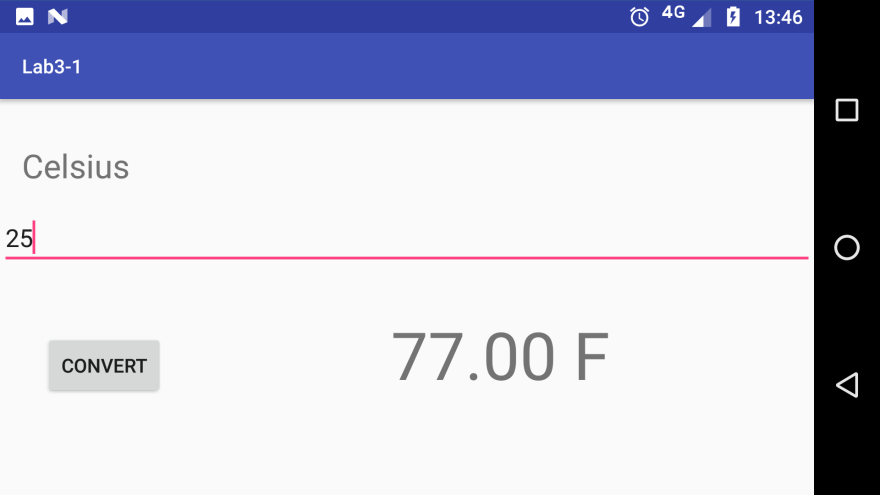
Lab 3

# Task 1

Main Activity:  


Distance Activity:  


Temperature Activity:  


## Key snippets:

Calling one activity from another with an intent.

private void btn\_Distance\_Clicked() {  
 //Set up the call to the new activity.  
 //1: Make an Intent (I figure it's symbolic of our intention to do a thing)  
 Intent i = new Intent();  
 //2: Set the class to the activity you wish you trigger next.  
 i.setClass(getApplicationContext(), com.example.s9726446.lab3\_1.DistanceActivity.class);  
 //3: Go ahead and start the activity!  
 startActivity(i);  
 //Don't forget to update the manifest file, with each new activity!  
}

Adding the new activty to the manifest

<activity android:name=".MainActivity">  
 <intent-filter>  
 <action android:name="android.intent.action.MAIN" />  
  
 <category android:name="android.intent.category.LAUNCHER" />  
 </intent-filter>  
 </activity>  
 <!-- Each new activity should be added here so the program knows to look for it, (even if it's an empty tag!)-->  
 <activity android:name=".DistanceActivity"></activity>  
 <activity android:name=".TemperatureActivity"></activity>  
</application>

