

Team Report

Android Development

William Blankenship and Jon Picchietti

Contents

Introduction	2
Health Calculator	3
Project Description.....	3
Features	3
Application Overview	4
Code	6
HealthCalculations.java	6
InputActivity.java	9
ResultActivity.java.....	11
main.xml	13
result.xml	15
dimensions.xml	16
strings.xml	17
AndroidManifest.xml	17
SimplyFractions	19
Project Description.....	19
Features	19
Application Overview	20
Code	20
Fraction.java.....	20
FractionArithmetic.java	23
PrimeNumbers.java	25
SimplyFractionsActivity.java	27
ResultActivity.java.....	28
fractionmain.xml.....	29
fractionresult.xml.....	30
dimensions.xml	31
strings.xml	31
AndroidManifest.xml	32
SoftKeyboard.....	33
Project Description.....	33
Features	33
Application Overview	33
Code	34
SoftKeyboard.java	34
qwerty.xml	43
symbols.xml	44

Introduction

Our team developed three independent projects during the course of this semester. Each project was an Android application that focused on a different feature of the Android Operating System. The main project we developed was a health calculator; the main focus of this application was to develop an application ready to be brought to market. The second project we developed was a simple application that did fraction arithmetic; the focus of this application was to first develop a stand-alone java library and then build an Android User Interface that implements the library (not altering any of the original library's code during the stage of developing the user interface). The final project our team worked on was developing a soft-keyboard for the Android Operating System.

Throughout the semester our team invested a significant amount of time into research and proto-types (very little of the proto-type code made it into our final project). We experimented with features such as opengl, interfacing with sensors, and working with SQLite. Our team decided that having a complete and fully functioning application ready to display to the class was more important to us than an application that implemented many features but was incomplete overall, therefore only 3 of our many projects made the cut at the end of the semester. Taking all things into consideration, our team invested over 100 hours tinkering with the HTC thunderbolt we were given by the department, playing around with code and making the most of the time we were given with it. We hope you enjoy viewing our code and using our applications as much as we enjoyed developing them.

Health Calculator

Project Description

This project was developed in response to the original BMI application assigned to our class. Our team saw many improvements that could be made to the original application; among these improvements were user-interface design and the quality of the content delivered to the user. Any application can take in the value of variables and output an equation's result to the screen; however this information isn't very useful to a user who doesn't know what to do with it.

Knowing a Body Mass Index doesn't really help the average app user, so our team decided to build an application that took the information the user was already giving us (such as weight, age, etc) and use it to give the user insight to their overall health, including a description of their weight class, a list of possible symptoms, recommended course of action to treat the weight issues if they exist, and tips for someone looking to take control of their body weight. In addition to the information we provided the user, we wanted to ensure navigating the application would feel natural to the user, and that the layout was aesthetically pleasing.

When building our application we focused on using all of the screen space available on a standard cell phone screen, and adding a scroll feature to allow phones with smaller screens to use the app as well. We also added a second activity for displaying the health calculations and information so the user can focus on the information without a cluttered user interface. A final touch we added to complete the user experience was to prevent the user from entering any value other than 0-9 in the EditText field, ensuring they could not accidentally cause the application to crash.

Features

- Streamlined User Interface
- Ability to calculate:
 - Body Mass Index
 - Basal Metabolic Rate
- Works for both Males and Females
- Takes input in both U.S. as well as Metric units of measurement.
- Uses the users BMI and BMR to offer the user the following information:
 - Description of the user's weight class
 - Possible symptoms a user of that weight class may experience
 - Treatment options for the user
 - Tips for maintaining a healthy weight

Application Overview

In this section we will present a general overview of the application's layout and functionality.

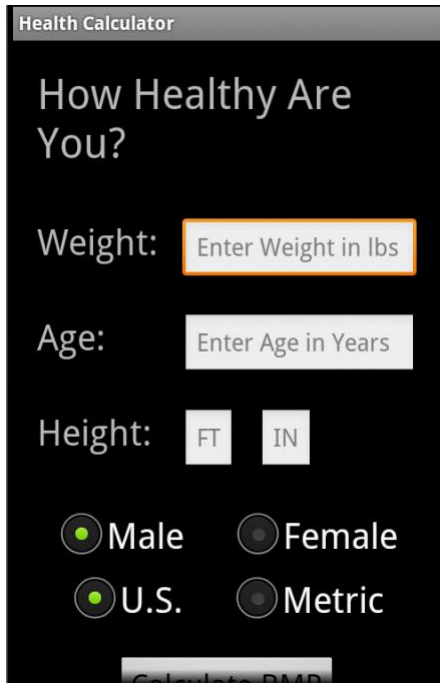


Figure 1 - InputActivity 1

Features

- The layout makes use of the entire screen
- When navigating the elements of the screen using the up-down arrows on the phone, the flow of navigation is controlled and natural (we dictate which elements you navigate to when you press the arrow keys)
- The user is able to enter their age, weight, height and gender
- The user can select between U.S. and Metric units



Figure 2 - InputActivity 2

Features

- When the user selects the Metric or U.S. RadioButton, the Hints for Weight and Height are updated.
- The entire layout is scrollable so any part of the layout that is off the screen can still be accessed.
- When Calculate BMR is pressed, this activity calls the ResultsActivity and passes the variables to it.

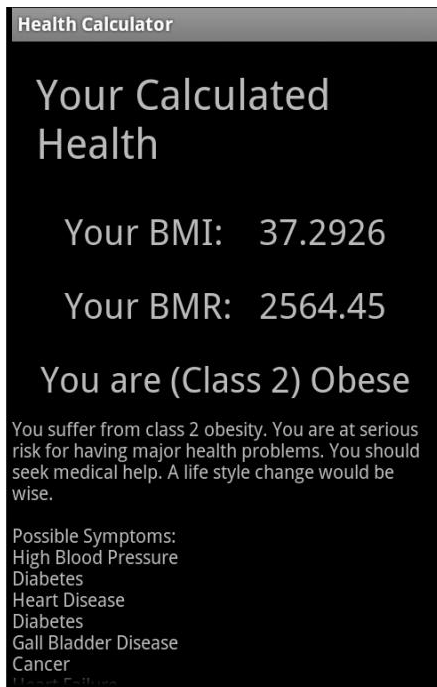


Figure 3 - ResultActivity 1

Features

- The user is presented with a BMI and BMR calculation.
- The user is told which weight class they belong in.
- A description of the weight class is provided.
- Possible Symptoms the user may experience are provided.

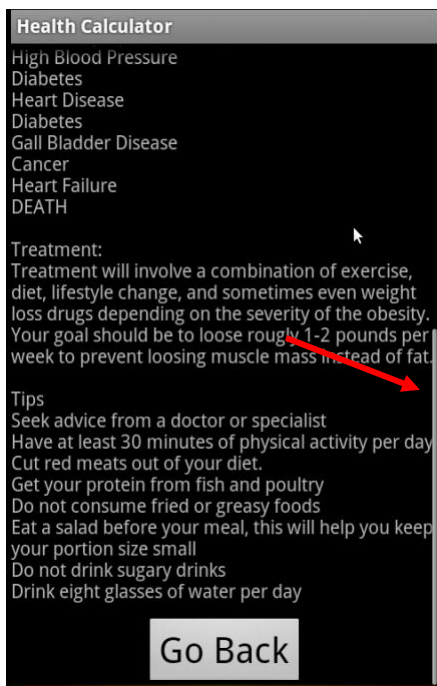


Figure 4 - ResultActivity 2

Features

- The user is provided with a possible course of treatment for either reducing or increasing their weight to achieve a normal BMI.
- The user is provided with a list of tips to help them during their treatment.
- The entire layout is contained in a scroll view to allow the generated content to extend off the screen.
- When the user presses the Go Back button, the activity pauses and calls InputActivity.

Code

HealthCalculations.java

This is the file that contains all the methods that handle the BMI and BMR calculations.

```
package com.awesomesauce.bmr;

/**
 * This class contains static method calls that deal with BMI and BMR
 * calculations
 *
 * @author William Blankenship
 * @date Nov 20, 2011
 * @email william.jblankenship(at)gmail.com
 */
public class HealthCalculations {

    /**
     * Calculates a persons Body Mass Index
     *
     * @param weight
     * The weight of the person
     * @param height
     * The height of the person in inches
     * @return
     * This function returns the BMI based on the values passed in
     */
    public static double calcBMI(int weight, int height)
    {
        //BMI = ( Weight in Pounds / ( Height in inches x Height in inches ) ) x 703
        return ((double)weight / ((double)height*(double)height)) * 703.0;
    }

    /**
     * Calculates a persons BMR
     *
     * @param weight
     * The weight of the person
     * @param age
     * The age of the person
     * @param height
     * The height of the person in inches
     * @param gender
     * The gender of the person (0=Male 1=Female)
     * @return
     * This function returns a double value representing the BMR of the
     * person
     */
    public static double calcBMR(int weight, int age, int height, int gender)
    {
        if(gender==0)
            return 66+(6.23*weight)+(12.7*height)-(6.8*age);
        else
            return 655+(4.35*weight)+(4.7*height)-(4.7*age);
    }

    /**
     * This function determines a person's weight class based on their BMI
     * value.
     *
     * @param BMI
     * The BMI of the person being evaluated
     * @return
     * This function returns a string identifying the weight class of the
     * person.
     */
    public static String isOverweight(double BMI)
    {

