Mobile App Design Project
Doodle App

Description:

This App takes user touch input and allows the user to draw colored lines on the screen with touch gestures. There will be a menu to allow the user to set Pen Color with Red, Green, Blue, and Alpha Channels. The user will also have a menu to change the thickness of the drawing pen.

Phase 1: Create the App Project

1. Start Eclipse and select New Project -> "Android Application Project"
2. Fill out the fields with the following:
    a. Application Name: DoodleApp
    b. Project Name: DoodleApp
    c. Package name: com.example.doodleapp
3. Click Next
4. Click Next at the Configure Project Screen
5. Click Next at the Configure Launcher Icon Screen.
6. Click Next at the Create Activity Screen.
7. Fill out the following fields in the New Blank Activity Screen
    a. Activity Name: Doodle
    b. LayoutName: activity_doodle
    c. Navigation Type: None
8. Click "Finish"

Phase 2: User Interface and XML Design:

We will need to create three XML files for this App:

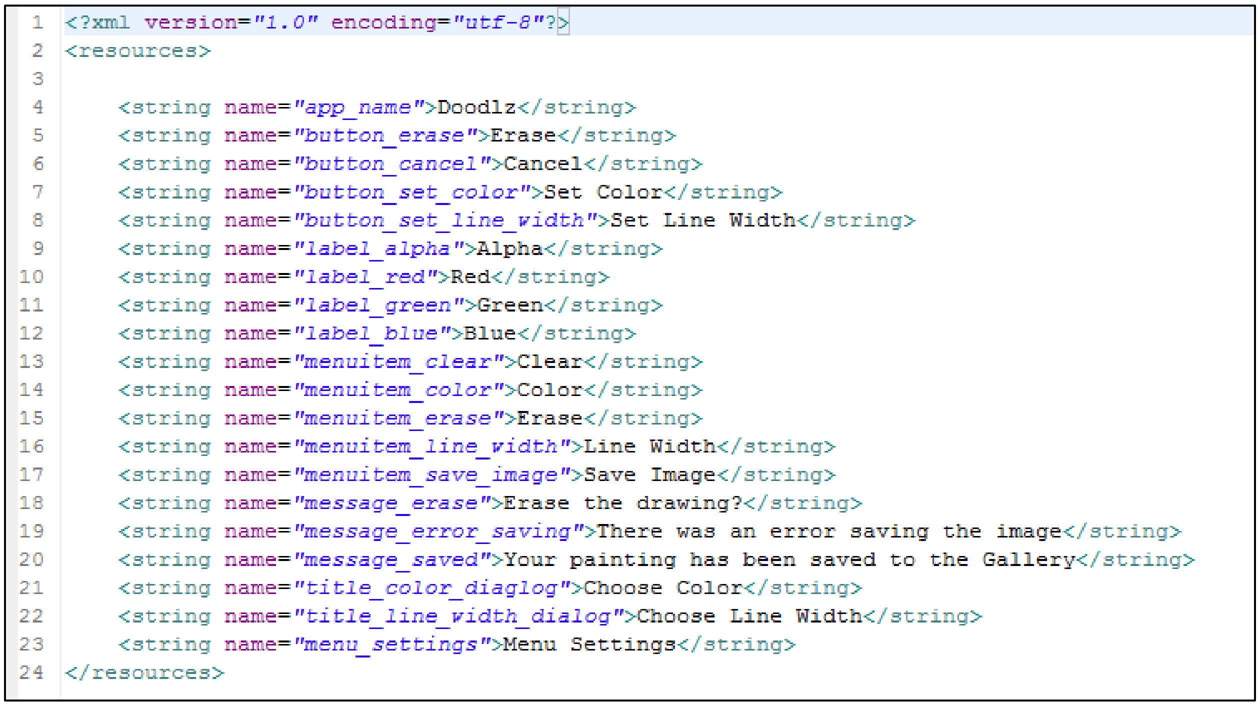
main.xml will hold the reference to the DoodleView object object which is a custom implementation of View

color_dialog.xml will hold the visual layout for the Color Settings.

width_dialog.xml will hold the visual layout for the Width Settings.

Process:

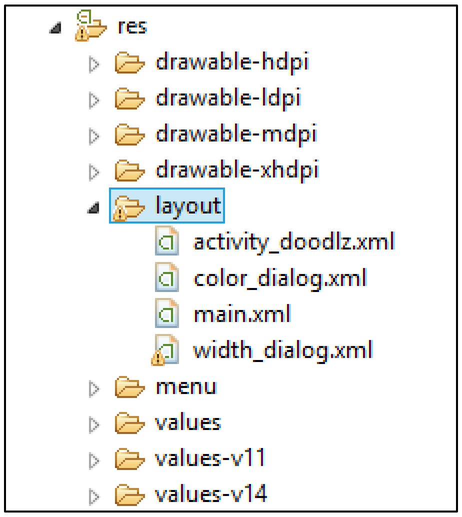
1. We need to set the String Resources for the App. Go to res/values and select the strings.xml file.
2. Delete all text within the strings.xml file and then type the following to set the Strings used for in User Displays and Messages:



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

    <string name="app_name">Doodlz</string>
    <string name="button_erase">Erase</string>
    <string name="button_cancel">Cancel</string>
    <string name="button_set_color">Set Color</string>
    <string name="button_set_line_width">Set Line Width</string>
    <string name="label_alpha">Alpha</string>
    <string name="label_red">Red</string>
    <string name="label_green">Green</string>
    <string name="label_blue">Blue</string>
    <string name="menuitem_clear">Clear</string>
    <string name="menuitem_color">Color</string>
    <string name="menuitem_erase">Erase</string>
    <string name="menuitem_line_width">Line Width</string>
    <string name="menuitem_save_image">Save Image</string>
    <string name="message_erase">Erase the drawing?</string>
    <string name="message_error_saving">There was an error saving the image</string>
    <string name="message_saved">Your painting has been saved to the Gallery</string>
    <string name="title_color_diaglog">Choose Color</string>
    <string name="title_line_width_dialog">Choose Line Width</string>
    <string name="menu_settings">Menu Settings</string>
</resources>
```

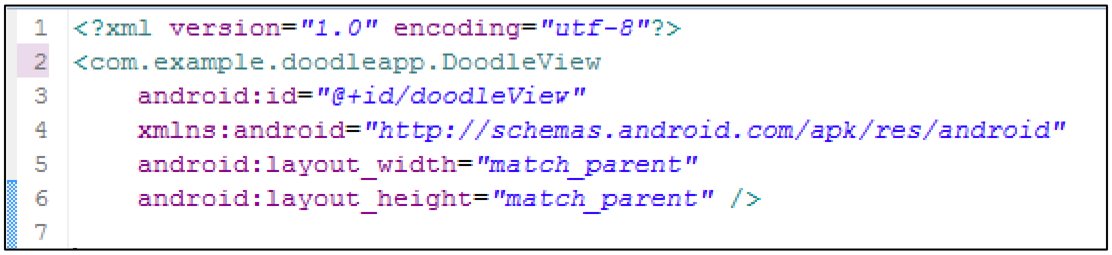
3. Go to res/layout and right click.
4. Select New -> Other -> Android XML File
5. Name the file main.xml

6. Create two additional xml files:
    a. color_dialog.xml
    b. width_dialog.xml



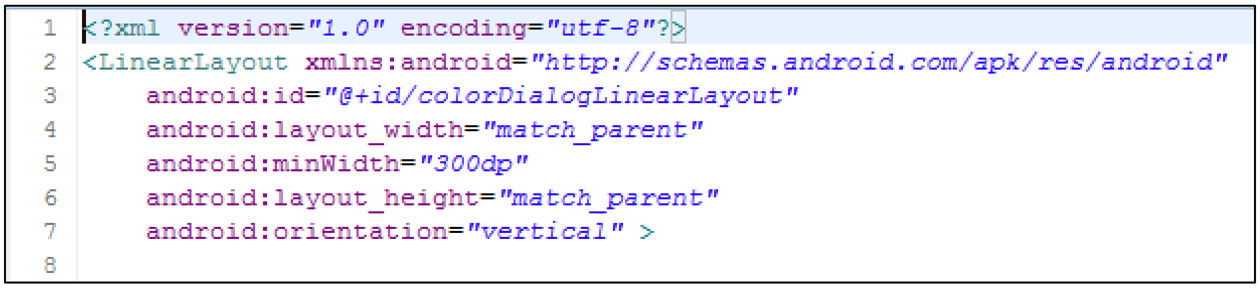
```
res
    drawable-hdpi
    drawable-ldpi
    drawable-mdpi
    drawable-xhdpi
    layout
        activity_doodlz.xml
        color_dialog.xml
        main.xml
        width_dialog.xml
    menu
    values
    values-v11
    values-v14
```

7. Select the main.xml file.
8. Erase any code in the main.xml file and type the following Code to reference the custom DoodleView object:



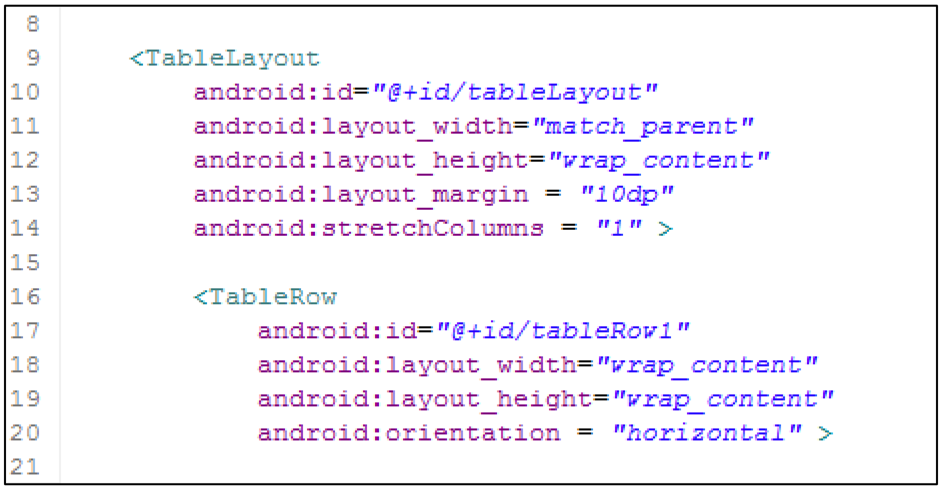
```
<?xml version="1.0" encoding="utf-8"?>
<com.example.doodleapp.DoodleView
    android:id="@+id/doodleView"
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent" />
```

9. Select the color_dialog.xml file. Erase any code in that file.
10. Type the following to set up the Linear Layout:



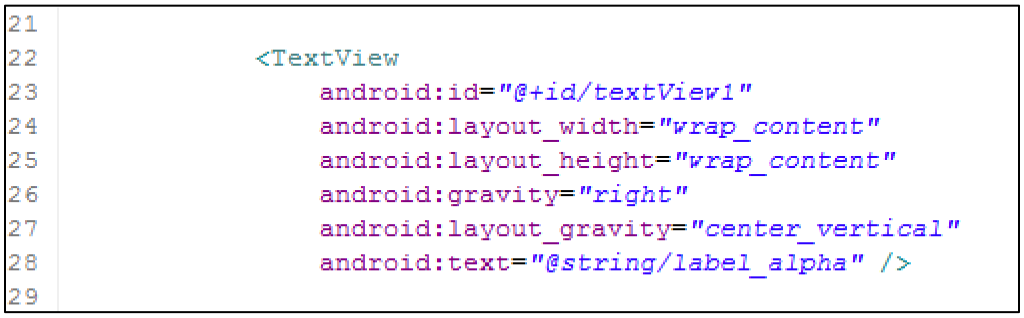
```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/colorDialogLinearLayout"
    android:layout_width="match_parent"
    android:minWidth="300dp"
    android:layout_height="match_parent"
    android:orientation="vertical" >
```

11. We now need to set up a Table Layout. Type the following to start the table and add the first row:



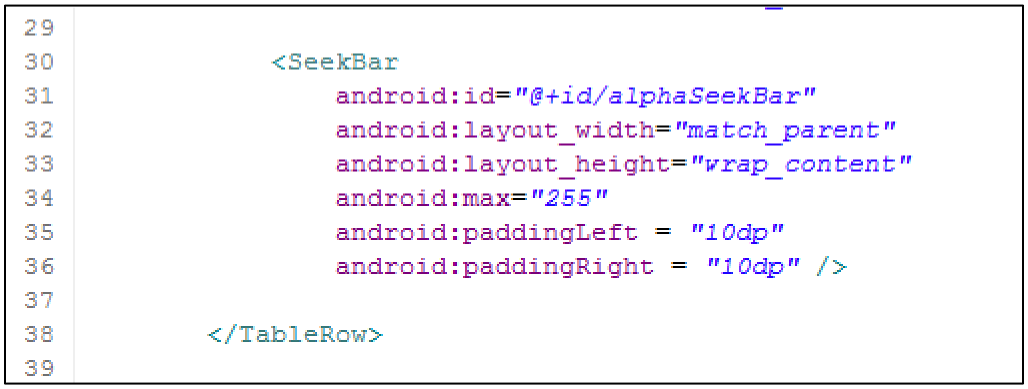
```
    <TableLayout
        android:id="@+id/tableLayout"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_margin = "10dp"
        android:stretchColumns = "1" >

        <TableRow
            android:id="@+id/tableRow1"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:orientation = "horizontal" >

```

12. Type the following to add a TextView object to the TableRow:



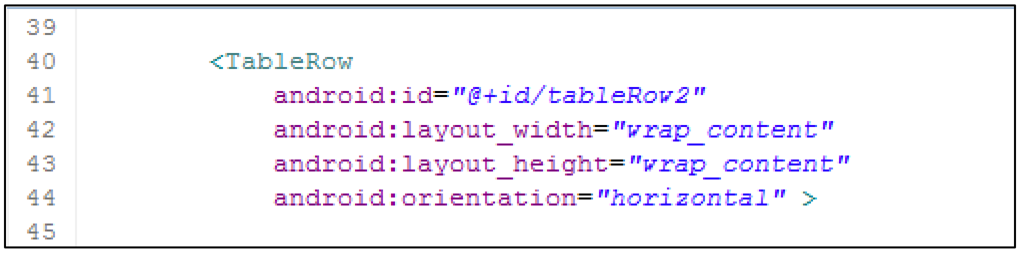
```

            <TextView
                android:id="@+id/textView1"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:gravity="right"
                android:layout_gravity="center_vertical"
                android:text="@string/label_alpha" />

```

13. Type the following to add a SeekBar Object and close the Table Row



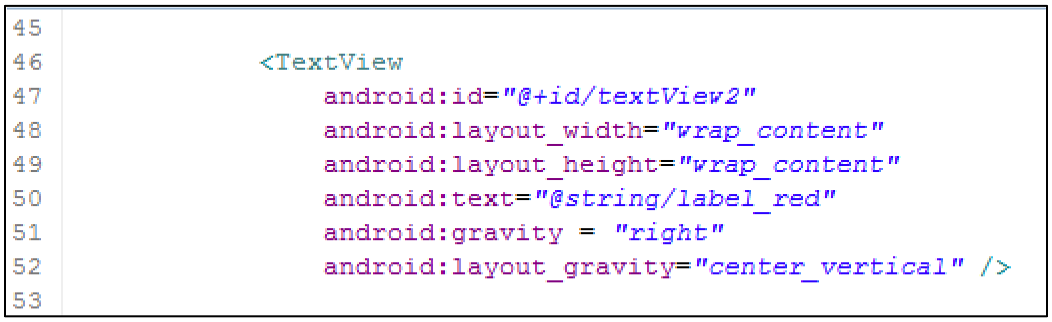
```

            <SeekBar
                android:id="@+id/alphaSeekBar"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:max="255"
                android:paddingLeft = "10dp"
                android:paddingRight = "10dp" />