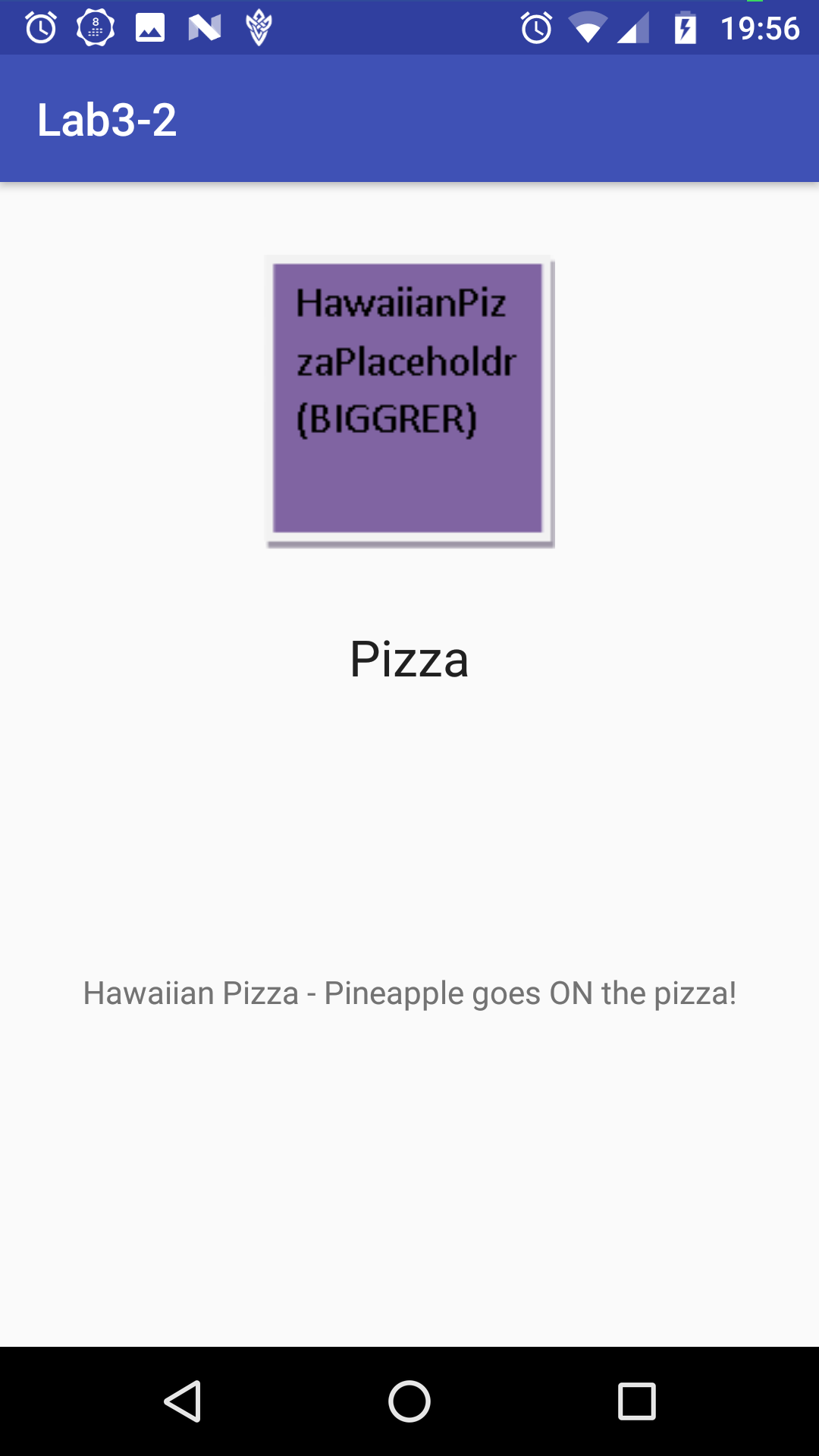
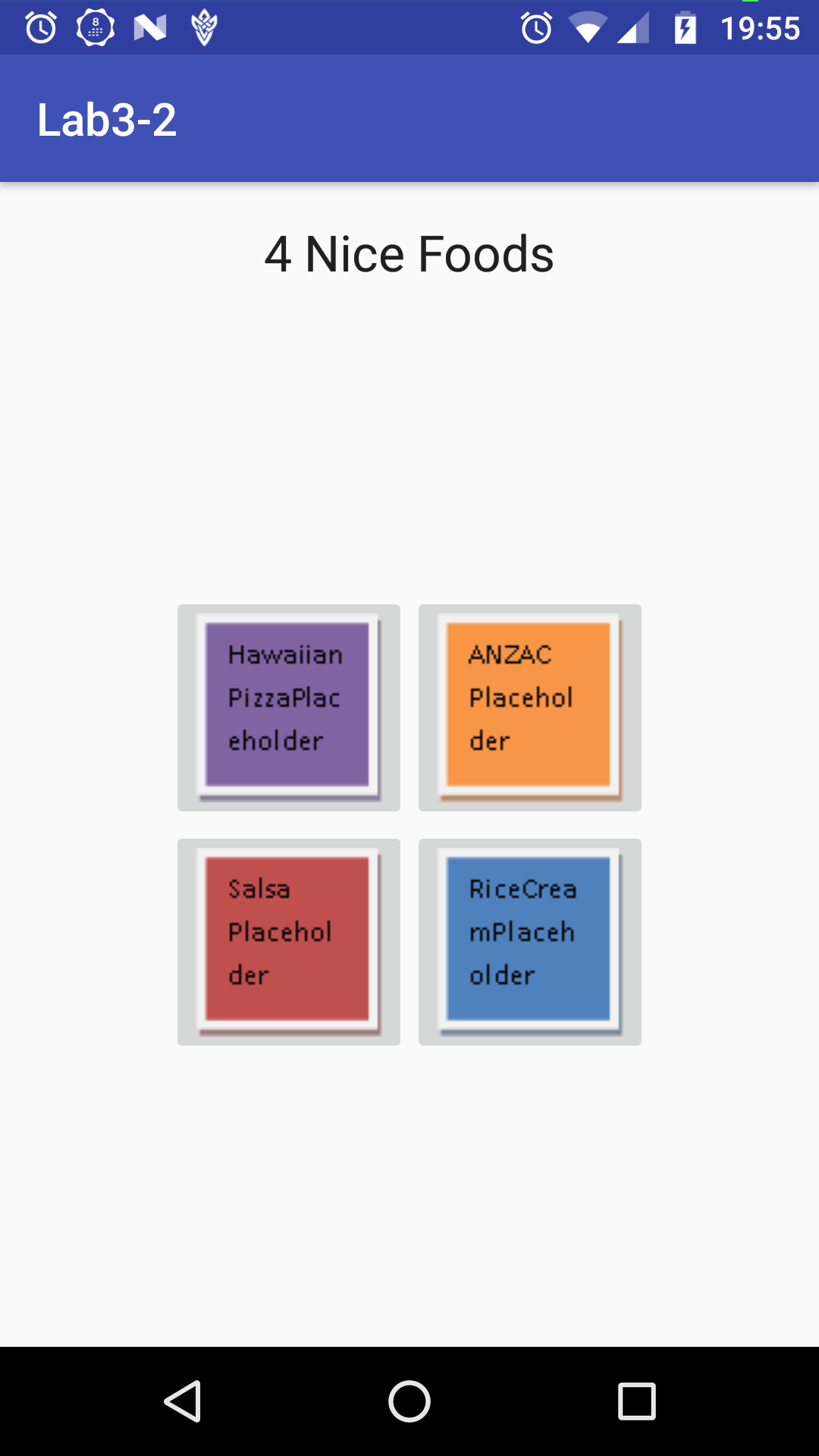