```

```

        if(BMI <= 18.5)
            return "You are Under Weight";
        else if(BMI < 25)
            return "You are Normal Weight";
        else if(BMI < 30)
            return "You are Overweight";
        else if(BMI < 35)
            return "You are (Class 1) Obese";
        else if(BMI < 40)
            return "You are (Class 2) Obese";
        else
            return "Morbidly Obese";
    }

/**
 * Returns a description of the weight class a person belongs to based
 * on their BMI
 *
 * @param BMI
 * The BMI of the person in question
 * @return
 * A string containing a description of the person's weight class
 */
public static String diagnostic(double BMI)
{
    if(BMI <= 18.5)
        return "Your BMI indicates you may be underweight. Seek adv" +
            "ice from a physician to see if you need to gain weight" +
            ".";
    else if(BMI < 25)
        return "Your BMI indicates your weight is right were it sho" +
            "uld be.";
    else if(BMI < 30)
        return "Your BMI indicates you are overweight. You would be" +
            "nefit from finding a healthy way to lower your overall" +
            " weight.";
    else if(BMI < 35)
        return "You suffer from class 1 obesity. You are at risk fo" +
            "serious health problems. Consult your doctor and consi" +
            "der reducing your weight 5-10%.";
    else if(BMI < 40)
        return "You suffer from class 2 obesity. You are at serious" +
            " risk for having major health problems. You should seek" +
            " medical help. A life style change would be w" +
            "ise.";
    else
        return "You are morbidly obese. Put down the droid and imme" +
            "diately contact your doctor. You need IMMEDIATE MEDICA" +
            "L ATTENTION. More then likely you suffer from a medica" +
            "l condition that has gone untreated. We wish you the b" +
            "est. God Speed.";
}

/**
 * This function generates a string of symptoms a person with a
 * specific BMI may experience.
 *
 * @param BMI
 * The BMI of the person
 * @return
 * This function returns a string containing a list of symptoms
 */
public static String symptoms(double BMI)
{
    String answer = "Possible Symptoms:\n";
    if(BMI <= 18.5)
    {
        answer += "Lower Immune System\nDissapeara" +
            "ance of Periods in women\nBone Loss\nMalnutrition\n\nT" +
            "he lower your BMI the greater your risk of developing " +
            "side effects.";
    }
}

```



```

    }
    else if(BMI < 25)
    {
        answer += "Good Health\nExcessive Happiness\nThe unexplai" +
            "nable urge to save random kittens from trees";
    }
    if(BMI >= 25)
    {
        answer += "High Blood Pressure\nDiabetes\nHeart Disease";
    }
    if(BMI >= 30)
    {
        answer += "\nDiabetes\nGall Bladder Disease\nCancer";
    }
    if(BMI >= 35)
    {
        answer += "\nHeart Failure\nDEATH";
    }
    return answer;
}

/**
 * This function determines a treatment plan for someone with a
 * specific BMI
 *
 * @param BMI
 * The BMI of the person
 * @return
 * A string containing treatment options for the patient
 */
public static String treatment(double BMI)
{
    if(BMI <= 18.5)
        return "Treatment:\nYou need to put on weight, however you " +
            "need to be cautious and not gain weight in the form of" +
            " excess fat. Increase your calorie intake and begin a w" +
            "orkout routine that will help convert those calories i" +
            "nto muscle instead of fat. A good target for weight ga" +
            "in is generally 1/2 pound per week, however this varie" +
            "s between individuals.\n\nTips:\nIncrease Calorie Inta" +
            "ke\nStick to an Excercise Routine\nDrink 6-8 Glasses o" +
            "f water a day\nEat Frequent small meals instead of 3 l" +
            "arge meals per day";
    else if(BMI < 25)
        return "";
    else if(BMI < 30)
        return "Treatment:\nSet a goal of loosing about 4-8 pounds " +
            "per month. Consume less calories than your BMR recomme" +
            "nds to loose weight. Do not overdue this, the less you" +
            " eat, the lower your BMR becomes. Starving yourself wil" +
            "l only cause malnutrition and make loosing weight hard" +
            "er. Increasing your daily physical activity will burn " +
            "additional calories making weight loss easier.\n\nTips" +
            ":\nCut red meats out of your diet.\nGet your protein f" +
            "rom fish and poultry\nDo not consume fried or greasy f" +
            "oods\nEat a salad before your meal, this will help you" +
            " keep your portion size small\nDo not drink sugary drin" +
            "ks\nDrink eight glasses of water per day";
    else
        return "Treatment:\nTreatment will involve a combination of" +
            " exercise, diet, lifestyle change, and sometimes even w" +
            "eight loss drugs depending on the severity of the obes" +
            "ity. Your goal should be to loose rougly 1-2 pounds pe" +
            "r week to prevent loosing muscle mass instead of fat. " +
            "\n\nTips\nSeek advice from a doctor or specialist\nHav" +
            "e at least 30 minutes of physical activity per day"+
            "\nCut red meats out of your diet.\nGet your protein f" +
            "rom fish and poultry\nDo not consume fried or greasy f" +
            "oods\nEat a salad before your meal, this will help you" +
            " keep your portion size small\nDo not drink sugary drin" +
            "ks\nDrink eight glasses of water per day";
}

```

```
}  
}
```

InputActivity.java

This file contains all the code for the first activities' interface (where the user inputs all of the data).

```
package com.awesomesauce.bmr;  
  
import android.app.Activity;  
import android.os.Bundle;  
import android.view.*;  
import android.widget.*;  
import android.content.Intent;  
  
/**  
 * This java file controls the main activity which receives input from the  
 * user.  
 *  
 * @author William Blankenship  
 * @email william.jblankenship(at)gmail.com  
 * @date 11/12/11  
 */  
public class InputActivity extends Activity {  
    @Override  
    public void onCreate(Bundle savedInstanceState) {  
        //This code runs when the program is first started on the phone  
        super.onCreate(savedInstanceState);  
        //We set the layout of this activity to main.xml  
        setContentView(R.layout.main);  
        populateHints();  
    }  
    @Override  
    protected void onStart() {  
        super.onStart();  
        populateHints();  
    }  
    @Override  
    protected void onResume() {  
        /* This code is ran every time the activity resumes (I.E. you  
         * return to this activity from another activity or application  
         */  
        super.onResume();  
        populateHints();  
    }  
    @Override  
    protected void onPause() {  
        super.onPause();  
        // Another activity is taking focus (this activity is about to be  
        // "paused").  
    }  
    @Override  
    protected void onStop() {  
        super.onStop();  
        // The activity is no longer visible (it is now "stopped")  
    }  
    @Override  
    protected void onDestroy() {  
        super.onDestroy();  
        // The activity is about to be destroyed.  
    }  
  
    /**  
     * This takes the user's input, calculates the BMI and BMR values,  
     * then passes these values to a new activity that will display them.  
     */  
    public void showResult(View v)  
    {  
        Intent intent = new Intent(this, ResultActivity.class);
```

```

    /* The next lines will retrieve the users input from the main.xml
    * layout file.*/
    EditText _weight = (EditText)findViewById(R.id.weightInput);
    EditText _age = (EditText)findViewById(R.id.ageInput);
    EditText _feet = (EditText)findViewById(R.id.feet);
    EditText _inches = (EditText)findViewById(R.id.inches);
    RadioButton _male = (RadioButton)findViewById(R.id.male);
    RadioButton _female = (RadioButton)findViewById(R.id.female);
    /* We set the default for all variables to 0 in case the user did not
    * enter values into the form.
    */
    int weight = 0;
    int age = 0;
    int gender = 0;
    int feet = 0;
    int inches = 0;

    //We check to make sure the user entered data then fill the variables
    if(_weight.getText().length() > 0)
        weight = Integer.parseInt(_weight.getText().toString());
    if(_age.getText().length() > 0)
        age = Integer.parseInt(_age.getText().toString());
    if(_male.isChecked())
    {
        gender = 0;
    }
    else if(_female.isChecked())
    {
        gender = 1;
    }
    if(_feet.getText().length() > 0)
        feet = Integer.parseInt(_feet.getText().toString());
    if(_inches.getText().length() > 0)
        inches = Integer.parseInt(_inches.getText().toString());
    /*End retrieving user's input*/

    /*We detect if the user has entered a metric unit or a US unit
    * of measurement*/
    int height = 0;
    RadioButton metric = (RadioButton)findViewById(R.id.metric);
    if(metric.isChecked())
    {
        /*If the unit is metric, we convert m to cm then add to cm
        * then convert to inches. We also convert kg to lbs*/
        height = (int) Math.ceil((feet*100+inches)*0.393700787);
        weight = (int) Math.ceil(weight*2.20462262);
    }
    else
    {
        /* Otherwise we convert to feet to inches then add to the value of
        * inches.
        */
        height= (feet*12) + inches;
    }

    //We then pass the user's input into the results activity
    intent.putExtra("weight", weight);
    intent.putExtra("age", age);
    intent.putExtra("gender", gender);
    intent.putExtra("height", height);
    startActivity(intent);
}

/**
 * Populates TextEdit hints that rely on user input
 *
 * @author William Blankenship
 */
public void populateHints() {
    /* We declare the metric RadioButton and feet, inches EditText

```

```

    * from main.xml so we can read and manipulate their values
    */
    RadioButton metric = (RadioButton)findViewById(R.id.metric);
    EditText _feet = (EditText)findViewById(R.id.feet);
    EditText _inches = (EditText)findViewById(R.id.inches);
    EditText _weight = (EditText)findViewById(R.id.weightInput);

    //We check to see if metric is checked
    if(metric.isChecked())
    {
        //Set the hint of feet and inches to reflect the metric system
        _feet.setHint("M");
        _inches.setHint("CM");
        _weight.setHint("Enter Weight in kg");
    }
    else
    {
        //Set the hint of feet and inches to reflect the US system
        _feet.setHint("FT");
        _inches.setHint("IN");
        _weight.setHint("Enter Weight in LBS");
    }
}
/**
 * The user is requesting to enter their height and weight in the U.S.
 * system.
 * @author William Blankenship
 */
public void toUS(View v) {
    EditText _weight = (EditText)findViewById(R.id.weightInput);
    EditText _feet = (EditText)findViewById(R.id.feet);
    EditText _inches = (EditText)findViewById(R.id.inches);

    _feet.setHint("FT");
    _inches.setHint("IN");
    _weight.setHint("Enter Weight in LBS");
}

/**
 * The user is requesting to enter their height in weight in the
 * metric system.
 * @author William Blankenship
 */
public void toMetric(View v) {
    EditText _feet = (EditText)findViewById(R.id.feet);
    EditText _inches = (EditText)findViewById(R.id.inches);
    EditText _weight = (EditText)findViewById(R.id.weightInput);

    _feet.setHint("M");
    _inches.setHint("CM");
    _weight.setHint("Enter Weight in kg");
}
}

