

Mobile Application Development using SOFIA

Kosta Damevski

Virginia State University

Google CS4HS Workshop

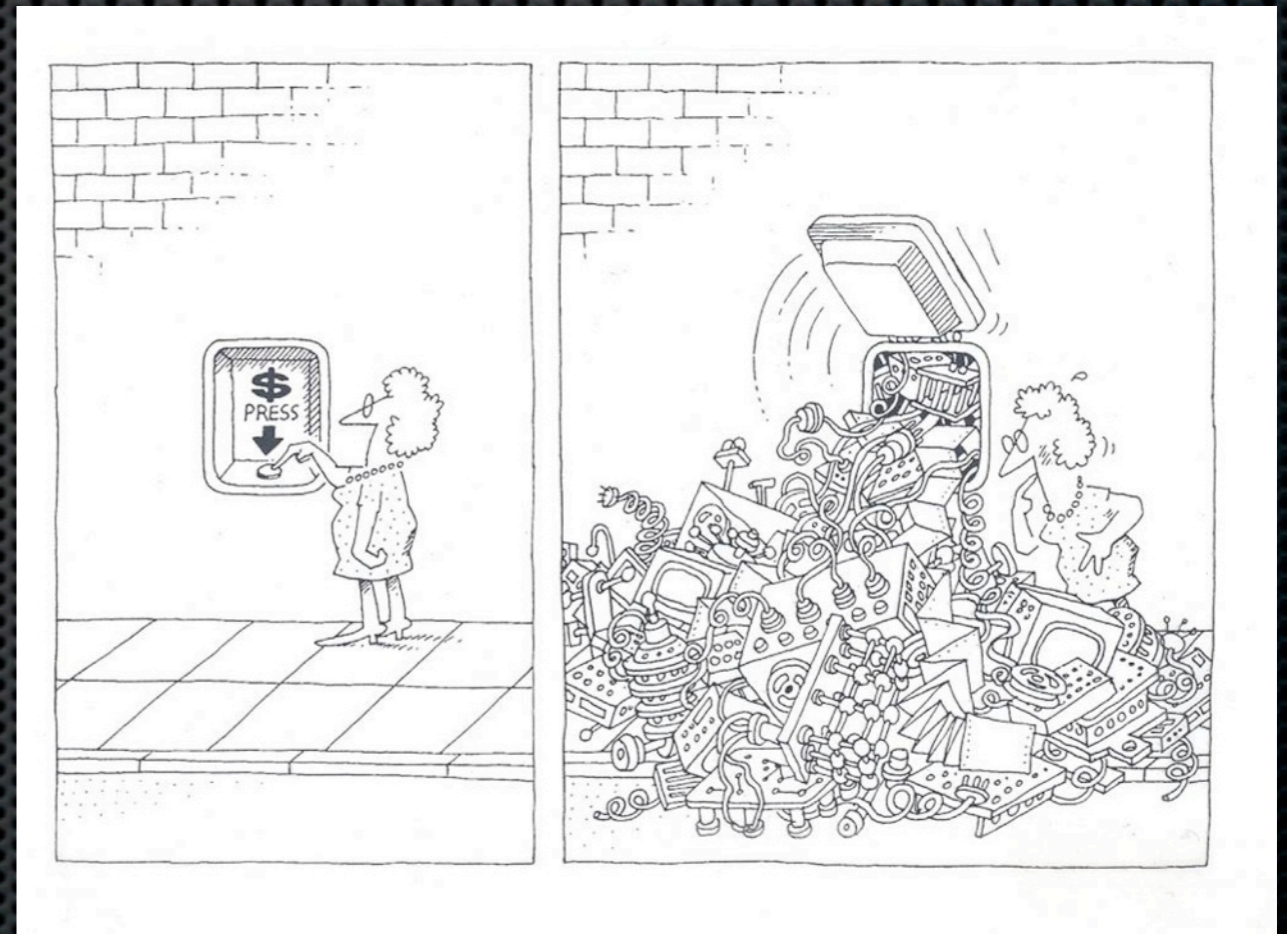
CS4HS Day 2

- ✦ Welcome, again!
- ✦ This workshop is about Android mobile application
 - ✦ specifically, about sensor-driven apps
- ✦ Some of you may want to teach Android in Java
 - ✦ most flexible way
 - ✦ good to know



Android with Java

- AppInventor
 - makes app development a lot easier
 - some limitations (project size, limited feature set)
- Mobile application development using the Android ADK is for the pros
 - Java, but at its worst



Android Basic Concepts

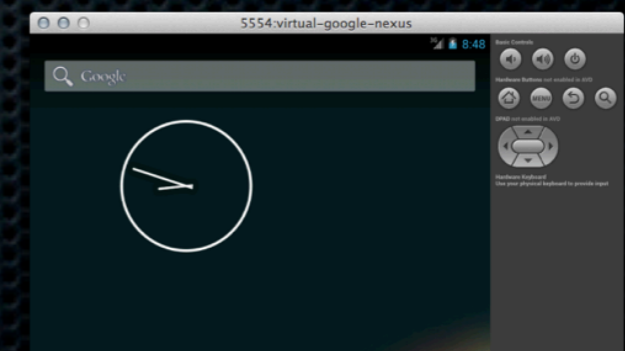
- ✦ Activities
 - ✦ communicate via Intents
 - ✦ have a strange life-cycle
- ✦ Event-driven Programming
 - ✦ no main method

Button.onClick
onCreate
onStart
onPause

...



intent



Introducing SOFIA

- ✦ Simplified Open Framework for Inventive Android applications
- ✦ Library that helps to make some of the Android API more accessible to students
 - ✦ Builds on top of the Android ADT
 - ✦ Target audience is CS1/CS2 classes
- ✦ Developed by Dr. Stephen Edwards and his team at Virginia Tech

Why SOFIA?