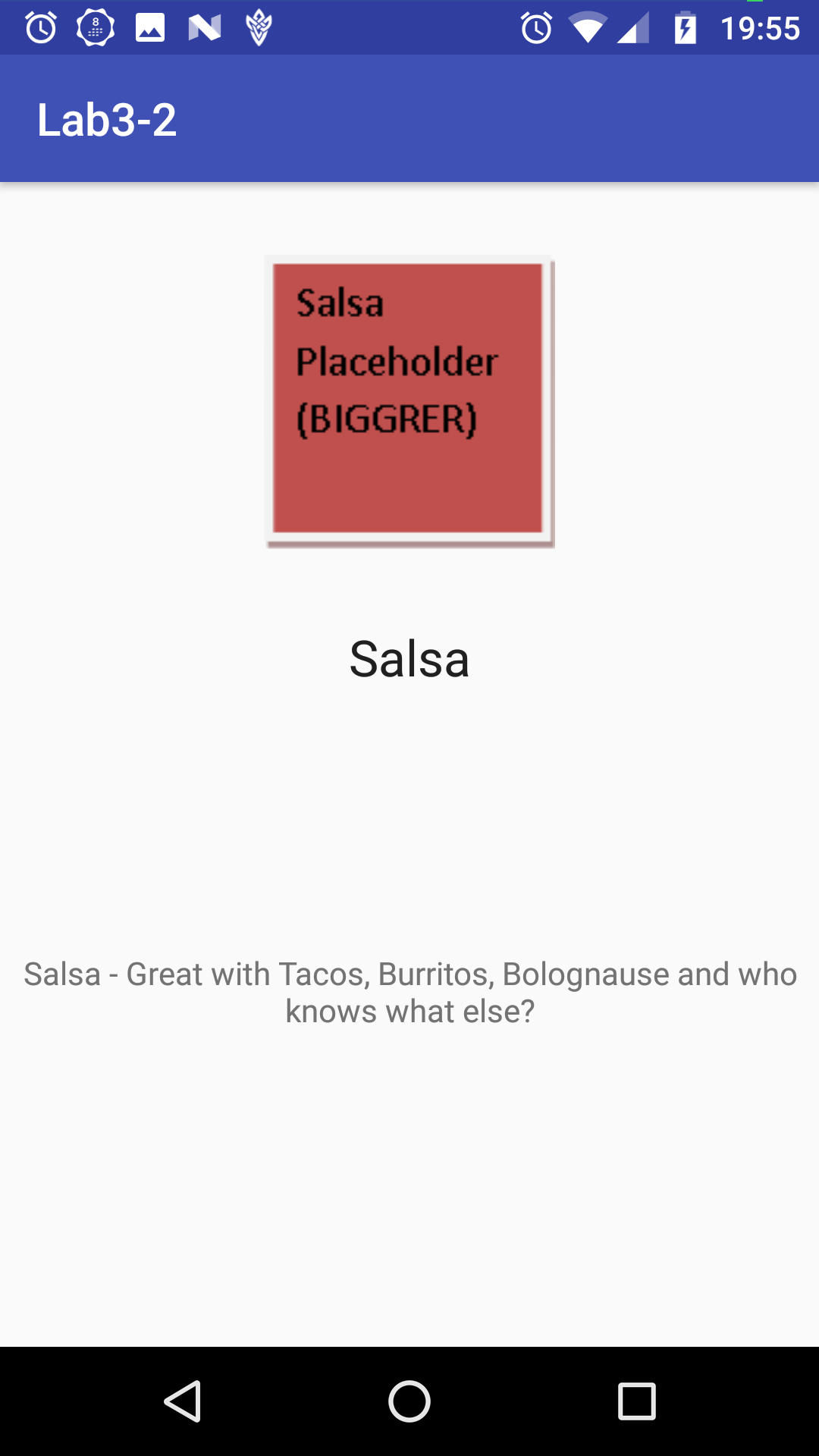
Issues with Orientation changes are resolved through use of bundles:

*/\*\* Reads the bundle and if it isn't null, returns the two saved values (input and output)  
 \** ***@param*** *b a bundle that might be null (eg: first run) or have values saved from a previous orientation.  
 \*/*private void restoreState(Bundle b){  
 if (b == null) return;  
 // Searching for the view manually rather than saving a local variable saves on memory.  
 ((TextView)findViewById(R.id.*celsiusTextView*)).setText(b.getString("celsius"));  
 ((TextView)findViewById(R.id.*convertedTempTextView*)).setText(b.getString("fahrenheit"));  
}  
  
*/\*\* Saves the current display if the screen orientation changes.  
 \** ***@param*** *b a bundle to save things into.  
 \*/*@Override  
protected void onSaveInstanceState(Bundle b){  
 b.putString("celsius", ((TextView)findViewById(R.id.*celsiusTextView*)).getText().toString());  
 b.putString("fahrenheit", ((TextView)findViewById(R.id.*convertedTempTextView*)).getText().toString());  
 super.onSaveInstanceState(b);  
}

# Task 2:

## Screenshots of the program running:

Program begins with the Main activity showing four [placeholder] images.  
Tapping one takes you to a second activity with more information about the tapped option.



## Code Snippets:

MainActivity.java: (Each click calls the same launching method, albeit with a different variable)

*/\*\* Handle Click \*\*/*private View.OnClickListener onPizzaClick = new View.OnClickListener()  
{  
 public void onClick(View v)  
 {  
 foodClick(0);  
 }  
};

MainActivity.java: (Setting up the information)

*/\*\*  
 \* Calls the display and tells it which images to load.  
 \** ***@param*** *item int representing which option was picked.  
 \* 0 - Pizza  
 \* 1 - ANZAC  
 \* 2 - Rice Cream  
 \* 3 - Salsa  
 \*/*private void foodClick(int item){  
 Intent i = new Intent();  
 i.setClass(getApplicationContext(), com.example.s9726446.lab3\_2.DisplayActivity.class);  
 // adds an extra variable to the intent, telling the next activity which item to show.   
 i.putExtra("item", item);  
 startActivity(i);  
}