```

ResultActivity.java

This file contains all of the code for the activity that displays the Health results.

```

package com.awesomesauce.bmr;

import java.text.DecimalFormat;

import android.app.Activity;
import android.os.Bundle;
import android.view.View;
import android.widget.TextView;

```

```

/**
 * This class controls that activity that displays the Health Results
 * based on the data received from InputActivity.java
 *
 * @author William Blankenship
 * @date Nov 18, 2011
 * @email william.jblankenship(at)gmail.com
 */
public class ResultActivity extends Activity{

    /**
     * Global Variables used to store the BMI and BMR rates
     */
    TextView bmiString;
    TextView bmrString;
    TextView weightAnalasys;
    TextView weightClass;

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.result);

        //Extras will receive all data passed from the previous activity
        Bundle extras = getIntent().getExtras();

        int weight = extras.getInt("weight");
        int age = extras.getInt("age");
        int gender = extras.getInt("gender");
        int height = extras.getInt("height");

        DecimalFormat f = new DecimalFormat("#.####");

        bmiString = (TextView)findViewById(R.id.bmi);
        String bmi = f.format(HealthCalculations.calcBMI(weight, height));
        bmiString.setText(bmi);

        bmrString = (TextView)findViewById(R.id.bmr);
        String bmr = f.format(HealthCalculations.calcBMR(weight, age, height, gender));
        bmrString.setText(bmr);

        weightClass = (TextView)findViewById(R.id.weightClass);
        weightClass.setText(HealthCalculations.isOverweight(Double.parseDouble(bmi)));

        weightAnalasys = (TextView)findViewById(R.id.weightAnalasys);
        String diagnosis = HealthCalculations.diagnostic(Double.parseDouble(bmi)) +
            "\n\n" + HealthCalculations.symptoms(Double.parseDouble(bmi)) +
            "\n\n" + HealthCalculations.treatment(Double.parseDouble(bmi));
        weightAnalasys.setText(diagnosis);

    }

    @Override
    protected void onStart() {
        super.onStart();
        // The activity is about to become visible.
    }

    @Override
    protected void onResume() {
        super.onResume();
        // The activity has become visible (it is now "resumed").
    }

    @Override
    protected void onPause() {
        super.onPause();
        // Another activity is taking focus (this activity is about to be "paused").
    }

    @Override
    protected void onStop() {
        super.onStop();
        // The activity is no longer visible (it is now "stopped")
    }
}

```

```

    }
    @Override
    protected void onDestroy() {
        super.onDestroy();
        // The activity is about to be destroyed.
    }

    public void exitResults(View V) {
        finish();
    }
}

```

main.xml

This is the xml file that constructs the main activity. Note how the entire layout is wrapped with a scrollview, this allows all elements in the layout to scroll vertically if they exceed the length of the screen.

```

<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent">
    <TextView
        android:id="@+id/Greeting"
        android:text="@string/Greeting"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentTop="true"
        android:layout_centerHorizontal="true"
        android:textSize="@+dimen/greetingFontSize"
        android:padding="@+dimen/spacing" />
    <TextView
        android:id="@+id/weight"
        android:text="@string/weight"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@+id/Greeting"
        android:textSize="@+dimen/fontSize"
        android:padding="@+dimen/spacing" />
    <EditText
        android:id="@+id/weightInput"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="@+dimen/spacing"
        android:layout_marginBottom="@+dimen/spacing"
        android:layout_marginRight="@+dimen/spacing"
        android:layout_toRightOf="@+id/weight"
        android:layout_below="@+id/Greeting"
        android:inputType="phone"
        android:digits="1234567890" />
    <TextView
        android:id="@+id/age"
        android:text="@string/age"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@+id/weight"
        android:textSize="@+dimen/fontSize"
        android:padding="@+dimen/spacing" />
    <EditText
        android:id="@+id/ageInput"
        android:hint="@string/ageInput"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="@+dimen/spacing"
        android:layout_marginRight="@+dimen/spacing"
        android:layout_marginBottom="@+dimen/spacing"
        android:layout_below="@+id/weight"
        android:layout_toRightOf="@+id/weight"
        android:inputType="phone"
        android:digits="1234567890" />

```

```

<TextView
    android:id="@+id/height"
    android:text="@string/height"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_below="@+id/age"
    android:textSize="@+dimen/fontSize"
    android:padding="@+dimen/spacing" />
<EditText
    android:id="@+id/feet"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginTop="@+dimen/spacing"
    android:layout_marginBottom="@+dimen/spacing"
    android:layout_below="@+id/age"
    android:layout_toRightOf="@+id/weight"
    android:inputType="phone"
    android:nextFocusDown="@+id/inches"
    android:nextFocusLeft="@+id/age"
    android:nextFocusRight="@+id/inches"
    android:digits="1234567890" />
<EditText
    android:id="@+id/inches"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginTop="@+dimen/spacing"
    android:layout_marginBottom="@+dimen/spacing"
    android:layout_marginRight="@+dimen/spacing"
    android:layout_below="@+id/age"
    android:layout_toRightOf="@+id/feet"
    android:layout_marginLeft="@+dimen/spacing"
    android:inputType="phone"
    android:nextFocusUp="@+id/feet"
    android:nextFocusDown="@+id/male"
    android:digits="1234567890" />
<RadioGroup
    android:id="@+id/gender"
    android:orientation="horizontal"
    android:layout_centerHorizontal="true"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_below="@+id/height"
    android:paddingTop="@+dimen/spacing"
    android:checkedButton="@+id/male">
    <RadioButton
        android:id="@+id/male"
        android:text="@string/male"
        android:textSize="@+dimen/fontSize"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginRight="@+dimen/spacing"
        android:nextFocusUp="@+id/inches"
        android:nextFocusDown="@+id/female" />
    <RadioButton
        android:id="@+id/female"
        android:text="@string/female"
        android:textSize="@+dimen/fontSize"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:nextFocusUp="@+id/male"
        android:nextFocusDown="@+id/us"
        android:layout_marginLeft="@+dimen/spacing" />
</RadioGroup>
<RadioGroup
    android:id="@+id/units"
    android:orientation="horizontal"
    android:layout_centerHorizontal="true"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_below="@+id/gender"
    android:checkedButton="@+id/us" >

```

```

        <RadioButton
            android:id="@+id/us"
            android:text="@string/us"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:textSize="@dimen/fontSize"
            android:layout_marginRight="@dimen/spacing"
            android:onClick="toUS"
            android:nextFocusUp="@id/female"
            android:nextFocusDown="@+id/metric" />
        <RadioButton
            android:id="@+id/metric"
            android:text="@string/metric"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:textSize="@dimen/fontSize"
            android:layout_marginLeft="@dimen/spacing"
            android:onClick="toMetric"
            android:nextFocusUp="@id/us"
            android:nextFocusDown="@+id/button1" />
    </RadioGroup>

    <Button
        android:id="@+id/button1"
        android:text="@string/button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@+id/units"
        android:layout_marginTop="@dimen/spacing"
        android:textSize="@+dimen/buttonFontSize"
        android:onClick="showResult"
        android:layout_centerHorizontal="true" />
</RelativeLayout>
</ScrollView>

```

result.xml

This xml documents constructs the activity that displays the BMI and BMR results. Again, take note of how the entire layout is encapsulated in a scrollview.

```

<?xml version="1.0" encoding="utf-8"?>
<!--
@author William Blankenship
@email william.jblankenship(at) gmail.com
@date 11/12/2011
-->
<ScrollView
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content">
    <RelativeLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content">

        <TextView
            android:id="@+id/resultGreeting"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="@string/resultGreeting"
            android:textSize="14pt"
            android:padding="@+dimen/spacing"
            android:layout_alignParentTop="true"
            android:layout_centerHorizontal="true" />

        <TableLayout
            android:id="@+id/resultTable"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_centerHorizontal="true"
            android:layout_below="@+id/resultGreeting">
            <TableRow>

```



```

        <TextView
            android:id="@+id/bmiMessage"
            android:text="@string/bmiMessage"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:textSize="@+dimen/fontSize"
            android:padding="@+dimen/resultSpacing" />
        <TextView
            android:id="@+id/bmi"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:textSize="@+dimen/fontSize"
            android:padding="@+dimen/resultSpacing" />
    </TableRow>
    <TableRow>
        <TextView
            android:id="@+id/bmrMessage"
            android:text="@string/bmrMessage"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:textSize="@+dimen/fontSize"
            android:padding="@+dimen/resultSpacing" />
        <TextView
            android:id="@+id/bmr"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:textSize="@+dimen/fontSize"
            android:padding="@+dimen/resultSpacing" />
    </TableRow>
</TableLayout>
<TextView
    android:id="@+id/weightClass"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:textSize="@+dimen/fontSize"
    android:padding="@+dimen/resultSpacing"
    android:layout_below="@+id/resultTable"
    android:layout_centerHorizontal="true" />
<TextView
    android:id="@+id/weightAnalsys"
    android:layout_below="@+id/weightClass"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content" />
<Button
    android:id="@+id/goBack"
    android:text="@string/goBack"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:textSize="@+dimen/fontSize"
    android:layout_margin="@+dimen/resultSpacing"
    android:layout_centerHorizontal="true"
    android:layout_below="@+id/weightAnalsys"
    android:onClick="exitResults" />
</RelativeLayout>
</ScrollView>

```

dimensions.xml

This document contains the dimension variables for all of the elements in the layout of the application (both the results activity and the main activity). If you want to change the font sizes or spacing on any of the activities, instead of having to go through and change a bunch of hardcoded values, you can simply change one variable here and it is reflected throughout the entire application.

```

<?xml version="1.0" encoding="utf-8"?>
<!--
@author William Blankenship
@email william.jblankenship(at)gmail.com
@date 11/12/11