- Even basic features of the Android API require some of the nastier features of Java (e.g. anonymous inner classes or type casting)

```
Button buttonOne = (Button) findViewById(R.id.ButtonOne);
buttonOne.setOnClickListener(new OnClickListener() {
    public void onClick(View arg0) {
        Toast.makeText(this, "Hello", Toast.LENGTH_LONG).show();
    }
});
```

Why SOFIA?

- ✦ Android App lifecycle adds complexity too
 - ✦ Apps commonly stay resident in memory
 - ✦ User can switch between apps at any point
 - ✦ Moving from screen to screen requires a fair amount of indirection and callbacks (events)

```
Intent i = new Intent(this, ActivityTwo.class);  
startActivity(i);
```

SOFIA Helps

- ✦ Provides a simpler Java API for Android App development
 - ✦ Simplifies development by reducing the clutter
 - ✦ Handles most of the common use scenarios
 - ✦ actions on UI elements
 - ✦ cleaner screen to screen communication
 - ✦ ample graphics and gaming support

Let's Explore SOFIA with these

1. **Tip Calculator App:** Learn some SOFIA basics
2. **Human Activity Classification App:** Our crowning achievement



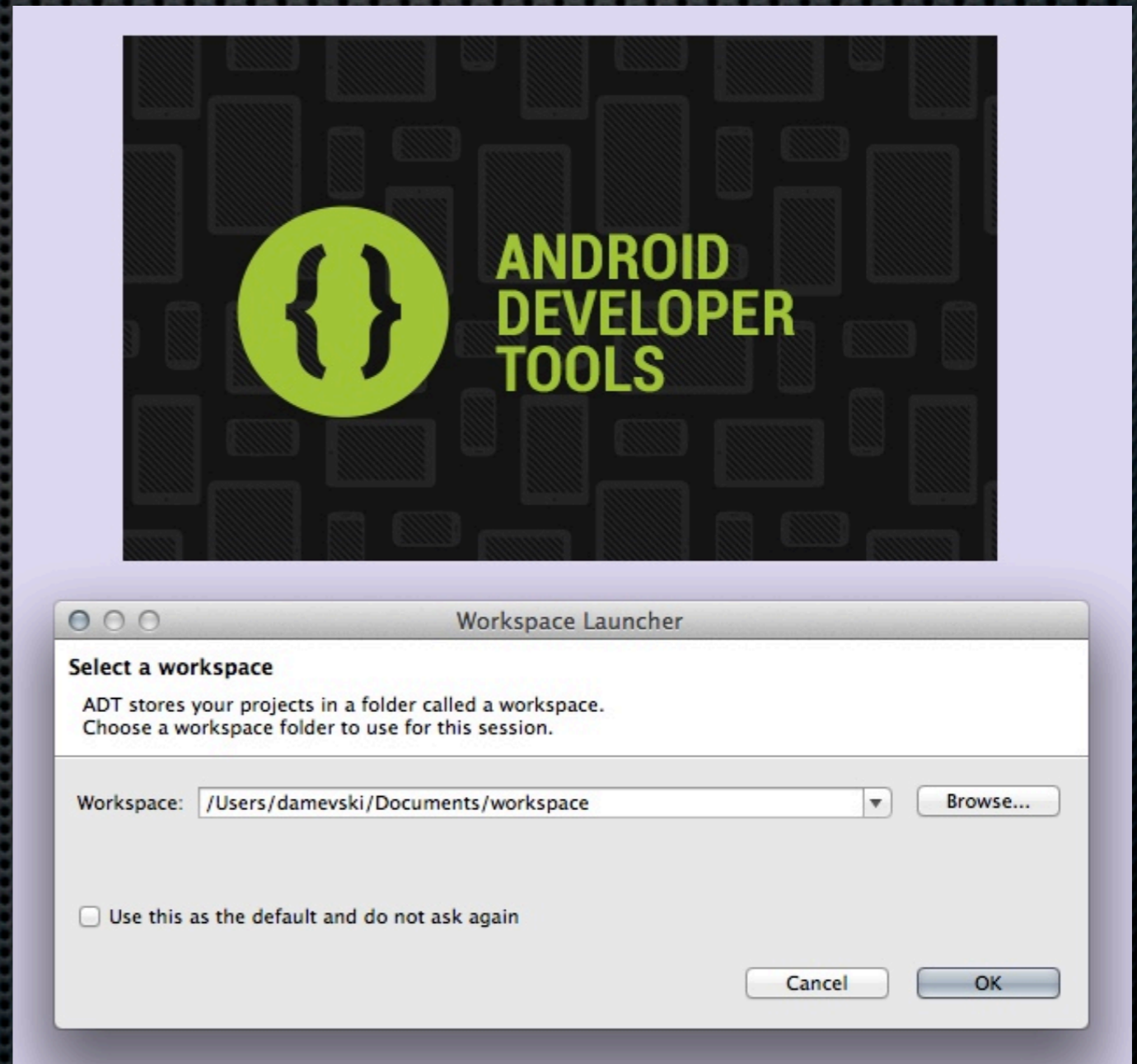
Part I: Tip Calculator

Getting to know the basics of Android and SOFIA

1. Starting the ADT

<http://developer.android.com/sdk>

- no installation needed, just download and run
- choose a workspace directory where you want the project files deposited



ADT Environment

The screenshot displays the ADT (Android Studio) environment. The top toolbar contains various icons, with the 'Run' icon (a green play button) circled in red. The left sidebar shows the 'Package Explorer' and 'Type Hierarchy' views, with the 'Type Hierarchy' view also circled in red. The main editor area shows a Java file named 'TipCalculatorScreen.java' with the following code:

```
// Setting the bill amount causes the observers to be notified, so the
// changeWasObserved method below will perform the actual updating of
// the widgets.

tipModel.setBillAmount(amount);
}

// -----
/**
 * Called when the 15% tip button is clicked.
 */
public void tip15Clicked()
{
    // Setting the tip rate causes the observers to be notified, so the
    // changeWasObserved method below will perform the actual updating of
    // the widgets.

    tipModel.setTipRate(0.15f);
}

// -----
/**
 * Called when the 18% tip button is clicked.
 */
public void tip18Clicked()
```