        </TableRow>

```

14. Now begin a 2nd Table Row:



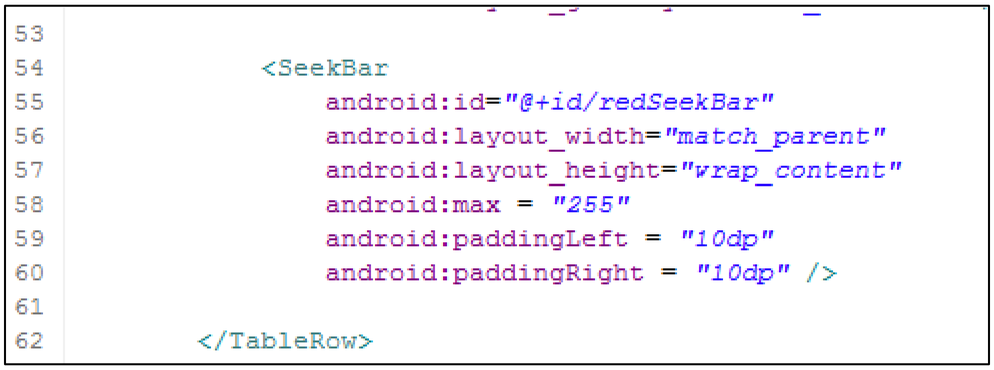
```
        <TableRow
            android:id="@+id/tableRow2"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:orientation="horizontal" >
```

15. Add a TextView Object to that row for the Red color component:



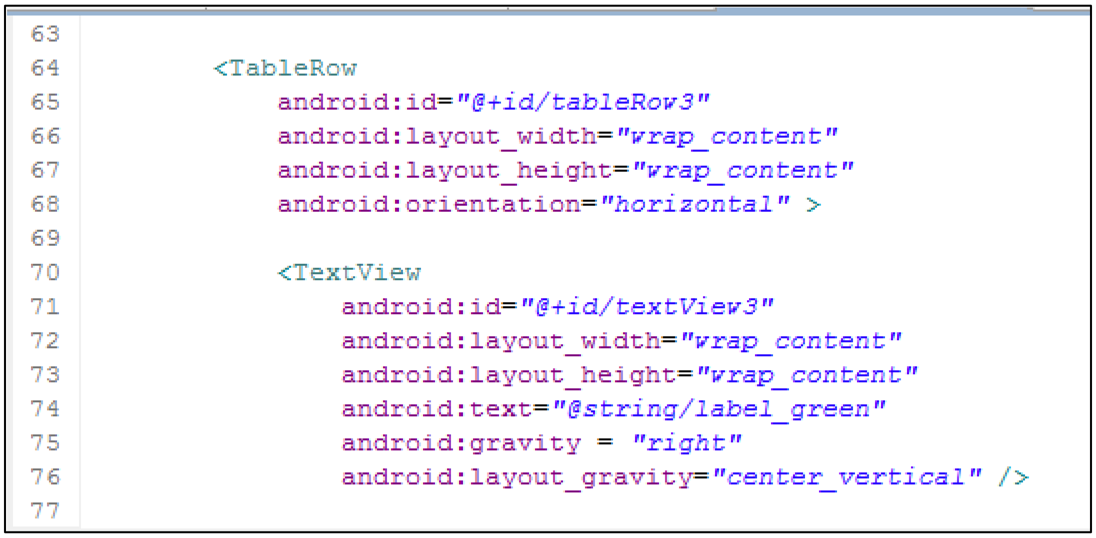
```
            <TextView
                android:id="@+id/textView2"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="@string/label_red"
                android:gravity = "right"
                android:layout_gravity="center_vertical" />
```

16. Add a SeekBar object for color input:



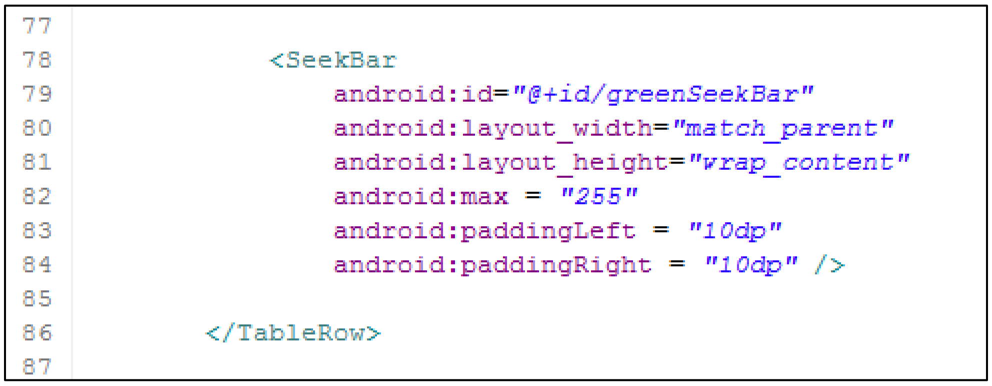
```
            <SeekBar
                android:id="@+id/redSeekBar"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:max = "255"
                android:paddingLeft = "10dp"
                android:paddingRight = "10dp" />

        </TableRow>
```

17. Start the third TableRow with its TextView Object:



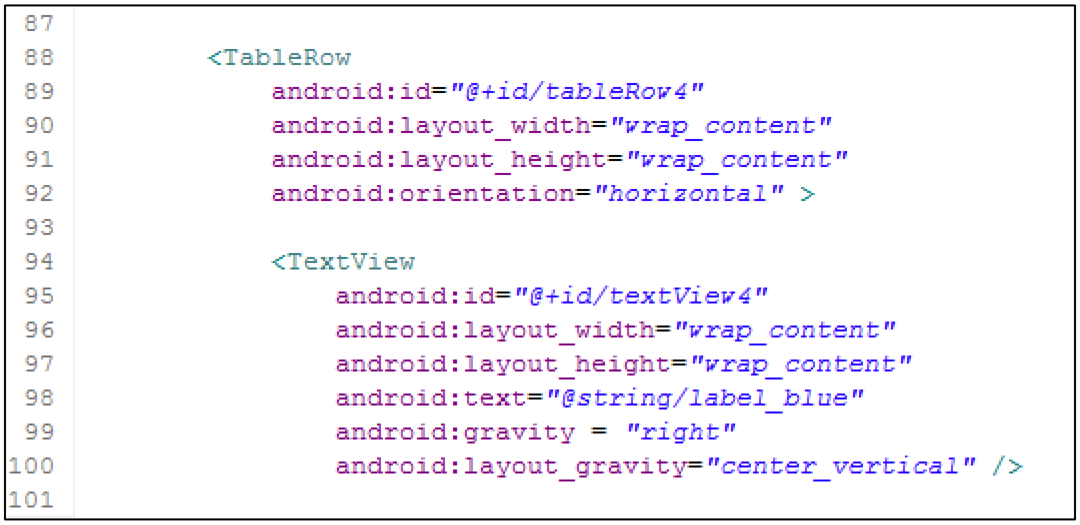
```xml

        <TableRow
            android:id="@+id/tableRow3"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:orientation="horizontal" >

            <TextView
                android:id="@+id/textView3"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="@string/label_green"
                android:gravity = "right"
                android:layout_gravity="center_vertical" />

```

18. Add the SeekBar for the Third Row:



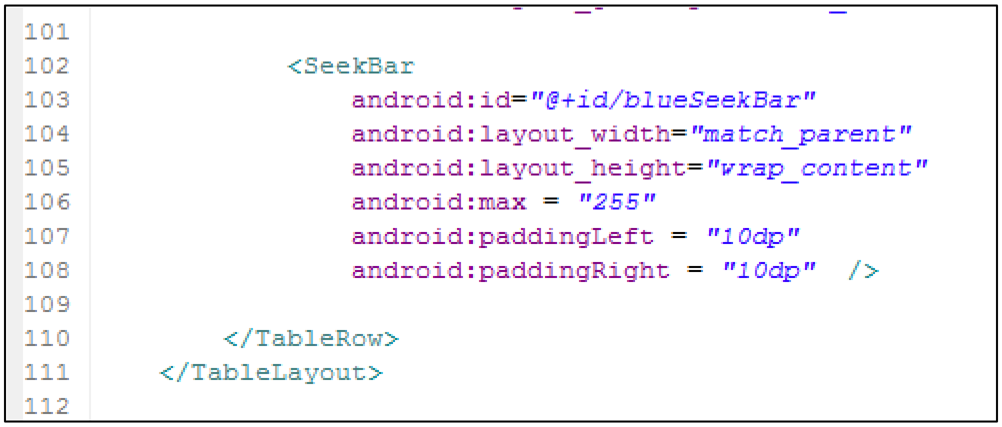
```xml

            <SeekBar
                android:id="@+id/greenSeekBar"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:max = "255"
                android:paddingLeft = "10dp"
                android:paddingRight = "10dp" />

        </TableRow>

```

19. Add a fourth TableRow and Text View Object for the Blue color component:



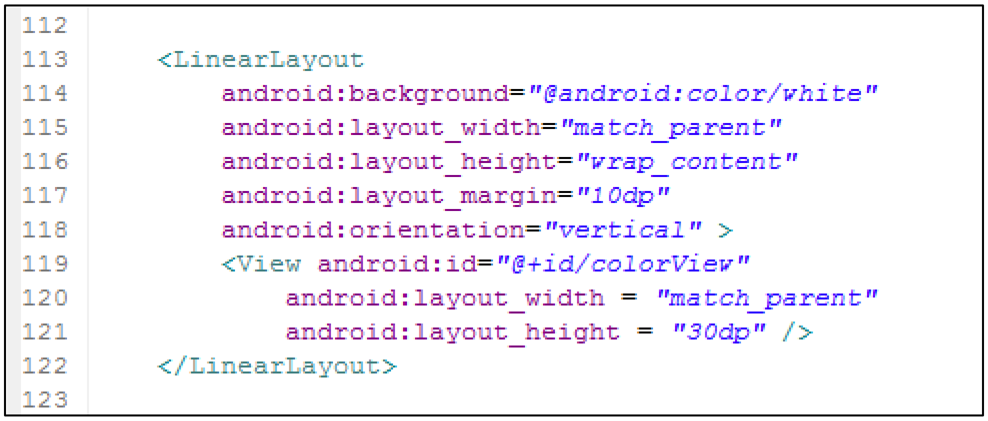
```

        <TableRow
            android:id="@+id/tableRow4"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:orientation="horizontal" >

            <TextView
                android:id="@+id/textView4"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="@string/label_blue"
                android:gravity = "right"
                android:layout_gravity="center_vertical" />

```

20. Add the SeekBar Object and Close the Row and the Table:



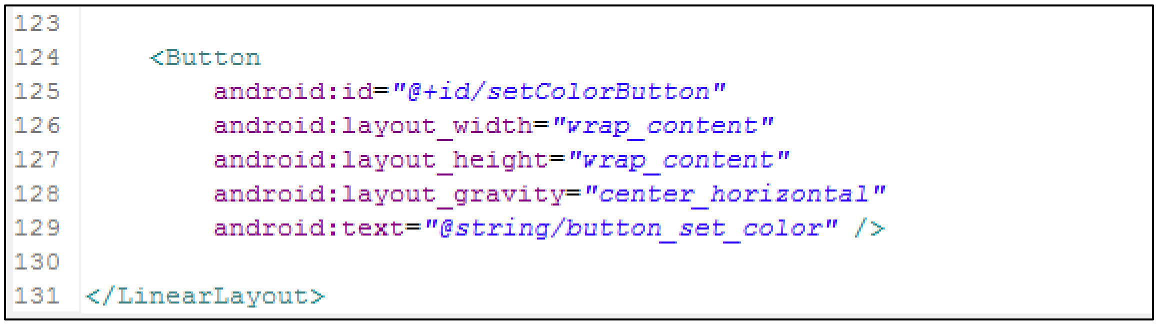
```

            <SeekBar
                android:id="@+id/blueSeekBar"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:max = "255"
                android:paddingLeft = "10dp"
                android:paddingRight = "10dp"  />

        </TableRow>
    </TableLayout>

```

21. Add a LinearLayout with an embedded View object:



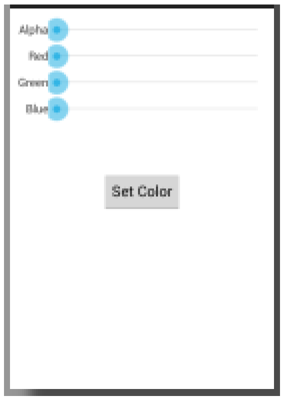
```

    <LinearLayout
        android:background="@android:color/white"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_margin="10dp"
        android:orientation="vertical" >
        <View android:id="@+id/colorView"
            android:layout_width = "match_parent"
            android:layout_height = "30dp" />
    </LinearLayout>

```

22. Add a Button Object and finish the LinearLayout:

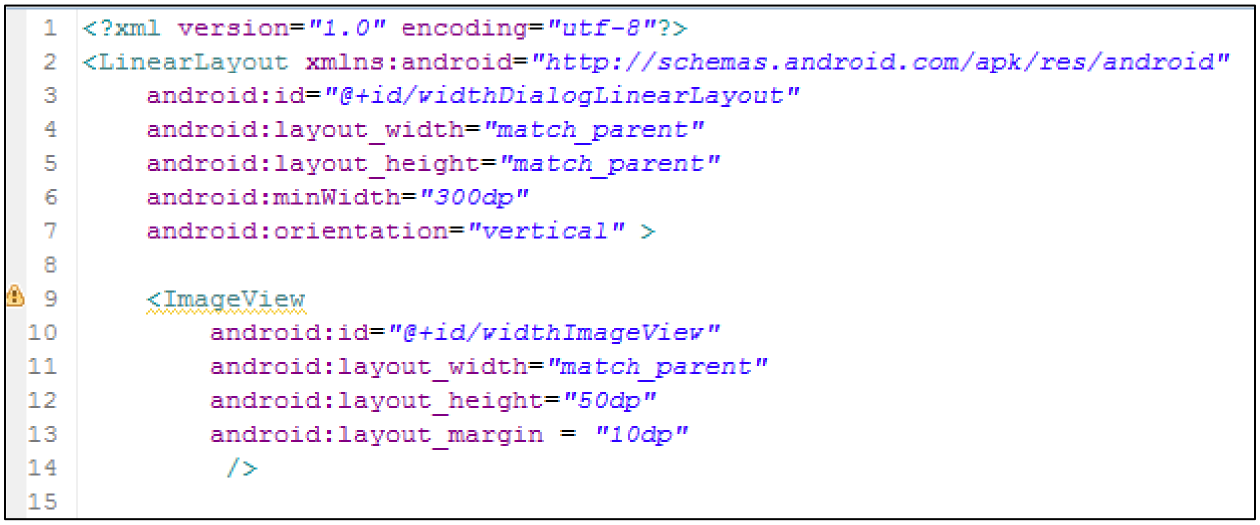


```

    <Button
        android:id="@+id/setColorButton"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center_horizontal"
        android:text="@string/button_set_color" />

</LinearLayout>
```

23. The finished Layout should look like this:

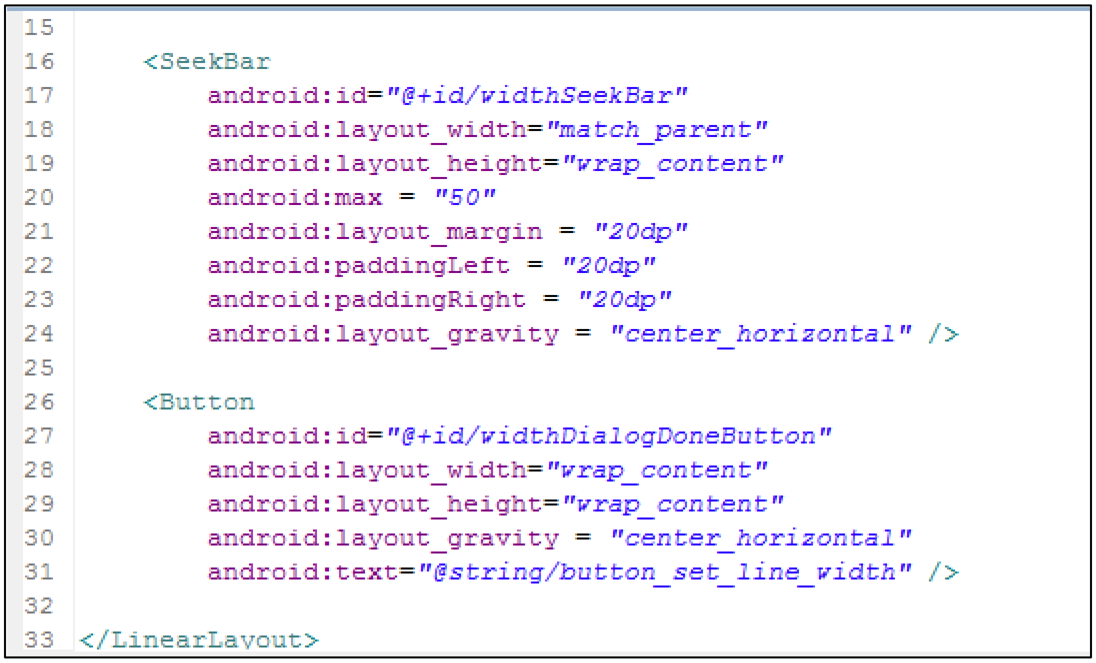


24. Select the width_dialog.xml file.
25. Delete any text and start the LinearLayout and ImageView Object xml code:



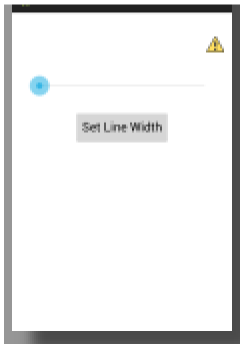
```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/widthDialogLinearLayout"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:minWidth="300dp"
    android:orientation="vertical" >