```

```
-->

<resources>

    <dimen name="fontSize">12pt</dimen>
    <dimen name="greetingFontSize">14pt</dimen>
    <dimen name="buttonFontSize">10pt</dimen>
    <dimen name="spacing">18dp</dimen>
    <dimen name="resultSpacing">10dp</dimen>

</resources>
```

strings.xml

All of the text found throughout the application can be found in this one convenient location (with the exception of the hints generated based on user input). When you want to change the text somewhere in the application, instead of thumbing through pages of code you can come to this convenient xml document where everything is easy to locate and has been given intuitive names.

```
<?xml version="1.0" encoding="utf-8"?>
<!--
@author William Blankenship
@email william.jblankenship(at) gmail.com
@date 11/12/11
-->
<resources>

    <string name="app_name">Health Calculator</string>
    <string name="Greeting">How Healthy Are You?</string>
    <string name="weight">Weight:</string>
    <string name="weightInput">Enter Weight in LBS</string>
    <string name="ageInput">Enter Age in Years</string>
    <string name="us">U.S.</string>
    <string name="metric">Metric</string>
    <string name="age">Age:</string>
    <string name="height">Height:</string>
    <string name="male">Male</string>
    <string name="female">Female</string>
    <string name="button">Calculate BMR</string>
    <string name="bmiMessage">Your BMI:</string>
    <string name="bmrMessage">Your BMR:</string>
    <string name="resultGreeting">Your Calculated Health</string>
    <string name="goBack">Go Back</string>

</resources>
```

AndroidManifest.xml

This document contains the information for the two activities in the application along with the information other applications can use to call our application from an external app.

```
<?xml version="1.0" encoding="utf-8"?>
<!--
@author William Blankenship
@email william.jblankenship(at) gmail.com
@date 11/12/11
-->
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.awesomesauce.bmr"
    android:versionCode="1"
    android:versionName="sunshine" >

    <uses-sdk android:minSdkVersion="10" />

    <application
        android:icon="@drawable/healthicon"
```

```
        android:label="@string/app_name" >
        <activity
            android:label="@string/app_name"
            android:name=".InputActivity" >
            <intent-filter >
                <action android:name="android.intent.action.MAIN" />
                <action android:name="com.awesomesauce.bmr.start_app" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity
            android:name=".ResultActivity">

        </activity>
    </application>

</manifest>
```

SimplyFractions

Project Description

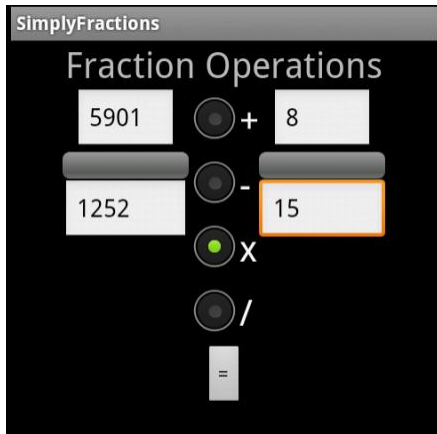
This application was developed to play around with external libraries. Our team first developed a library that implemented a vast majority of the topics we have covered in class. The library was able to do arithmetic on fractions by generating a list of prime numbers and using that list to factor, simplify, and find common denominators of fractions. After building the library we decided to port the application to android as practice. We built a layout that would allow the user to create two fractions and select the operation they wanted to perform, then we passed all of the data to our library and it created a new fraction to store the result in. We then displayed the result in a separate activity. The main goal of this application was not to develop a well built interface or complete solution, but to develop a java library that had full functionality and then prove how flexible it was by porting it to the android OS.

Features

- Built on a java library that has a full feature set
- Ability to perform fraction arithmetic
 - Addition
 - Subtraction
 - Multiplication
 - Division
- Works with both positive and negative integers in the numerator and denominator
- Simplifies the resulting fraction
- Able to simplify fractions much larger than most TI calculators can handle.
 - The application can handle a range of values between 2,147,483,648 and -2,147,483,648 inclusive.
 - This range can easily be extended to -9,223,372,036,854,775,808 to 9,223,372,036,854,775,807 by changing every occurrence of int to long in the fractions library. (We chose not to do this in order to increase the speed of the application and preserve resources on the phone).

Application Overview

In this section we will present a general overview of the application's layout and functionality.



Features

- The user can enter two fractions
- The user is able to choose between four mathematical operations to perform on the two fractions supplied



Features

- The resulting fraction produced from the operation performed in the previous activity is displayed
- The result is simplified

Code

Fraction.java

This file is part of the com.blankenship.fraction package. It is a data structure representing a fraction in memory.

```
package com.blankenship.fraction;

/**
 * This class is a data structure designed to store a fraction with a
 * numerator and denominator
 *
 * @author William Blankenship
 */
public final class Fraction extends PrimeNumbers{

    /**
     * To whom it may concern:
     * Is there any reason why Java does not support operator
     * overloading? It would make the following code look a lot cleaner.
     * fraction.add(fraction) would simply be fraction+fraction.
     * Just felt like ranting about that, I miss c++'s versatility.
     */
}
```

```

private int NUMERATOR;
private int DENOMINATOR;

/**
 * This constructor accepts 3 variables, accuracy, numerator, and
 * denominator. The numerator and denominator variables will be used
 * to construct the fraction. The accuracy represents the maximum
 * value of the prime numbers that will be used to factor the
 * numerator and denominator when reducing the fraction (for example
 * an accuracy of 3 would use {2,3} when reducing the fraction.
 *
 * -->An Important thing to note is all objects of type fraction share
 * the same list of prime numbers, the largest accuracy of all
 * fractions used in your program will be used when reducing fractions
 */
public Fraction(int accuracy, int numerator, int denominator)
{
    setFraction(numerator,denominator);
    if(getPrime(numberOfPrimes()-1)<accuracy)
    {
        genPrimes(accuracy);
    }
    reduceFraction();
}

/**
 * This constructor accepts 2 variables: numerator, and
 * denominator. The numerator and denominator variables will be used
 * to construct the fraction. The assumed accuracy of this fraction
 * will be 100
 */
public Fraction(int numerator, int denominator)
{
    setFraction(numerator,denominator);
    if(getPrime(numberOfPrimes()-1)<100)
    {
        genPrimes(100);
    }
}

/**
 * This constructor accepts 1 variable: wholeNumber. wholeNumber will
 * be used as the numerator and the fraction will have a denominator
 * of 1. The assumed accuracy of this fraction be 100
 */
public Fraction(int wholeNumber)
{
    setFraction(wholeNumber);
    if(getPrime(numberOfPrimes()-1)<100)
    {
        genPrimes(100);
    }
    reduceFraction();
}

/**
 * This function sets the numerator and denominator of the calling
 * object to the 2 variables provided.
 */
public void setFraction(int numerator, int denominator)
{
    setNumerator(numerator);
    setDenominator(denominator);
    reduceFraction();
}

/**
 * This function sets the numerator of the calling object to the
 * variable provided, the denominator is assumed to be 1.
 */
public void setFraction(int wholeNumber)

```

```

{
    setNumerator(wholeNumber);
    setDenominator(1);
}

/**
 * This function sets the numerator of the fraction to the variable
 * provided.
 */
public void setNumerator(int numerator)
{
    NUMERATOR = numerator;
}

/**
 * This function sets the denominator of the fraction to the variable
 * provided. If a 0 (zero) value is passed to this function it will
 * print an error message to the system's default output device and it
 * will set the denominator of the fraction to 1 to prevent division
 * by zero errors later in the program.
 */
public void setDenominator(int denominator)
{
    if(denominator!=0){DENOMINATOR = denominator;}
    else
    {
        System.out.println("Division By Zero Error!\n"+
            "Setting Denominator to 1 to avoid caos!");
        DENOMINATOR=1;
    }
}

/**
 * This function returns the value of the fraction's numerator.
 */
public int getNumerator()
{
    return NUMERATOR;
}

/**
 * This function returns the value of the fraction's denominator.
 */
public int getDenominator()
{
    return DENOMINATOR;
}

/**
 * This function will return the decimal equivalent of the fraction
 * as a double variable.
 */
public double getDouble()
{
    return ((double)getNumerator()/ (double)getDenominator());
}

/**
 * This function will return the decimal equivalent of the fraction
 * as a floating point variable.
 */
public float getFloat()
{
    return ((float)getNumerator()/ (float)getDenominator());
}

/**
 * This function reduces the calling object's numerator and
 * denominator by eliminating their common prime factors.
 */
protected void reduceFraction()

```

```

{
    int nArray[] = factor(this.getNumerator());
    int dArray[] = factor(this.getDenominator());
    int nLength = nArray.length;
    int dLength = dArray.length;
    int nArraySeek = 0;
    int dArraySeek = 0;

    while (nArraySeek < nLength && dArraySeek < dLength)
    {
        if (nArray[nArraySeek] == dArray[dArraySeek])
        {
            this.setNumerator(getNumerator() / nArray[nArraySeek]);
            this.setDenominator(getDenominator() / dArray[dArraySeek]);
            nArraySeek++;
            dArraySeek++;
        }
        else if (nArray[nArraySeek] < dArray[dArraySeek])
        {
            nArraySeek++;
        }
        else if (nArray[nArraySeek] > dArray[dArraySeek])
        {
            dArraySeek++;
        }
    }
}
}

```

FractionArithmetic.java

This file is part of the com.blankenship.fraction package. This class contains static functions that are used to perform mathematical operations on Fraction objects.

```

package com.blankenship.fraction;

/**
 * This class contains several static functions that can be used to
 * perform arithmetic on Fraction objects
 *
 * @author William Blankenship
 */
public class FractionArithmetic {

    /**
     * This function multiplies fraction1 by fraction2
     */
    static public Fraction multiply(Fraction fraction1, Fraction fraction2)
    {
        return new
Fraction(fraction1.getNumerator()*fraction2.getNumerator(),fraction1.getDenominator()*fraction2.g
etDenominator());
    }

    /**
     * This function multiplies fraction1 by the variable wholeNumber
     */
    static public Fraction multiply(Fraction fraction, int wholeNumber)
    {
        Fraction result = new
Fraction(wholeNumber*fraction.getNumerator(),fraction.getDenominator());
        result.reduceFraction();
        return result;
    }

    /**
     * This function will add the fraction fraction2 to fraction1
     */
    static public Fraction add(Fraction fraction1, Fraction fraction2)