The text 'editing space' is overlaid in red on the code editor. The bottom panel shows the 'LogCat' console with a search bar and a table of log messages. The text 'Console and Log Cat' is overlaid in red on the console area.

projects

editing space

Console and Log Cat

Lev	Time	PID	TID	Application	Tag	Text
-----	------	-----	-----	-------------	-----	------

Writable | Smart Insert | 70 : 6 | 53M of 120M

2. Grab Workshop Package

- ✦ We've packaged the SOFIA library as well as some projects will make use of later
- ✦ Please download the package
 - ✦ <http://sysnetgrp.net/academy/cs4hs>
- ✦ Go to `File->Import->General->Existing Projects into Workspace` and follow the wizard to get all of the projects into your ADT workspace

3. Creating a Virtual Device

- ✦ Click on the Android Virtual Device Manager...



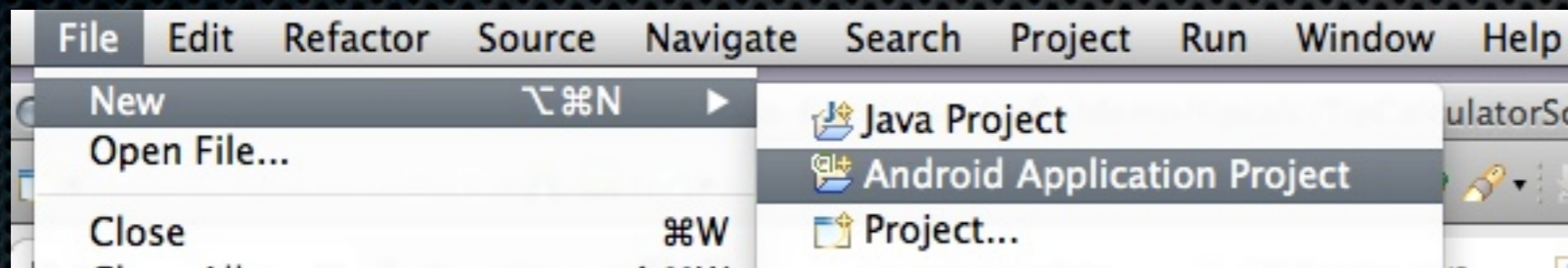
- ✦ Create a new AVD...



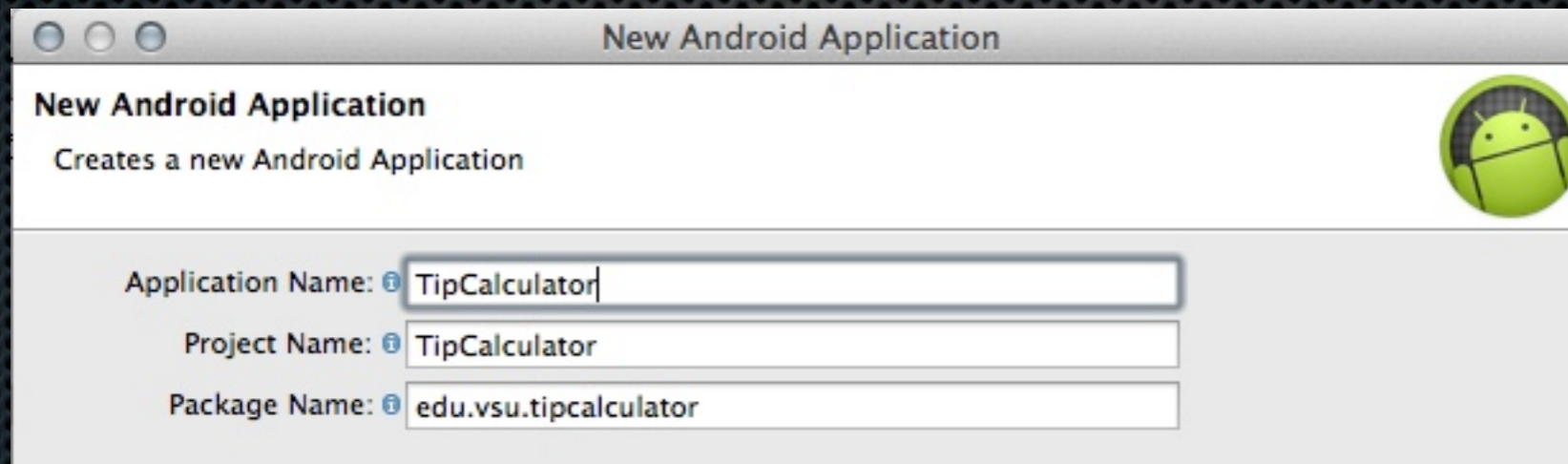
- ✦ Start the AVD (takes a while, grab some coffee)