    <ImageView
        android:id="@+id/widthImageView"
        android:layout_width="match_parent"
        android:layout_height="50dp"
        android:layout_margin = "10dp"
         />

```

26. Finish the Code with the SeekBar and Button Objects:



```

    <SeekBar
        android:id="@+id/widthSeekBar"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:max = "50"
        android:layout_margin = "20dp"
        android:paddingLeft = "20dp"
        android:paddingRight = "20dp"
        android:layout_gravity = "center_horizontal" />

    <Button
        android:id="@+id/widthDialogDoneButton"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity = "center_horizontal"
        android:text="@string/button_set_line_width" />

</LinearLayout>
```

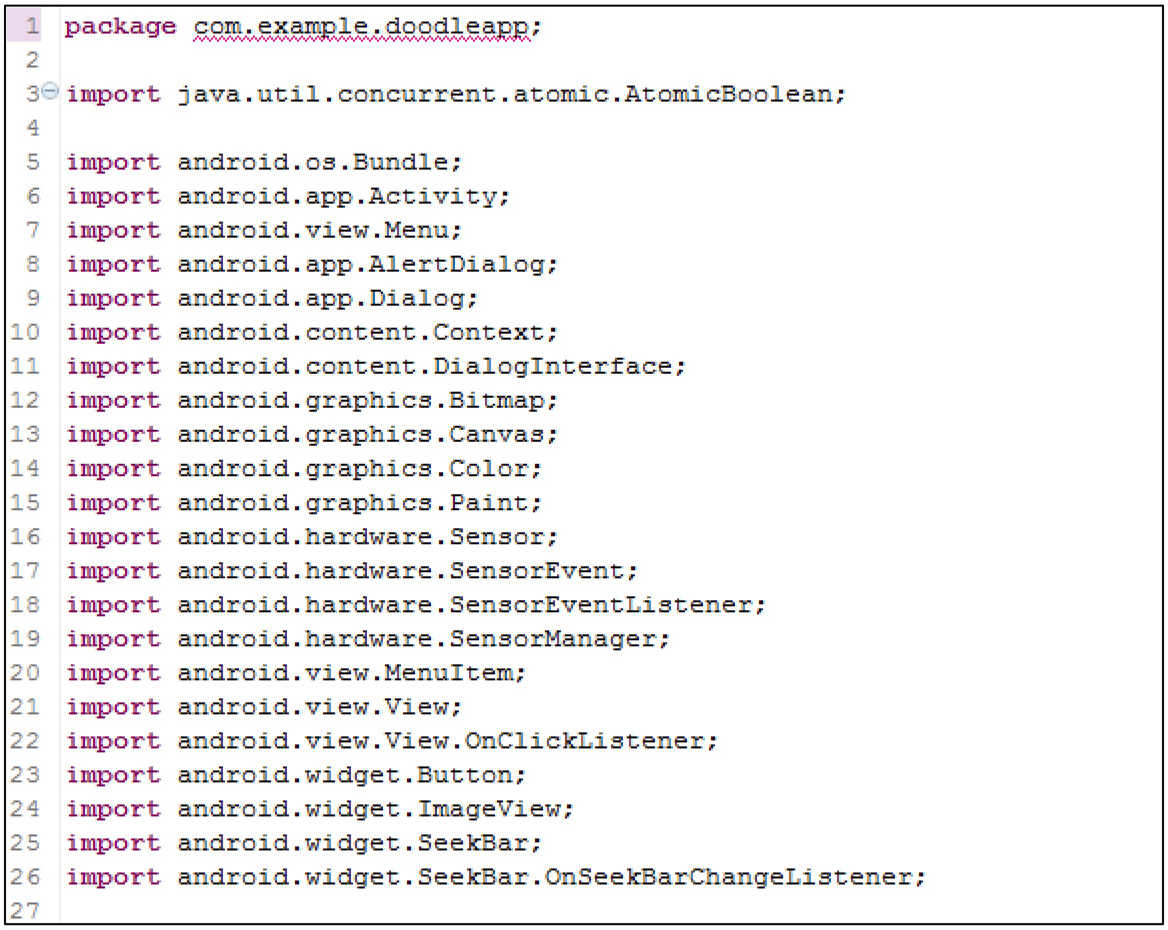
27. The Finished width_dialog.xml should look like this:



Phase 3: Writing the Code for Doodle.java

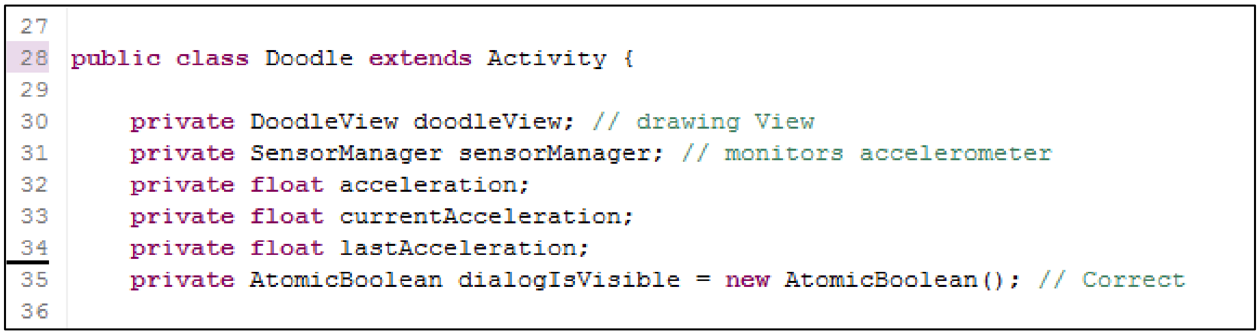
Process:

1. Go to the src/Doodle.java file and remove the existing code. We need to start the file with the package name and then the import statements. Write the following code from Lines 1 to 27



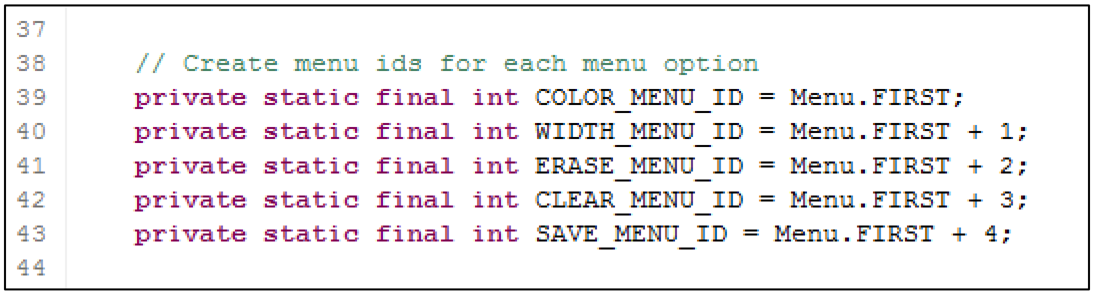
```
 1  package com.example.doodleapp;
 2
 3  import java.util.concurrent.atomic.AtomicBoolean;
 4
 5  import android.os.Bundle;
 6  import android.app.Activity;
 7  import android.view.Menu;
 8  import android.app.AlertDialog;
 9  import android.app.Dialog;
10  import android.content.Context;
11  import android.content.DialogInterface;
12  import android.graphics.Bitmap;
13  import android.graphics.Canvas;
14  import android.graphics.Color;
15  import android.graphics.Paint;
16  import android.hardware.Sensor;
17  import android.hardware.SensorEvent;
18  import android.hardware.SensorEventListener;
19  import android.hardware.SensorManager;
20  import android.view.MenuItem;
21  import android.view.View;
22  import android.view.View.OnClickListener;
23  import android.widget.Button;
24  import android.widget.ImageView;
25  import android.widget.SeekBar;
26  import android.widget.SeekBar.OnSeekBarChangeListener;
27
```

2. Begin to define the fields for the Doodle class.



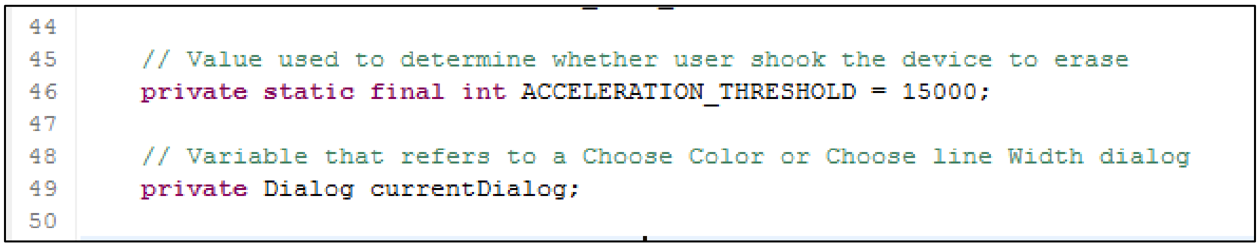
```

public class Doodle extends Activity {

    private DoodleView doodleView; // drawing View
    private SensorManager sensorManager; // monitors accelerometer
    private float acceleration;
    private float currentAcceleration;
    private float lastAcceleration;
    private AtomicBoolean dialogIsVisible = new AtomicBoolean(); // Correct

```

3. Write the constant id values for menu construction



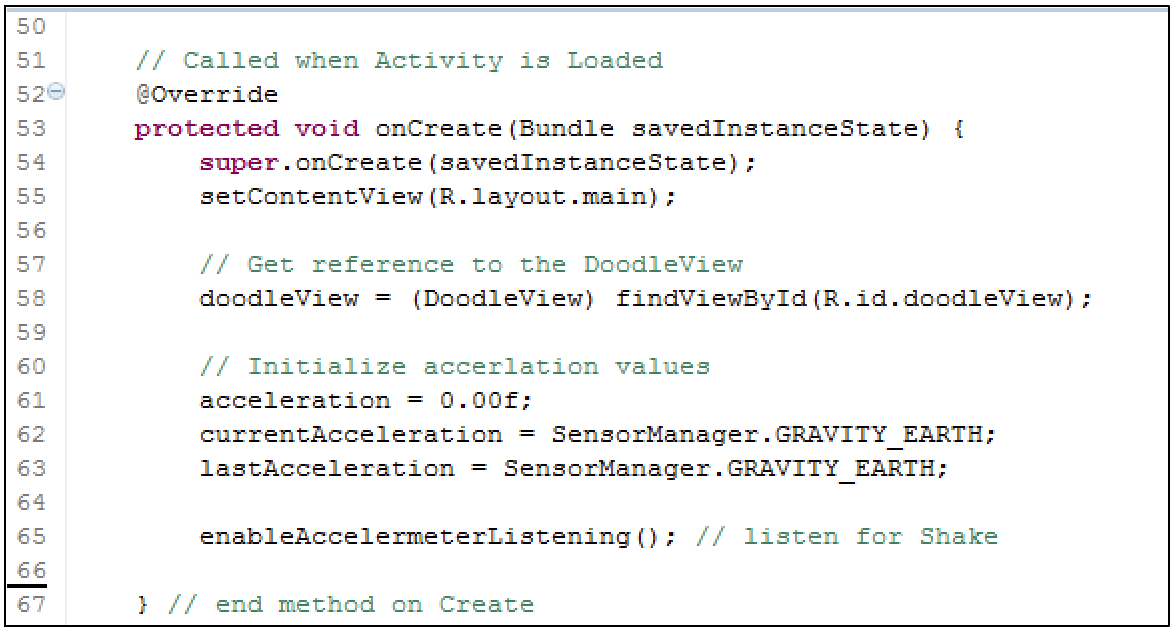
```

    // Create menu ids for each menu option
    private static final int COLOR_MENU_ID = Menu.FIRST;
    private static final int WIDTH_MENU_ID = Menu.FIRST + 1;
    private static final int ERASE_MENU_ID = Menu.FIRST + 2;
    private static final int CLEAR_MENU_ID = Menu.FIRST + 3;
    private static final int SAVE_MENU_ID = Menu.FIRST + 4;

```

4. Finish the field declaration with a constant for acceleration threshold and a Dialog object



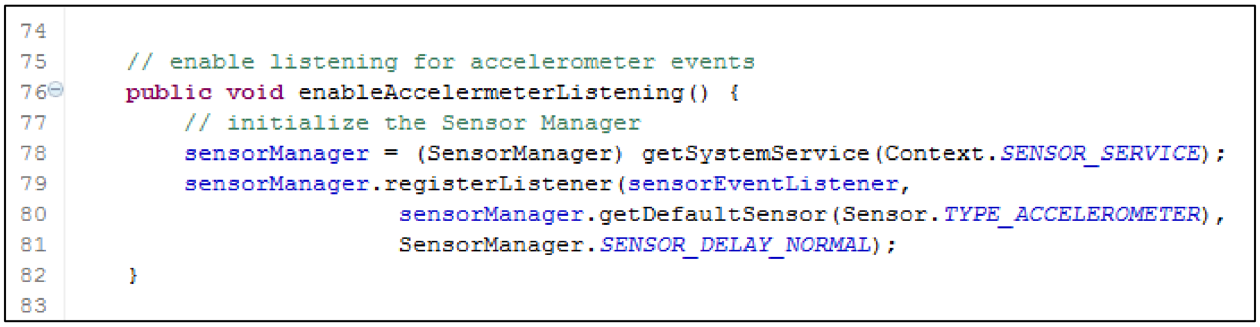
```

    // Value used to determine whether user shook the device to erase
    private static final int ACCELERATION_THRESHOLD = 15000;

    // Variable that refers to a Choose Color or Choose line Width dialog
    private Dialog currentDialog;

```

5. Write the onCreate() method. This acts similar to the constructor in an Android activity.



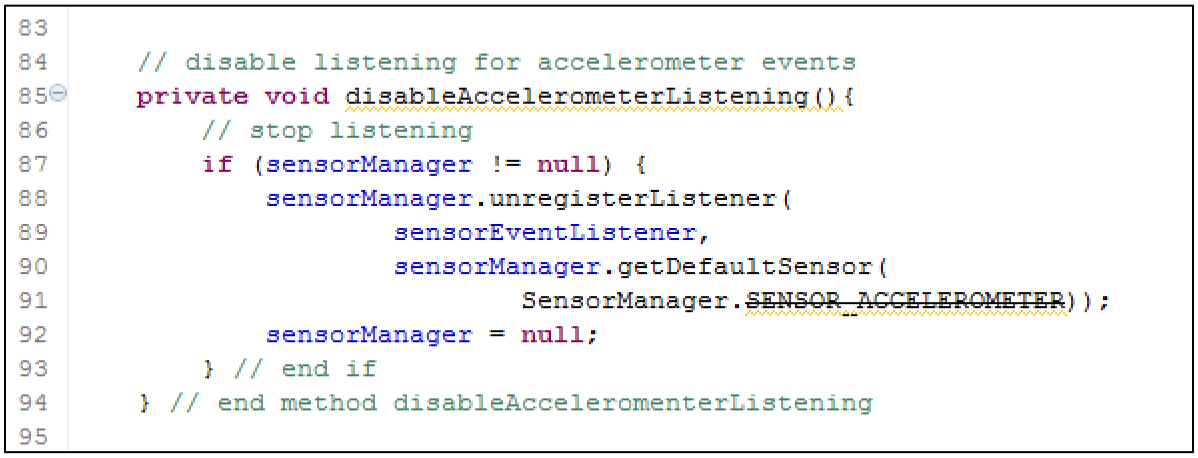
```

    // Called when Activity is Loaded
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

        // Get reference to the DoodleView
        doodleView = (DoodleView) findViewById(R.id.doodleView);

        // Initialize accerlation values
        acceleration = 0.00f;
        currentAcceleration = SensorManager.GRAVITY_EARTH;
        lastAcceleration = SensorManager.GRAVITY_EARTH;

        enableAccelermeterListening(); // listen for Shake

    } // end method on Create
```

6. Write the method to enable the accelerometer listener. This allows the program to 'listen' to the sensors in the device to read values.