```



```

        {
            Fraction uncommonDenominators = new
Fraction(fraction2.getDenominator(), fraction1.getDenominator());

            fraction2.setNumerator(fraction2.getNumerator()*uncommonDenominators.getDenominator());

            fraction2.setDenominator(fraction2.getDenominator()*uncommonDenominators.getDenominator()
);

            fraction1.setNumerator(fraction1.getNumerator()*uncommonDenominators.getNumerator());

            Fraction result = new
Fraction((fraction2.getNumerator()+fraction1.getNumerator()), fraction2.getDenominator());
            result.reduceFraction();

            return result;
        }

/**
 * This function will add the number wholeNumber to fraction
 */
static public Fraction add(Fraction fraction, int wholeNumber)
{
    return add(fraction, new Fraction(wholeNumber,1));
}

/**
 * This function subtracts the fraction fraction2 from fraction1.
 */
static public Fraction subtract(Fraction fraction1, Fraction fraction2)
{
    Fraction uncommonDenominators = new
Fraction(fraction1.getDenominator(), fraction2.getDenominator());

    fraction1.setNumerator(fraction1.getNumerator()*uncommonDenominators.getDenominator());

    fraction1.setDenominator(fraction1.getDenominator()*uncommonDenominators.getDenominator()
);

    fraction2.setNumerator(fraction2.getNumerator()*uncommonDenominators.getNumerator());

    Fraction result = new Fraction((fraction1.getNumerator()-
fraction2.getNumerator()), fraction1.getDenominator());
    result.reduceFraction();

    return result;
}

/**
 * This function will subtract wholeNumber from fraction1.
 */
static public Fraction subtract(Fraction fraction, int wholeNumber)
{
    return subtract(fraction, new Fraction(wholeNumber,1));
}

/**
 * This function will divide fraction1 by fraction2
 */
static public Fraction divide(Fraction fraction1, Fraction fraction2)
{
    return multiply(fraction1, new
Fraction(fraction2.getDenominator(), fraction2.getNumerator()));
}

/**
 * This function will divide the calling fraction by wholeNumber
 */
static public Fraction divide (Fraction fraction, int wholeNumber)

```

```

    {
        return multiply(fraction, new Fraction(1,wholeNumber));
    }
}

```

PrimeNumbers.java

This file is part of the com.blankenship.fraction package. This function is a data structure used to generate a list of prime numbers up to a set integer and use that list of prime numbers to factor any integer passed to the function.

```

package com.blankenship.fraction;

/**
 * This class only needs to be declared once in a program. It is used to
 * generate a static array of prime integers. Seeing as tampering with the
 * array of prime numbers can wreak havoc on our library, we take great
 * care to ensure no code outside of our library can change the values
 * stored in the array.
 *
 * @author William Blankenship
 */
abstract class PrimeNumbers {

    /**
     * All objects that are of type primeNumbers or extends primeNumbers
     * will share a single array of prime numbers to reduce memory
     * consumed by the program.
     */
    protected static int PRIMES[] = {2,3};

    /**
     * This function will generate a list of prime numbers less than or
     * equal to the variable numberOfPrimes.
     *
     * @author William Blankenship
     */
    public void genPrimes(int numberOfPrimes)
    {
        int currentNumber;
        if(PRIMES.length>0)
            currentNumber = PRIMES[PRIMES.length-1];
        else
            currentNumber = 0;
        while(currentNumber<numberOfPrimes)
        {
            int arrayIndex = PRIMES.length-1;
            int arraySeek=0;
            while(arraySeek<=arrayIndex&&currentNumber%PRIMES[arrayIndex-
arraySeek] !=0)
            {
                arraySeek++;
            }
            if(arraySeek>arrayIndex)
            {
                addToPrimes(currentNumber);
                arrayIndex++;
            }
            currentNumber++;
            arraySeek=1;
        }

        private void addToPrimes(int numberToAdd) {
            int length = PRIMES.length;
            int[] temp = new int[length+1];

```

```

        for(int i=0;i<length;i++) {
            temp[i] = PRIMES[i];
        }
        temp[length] = numberToAdd;
        PRIMES = temp.clone();
    }

    /**
     * This function will factor any number passed to it using the list
     * of generated prime numbers stored in the array PRIMES[]
     *
     * @author William Blankenship
     */
    public int[] factor(int number)
    {
        int factors[] = new int[0];
        int length=PRIMES.length;
        int arraySeek=0;
        while(arraySeek<length&&number!=1)
        {
            if(number%PRIMES[arraySeek]==0)
            {
                number /= PRIMES[arraySeek];
                int arrayLength = factors.length;
                int temp[] = new int[arrayLength+1];
                for(int i=0; i<arrayLength;i++)
                {
                    temp[i]=factors[i];
                }
                temp[arrayLength]=PRIMES[arraySeek];
                factors = temp;
                arraySeek=0;
            }
            else
            {
                arraySeek++;
            }
        }
        if(number!=1)
        {
            int arrayLength = factors.length;
            int temp[] = new int[arrayLength+1];
            for(int i=0; i<arrayLength;i++)
            {
                temp[i]=factors[i];
            }
            temp[arrayLength]=number;
            factors = temp;
        }
        return factors;
    }

    /**
     * This function will return the number of prime numbers stored in
     * the array PRIMES[] since PRIMES[] can not be accessed directly
     * outside of the package.
     */
    public int numberOfPrimes()
    {
        return PRIMES.length;
    }

    public int getPrime(int index)
    {
        if(index < numberOfPrimes() && index > 0)
            return PRIMES[index];
        else
            return 0;
    }

```

```
}
```

SimplyFractionsActivity.java

This function controls the main activity where the user enters fractions and selects the mathematical operation to perform on the fractions.

```
/**
 * @author William Blankenship
 * @date November 18, 2011
 * @course cs202
 */

package com.blankenship.simplyFractions;

import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.EditText;
import android.widget.RadioButton;
import com.blankenship.fraction.*;

public class SimplyFractionsActivity extends Activity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.fractionmain);
    }

    public void calculateResult(View v)
    {
        Intent intent = new Intent(this, ResultActivity.class);

        EditText _num1 = (EditText)findViewById(R.id.numerator1);
        EditText _den1 = (EditText)findViewById(R.id.denominator1);
        EditText _num2 = (EditText)findViewById(R.id.numerator2);
        EditText _den2 = (EditText)findViewById(R.id.denominator2);
        RadioButton _add = (RadioButton)findViewById(R.id.add);
        RadioButton _subtract = (RadioButton)findViewById(R.id.subtract);
        RadioButton _multiply = (RadioButton)findViewById(R.id.multiply);
        RadioButton _divide = (RadioButton)findViewById(R.id.divide);
        int numerator1 = 0;
        int denominator1 = 1;
        int numerator2 = 0;
        int denominator2 = 2;
        if(_num1.getText().toString().length()>0)
            numerator1 = Integer.parseInt(_num1.getText().toString());
        if(_den1.getText().toString().length()>0)
            denominator1 = Integer.parseInt(_den1.getText().toString());
        if(_num2.getText().toString().length()>0)
            numerator2 = Integer.parseInt(_num2.getText().toString());
        if(_den2.getText().toString().length()>0)
            denominator2 = Integer.parseInt(_den2.getText().toString());
        int accuracy = (int)Math.max(Math.sqrt(numerator1), Math.sqrt(numerator2));
        Fraction first = new Fraction(accuracy,numerator1, denominator1);
        Fraction second = new Fraction(accuracy,numerator2, denominator2);

        Fraction result;
        if(_add.isChecked())
        {
            result = FractionArithmetic.add(first, second);
        }
        else if(_subtract.isChecked())
        {
            result = FractionArithmetic.subtract(first, second);
        }
        else if(_multiply.isChecked())
        {

```

```

        result = FractionArithmetic.multiply(first, second);
    }
    else if(_divide.isChecked())
    {
        result = FractionArithmetic.divide(first, second);
    }
    else
        result = new Fraction(1);

    intent.putExtra("numerator", result.getNumerator());
    intent.putExtra("denominator", result.getDenominator());

    startActivity(intent);
}
}

```

ResultActivity.java

This function controls the result activity where the program displays the resulting fraction.

```

/**
 * @author William Blankenship
 * @date Nov 24, 2011
 * @course cs202
 */

package com.blankenship.simplyFractions;

import android.app.Activity;
import android.os.Bundle;
import android.view.View;
import android.widget.TextView;

public class ResultActivity extends Activity{
    @Override
    public void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.fractionresult);

        Bundle extras = getIntent().getExtras();

        int numerator = extras.getInt("numerator");
        int denominator = extras.getInt("denominator");

        TextView _num = (TextView)findViewById(R.id.numAnswer);
        TextView _den = (TextView)findViewById(R.id.denominatorAnswer);

        _num.setText(Integer.toString(numerator));
        _den.setText(Integer.toString(denominator));
    }

    public void exitResult(View v)
    {
        finish();
    }
}

```

fractionmain.xml

This is the xml document that constructs the main activity. It has 4 TextEdit fields to retrieve the numerator and denominator for both fractions. It uses progress bars to separate numerators and denominators (using these elements for the bar was the path of least resistance) and RadioButtons to determine which operation to perform.

```
<?xml version="1.0" encoding="utf-8"?>

<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent" >

    <TextView
        android:id="@+id/greeting"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentTop="true"
        android:layout_centerHorizontal="true"
        android:text="@string/name"
        android:textSize="@+dimen/fontSize" />

    <RadioGroup
        android:id="@+id/opperations"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@+id/greeting"
        android:layout_centerHorizontal="true"
        android:orientation="vertical"
        android:checkedButton="@+id/add" >

        <RadioButton
            android:id="@+id/add"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="+"
            android:textSize="@+dimen/fontSize"/>

        <RadioButton
            android:id="@+id/subtract"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="-"
            android:textSize="@+dimen/fontSize" />

        <RadioButton
            android:id="@+id/multiply"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="x"
            android:textSize="@+dimen/fontSize" />

        <RadioButton
            android:id="@+id/divide"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="/"
            android:textSize="@+dimen/fontSize" />
    </RadioGroup>

    <LinearLayout
        android:orientation="vertical"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_toLeftOf="@+id/opperations"
        android:layout_below="@+id/greeting" >

        <EditText
```

```

        android:id="@+id/numerator1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:hint="Num 1"
        android:inputType="text" />
<ProgressBar
    android:id="@+id/bar1"
    style="?android:attr/progressBarStyleHorizontal"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content" />
<EditText
    android:id="@+id/denominator1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_gravity="center"
    android:hint="Denom 1"
    android:inputType="text" />

</LinearLayout>

<LinearLayout
    android:id="@+id/frac2"
    android:orientation="vertical"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_toRightOf="@+id/operations"
    android:layout_below="@+id/greeting">

    <EditText
        android:id="@+id/numerator2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:hint="Num 2"
        android:inputType="text" />

    <ProgressBar
        android:id="@+id/bar2"
        style="?android:attr/progressBarStyleHorizontal"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content" />

    <EditText
        android:id="@+id/denominator2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:hint="Denom 2"
        android:inputType="text" />

</LinearLayout>
<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="="
    android:layout_below="@+id/operations"
    android:layout_centerHorizontal="true"
    android:layout_centerVertical="true"
    android:onClick="calculateResult"/>
</RelativeLayout>