4. Creating a New Project

- Go to **File**→**New**→**Android Application Project**

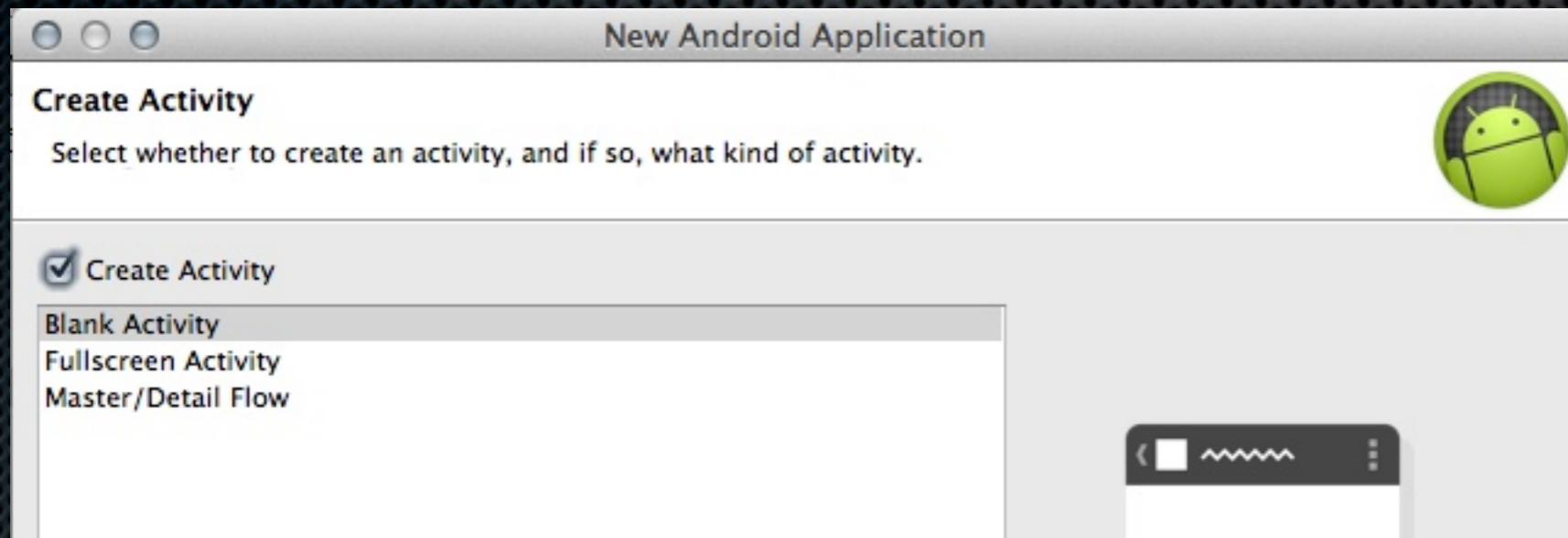


- Use **TipCalculator** as the application name

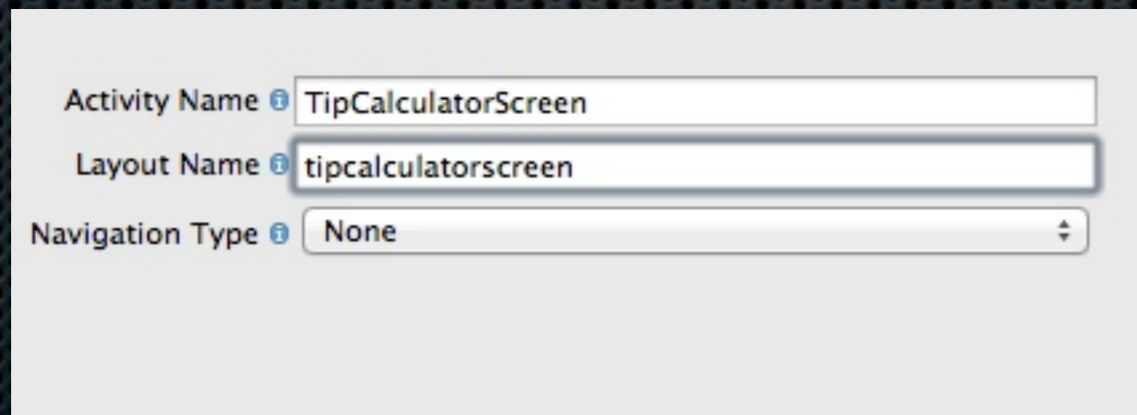


4. Create a New Project Cont.

- ✦ Create a blank activity



- ✦ Important: Activity and layout name have to match, and layout must be in lowercase