```

    // enable listening for accelerometer events
    public void enableAccelermeterListening() {
        // initialize the Sensor Manager
        sensorManager = (SensorManager) getSystemService(Context.SENSOR_SERVICE);
        sensorManager.registerListener(sensorEventListener,
                      sensorManager.getDefaultSensor(Sensor.TYPE_ACCELEROMETER),
                      SensorManager.SENSOR_DELAY_NORMAL);
    }

```

7. This method unregisters the accelerometer when the Activity is terminated.

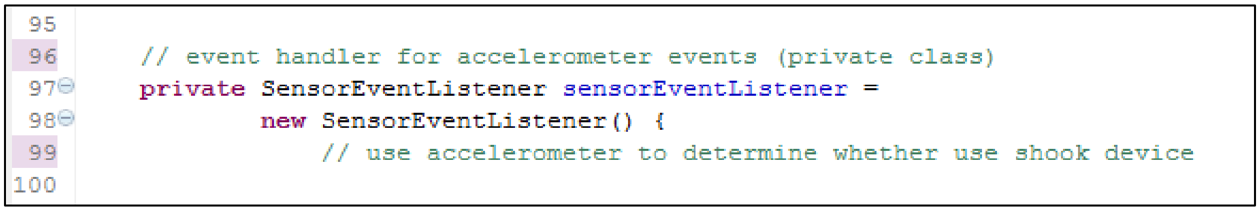


```

    // disable listening for accelerometer events
    private void disableAccelerometerListening(){
        // stop listening
        if (sensorManager != null) {
            sensorManager.unregisterListener(
                    sensorEventListener,
                    sensorManager.getDefaultSensor(
                            SensorManager.~~SENSOR_ACCELEROMETER~~));
            sensorManager = null;
        } // end if
    } // end method disableAcceleromenterListening

```

8. We will now write a private inner class of the type SensorEventListener. This listener will take the phone's event (in this case, shaking) and return values corresponding to the x, y, and z acceleration. The SensorEvent object has a public array called 'values'. The values array holds the following data:
   a. values[0] -> The X direction acceleration.
   b. values[1] -> The Y direction acceleration.
   c. values[2] -> The Z direction acceleration.

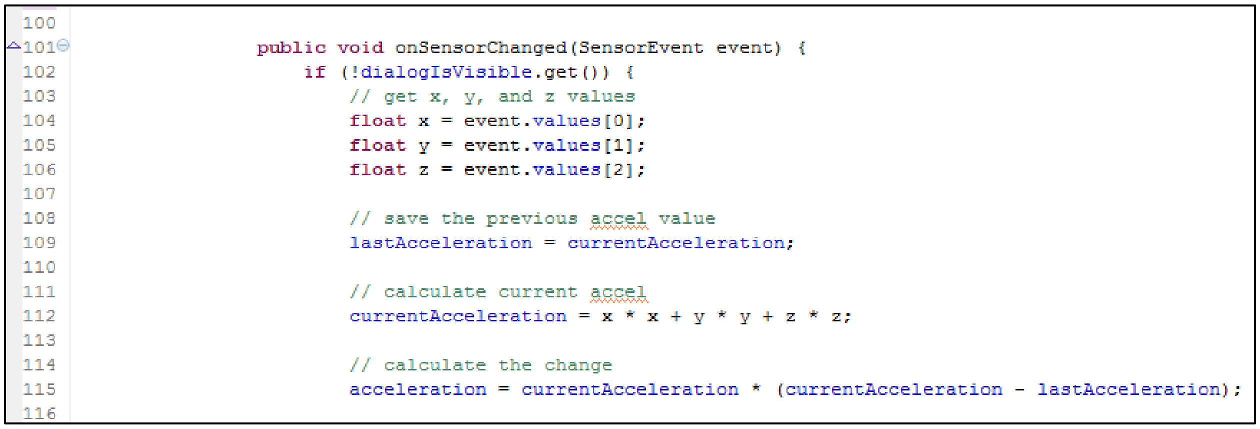
   We will pull this data from the shaking event to calculate the total acceleration. If the acceleration is greater than the threshold, we will call the method to erase the picture.

   Type the following to start the class:



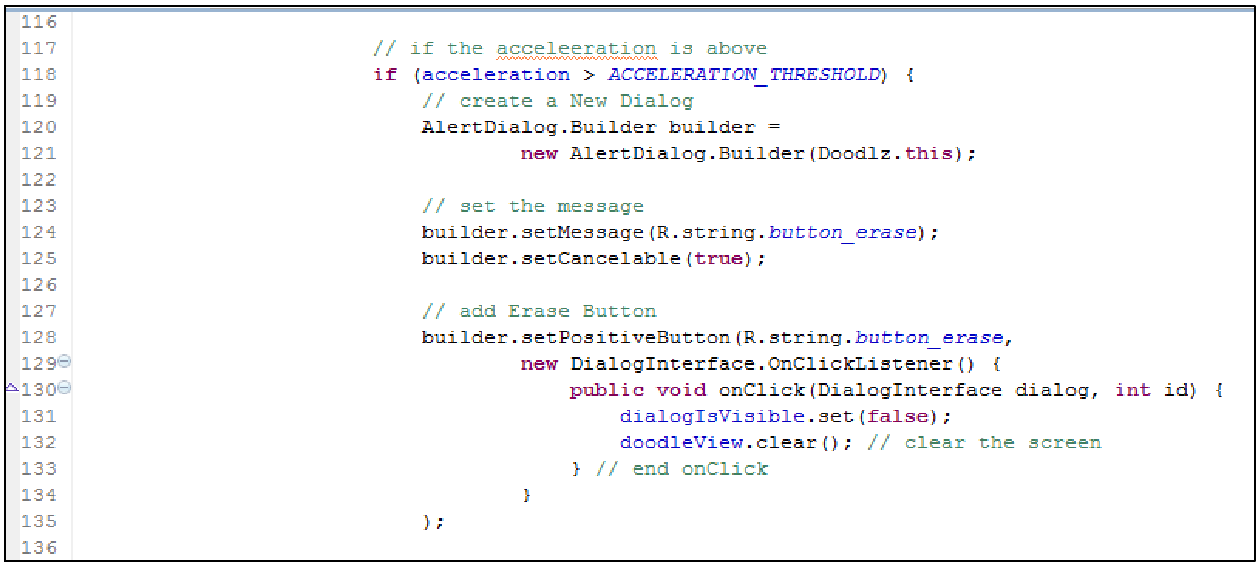
```

    // event handler for accelerometer events (private class)
    private SensorEventListener sensorEventListener =
            new SensorEventListener() {
                // use accelerometer to determine whether use shook device

```

9. Start the onSensorChanged method to begin calculating the acceleration.



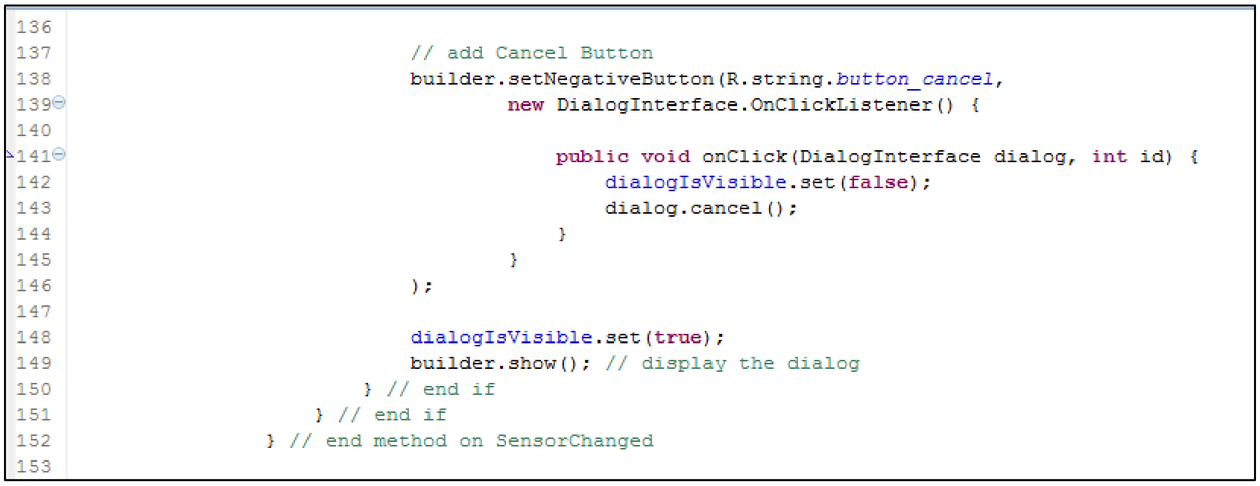
```
                public void onSensorChanged(SensorEvent event) {
                    if (!dialogIsVisible.get()) {
                        // get x, y, and z values
                        float x = event.values[0];
                        float y = event.values[1];
                        float z = event.values[2];

                        // save the previous accel value
                        lastAcceleration = currentAcceleration;

                        // calculate current accel
                        currentAcceleration = x * x + y * y + z * z;

                        // calculate the change
                        acceleration = currentAcceleration * (currentAcceleration - lastAcceleration);

```

10. Add the conditional statement covering of the phone is shaking:



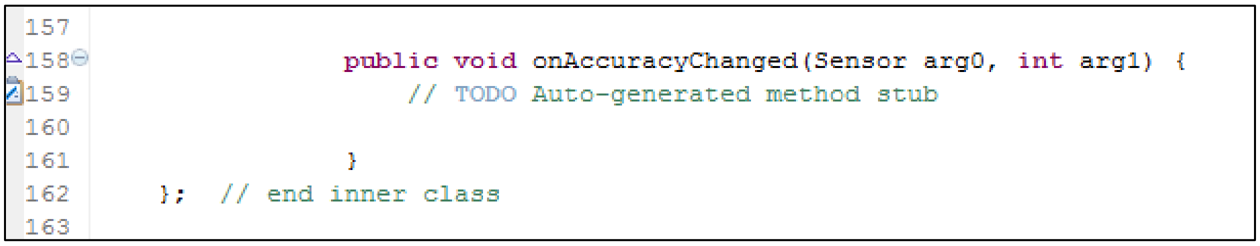
```

                        // if the acceleeration is above
                        if (acceleration > ACCELERATION_THRESHOLD) {
                            // create a New Dialog
                            AlertDialog.Builder builder =
                                    new AlertDialog.Builder(Doodlz.this);

                            // set the message
                            builder.setMessage(R.string.button_erase);
                            builder.setCancelable(true);

                            // add Erase Button
                            builder.setPositiveButton(R.string.button_erase,
                                    new DialogInterface.OnClickListener() {
                                        public void onClick(DialogInterface dialog, int id) {
                                            dialogIsVisible.set(false);
                                            doodleView.clear(); // clear the screen
                                        } // end onClick
                                    }
                            );

```

11. Finish the onSensorChanged method:



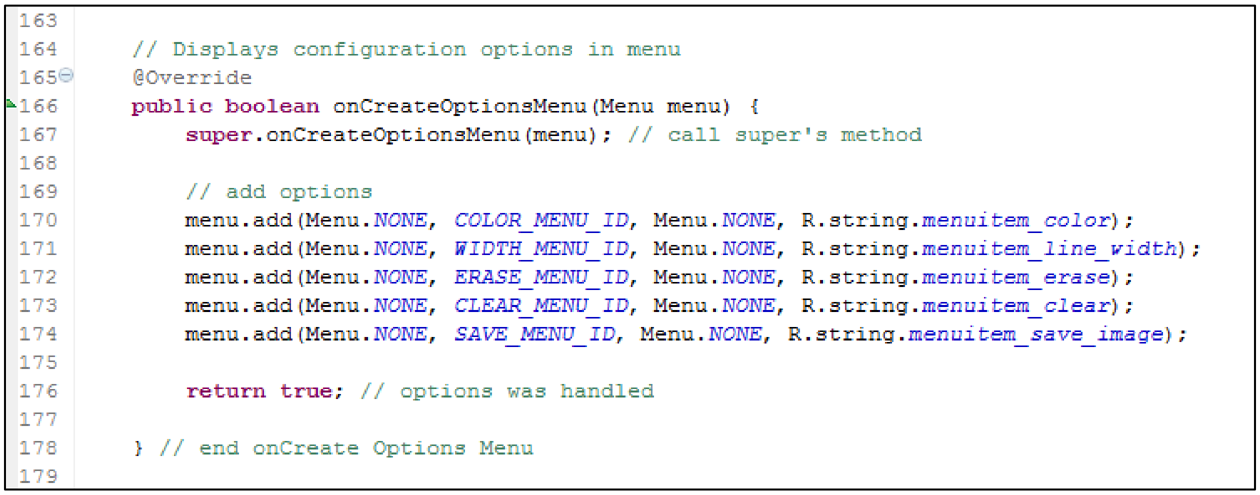
```
            // add Cancel Button
            builder.setNegativeButton(R.string.button_cancel,
                    new DialogInterface.OnClickListener() {

                        public void onClick(DialogInterface dialog, int id) {
                            dialogIsVisible.set(false);
                            dialog.cancel();
                        }
                    }
            );

            dialogIsVisible.set(true);
            builder.show(); // display the dialog
        } // end if
    } // end if
} // end method on SensorChanged
```

12. Finish the inner class with a required onAccuracyChanged method (we leave this empty) and closing the inner class:



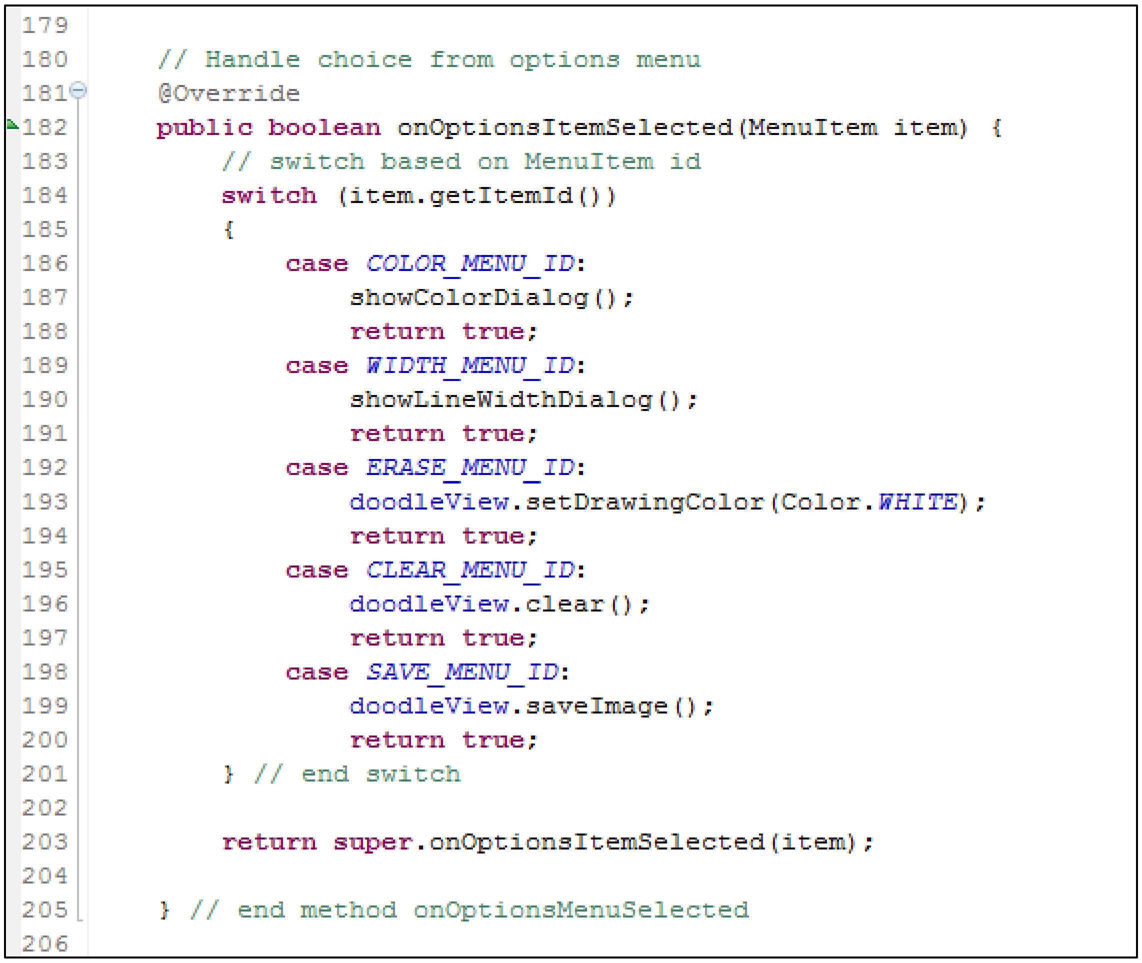
```
            public void onAccuracyChanged(Sensor arg0, int arg1) {
                // TODO Auto-generated method stub

            }
};   // end inner class
```

13. Write the method onCreateOptionsMenu that assigns the menu strings to the menu object



```
// Displays configuration options in menu
@Override
public boolean onCreateOptionsMenu(Menu menu) {
    super.onCreateOptionsMenu(menu); // call super's method

    // add options
    menu.add(Menu.NONE, COLOR_MENU_ID, Menu.NONE, R.string.menuitem_color);
    menu.add(Menu.NONE, WIDTH_MENU_ID, Menu.NONE, R.string.menuitem_line_width);
    menu.add(Menu.NONE, ERASE_MENU_ID, Menu.NONE, R.string.menuitem_erase);
    menu.add(Menu.NONE, CLEAR_MENU_ID, Menu.NONE, R.string.menuitem_clear);
    menu.add(Menu.NONE, SAVE_MENU_ID, Menu.NONE, R.string.menuitem_save_image);

    return true; // options was handled

} // end onCreate Options Menu
```

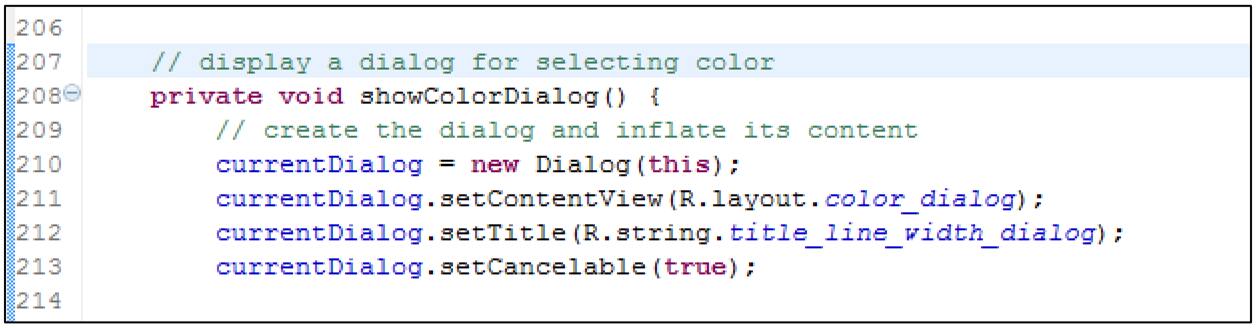
14. We will use a switch / case structure to handle the options from the Options Menu. Each case will call a different method we will define within the Doodle or the doodleView objects.