DisplayActivity.java: (Using this information to choose what to display)

private void initialiseUI(){  
 //get intent data. Only one thing is passed so not bothering with a local bundle.  
 int item = getIntent().getExtras().getInt("item");  
 String title;  
 String description;  
 Drawable draw;  
 //use passed value.  
 switch (item){  
 case 0:  
 title = "Pizza";  
 description = getResources().getString(R.string.*descr\_pizza*);  
 draw = getResources().getDrawable(R.drawable.*pizza\_biggerer*);  
 break;  
 case 1:

activity\_display.xml: (leaving the content - where possible – to the code)

<!--Source can't be blank so default image used: -->  
<ImageView  
 android:id="@+id/imgOutput"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 app:srcCompat="@drawable/anzac\_biggerer"  
 android:layout\_marginRight="8dp"  
 app:layout\_constraintRight\_toRightOf="parent"  
 android:layout\_marginLeft="8dp"  
 app:layout\_constraintLeft\_toLeftOf="parent"  
 app:layout\_constraintTop\_toTopOf="parent"  
 android:layout\_marginTop="32dp" />  
  
<!--Content specified in code.-->  
<TextView  
 android:id="@+id/txtOutputTitle"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginLeft="8dp"  
 android:layout\_marginRight="8dp"  
 android:layout\_marginTop="32dp"  
 android:textAppearance="@android:style/TextAppearance.Large"  
 app:layout\_constraintLeft\_toLeftOf="@+id/imgOutput"  
 app:layout\_constraintRight\_toRightOf="@+id/imgOutput"  
 app:layout\_constraintTop\_toBottomOf="@+id/imgOutput" />

Strings.xml: (Where all the text is stored)

<resources>  
 <string name="app\_name">Lab3-2</string>  
 <string name="lbl\_header">4 Nice Foods</string>  
 <string name="descr\_salsa">Salsa - Great with Tacos, Burritos, Bolognause and who knows what else?</string>  
 <string name="descr\_anzac">ANZAC Biscuit - lasts forever (not really but still a really long time)</string>  
 <string name="descr\_rice">Creamed Rice - only 99c a can! Also very delicious.</string>  
 <string name="descr\_pizza">Hawaiian Pizza - Pineapple goes ON the pizza!</string>  
 <string name="capt\_ANZAC">ANZAC Biscuit</string>  
 <string name="capt\_Pizza">Pizza with Pineapple</string>  
 <string name="capt\_Rice">Rice Cream</string>  
 <string name="capt\_Salsa">Salsa</string>  
</resources>

# Task 3:

1. Intent messaging is considered a late run-time binding between components as
   * Intents are generated at runtime, rather than (for instance) during compilation.
   * They also serve as a ‘bridge’ between activities, allowing them to share information with each other.
2. The contents of a passive data structure (Intent) are
   * In a word: Elsewhere.
   * In more words: Determined at runtime.
3. The word ‘passive’ is used to describe intents as like other Passive Data Structures, the logic for data manipulation and integrity is contained elsewhere.
4. “[An Intent holds an] abstract description of an operation to be performed.”
   * This means that an Intent only contains a summary of a given operation, such as holding two numbers and an operator for –another- object to make use of.
   * To cite earlier code for an example:

private void foodClick(int item){  
 Intent i = new Intent();  
 i.setClass(getApplicationContext(), com.example.s9726446.lab3\_2.DisplayActivity.class);  
 i.putExtra("item", item);

Here an intent is given an item. It doesn’t know what to do with it, or what it’s for.

private void initialiseUI(){  
 //get intent data. Only one thing is passed so not bothering with a local bundle.  
 int item = getIntent().getExtras().getInt("item");

switch (item){  
 case 0:

Here another object receives that item, and then uses it to determine an outcome.

# Appendix I:

General Observations:

* As an aside, I realise I could’ve simply done the selection logic in the main activity and added the image, title and description to the intent, however I thought it might be better using smaller intents. (That and I like writing switch statements!)
  + Since discussed with Isuru.
* Placeholder images were text boxes with different default themes chosen, then copied and enlarged for the larger versions. Each was then saved as a png file.