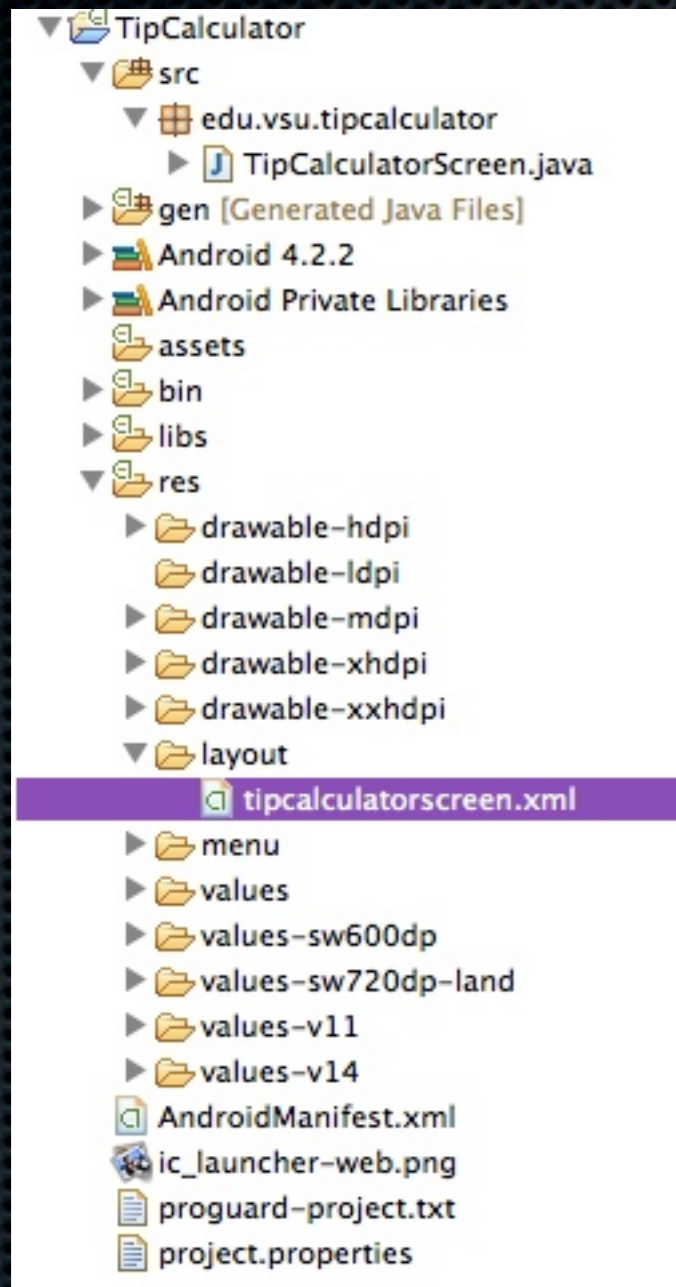


Activity Name

Layout Name

Navigation Type

Structure of an Android Project



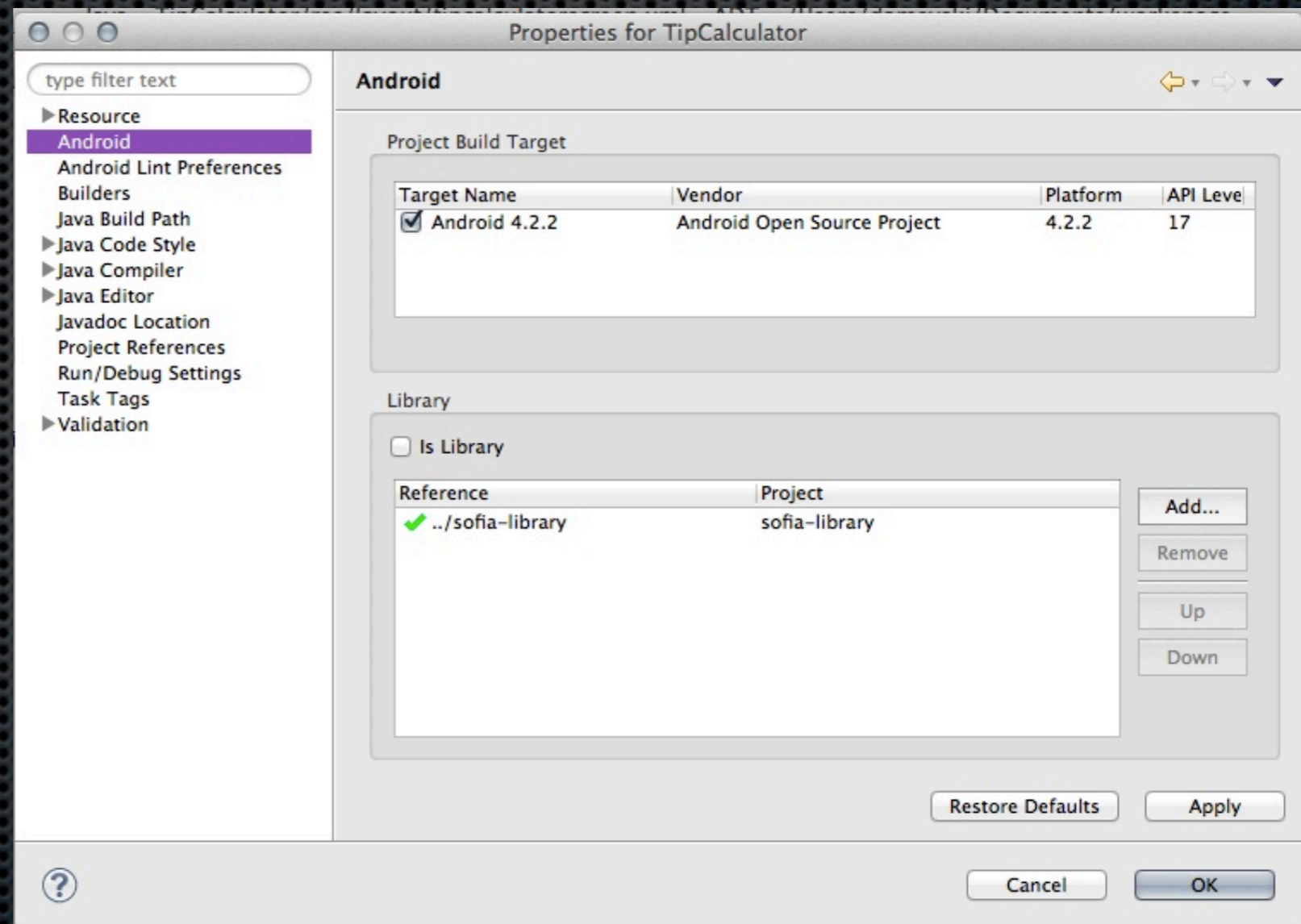
Java program

images

XML file that describes screen layout

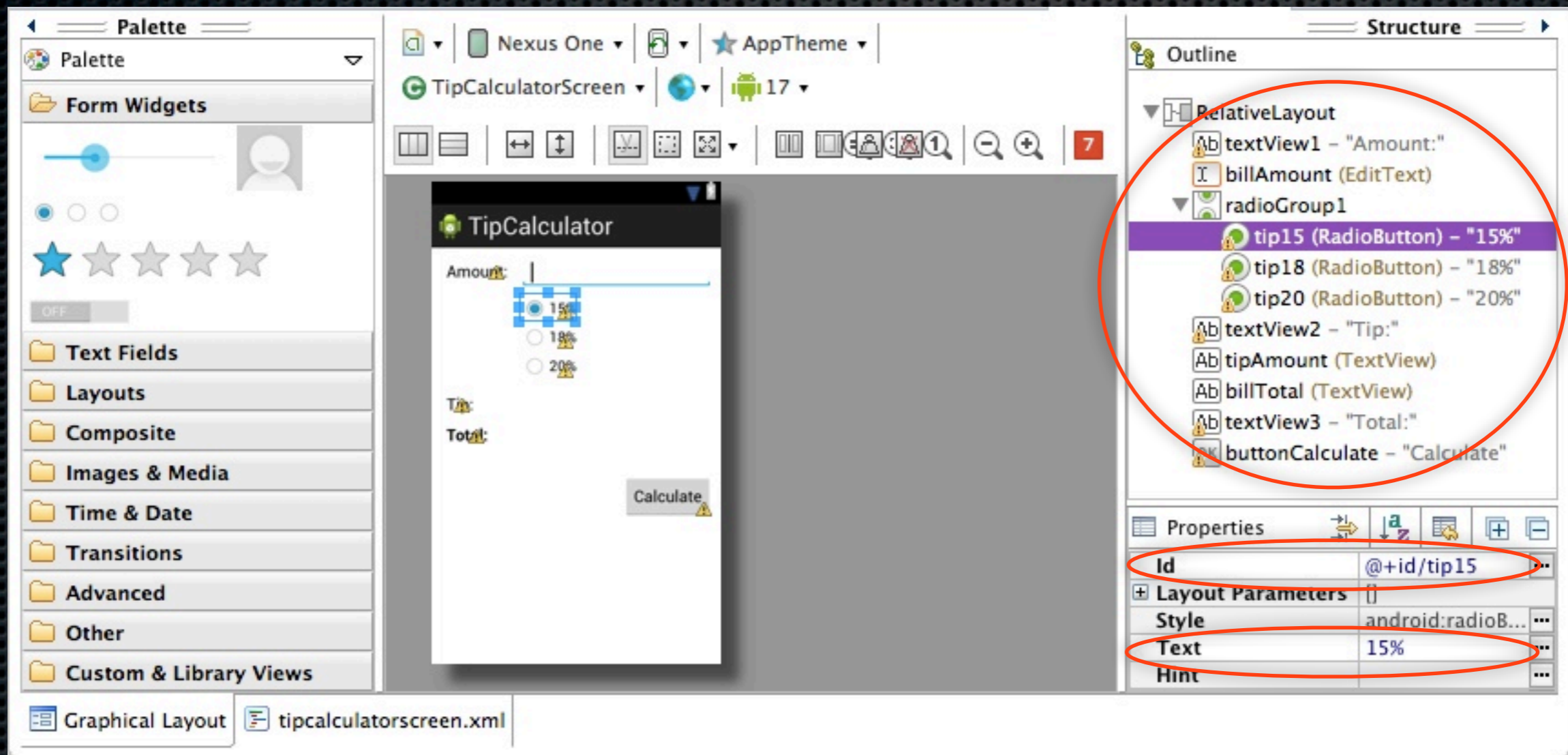