```

    // Handle choice from options menu
    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        // switch based on MenuItem id
        switch (item.getItemId())
        {
            case COLOR_MENU_ID:
                showColorDialog();
                return true;
            case WIDTH_MENU_ID:
                showLineWidthDialog();
                return true;
            case ERASE_MENU_ID:
                doodleView.setDrawingColor(Color.WHITE);
                return true;
            case CLEAR_MENU_ID:
                doodleView.clear();
                return true;
            case SAVE_MENU_ID:
                doodleView.saveImage();
                return true;
        } // end switch

        return super.onOptionsItemSelected(item);

    } // end method onOptionsMenuSelected

```

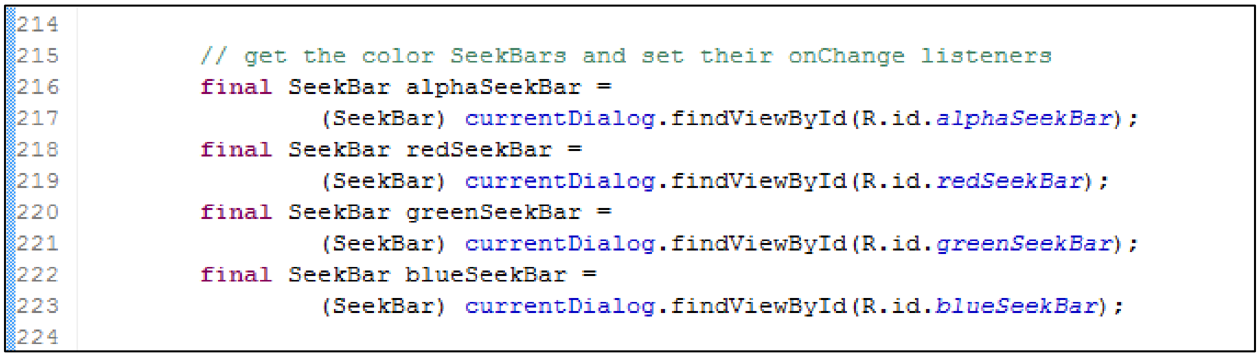
15. The showColorDialog() method builds the interface to allow the user to select the color of the drawing line with 4 slider values:
    a. Alpha -> The transparency of the pixels
    b. Red -> The intensity of red in the pixel
    c. Green -> The intensity of green in the pixel
    d. Blue -> The intensity of blue in the pixel

    Start the showColorDialog() method by defining and connecting the currentDialog instance to the layout color_dialog.xml file



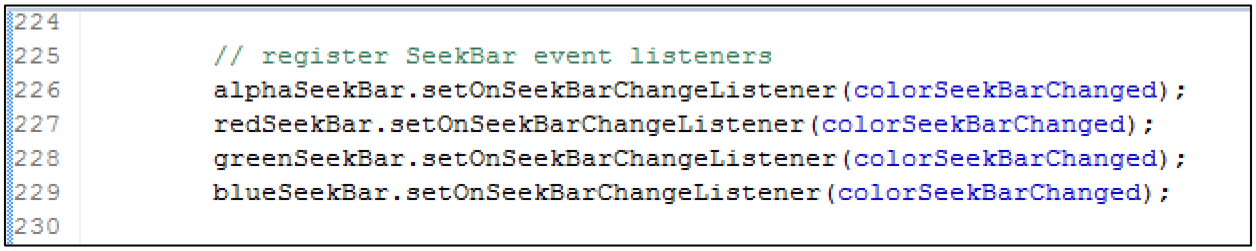
```

    // display a dialog for selecting color
    private void showColorDialog() {
        // create the dialog and inflate its content
        currentDialog = new Dialog(this);
        currentDialog.setContentView(R.layout.color_dialog);
        currentDialog.setTitle(R.string.title_line_width_dialog);
        currentDialog.setCancelable(true);

```

16. Connect the seekBar objects to their respective xml representations



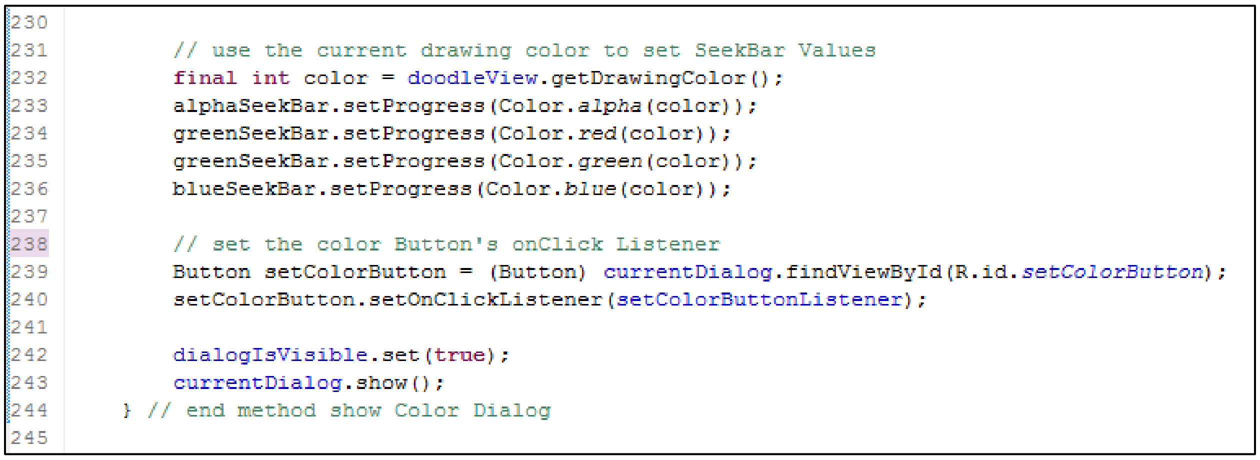
```
        // get the color SeekBars and set their onChange listeners
        final SeekBar alphaSeekBar =
                (SeekBar) currentDialog.findViewById(R.id.alphaSeekBar);
        final SeekBar redSeekBar =
                (SeekBar) currentDialog.findViewById(R.id.redSeekBar);
        final SeekBar greenSeekBar =
                (SeekBar) currentDialog.findViewById(R.id.greenSeekBar);
        final SeekBar blueSeekBar =
                (SeekBar) currentDialog.findViewById(R.id.blueSeekBar);
```

17. Register the seekBar event listeners



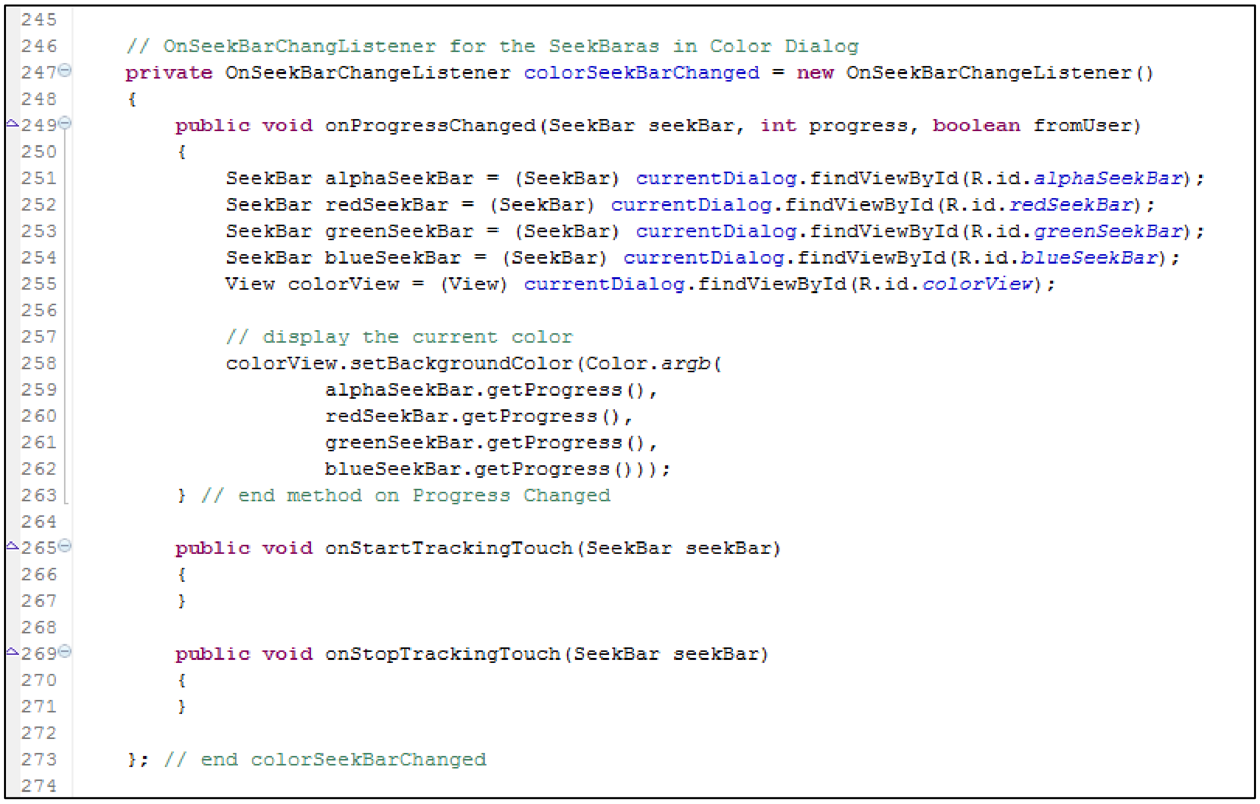
```
        // register SeekBar event listeners
        alphaSeekBar.setOnSeekBarChangeListener(colorSeekBarChanged);
        redSeekBar.setOnSeekBarChangeListener(colorSeekBarChanged);
        greenSeekBar.setOnSeekBarChangeListener(colorSeekBarChanged);
        blueSeekBar.setOnSeekBarChangeListener(colorSeekBarChanged);
```

18. Fetch the current color and set the progress of the seek bars to the corresponding value. We will also set the Button event listener and close the color dialog method



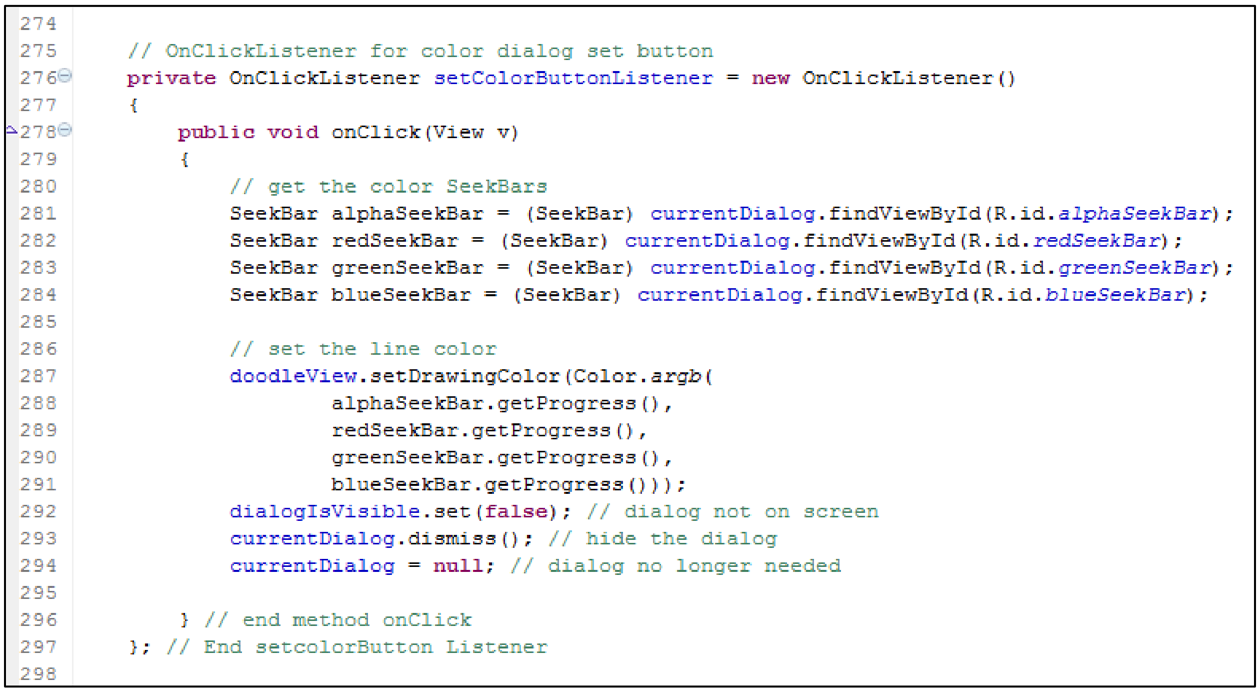
```
        // use the current drawing color to set SeekBar Values
        final int color = doodleView.getDrawingColor();
        alphaSeekBar.setProgress(Color.alpha(color));
        greenSeekBar.setProgress(Color.red(color));
        greenSeekBar.setProgress(Color.green(color));
        blueSeekBar.setProgress(Color.blue(color));

        // set the color Button's onClick Listener
        Button setColorButton = (Button) currentDialog.findViewById(R.id.setColorButton);
        setColorButton.setOnClickListener(setColorButtonListener);

        dialogIsVisible.set(true);
        currentDialog.show();
    } // end method show Color Dialog
```

19. Now build the inner class OnSeekBarChangeListener to react to changes in the seekBars.



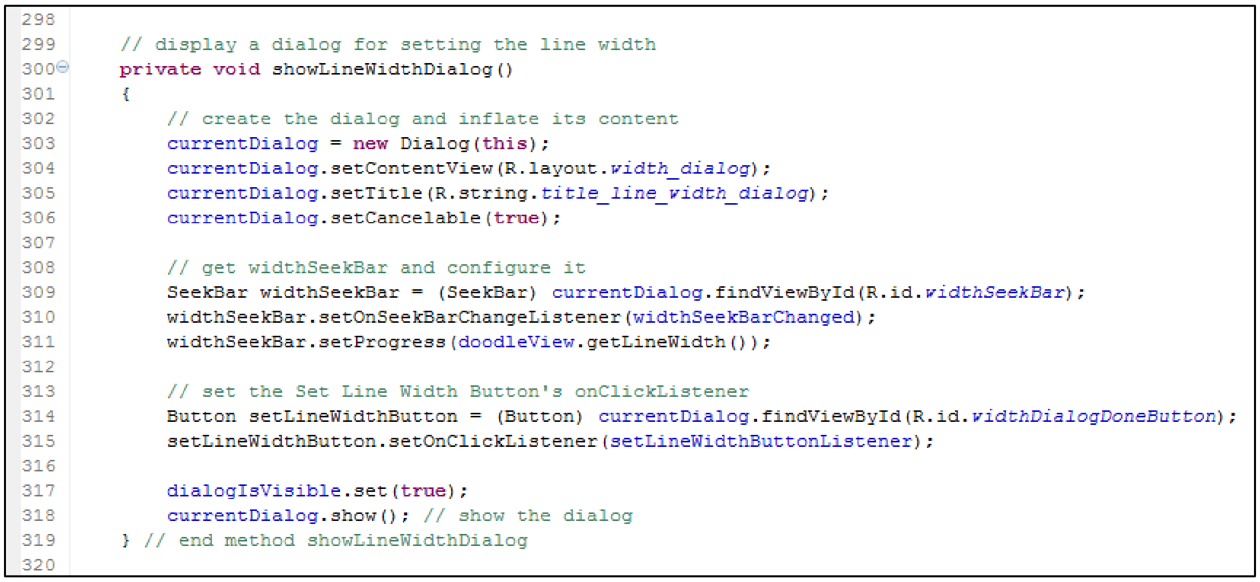
```java