```

fractionresult.xml

This xml document constructs the result activity. If you notice the TextViews do not have their text values initialized, this is because ResultActivity.java sets these values based on generated content.

```

<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent" >

```

```

<LinearLayout
    android:id="@+id/answer"
    android:orientation="vertical"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_centerInParent="true">

    <TextView
        android:id="@+id/numAnswer"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:textSize="18pt" />

    <ProgressBar
        android:id="@+id/barAnswer"
        style="?android:attr/progressBarStyleHorizontal"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content" />

    <TextView
        android:id="@+id/denominatorAnswer"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:textSize="18pt" />

</LinearLayout>

<TextView
    android:layout_above="@+id/answer"
    android:layout_centerHorizontal="true"
    android:textSize="18pt"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Result"/>

<Button
    android:text="GoBack"
    android:onClick="exitResult"
    android:layout_below="@+id/answer"
    android:layout_centerHorizontal="true"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content" />

</RelativeLayout>

```

dimensions.xml

This file contains a single variable that controls the font size of all text in the application.

```

<?xml version="1.0" encoding="utf-8"?>
<resources>
    <dimen name="fontSize">12pt</dimen>
</resources>

```

strings.xml

This document contains the name of the application and one string value.

```

<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="name">Fraction Operations</string>
    <string name="app_name">SimplyFractions</string>
</resources>

```


AndroidManifest.xml

This file contains the information required for the program to start both activities as well as the default activity to display when the program runs.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.blankenship.simplyFractions"
    android:versionCode="1"
    android:versionName="1.0" >

    <uses-sdk android:minSdkVersion="10" />

    <application
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name" >
        <activity
            android:label="@string/app_name"
            android:name=".SimplyFractionsActivity" >
            <intent-filter >
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity
            android:label="@string/app_name"
            android:name=".ResultActivity">

        </activity>
    </application>

</manifest>
```

SoftKeyboard

Project Description

We started this project specifically for the presentation in class. We didn't like the look of the default android keyboard or the key layout so we decided to build our own. Building a keyboard requires a lot of tedious code therefore we started with the basic android keyboard built by Google as a demo and gutted the code rebuilding it for our needs.



By gutting the code, we learned a lot about working with code written by other programmers, a skill that will be invaluable when we enter the industry. As we only worked with several of the files, we will only include the source code of the files we altered and make note of the changes we made. This keyboard is not ready for production use; it will only work on specific screen sizes and does not have any Latin symbols. The sole purpose of this project was to build a keyboard that looked nice and complimented the aesthetic features of our HealthCalculator during the in-class presentation.

Features

- Allows the user to focus on entering numbers into programs such as calculators without other symbols taking up screen space.
- A nice shade of gray that doesn't clash with the color scheme of our applications
- Two different keyboards can be called, one with numbers in ascending order and one with numbers in descending order.
- Allows the user to retract the keyboard unlike the default phone input.

Application Overview

In this section we will detail the features of the keyboard.

	<p><i>Features</i></p> <ul style="list-style-type: none">• Doesn't include any symbols the user will not need while using our app• Allows the user to retract the keyboard unlike the default phone input• Allows the user to delete text
	<p><i>Features</i></p> <ul style="list-style-type: none">• The keyboard can be in ascending or descending order

Code

In this section we will include only the files we altered. For the full source code refer to either Google's demo (comes with the Android SDK) or the .zip archive we will be sending with this document.

SoftKeyboard.java

In this file, we changed the phone input to use the qwerty layout so we could choose which of our two layouts we wanted to utilize from our applications by selecting either "text" or "phone" as an input method.

```
/*
 * Copyright (C) 2008-2009 Google Inc.
 *
 * Licensed under the Apache License, Version 2.0 (the "License"); you may not
 * use this file except in compliance with the License. You may obtain a copy of
 * the License at
 *
 * http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS, WITHOUT
 * WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the
 * License for the specific language governing permissions and limitations under
 * the License.
 */

package com.example.android.softkeyboard;

import android.inputmethodservice.InputMethodService;
import android.inputmethodservice.Keyboard;
import android.inputmethodservice.KeyboardView;
import android.text.method.MetaKeyKeyListener;
import android.util.Log;
import android.view.KeyCharacterMap;
import android.view.KeyEvent;
import android.view.View;
import android.view.inputmethod.CompletionInfo;
import android.view.inputmethod.EditorInfo;
import android.view.inputmethod.InputConnection;
import android.view.inputmethod.InputMethodManager;

import java.util.ArrayList;
import java.util.List;

/**
 * Example of writing an input method for a soft keyboard. This code is
 * focused on simplicity over completeness, so it should in no way be considered
 * to be a complete soft keyboard implementation. Its purpose is to provide
 * a basic example for how you would get started writing an input method, to
 * be fleshed out as appropriate.
 * @author Jon Picchietti
 */
@SuppressWarnings("unused")
public class SoftKeyboard extends InputMethodService
    implements KeyboardView.OnKeyboardActionListener {
    static final boolean DEBUG = false;

    /**
     * This boolean indicates the optional example code for performing
     * processing of hard keys in addition to regular text generation
     * from on-screen interaction. It would be used for input methods that
     * perform language translations (such as converting text entered on
     * a QWERTY keyboard to Chinese), but may not be used for input methods
     * that are primarily intended to be used for on-screen text entry.
     */
    static final boolean PROCESS_HARD_KEYS = true;
```

```

private KeyboardView mInputView;
private CandidateView mCandidateView;
private CompletionInfo[] mCompletions;

private StringBuilder mComposing = new StringBuilder();
private boolean mPredictionOn;
private boolean mCompletionOn;
private int mLastDisplayWidth;
private boolean mCapsLock;
private long mLastShiftTime;
private long mMetaState;

private LatinKeyboard mSymbolsKeyboard;
private LatinKeyboard mSymbolsShiftedKeyboard;
private LatinKeyboard mQwertyKeyboard;

private LatinKeyboard mCurKeyboard;

private String mWordSeparators;

/**
 * Main initialization of the input method component. Be sure to call
 * to super class.
 */
@Override public void onCreate() {
    super.onCreate();
    mWordSeparators = getResources().getString(R.string.word_separators);
}

/**
 * This is the point where you can do all of your UI initialization. It
 * is called after creation and any configuration change.
 */
@Override public void onInitializeInterface() {
    if (mQwertyKeyboard != null) {
        // Configuration changes can happen after the keyboard gets recreated,
        // so we need to be able to re-build the keyboards if the available
        // space has changed.
        int displayWidth = getMaxWidth();
        if (displayWidth == mLastDisplayWidth) return;
        mLastDisplayWidth = displayWidth;
    }
    mQwertyKeyboard = new LatinKeyboard(this, R.xml.qwerty);
    mSymbolsKeyboard = new LatinKeyboard(this, R.xml.symbols);
    mSymbolsShiftedKeyboard = new LatinKeyboard(this, R.xml.symbols_shift);
}

/**
 * Called by the framework when your view for creating input needs to
 * be generated. This will be called the first time your input method
 * is displayed, and every time it needs to be re-created such as due to
 * a configuration change.
 */
@Override public View onCreateInputView() {
    mInputView = (KeyboardView) getLayoutInflater().inflate(
        R.layout.input, null);
    mInputView.setOnKeyboardActionListener(this);
    mInputView.setKeyboard(mQwertyKeyboard);
    return mInputView;
}

/**
 * Called by the framework when your view for showing candidates needs to
 * be generated, like {@link #onCreateInputView}.
 */
@Override public View onCreateCandidatesView() {
    mCandidateView = new CandidateView(this);
    mCandidateView.setService(this);
    return mCandidateView;
}

```

```

/**
 * This is the main point where we do our initialization of the input method
 * to begin operating on an application. At this point we have been
 * bound to the client, and are now receiving all of the detailed information
 * about the target of our edits.
 */
@Override public void onStartInput(EditorInfo attribute, boolean restarting) {
    super.onStartInput(attribute, restarting);

    // Reset our state. We want to do this even if restarting, because
    // the underlying state of the text editor could have changed in any way.
    mComposing.setLength(0);
    updateCandidates();

    if (!restarting) {
        // Clear shift states.
        mMetaState = 0;
    }

    mPredictionOn = false;
    mCompletionOn = false;
    mCompletions = null;

    // We are now going to initialize our state based on the type of
    // text being edited.
    switch (attribute.inputType & EditorInfo.TYPE_MASK_CLASS) {
        case EditorInfo.TYPE_CLASS_NUMBER:
        case EditorInfo.TYPE_CLASS_DATETIME:
            // Numbers and dates default to the symbols keyboard, with
            // no extra features.
            mCurKeyboard = mSymbolsKeyboard;
            break;

        case EditorInfo.TYPE_CLASS_PHONE:
            // Phones will also default to the symbols keyboard, though
            // often you will want to have a dedicated phone keyboard.
            mCurKeyboard = mQwertyKeyboard;
            break;

        case EditorInfo.TYPE_CLASS_TEXT:
            // This is general text editing. We will default to the
            // normal alphabetic keyboard, and assume that we should
            // be doing predictive text (showing candidates as the
            // user types).
            mCurKeyboard = mSymbolsKeyboard;
            mPredictionOn = true;