5. Link to the SOFIA library

- ✦ Launch the project menu: right-click on the project and go to `properties`
- ✦ Add the `sofia-library`



6. Draw the App Layout

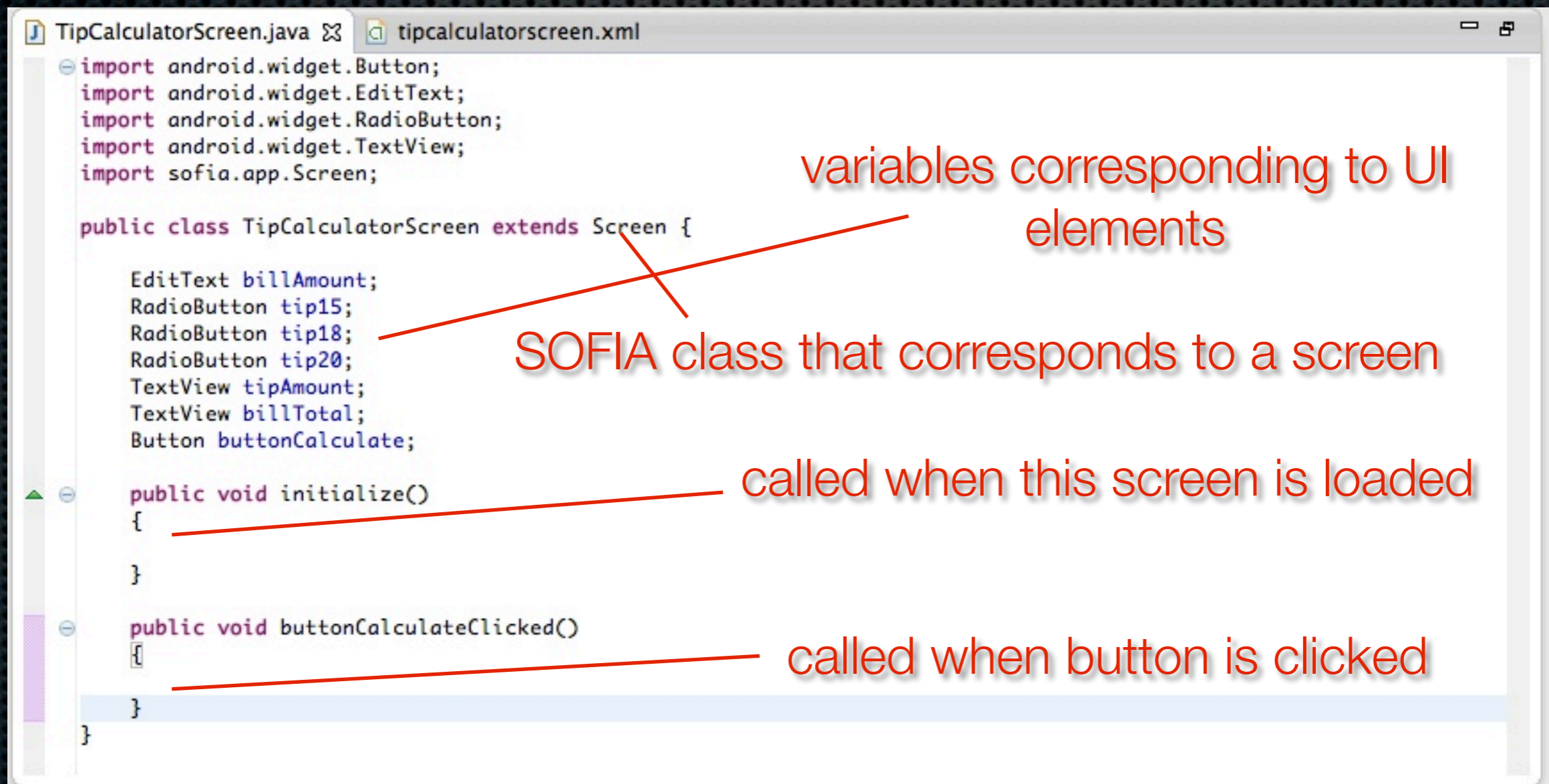
- ✦ Draw a screen that looks similar to this:



- ✦ Make sure you remember the element IDs

7. SOFIA-fy your Java code

- ✦ Go to `TipCalculatorScreen.java`
 - ✦ getting rid of the default Activity code



```
TipCalculatorScreen.java  tipcalculatorscreen.xml
import android.widget.Button;
import android.widget.EditText;
import android.widget.RadioButton;
import android.widget.TextView;
import sofia.app.Screen;

public class TipCalculatorScreen extends Screen {

    EditText billAmount;
    RadioButton tip15;
    RadioButton tip18;
    RadioButton tip20;
    TextView tipAmount;
    TextView billTotal;
    Button buttonCalculate;

    public void initialize()
    {

    }

    public void buttonCalculateClicked()
    {

    }
}
```

variables corresponding to UI elements

SOFIA class that corresponds to a screen

called when this screen is loaded

called when button is clicked

8. Now we can code freely

```
*TipCalculatorScreen.java  tipcalulatorscreen.xml

public void buttonCalculateClicked()
{
    double oweTip = 0.0;
    double oweFood = 0.0;
    try {
        oweFood = Double.parseDouble(billAmount.getText().toString());
    }
    catch (NumberFormatException nfe)
    {
        Toast.makeText(this, "Cannot understand bill amount!", Toast.LENGTH_LONG).show();
    }

    if(tip15.isChecked())
    {
        oweTip = oweFood * 0.15;
    }
    else if(tip18.isChecked())
    {
        oweTip = oweFood * 0.18;
    }
    else if(tip20.isChecked())
    {
        oweTip = oweFood * 0.20;
    }
    tipAmount.setText(String.valueOf(oweTip));
    billTotal.setText(String.valueOf(oweFood + oweTip));
}
}
```

refers to class variables which refer to UI elements

9. Running the app

- ✦ Right click project, Run As -> Android Application
 - ✦ This should start the simulator
 - ✦ If a phone is connected, it should give you an option to execute on either the phone or the simulator

Done. Let's talk.

- ✦ Do all of these steps make sense?
 - ✦ The ADK is based on a professional Java IDE, called Eclipse, which has a lot of features
 - ✦ Probably at least two ways for doing each of the steps I mentioned so far
- ✦ Do you understand where SOFIA helped?



Part II: Human Activity Classification

Porting the ApplInventor application to Java, plus a couple of new SOFIA tricks

Human Activity Classification

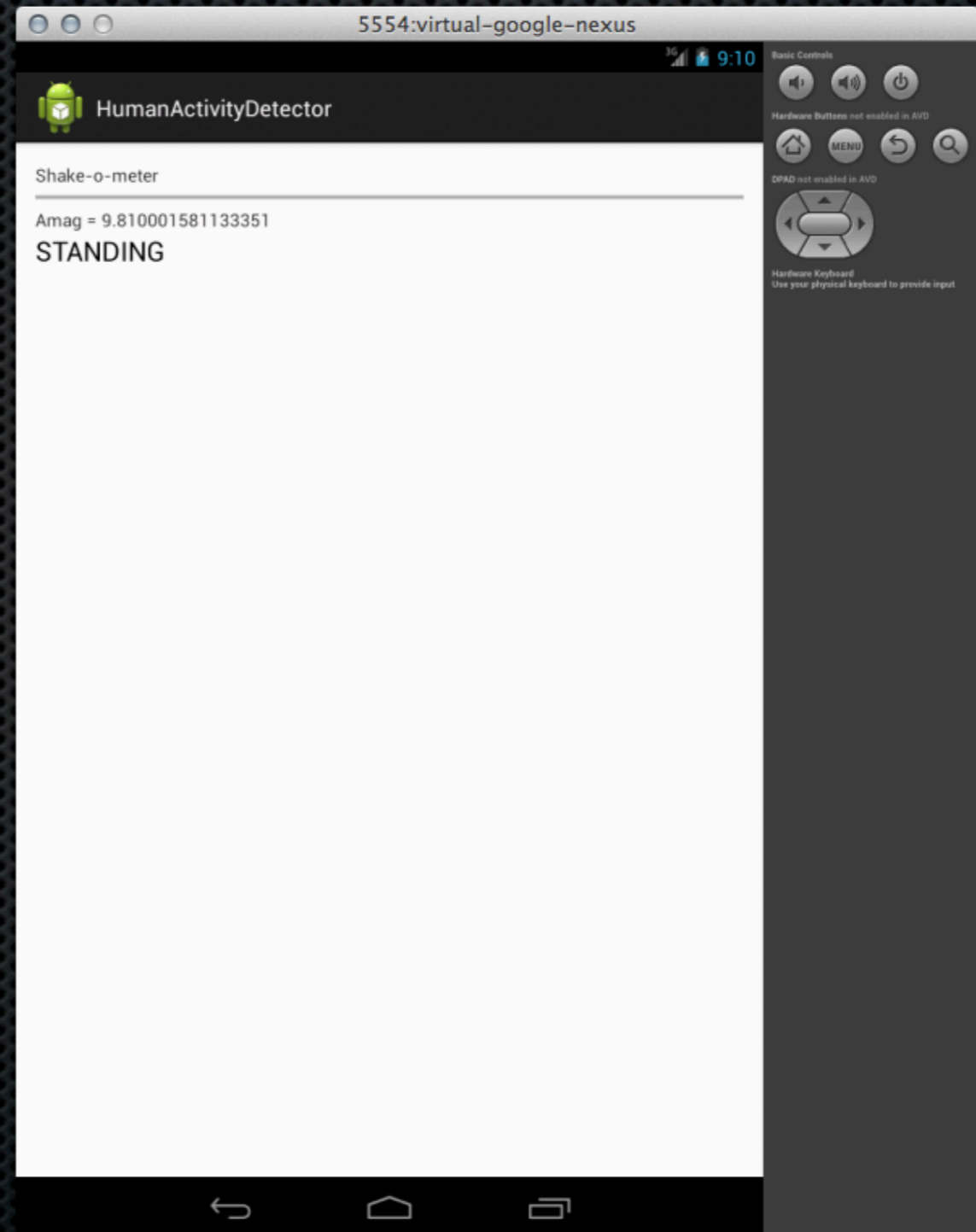
- ✦ Compared to yesterday, let's make a version that is:
 - ✦ real-time, not stop and compute
 - ✦ fast enough to get data at a finer granularity
 - ✦ a basis for a more complicated app
 - ✦ calorie counter
 - ✦ fall detector
- ✦ Give you a little bit of practice with ADK and SOFIA