    // OnSeekBarChangListener for the SeekBaras in Color Dialog
    private OnSeekBarChangeListener colorSeekBarChanged = new OnSeekBarChangeListener()
    {
        public void onProgressChanged(SeekBar seekBar, int progress, boolean fromUser)
        {
            SeekBar alphaSeekBar = (SeekBar) currentDialog.findViewById(R.id.alphaSeekBar);
            SeekBar redSeekBar = (SeekBar) currentDialog.findViewById(R.id.redSeekBar);
            SeekBar greenSeekBar = (SeekBar) currentDialog.findViewById(R.id.greenSeekBar);
            SeekBar blueSeekBar = (SeekBar) currentDialog.findViewById(R.id.blueSeekBar);
            View colorView = (View) currentDialog.findViewById(R.id.colorView);

            // display the current color
            colorView.setBackgroundColor(Color.argb(
                    alphaSeekBar.getProgress(),
                    redSeekBar.getProgress(),
                    greenSeekBar.getProgress(),
                    blueSeekBar.getProgress()));
        } // end method on Progress Changed

        public void onStartTrackingTouch(SeekBar seekBar)
        {
        }

        public void onStopTrackingTouch(SeekBar seekBar)
        {
        }

    }; // end colorSeekBarChanged

```

20. Build the OnClickListener inner class to handle the button click event:



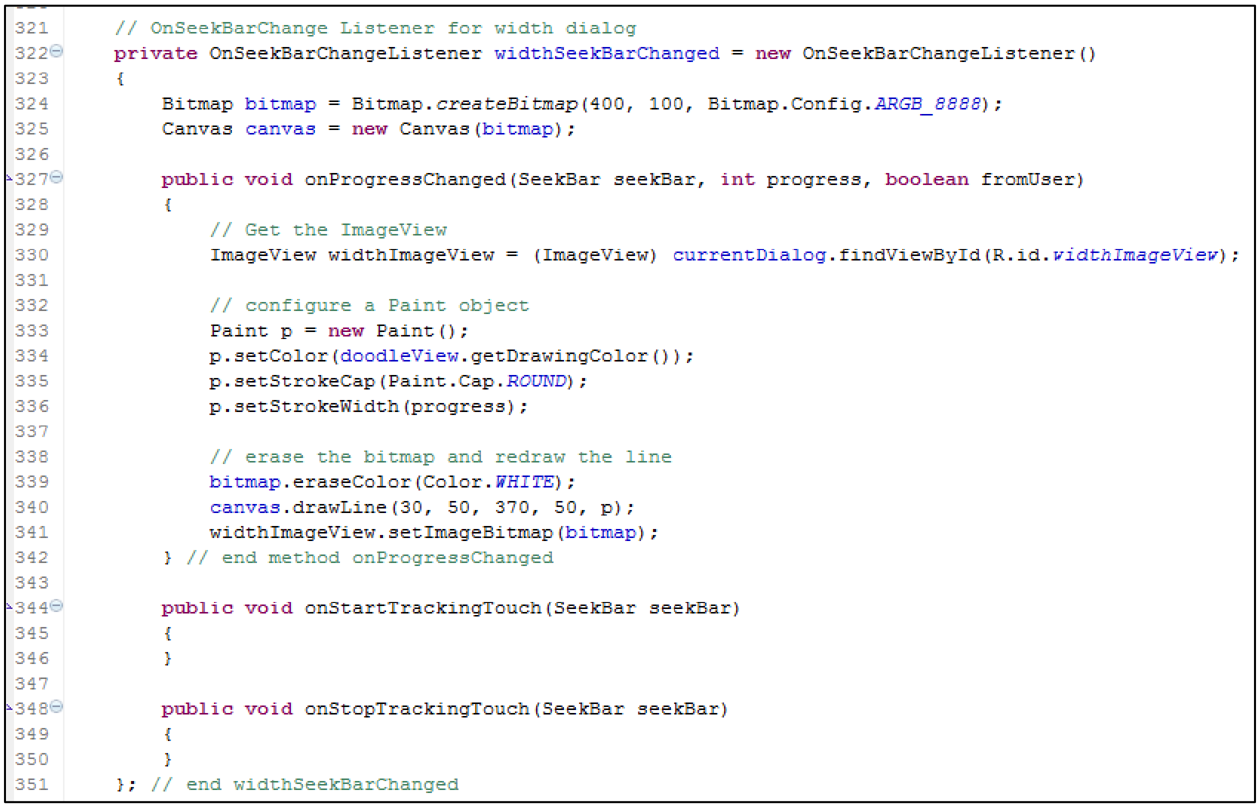
```java

    // OnClickListener for color dialog set button
    private OnClickListener setColorButtonListener = new OnClickListener()
    {
        public void onClick(View v)
        {
            // get the color SeekBars
            SeekBar alphaSeekBar = (SeekBar) currentDialog.findViewById(R.id.alphaSeekBar);
            SeekBar redSeekBar = (SeekBar) currentDialog.findViewById(R.id.redSeekBar);
            SeekBar greenSeekBar = (SeekBar) currentDialog.findViewById(R.id.greenSeekBar);
            SeekBar blueSeekBar = (SeekBar) currentDialog.findViewById(R.id.blueSeekBar);

            // set the line color
            doodleView.setDrawingColor(Color.argb(
                    alphaSeekBar.getProgress(),
                    redSeekBar.getProgress(),
                    greenSeekBar.getProgress(),
                    blueSeekBar.getProgress()));
            dialogIsVisible.set(false); // dialog not on screen
            currentDialog.dismiss(); // hide the dialog
            currentDialog = null; // dialog no longer needed

        } // end method onClick
    }; // End setcolorButton Listener

```

21. We will now build the Dialog to change the line Width. This is similar to the process for the Line Color menu. We will start with the showLineWidthDialog() method



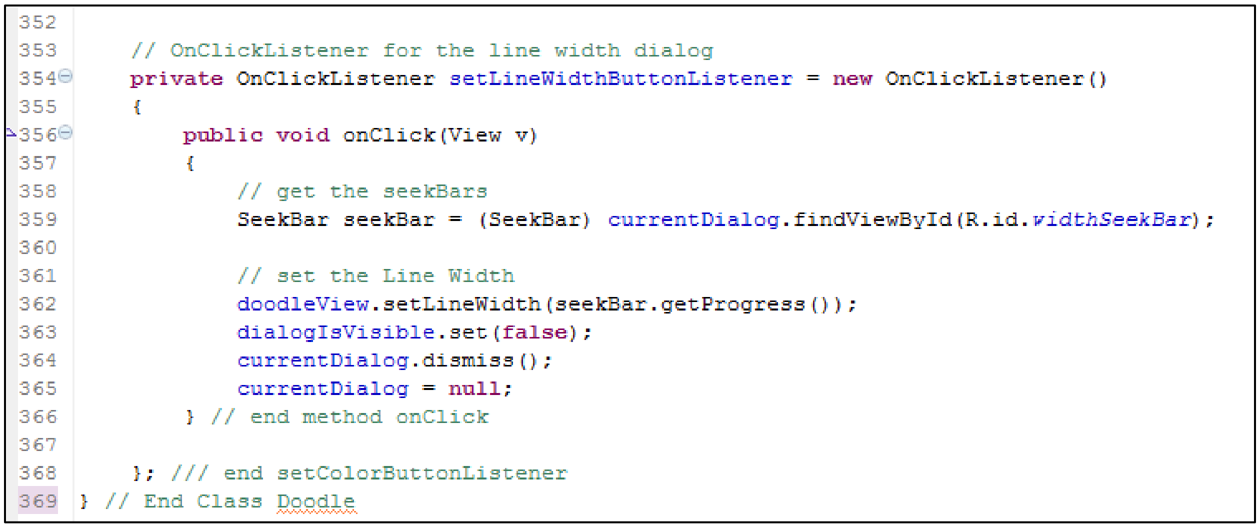
```java

    // display a dialog for setting the line width
    private void showLineWidthDialog()
    {
        // create the dialog and inflate its content
        currentDialog = new Dialog(this);
        currentDialog.setContentView(R.layout.width_dialog);
        currentDialog.setTitle(R.string.title_line_width_dialog);
        currentDialog.setCancelable(true);

        // get widthSeekBar and configure it
        SeekBar widthSeekBar = (SeekBar) currentDialog.findViewById(R.id.widthSeekBar);
        widthSeekBar.setOnSeekBarChangeListener(widthSeekBarChanged);
        widthSeekBar.setProgress(doodleView.getLineWidth());

        // set the Set Line Width Button's onClickListener
        Button setLineWidthButton = (Button) currentDialog.findViewById(R.id.widthDialogDoneButton);
        setLineWidthButton.setOnClickListener(setLineWidthButtonListener);

        dialogIsVisible.set(true);
        currentDialog.show(); // show the dialog
    } // end method showLineWidthDialog

```

22. We will write the inner class for the width seek bar.



```java
    // OnSeekBarChange Listener for width dialog
    private OnSeekBarChangeListener widthSeekBarChanged = new OnSeekBarChangeListener()
    {
        Bitmap bitmap = Bitmap.createBitmap(400, 100, Bitmap.Config.ARGB_8888);
        Canvas canvas = new Canvas(bitmap);

        public void onProgressChanged(SeekBar seekBar, int progress, boolean fromUser)
        {
            // Get the ImageView
            ImageView widthImageView = (ImageView) currentDialog.findViewById(R.id.widthImageView);

            // configure a Paint object
            Paint p = new Paint();
            p.setColor(doodleView.getDrawingColor());
            p.setStrokeCap(Paint.Cap.ROUND);
            p.setStrokeWidth(progress);

            // erase the bitmap and redraw the line
            bitmap.eraseColor(Color.WHITE);
            canvas.drawLine(30, 50, 370, 50, p);
            widthImageView.setImageBitmap(bitmap);
        } // end method onProgressChanged

        public void onStartTrackingTouch(SeekBar seekBar)
        {
        }

        public void onStopTrackingTouch(SeekBar seekBar)
        {
        }
    }; // end widthSeekBarChanged
```

23. We will finish with the inner class for the Button Click listener. We will also end the Class Doodle.



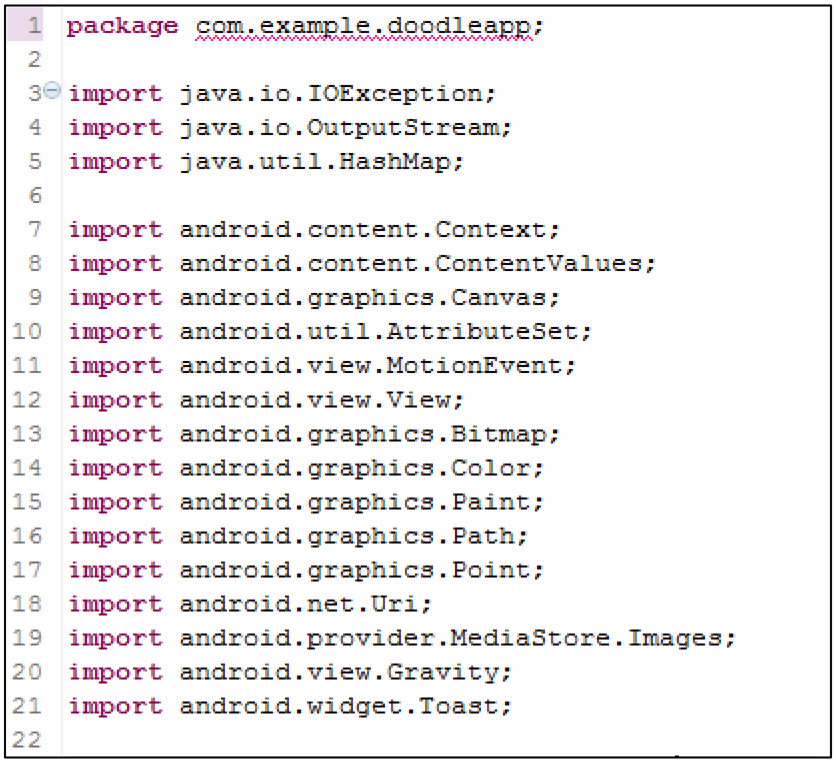
```

    // OnClickListener for the line width dialog
    private OnClickListener setLineWidthButtonListener = new OnClickListener()
    {
        public void onClick(View v)
        {
            // get the seekBars
            SeekBar seekBar = (SeekBar) currentDialog.findViewById(R.id.widthSeekBar);

            // set the Line Width
            doodleView.setLineWidth(seekBar.getProgress());
            dialogIsVisible.set(false);
            currentDialog.dismiss();
            currentDialog = null;
        } // end method onClick