            // We now look for a few special variations of text that will
            // modify our behavior.
            int variation = attribute.inputType & EditorInfo.TYPE_MASK_VARIATION;
            if (variation == EditorInfo.TYPE_TEXT_VARIATION_PASSWORD ||
                variation == EditorInfo.TYPE_TEXT_VARIATION_VISIBLE_PASSWORD) {
                // Do not display predictions / what the user is typing
                // when they are entering a password.
                mPredictionOn = false;
            }

            if (variation == EditorInfo.TYPE_TEXT_VARIATION_EMAIL_ADDRESS
                || variation == EditorInfo.TYPE_TEXT_VARIATION_URI
                || variation == EditorInfo.TYPE_TEXT_VARIATION_FILTER) {
                // Our predictions are not useful for e-mail addresses
                // or URIs.
                mPredictionOn = false;
            }

            if ((attribute.inputType & EditorInfo.TYPE_TEXT_FLAG_AUTO_COMPLETE) != 0) {
                // If this is an auto-complete text view, then our predictions
                // will not be shown and instead we will allow the editor
                // to supply their own. We only show the editor's
                // candidates when in fullscreen mode, otherwise relying
                // on it displaying its own UI.
            }
    }
}

```

```

        mPredictionOn = false;
        mCompletionOn = isFullscreenMode();
    }

    // We also want to look at the current state of the editor
    // to decide whether our alphabetic keyboard should start out
    // shifted.
    updateShiftKeyState(attribute);
    break;

default:
    // For all unknown input types, default to the alphabetic
    // keyboard with no special features.
    mCurKeyboard = mQwertyKeyboard;
    updateShiftKeyState(attribute);
}

// Update the label on the enter key, depending on what the application
// says it will do.
mCurKeyboard.setImeOptions(getResources(), attribute.imeOptions);
}

/**
 * This is called when the user is done editing a field. We can use
 * this to reset our state.
 */
@Override public void onFinishInput() {
    super.onFinishInput();

    // Clear current composing text and candidates.
    mComposing.setLength(0);
    updateCandidates();

    // We only hide the candidates window when finishing input on
    // a particular editor, to avoid popping the underlying application
    // up and down if the user is entering text into the bottom of
    // its window.
    setCandidatesViewShown(false);

    mCurKeyboard = mQwertyKeyboard;
    if (mInputView != null) {
        mInputView.closing();
    }
}

@Override public void onStartInputView(EditorInfo attribute, boolean restarting) {
    super.onStartInputView(attribute, restarting);
    // Apply the selected keyboard to the input view.
    mInputView.setKeyboard(mCurKeyboard);
    mInputView.closing();
}

/**
 * Deal with the editor reporting movement of its cursor.
 */
@Override public void onUpdateSelection(int oldSelStart, int oldSelEnd,
    int newSelStart, int newSelEnd,
    int candidatesStart, int candidatesEnd) {
    super.onUpdateSelection(oldSelStart, oldSelEnd, newSelStart, newSelEnd,
        candidatesStart, candidatesEnd);

    // If the current selection in the text view changes, we should
    // clear whatever candidate text we have.
    if (mComposing.length() > 0 && (newSelStart != candidatesEnd
        || newSelEnd != candidatesEnd)) {
        mComposing.setLength(0);
        updateCandidates();
        InputConnection ic = getCurrentInputConnection();
        if (ic != null) {
            ic.finishComposingText();
        }
    }
}

```

```

    }
}

/**
 * This tells us about completions that the editor has determined based
 * on the current text in it. We want to use this in fullscreen mode
 * to show the completions ourselves, since the editor can not be seen
 * in that situation.
 */
@Override public void onDisplayCompletions(CompletionInfo[] completions) {
    if (mCompletionOn) {
        mCompletions = completions;
        if (completions == null) {
            setSuggestions(null, false, false);
            return;
        }

        List<String> stringList = new ArrayList<String>();
        for (int i=0; i<(completions != null ? completions.length : 0); i++) {
            CompletionInfo ci = completions[i];
            if (ci != null) stringList.add(ci.getText().toString());
        }
        setSuggestions(stringList, true, true);
    }
}

/**
 * This translates incoming hard key events in to edit operations on an
 * InputConnection. It is only needed when using the
 * PROCESS_HARD_KEYS option.
 */
private boolean translateKeyDown(int keyCode, KeyEvent event) {
    mMetaState = MetaKeyListener.handleKeyDown(mMetaState,
        keyCode, event);
    int c = event.getUnicodeChar(MetaKeyListener.getMetaState(mMetaState));
    mMetaState = MetaKeyListener.adjustMetaAfterKeypress(mMetaState);
    InputConnection ic = getCurrentInputConnection();
    if (c == 0 || ic == null) {
        return false;
    }

    boolean dead = false;

    if ((c & KeyCharacterMap.COMBINING_ACCENT) != 0) {
        dead = true;
        c = c & KeyCharacterMap.COMBINING_ACCENT_MASK;
    }

    if (mComposing.length() > 0) {
        char accent = mComposing.charAt(mComposing.length() - 1);
        int composed = KeyEvent.getDeadChar(accent, c);

        if (composed != 0) {
            c = composed;
            mComposing.setLength(mComposing.length() - 1);
        }
    }

    onKey(c, null);

    return true;
}

/**
 * Use this to monitor key events being delivered to the application.
 * We get first crack at them, and can either resume them or let them
 * continue to the app.
 */
@Override public boolean onKeyDown(int keyCode, KeyEvent event) {
    switch (keyCode) {
        case KeyEvent.KEYCODE_BACK:

```

```

        // The InputMethodService already takes care of the back
        // key for us, to dismiss the input method if it is shown.
        // However, our keyboard could be showing a pop-up window
        // that back should dismiss, so we first allow it to do that.
        if (event.getRepeatCount() == 0 && mInputView != null) {
            if (mInputView.handleBack()) {
                return true;
            }
        }
        break;

    case KeyEvent.KEYCODE_DEL:
        // Special handling of the delete key: if we currently are
        // composing text for the user, we want to modify that instead
        // of let the application to the delete itself.
        if (mComposing.length() > 0) {
            onKey(Keyboard.KEYCODE_DELETE, null);
            return true;
        }
        break;

    case KeyEvent.KEYCODE_ENTER:
        // Let the underlying text editor always handle these.
        return false;

    default:
        // For all other keys, if we want to do transformations on
        // text being entered with a hard keyboard, we need to process
        // it and do the appropriate action.
        if (PROCESS_HARD_KEYS) {
            if (keyCode == KeyEvent.KEYCODE_SPACE
                && (event.getMetaState() & KeyEvent.META_ALT_ON) != 0) {
                // A silly example: in our input method, Alt+Space
                // is a shortcut for 'android' in lower case.
                InputConnection ic = getCurrentInputConnection();
                if (ic != null) {
                    // First, tell the editor that it is no longer in the
                    // shift state, since we are consuming this.
                    ic.clearMetaKeyStates(KeyEvent.META_ALT_ON);
                    keyDownUp(KeyEvent.KEYCODE_A);
                    keyDownUp(KeyEvent.KEYCODE_N);
                    keyDownUp(KeyEvent.KEYCODE_D);
                    keyDownUp(KeyEvent.KEYCODE_R);
                    keyDownUp(KeyEvent.KEYCODE_O);
                    keyDownUp(KeyEvent.KEYCODE_I);
                    keyDownUp(KeyEvent.KEYCODE_D);
                    // And we consume this event.
                    return true;
                }
            }
            if (mPredictionOn && translateKeyDown(keyCode, event)) {
                return true;
            }
        }
    }

    return super.onKeyDown(keyCode, event);
}

/**
 * Use this to monitor key events being delivered to the application.
 * We get first crack at them, and can either resume them or let them
 * continue to the app.
 */
@Override public boolean onKeyUp(int keyCode, KeyEvent event) {
    // If we want to do transformations on text being entered with a hard
    // keyboard, we need to process the up events to update the meta key
    // state we are tracking.
    if (PROCESS_HARD_KEYS) {
        if (mPredictionOn) {
            mMetaState = MetaKeyListener.handleKeyUp(mMetaState,

```



```

        keyCode, event);
    }

    return super.onKeyUp(keyCode, event);
}

/**
 * Helper function to commit any text being composed in to the editor.
 */
private void commitTyped(InputConnection inputConnection) {
    if (mComposing.length() > 0) {
        inputConnection.commitText(mComposing, mComposing.length());
        mComposing.setLength(0);
        updateCandidates();
    }
}

/**
 * Helper to update the shift state of our keyboard based on the initial
 * editor state.
 */
private void updateShiftKeyState(EditorInfo attr) {
    if (attr != null
        && mInputView != null && mQwertyKeyboard == mInputView.getKeyboard()) {
        int caps = 0;
        EditorInfo ei = getCurrentInputEditorInfo();
        if (ei != null && ei.inputType != EditorInfo.TYPE_NULL) {
            caps = getCurrentInputConnection().getCursorCapsMode(attr.inputType);
        }
        mInputView.setShifted(mCapsLock || caps != 0);
    }
}

/**
 * Helper to determine if a given character code is alphabetic.
 */
private boolean isAlphabet(int code) {
    if (Character.isLetter(code)) {
        return true;
    } else {
        return false;
    }
}

/**
 * Helper to send a key down / key up pair to the current editor.
 */
private void keyDownUp(int keyEventCode) {
    getCurrentInputConnection().sendKeyEvent(
        new KeyEvent(KeyEvent.ACTION_DOWN, keyEventCode));
    getCurrentInputConnection().sendKeyEvent(
        new KeyEvent(KeyEvent.ACTION_UP, keyEventCode));
}

/**
 * Helper to send a character to the editor as raw key events.
 */
private void sendKey(int keyCode) {
    switch (keyCode) {
        case '\n':
            keyDownUp(KeyEvent.KEYCODE_ENTER);
            break;
        default:
            if (keyCode >= '0' && keyCode <= '9') {
                keyDownUp(keyCode - '0' + KeyEvent.KEYCODE_0);
            } else {
                getCurrentInputConnection().commitText(String.valueOf((char) keyCode), 1);
            }
            break;
    }
}

```