1. Create a Project for H.A.C.

- ✦ Call it HumanActivityClassification or something similar
 - ✦ create an empty activity, e.g.
HumanActivityClassificationActivity
- ✦ Right click on the project, go to Properties, Android and link in the SOFIA library

2. Draw a Uber Simple UI

- ✦ a progress bar
- ✦ a TextView to show the most recent value
 - ✦ as a “sanity” check
- ✦ a TextView to show the result of the classification



3. Import *AccelerometerReader*

- ✦ We provide you with a class to help read the Accelerometer (on the workshop web page)
 - ✦ it's not super complex, but saves time today
- ✦ You get 3 methods:

```
AccelerometerReader accelReader =  
    new AccelerometerReader(this);  
accelReader.getAx();  
accelReader.getAy();  
accelReader.getAz();
```

The AccelerometerReader needs a pointer to the current activity/screen

4. Instantiate the Important Vars

```
TipCalculatorScreen.java  tipcalculatorscreen.xml  AccelerometerReader.java  *HumanActivityDetectorScree
```

```
public class HumanActivityDetectorScreen extends Screen {  
    TextView dataSampleTextView;  
    TextView statusTextView;  
    ProgressBar progressBar;  
  
    AccelerometerReader accelReader;  
  
    final int WINDOW_SIZE = 25;  
    final int SAMPLING_RATE_IN_MS = 500;  
    int topIndex;  
    double [] dataWindow;  
  
    public void initialize()  
    {  
        topIndex = 0;  
        dataWindow = new double[WINDOW_SIZE];  
        accelReader = new AccelerometerReader(this);  
        Timer.callRepeatedly(this, "recordAcceleration", SAMPLING_RATE_IN_MS);  
    }  
  
    public void recordAcceleration()  
    {  
    }  
}
```

UI vars

useful stuff
(array,
constants)

SOFIA Timer - will call this
method at a specific frequency

Finish it!

- ✦ If you would like, try to finish this app by completing the code in the *recordAcceleration* method
 - ✦ or, grab a coffee and take a break
 - ✦ I will show you my version in a few minutes

My version of *recordAcceleration*

```
TipCalculatorScreen.java  tipcalculatorscreen.xml  AccelerometerReader.java  *HumanActivityDetectorScreen
```

```
public void recordAcceleration()
{
    double ax = accelReader.getAx();
    double ay = accelReader.getAy();
    double az = accelReader.getAz();
    double Amag = Math.sqrt(ax*ax + ay*ay + az*az);
    dataWindow[topIndex] = Amag;
    dataSampleTextView.setText("Amag = " + Amag);
    topIndex++;
    if(topIndex >= WINDOW_SIZE)
    {
        updateHumanActivity();
        topIndex = 0;
    }
}
```

I added another method `updateHumanActivity` (next slide)

My version of *updateHumanActivity*

```
TipCalculatorScreen.java  tipcalculatorscreen.xml  AccelerometerReader.java  HumanActivityDetectorScree  -  +

private void updateHumanActivity()
{
    double sum = 0.0;
    for(int i=0; i<WINDOW_SIZE; i++) {
        sum = sum + dataWindow[i];
    }

    double avg = sum / WINDOW_SIZE;
    double variance = 0.0;
    for(int i=0; i<WINDOW_SIZE; i++) {
        variance = variance + Math.pow(dataWindow[i] - avg, 2);
    }
    variance = variance / WINDOW_SIZE;

    int pctProgress = (int) ((variance / MAX_VARIANCE) * 100);
    progressBar.setProgress(pctProgress);
    if(pctProgress <= 33) {
        statusTextView.setText("STANDING");
    }
    else if(pctProgress <= 66) {
        statusTextView.setText("WALKING");
    }
    else {
        statusTextView.setText("RUNNING");
    }
}
}
```


Done. Let's talk.

- ✦ How's it going?
- ✦ Is this useful to you?

- ✦ We like these sensor-driven mobile apps in our classes? What do you think about them?

Extensions to Human Activity App

- ✦ Measure the accuracy of the classification
 - ✦ compare the app's result to a human response
 - ✦ how often did I say walking if the user said he was walking?
- ✦ Better classification algorithm
 - ✦ Hidden Markov Model is one that people have used when doing this for real
- ✦ Add of the other add on scenarios we discussed

Other sensor-driven apps

- ✦ We have tried most of these in our classes
 - ✦ Baby toy (timing)
 - ✦ Driver detector (sound, frequency filtering)
 - ✦ Parking helper (GPS)
 - ✦ Security access device (finite state machines, reliability)

There is more on SOFIA...

- ✦ SOFIA is nice for game design
 - ✦ lots of physics and animation APIs
 - ✦ sadly, we didn't even mention those today
- ✦ Android ADK is continuously evolving and getting better
 - ✦ a new IDE, based on IntelliJ was just introduced at the Google I/O conferent

Thanks

- ✦ You guys for attending, listening and participating
- ✦ Google for the funding
- ✦ Dr. Stephen Edwards for pointing us to SOFIA materials
 - ✦ <http://sofia.cs.vt.edu/sigcse2013>