    }; /// end setColorButtonListener
} // End Class Doodle
```

Phase 4: Writing the Code for the DoodleView class

DoodleView extends the Android View class to provide a region on the screen to call drawing and display graphics. In this class we will fill HashMap objects with correct drawing and past drawings. The class will then iterate through these HashMaps to redraw all the lines as the user touches and drags on the screen.

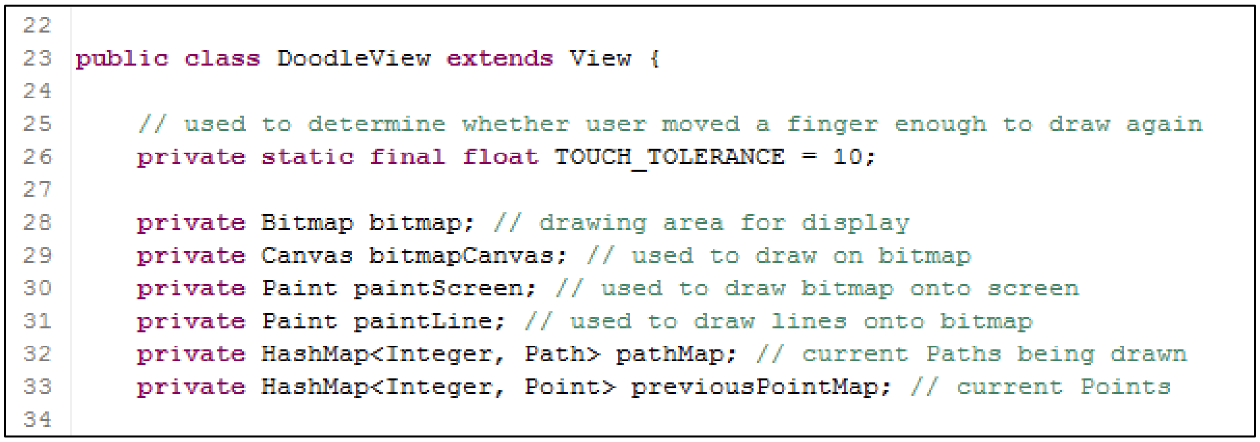
Process:

1. Click on Doodle/src and right click. Select New -> Class and name the Class DoodleView.java
2. Go to the DoodleView.java class and write the code to import the needed Android and java classes (Lines 1 to 22)



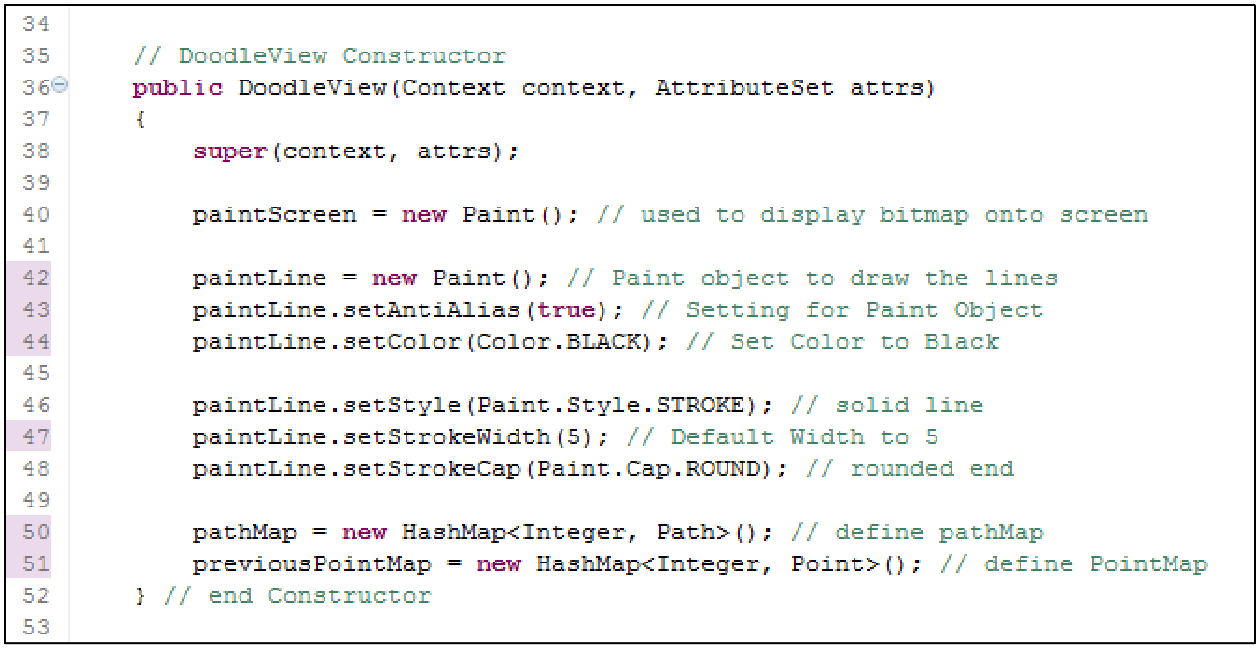
```
1  package com.example.doodleapp;
2
3  import java.io.IOException;
4  import java.io.OutputStream;
5  import java.util.HashMap;
6
7  import android.content.Context;
8  import android.content.ContentValues;
9  import android.graphics.Canvas;
10 import android.util.AttributeSet;
11 import android.view.MotionEvent;
12 import android.view.View;
13 import android.graphics.Bitmap;
14 import android.graphics.Color;
15 import android.graphics.Paint;
16 import android.graphics.Path;
17 import android.graphics.Point;
18 import android.net.Uri;
19 import android.provider.MediaStore.Images;
20 import android.view.Gravity;
21 import android.widget.Toast;
22
```

3. Define the Fields for the DoodleView class. We will also declare that DoodleView will extend the Android View class. (Lines 22 to 34)



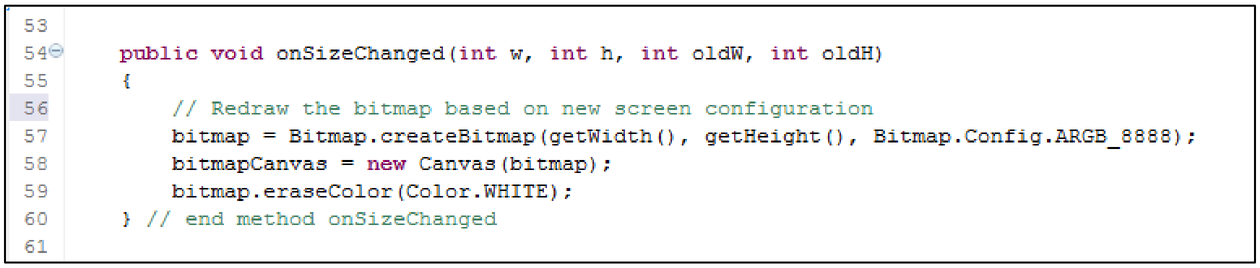
```
22
23 public class DoodleView extends View {
24
25     // used to determine whether user moved a finger enough to draw again
26     private static final float TOUCH_TOLERANCE = 10;
27
28     private Bitmap bitmap; // drawing area for display
29     private Canvas bitmapCanvas; // used to draw on bitmap
30     private Paint paintScreen; // used to draw bitmap onto screen
31     private Paint paintLine; // used to draw lines onto bitmap
32     private HashMap<Integer, Path> pathMap; // current Paths being drawn
33     private HashMap<Integer, Point> previousPointMap; // current Points
34
```

4. Because DoodleView is a java class and not an activity, we will use a standard constructor to initiate the fields



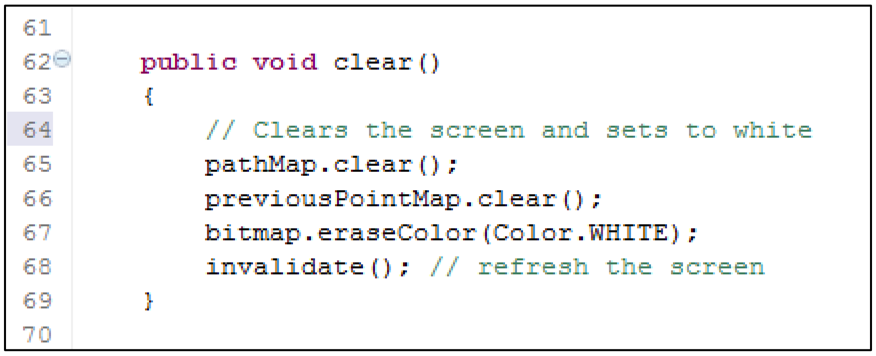
```
34
35     // DoodleView Constructor
36     public DoodleView(Context context, AttributeSet attrs)
37     {
38         super(context, attrs);
39
40         paintScreen = new Paint(); // used to display bitmap onto screen
41
42         paintLine = new Paint(); // Paint object to draw the lines
43         paintLine.setAntiAlias(true); // Setting for Paint Object
44         paintLine.setColor(Color.BLACK); // Set Color to Black
45
46         paintLine.setStyle(Paint.Style.STROKE); // solid line
47         paintLine.setStrokeWidth(5); // Default Width to 5
48         paintLine.setStrokeCap(Paint.Cap.ROUND); // rounded end
49
50         pathMap = new HashMap<Integer, Path>(); // define pathMap
51         previousPointMap = new HashMap<Integer, Point>(); // define PointMap
52     } // end Constructor
53
```

5. The onSizeChanged() method will redraw the screen when the orientation of the device is changed.



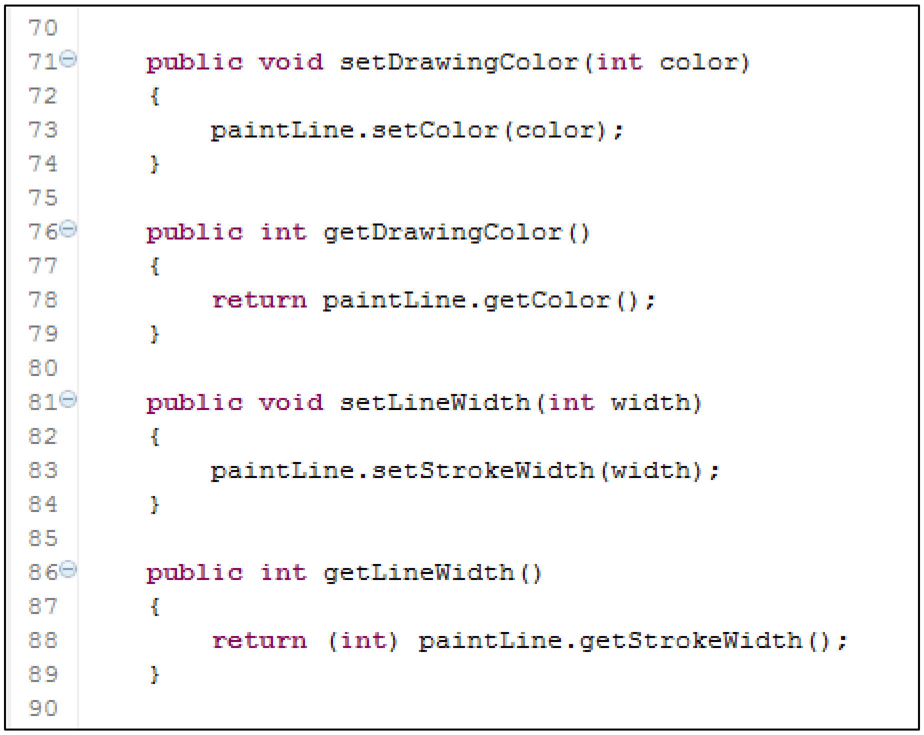
```java

    public void onSizeChanged(int w, int h, int oldW, int oldH)
    {
        // Redraw the bitmap based on new screen configuration
        bitmap = Bitmap.createBitmap(getWidth(), getHeight(), Bitmap.Config.ARGB_8888);
        bitmapCanvas = new Canvas(bitmap);
        bitmap.eraseColor(Color.WHITE);
    } // end method onSizeChanged

```

6. The clear() method will reset the screen and set the color to White.



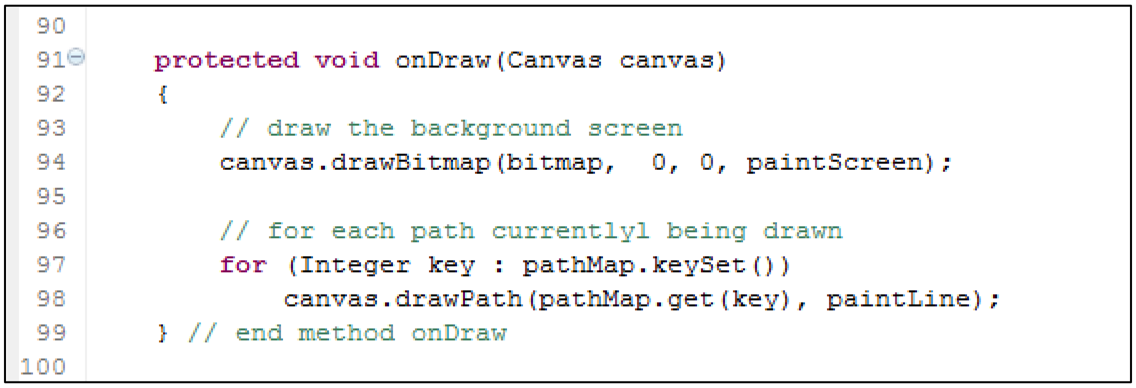
```java

    public void clear()
    {
        // Clears the screen and sets to white
        pathMap.clear();
        previousPointMap.clear();
        bitmap.eraseColor(Color.WHITE);
        invalidate(); // refresh the screen
    }

```

7. The next 4 methods act as modifiers and accessors to the paintLine object color and width data.



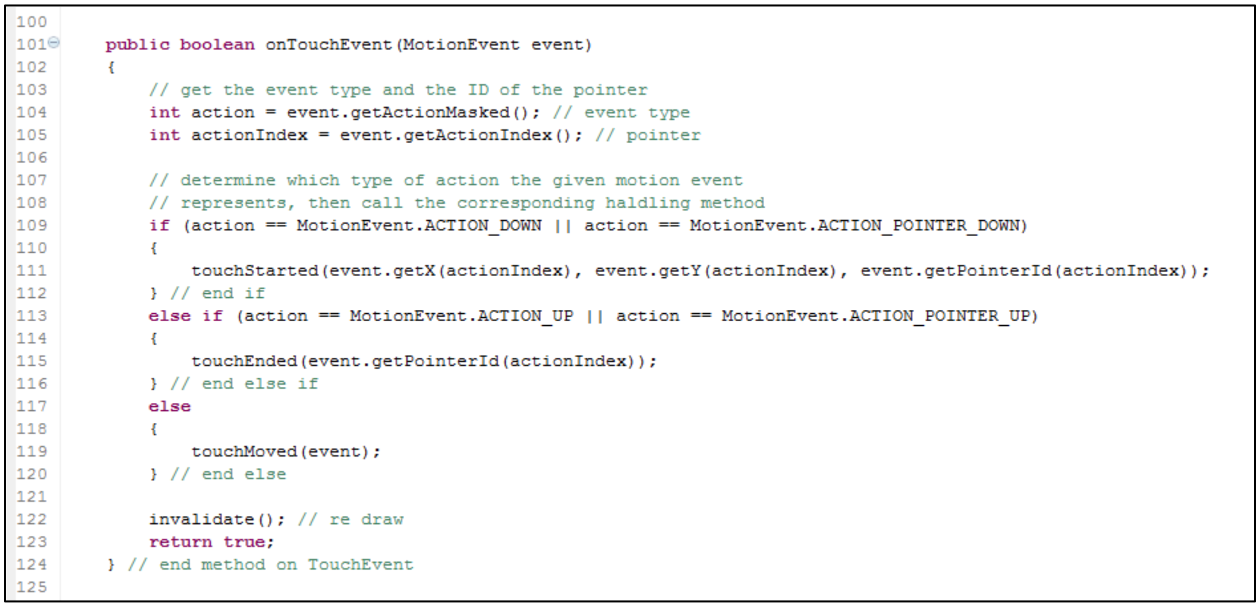
```java

    public void setDrawingColor(int color)
    {
        paintLine.setColor(color);
    }

    public int getDrawingColor()
    {
        return paintLine.getColor();
    }

    public void setLineWidth(int width)
    {
        paintLine.setStrokeWidth(width);
    }

    public int getLineWidth()
    {
        return (int) paintLine.getStrokeWidth();
    }

```

8. The onDraw() method does the work to iterate through the pathMap HashMap and paint the lines.



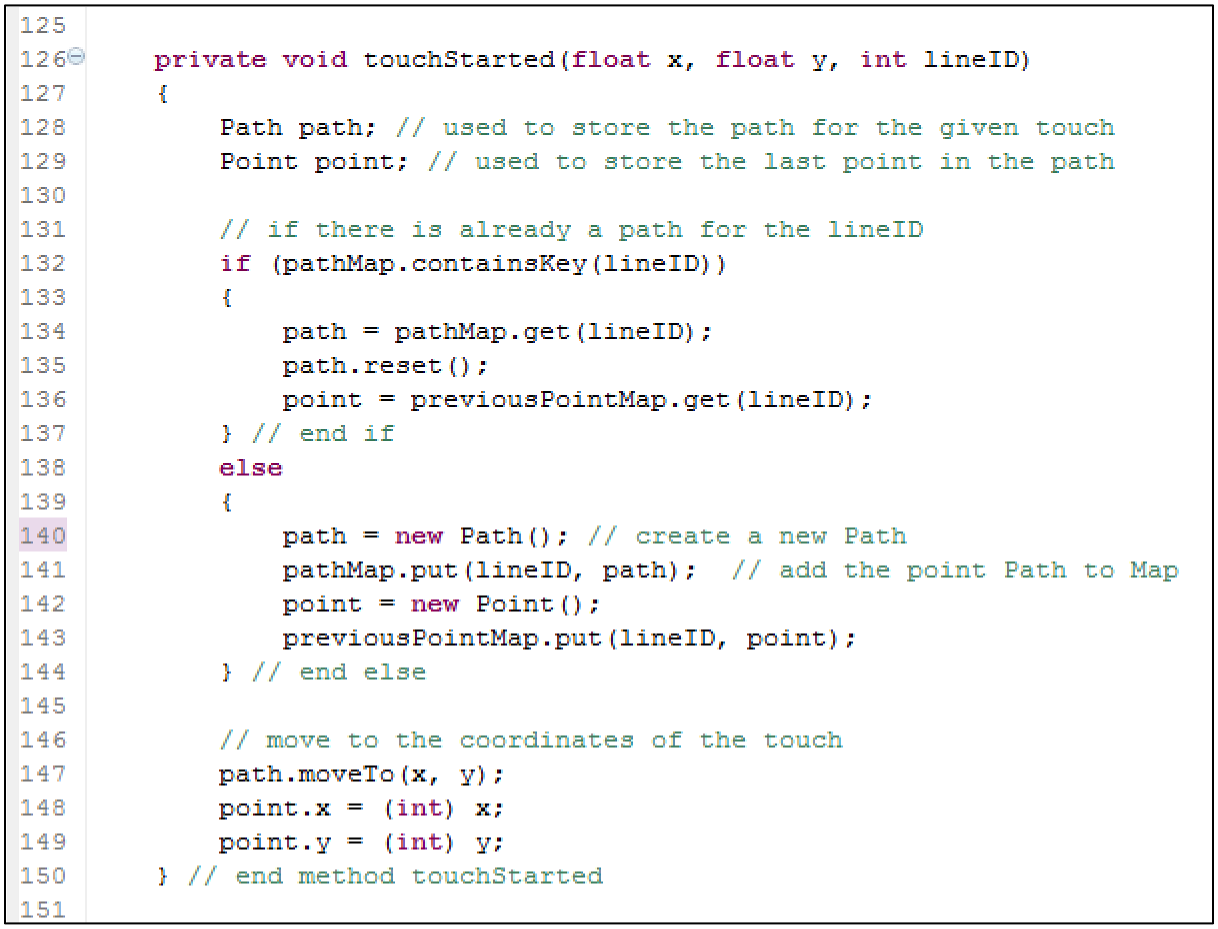
```java

    protected void onDraw(Canvas canvas)
    {
        // draw the background screen
        canvas.drawBitmap(bitmap,  0, 0, paintScreen);