```

}

// Implementation of KeyboardViewListener

public void onKey(int primaryCode, int[] keyCodes) {
    if (isWordSeparator(primaryCode)) {
        // Handle separator
        if (mComposing.length() > 0) {
            commitTyped(getCurrentInputConnection());
        }
        sendKey(primaryCode);
        updateShiftKeyState(getCurrentInputEditorInfo());
    } else if (primaryCode == Keyboard.KEYCODE_DELETE) {
        handleBackspace();
    } else if (primaryCode == Keyboard.KEYCODE_SHIFT) {
        handleShift();
    } else if (primaryCode == Keyboard.KEYCODE_CANCEL) {
        handleClose();
        return;
    } else if (primaryCode == LatinKeyboardView.KEYCODE_OPTIONS) {
        // Show a menu or somethin'
    } else if (primaryCode == Keyboard.KEYCODE_MODE_CHANGE
        && mInputView != null) {
        Keyboard current = mInputView.getKeyboard();
        if (current == mSymbolsKeyboard || current == mSymbolsShiftedKeyboard) {
            current = mQwertyKeyboard;
        } else {
            current = mSymbolsKeyboard;
        }
        mInputView.setKeyboard(current);
        if (current == mSymbolsKeyboard) {
            current.setShifted(false);
        }
    } else {
        handleCharacter(primaryCode, keyCodes);
    }
}

public void onText(CharSequence text) {
    InputConnection ic = getCurrentInputConnection();
    if (ic == null) return;
    ic.beginBatchEdit();
    if (mComposing.length() > 0) {
        commitTyped(ic);
    }
    ic.commitText(text, 0);
    ic.endBatchEdit();
    updateShiftKeyState(getCurrentInputEditorInfo());
}

/**
 * Update the list of available candidates from the current composing
 * text. This will need to be filled in by however you are determining
 * candidates.
 */
private void updateCandidates() {
    if (!mCompletionOn) {
        if (mComposing.length() > 0) {
            ArrayList<String> list = new ArrayList<String>();
            list.add(mComposing.toString());
            setSuggestions(list, true, true);
        } else {
            setSuggestions(null, false, false);
        }
    }
}

public void setSuggestions(List<String> suggestions, boolean completions,
    boolean typedWordValid) {
    if (suggestions != null && suggestions.size() > 0) {
        setCandidatesViewShown(true);
    }
}

```

```

    } else if (isExtractViewShown()) {
        setCandidatesViewShown(true);
    }
    if (mCandidateView != null) {
        mCandidateView.setSuggestions(suggestions, completions, typedWordValid);
    }
}

private void handleBackspace() {
    final int length = mComposing.length();
    if (length > 1) {
        mComposing.delete(length - 1, length);
        getCurrentInputConnection().setComposingText(mComposing, 1);
        updateCandidates();
    } else if (length > 0) {
        mComposing.setLength(0);
        getCurrentInputConnection().commitText("", 0);
        updateCandidates();
    } else {
        keyDownUp(KeyEvent.KEYCODE_DEL);
    }
    updateShiftKeyState(getCurrentInputEditorInfo());
}

private void handleShift() {
    if (mInputView == null) {
        return;
    }

    Keyboard currentKeyboard = mInputView.getKeyboard();
    if (mQwertyKeyboard == currentKeyboard) {
        // Alphabet keyboard
        checkToggleCapsLock();
        mInputView.setShifted(mCapsLock || !mInputView.isShifted());
    } else if (currentKeyboard == mSymbolsKeyboard) {
        mSymbolsKeyboard.setShifted(true);
        mInputView.setKeyboard(mSymbolsShiftedKeyboard);
        mSymbolsShiftedKeyboard.setShifted(true);
    } else if (currentKeyboard == mSymbolsShiftedKeyboard) {
        mSymbolsShiftedKeyboard.setShifted(false);
        mInputView.setKeyboard(mSymbolsKeyboard);
        mSymbolsKeyboard.setShifted(false);
    }
}

private void handleCharacter(int primaryCode, int[] keyCodes) {
    if (isInputViewShown()) {
        if (mInputView.isShifted()) {
            primaryCode = Character.toUpperCase(primaryCode);
        }
    }
    if (isAlphabet(primaryCode) && mPredictionOn) {
        mComposing.append((char) primaryCode);
        getCurrentInputConnection().setComposingText(mComposing, 1);
        updateShiftKeyState(getCurrentInputEditorInfo());
        updateCandidates();
    } else {
        getCurrentInputConnection().commitText(
            String.valueOf((char) primaryCode), 1);
    }
}

private void handleClose() {
    commitTyped(getCurrentInputConnection());
    requestHideSelf(0);
    mInputView.closing();
}

private void checkToggleCapsLock() {
    long now = System.currentTimeMillis();
    if (mLastShiftTime + 800 > now) {

```

```

        mCapsLock = !mCapsLock;
        mLastShiftTime = 0;
    } else {
        mLastShiftTime = now;
    }
}

private String getWordSeparators() {
    return mWordSeparators;
}

public boolean isWordSeparator(int code) {
    String separators = getWordSeparators();
    return separators.contains(String.valueOf((char) code));
}

public void pickDefaultCandidate() {
    pickSuggestionManually(0);
}

public void pickSuggestionManually(int index) {
    if (mCompletionOn && mCompletions != null && index >= 0
        && index < mCompletions.length) {
        CompletionInfo ci = mCompletions[index];
        getCurrentInputConnection().commitCompletion(ci);
        if (mCandidateView != null) {
            mCandidateView.clear();
        }
        updateShiftKeyState(getCurrentInputEditorInfo());
    } else if (mComposing.length() > 0) {
        // If we were generating candidate suggestions for the current
        // text, we would commit one of them here. But for this sample,
        // we will just commit the current text.
        commitTyped(getCurrentInputConnection());
    }
}

public void swipeRight() {
    if (mCompletionOn) {
        pickDefaultCandidate();
    }
}

public void swipeLeft() {
    handleBackspace();
}

public void swipeDown() {
    handleClose();
}

public void swipeUp() {
}

public void onPress(int primaryCode) {
}

public void onRelease(int primaryCode) {
}
}

```

qwerty.xml

This file contains the layout for the keyboard displaying characters in ascending order from top to bottom.

```

<?xml version="1.0" encoding="utf-8"?>
<!--
/*
**

```

```

** Copyright 2008, The Android Open Source Project
**
** Licensed under the Apache License, Version 2.0 (the "License");
** you may not use this file except in compliance with the License.
** You may obtain a copy of the License at
**
**     http://www.apache.org/licenses/LICENSE-2.0
**
** Unless required by applicable law or agreed to in writing, software
** distributed under the License is distributed on an "AS IS" BASIS,
** WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
** See the License for the specific language governing permissions and
** limitations under the License.
*/

REVISED BY: Jon Picchietti
-->

<Keyboard xmlns:android="http://schemas.android.com/apk/res/android"
    android:keyWidth="32%p"
    android:horizontalGap="3px"
    android:verticalGap="3px"
    android:keyHeight="@dimen/key_height"
    >

    <Row>
        <Key android:codes="49" android:keyLabel="1" android:keyEdgeFlags="left"/>
        <Key android:codes="50" android:keyLabel="2"/>
        <Key android:codes="51" android:keyLabel="3" android:keyEdgeFlags="right"/>
    </Row>

    <Row>
        <Key android:codes="52" android:keyLabel="4" android:keyEdgeFlags="left"/>
        <Key android:codes="53" android:keyLabel="5"/>
        <Key android:codes="54" android:keyLabel="6" android:keyEdgeFlags="right"/>
    </Row>

    <Row>
        <Key android:codes="55" android:keyLabel="7" android:keyEdgeFlags="left"/>
        <Key android:codes="56" android:keyLabel="8"/>
        <Key android:codes="57" android:keyLabel="9" android:keyEdgeFlags="right"/>
    </Row>

    <Row>
        <Key android:codes="-5" android:keyIcon="@drawable/sym_keyboard_delete"
            android:keyEdgeFlags="left" android:isRepeatable="true"/>
        <Key android:codes="48" android:keyLabel="0"/>
        <Key android:codes="-3" android:keyIcon="@drawable/sym_keyboard_done"
            android:keyEdgeFlags="right"/>
    </Row>

</Keyboard>

```

symbols.xml

This file contains the xml for the keyboard with keys in descending order from top to bottom.

```

<?xml version="1.0" encoding="utf-8"?>
<!--
/*
**
** Copyright 2008, The Android Open Source Project
**
** Licensed under the Apache License, Version 2.0 (the "License");
** you may not use this file except in compliance with the License.
** You may obtain a copy of the License at
**
**     http://www.apache.org/licenses/LICENSE-2.0
**
** Unless required by applicable law or agreed to in writing, software

```

```

** distributed under the License is distributed on an "AS IS" BASIS,
** WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
** See the License for the specific language governing permissions and
** limitations under the License.
*/
REVISED BY: Jon Picchietti
-->

<Keyboard xmlns:android="http://schemas.android.com/apk/res/android"
    android:keyWidth="32%p"
    android:horizontalGap="3px"
    android:verticalGap="3px"
    android:keyHeight="@dimen/key_height"
    >

    <Row>
        <Key android:codes="-5" android:keyIcon="@drawable/sym_keyboard_delete"
            android:keyEdgeFlags="left" android:isRepeatable="true"/>
        <Key android:codes="48" android:keyLabel="0"/>
        <Key android:codes="-3" android:keyIcon="@drawable/sym_keyboard_done"
            android:keyEdgeFlags="right"/>
    </Row>

    <Row>
        <Key android:codes="55" android:keyLabel="7" android:keyEdgeFlags="left"/>
        <Key android:codes="56" android:keyLabel="8"/>
        <Key android:codes="57" android:keyLabel="9" android:keyEdgeFlags="right"/>
    </Row>

    <Row>
        <Key android:codes="52" android:keyLabel="4" android:keyEdgeFlags="left"/>
        <Key android:codes="53" android:keyLabel="5"/>
        <Key android:codes="54" android:keyLabel="6" android:keyEdgeFlags="right"/>
    </Row>

    <Row>
        <Key android:codes="49" android:keyLabel="1" android:keyEdgeFlags="left"/>
        <Key android:codes="50" android:keyLabel="2"/>
        <Key android:codes="51" android:keyLabel="3" android:keyEdgeFlags="right"/>
    </Row>
</Keyboard>

```