        // for each path currentlyl being drawn
        for (Integer key : pathMap.keySet())
            canvas.drawPath(pathMap.get(key), paintLine);
    } // end method onDraw

```

9. The onTouchEvent() method listens to the user touch events and depending on the type of touch (Touch Down, Touch Up, and Touch Moved)



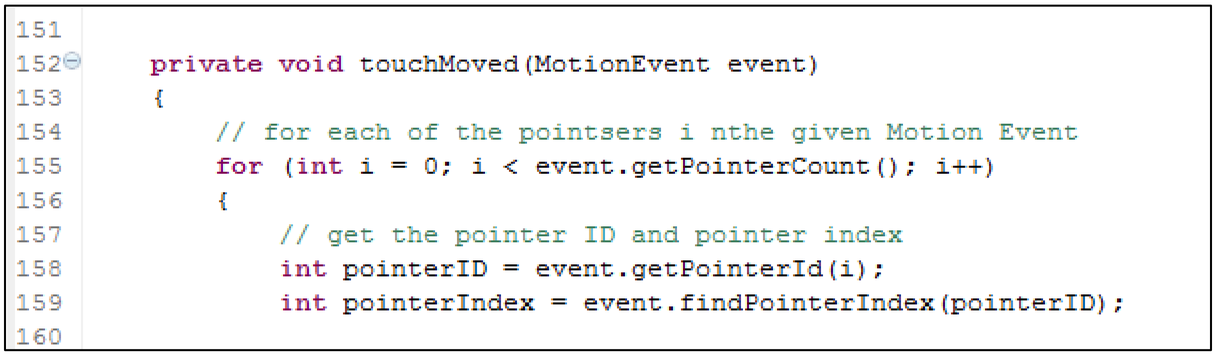
```java

    public boolean onTouchEvent(MotionEvent event)
    {
        // get the event type and the ID of the pointer
        int action = event.getActionMasked(); // event type
        int actionIndex = event.getActionIndex(); // pointer

        // determine which type of action the given motion event
        // represents, then call the corresponding haldling method
        if (action == MotionEvent.ACTION_DOWN || action == MotionEvent.ACTION_POINTER_DOWN)
        {
            touchStarted(event.getX(actionIndex), event.getY(actionIndex), event.getPointerId(actionIndex));
        } // end if
        else if (action == MotionEvent.ACTION_UP || action == MotionEvent.ACTION_POINTER_UP)
        {
            touchEnded(event.getPointerId(actionIndex));
        } // end else if
        else
        {
            touchMoved(event);
        } // end else

        invalidate(); // re draw
        return true;
    } // end method on TouchEvent

```

10. The touchStarted() method will initiate a new Path and put the path in the pathMap HashMap.



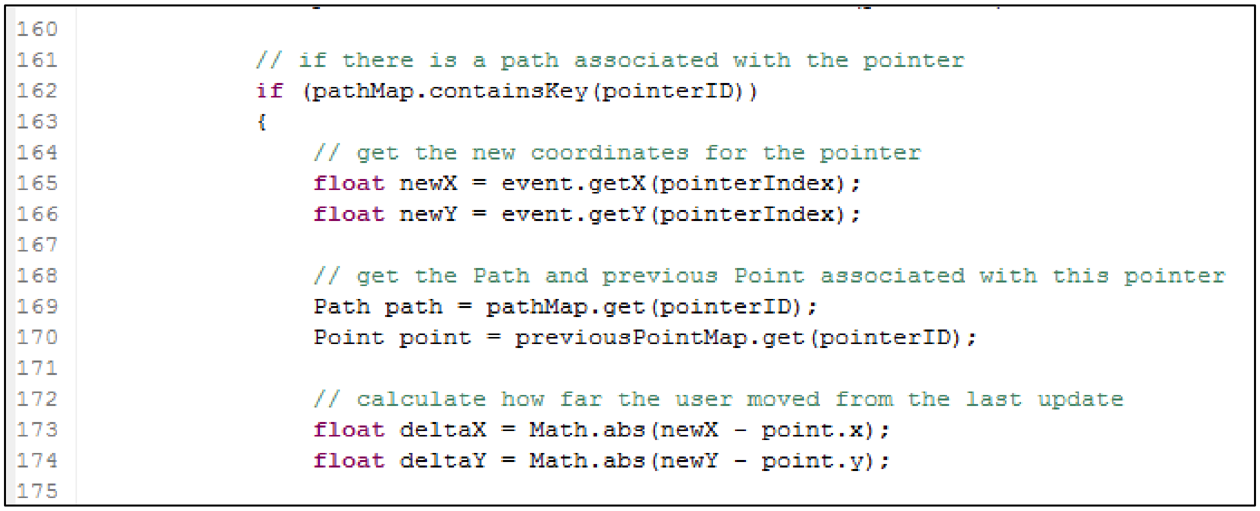
```java

    private void touchStarted(float x, float y, int lineID)
    {
        Path path; // used to store the path for the given touch
        Point point; // used to store the last point in the path

        // if there is already a path for the lineID
        if (pathMap.containsKey(lineID))
        {
            path = pathMap.get(lineID);
            path.reset();
            point = previousPointMap.get(lineID);
        } // end if
        else
        {
            path = new Path(); // create a new Path
            pathMap.put(lineID, path);  // add the point Path to Map
            point = new Point();
            previousPointMap.put(lineID, point);
        } // end else

        // move to the coordinates of the touch
        path.moveTo(x, y);
        point.x = (int) x;
        point.y = (int) y;
    } // end method touchStarted

```

11. The method touchMoved() stores the coordinates of a series of touch events. Start the code for the touchMoved() method



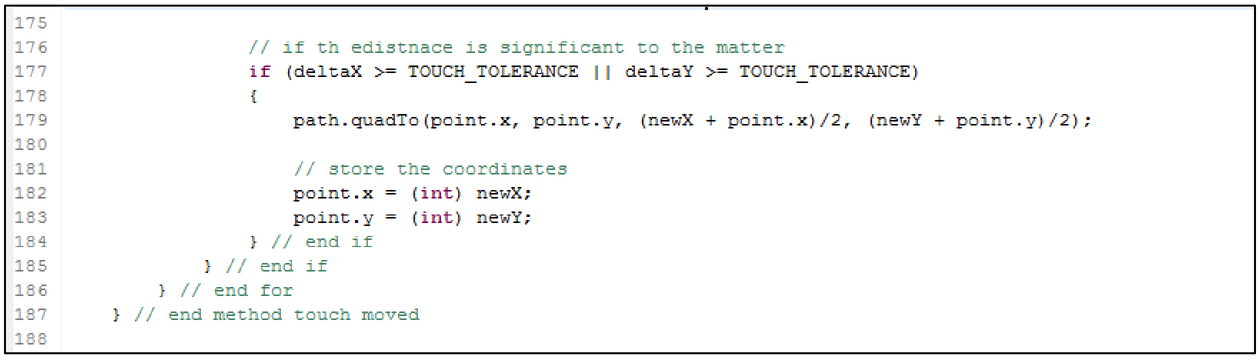
```java

    private void touchMoved(MotionEvent event)
    {
        // for each of the pointsers i nthe given Motion Event
        for (int i = 0; i < event.getPointerCount(); i++)
        {
            // get the pointer ID and pointer index
            int pointerID = event.getPointerId(i);
            int pointerIndex = event.findPointerIndex(pointerID);

```

12. Continue the touchMoved() method with the nested if statements from Lines 161 to 175



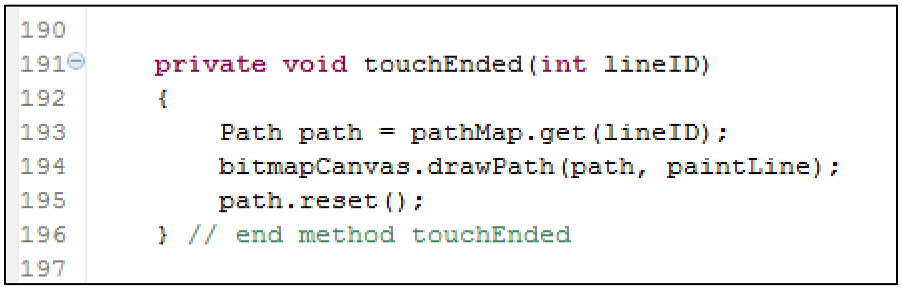
```
160
161             // if there is a path associated with the pointer
162             if (pathMap.containsKey(pointerID))
163             {
164                 // get the new coordinates for the pointer
165                 float newX = event.getX(pointerIndex);
166                 float newY = event.getY(pointerIndex);
167
168                 // get the Path and previous Point associated with this pointer
169                 Path path = pathMap.get(pointerID);
170                 Point point = previousPointMap.get(pointerID);
171
172                 // calculate how far the user moved from the last update
173                 float deltaX = Math.abs(newX - point.x);
174                 float deltaY = Math.abs(newY - point.y);
175
```

13. We add another nesting if to measure if the deltaX or deltaY is greater than the TOUCH_TOLERANCE constant.



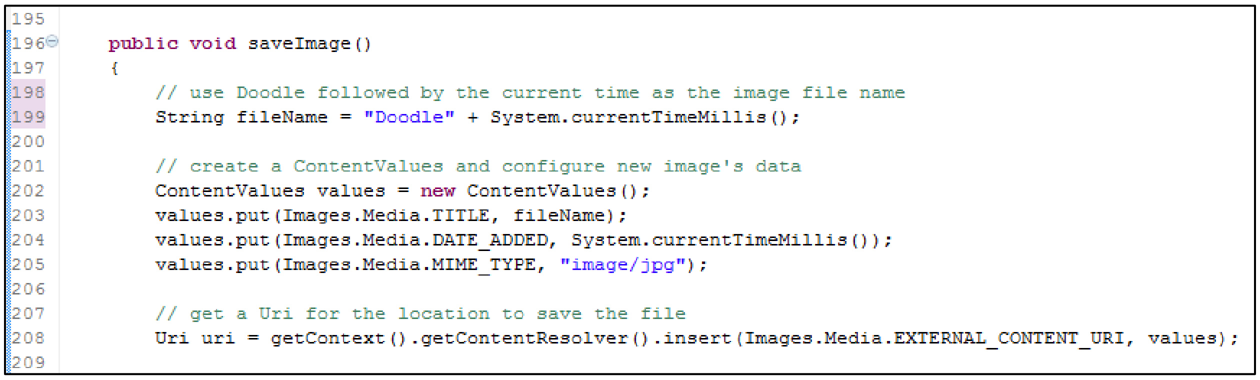
```
175
176                 // if th edistnace is significant to the matter
177                 if (deltaX >= TOUCH_TOLERANCE || deltaY >= TOUCH_TOLERANCE)
178                 {
179                     path.quadTo(point.x, point.y, (newX + point.x)/2, (newY + point.y)/2);
180
181                     // store the coordinates
182                     point.x = (int) newX;
183                     point.y = (int) newY;
184                 } // end if
185             } // end if
186         } // end for
187     } // end method touch moved
188
```

14. Write the method to handle when the touch ends.



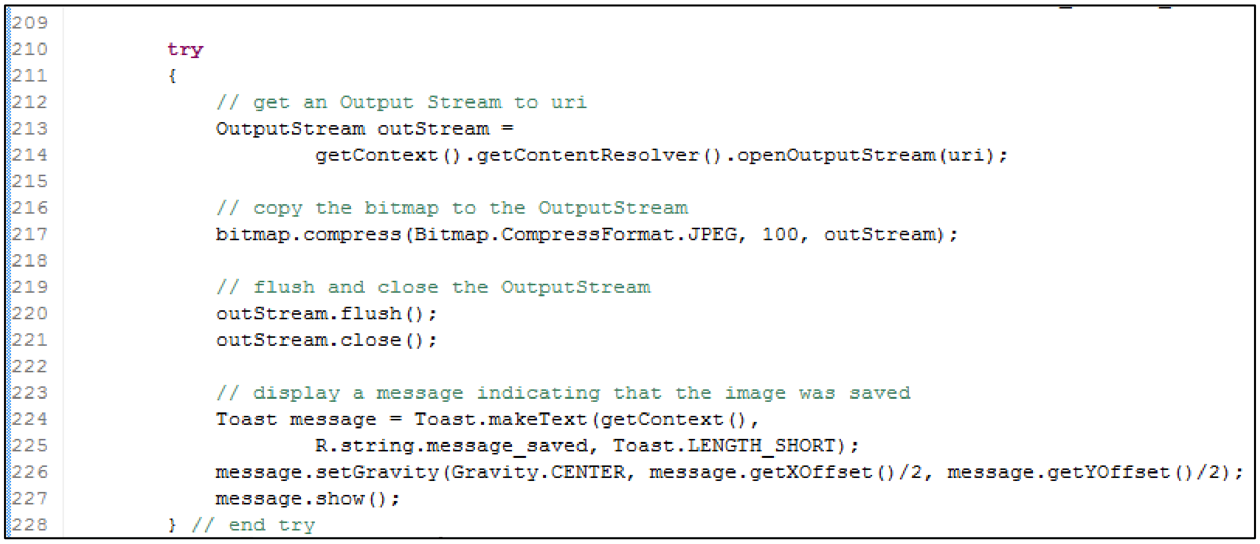
```
190
191     private void touchEnded(int lineID)
192     {
193         Path path = pathMap.get(lineID);
194         bitmapCanvas.drawPath(path, paintLine);
195         path.reset();
196     } // end method touchEnded
197
```

15. The saveImage() method will establish the filename and location for a current image to prepare for saving.



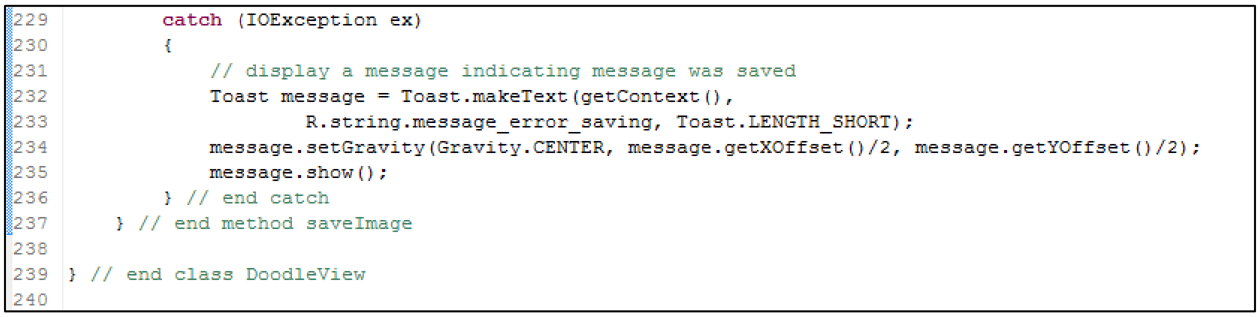
```java

    public void saveImage()
    {
        // use Doodle followed by the current time as the image file name
        String fileName = "Doodle" + System.currentTimeMillis();

        // create a ContentValues and configure new image's data
        ContentValues values = new ContentValues();
        values.put(Images.Media.TITLE, fileName);
        values.put(Images.Media.DATE_ADDED, System.currentTimeMillis());
        values.put(Images.Media.MIME_TYPE, "image/jpg");

        // get a Uri for the location to save the file
        Uri uri = getContext().getContentResolver().insert(Images.Media.EXTERNAL_CONTENT_URI, values);

```

16. The try / catch will save the picture file and provide a Toast feedback message for success or failure. Code the try:



```java

        try
        {
            // get an Output Stream to uri
            OutputStream outStream =
                    getContext().getContentResolver().openOutputStream(uri);

            // copy the bitmap to the OutputStream
            bitmap.compress(Bitmap.CompressFormat.JPEG, 100, outStream);

            // flush and close the OutputStream
            outStream.flush();
            outStream.close();

            // display a message indicating that the image was saved
            Toast message = Toast.makeText(getContext(),
                    R.string.message_saved, Toast.LENGTH_SHORT);
            message.setGravity(Gravity.CENTER, message.getXOffset()/2, message.getYOffset()/2);
            message.show();
        } // end try
```

17. Code the Catch and close the class DoodleView



```java
        catch (IOException ex)
        {
            // display a message indicating message was saved
            Toast message = Toast.makeText(getContext(),
                    R.string.message_error_saving, Toast.LENGTH_SHORT);
            message.setGravity(Gravity.CENTER, message.getXOffset()/2, message.getYOffset()/2);
            message.show();
        } // end catch
    } // end method saveImage

} // end class DoodleView

```

18. You are now finished the Doodle App. Save all files and download to an Android device to test.