```java
package com.gamification_research;

import java.util.Calendar;

/**
 * @author Jonathan Cassar
 * @Class Name: Calendar Screen
 * @Extends Activity
 * @TargetApi(Build.VERSION_CODES.HONEYCOMB)
 * @Description: The calendar screen is responsible for the handling of
calendar events. This class connects to the gmail calendar database and
extracts
 * user's events keyed by the user. Furthermore, it indicates to the user the
free slots in which the user can work on the dissertation.
 */
public class Calendar_screen extends Activity
{
    Data d;
    TextView txt0,txt1,txt2,txt3,txt4,txt5,txt6,txt7,
txt8,txt9,txt10,txt11,txt12,txt13,txt14,txt15,txt16,txt17,txt18,txt19,txt20,txt2
1,txt22,txt23,txt24;
    Time dstart, dend, all_day_start;
    Long start, end, day;
    Context c;
    Toast t;
    Calendar cal;
    int counter =-1;
    int yy,mm,dd,hr,min,sec;
    ContentResolver cr;
    String user,email;
    String[] events;
    int [] hrs;
    SharedPreferences getData;
    String[] evts_chron =
{"-1","-1","-1","-1","-1","-1","-1","-1","-1","-1","-1","-1","-1","-1","-1","-1
","-1","-1","-1","-1","-1","-1","-1","-1","-1"};

    //[1] ON CREATE
    /**
     * @Method name: onCreate
     * @param Bundle savedInstances
     * @return null
     * @Description: On Create method
     */
    protected void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.calendar_screen);

        //Get References
        txt0 = (TextView)findViewById(R.id.txt0_schedule_srn);
        txt1 = (TextView)findViewById(R.id.txt1_schedule_srn);
        txt2 = (TextView)findViewById(R.id.txt2_schedule_srn);
        txt3 = (TextView)findViewById(R.id.txt3_schedule_srn);
        txt4 = (TextView)findViewById(R.id.txt4_schedule_srn);
        txt5 = (TextView)findViewById(R.id.txt5_schedule_srn);
        txt6 = (TextView)findViewById(R.id.txt6_schedule_srn);
```

```
        txt7 = (TextView)findViewById(R.id.txt7_schedule_srn);
        txt8 = (TextView)findViewById(R.id.txt8_schedule_srn);
        txt9 = (TextView)findViewById(R.id.txt9_schedule_srn);
        txt10 = (TextView)findViewById(R.id.txt10_schedule_srn);
        txt11 = (TextView)findViewById(R.id.txt11_schedule_srn);
        txt12 = (TextView)findViewById(R.id.txt12_schedule_srn);
        txt13 = (TextView)findViewById(R.id.txt13_schedule_srn);
        txt14 = (TextView)findViewById(R.id.txt14_schedule_srn);
        txt15 = (TextView)findViewById(R.id.txt15_schedule_srn);
        txt16 = (TextView)findViewById(R.id.txt16_schedule_srn);
        txt17 = (TextView)findViewById(R.id.txt17_schedule_srn);
        txt18 = (TextView)findViewById(R.id.txt18_schedule_srn);
        txt19 = (TextView)findViewById(R.id.txt19_schedule_srn);
        txt20 = (TextView)findViewById(R.id.txt20_schedule_srn);
        txt21 = (TextView)findViewById(R.id.txt21_schedule_srn);
        txt22 = (TextView)findViewById(R.id.txt22_schedule_srn);
        txt23 = (TextView)findViewById(R.id.txt23_schedule_srn);
        txt24 = (TextView)findViewById(R.id.txt24_schedule_srn);

        //Methods
        get_data();
        convert();
        read_calendar();
        arrange_data();
        show_data();
        notification();
    }

    //[2] GET DATA
    /**
     * @Method name: get_data
     * @param null
     * @return null
     * @Description: Get the required data to be used by methods within the
class
     */
    public void get_data()
    {
        getData =
PreferenceManager.getDefaultSharedPreferences(getBaseContext());
        user = getData.getString("user_name", "User");
        email = getData.getString("email", "email");
    }

    //[3] CONVERT TIME TO MILLISECONDS
    /**
     * @Method name: convert
     * @param null
     * @return null
     * @Description: Extracts the system date and convert it to milliseconds
     */
    public void convert()
    {
        dstart = new Time();
        dend = new Time();
        all_day_start = new Time();
```

```
        dstart.setToNow();

        dstart.hour=0;
        dstart.minute=0;
        dstart.second=0;

        dend.set(dstart);
        dend.hour=dstart.hour+24;
        dend.minute=dstart.minute+59;
        dend.second=dstart.second+59;

        all_day_start.timezone=TimeZone.getDefault().toString();
        all_day_start.set(dstart.monthDay, dstart.month, dstart.year);

        start = dstart.toMillis(false);
        end = dend.toMillis(false) + 999;
        day = all_day_start.toMillis(false);
    }

    //[4] READ DATA FROM GMAIL CALENDAR
    /**
     * @Method name: read_calendar
     * @param null
     * @return null
     * @SuppressLint("InlinedApi")
     * @Description: Reads data from the gmail calendar through the use of an
SQL query.
     */
    public void read_calendar()
    {
        c = getApplicationContext();
        cr = c.getContentResolver();

        //Columns to query
        String [] COLS = new String[]{"calendar_id", "title","dtstart",
"dtend"};

        //Select statement
        String select =
                "((" + Calendars.ACCOUNT_NAME + " = ?) " +
                        "AND (" + Calendars.OWNER_ACCOUNT + "= ?) " +
                        "AND (" +
                        "((" + Events.DTSTART + ">= ?) " +
                        "AND (" + Events.DTSTART + "<= ?) " +
                        "AND (" + Events.ALL_DAY + "= ?) " +
                        ") " +
                        "OR ((" + Events.DTSTART + "= ?) " +
                        "AND (" + Events.ALL_DAY + "= ?)" +
                        ")" +
                        ")" +
                        ")";

        //Where clause
        String[] where = new String[] {email,email,start.toString(),
end.toString(), "0",day.toString(), "1" };

        //Query
```

```
        Cursor crsr =
cr.query(Uri.parse("content://com.android.calendar/events"),COLS,select,where,E
vents.DTSTART);

        //Get data
        crsr.moveToFirst();
        events = new String[crsr.getCount()];    //size of cursor i.e database
        hrs =  new int[crsr.getCount()];

        for (int i = 0; i < events.length; i++)
        {
            //Convert milliseconds date to dd/mm/yy hr:mm:ss
            cal = Calendar.getInstance();
            cal.setTimeInMillis(Long.parseLong(crsr.getString(2)));
            yy = cal.get(Calendar.YEAR);
            mm = cal.get(Calendar.MONTH)+1;
            dd = cal.get(Calendar.DAY_OF_MONTH);
            hr = cal.get(Calendar.HOUR_OF_DAY);
            min = cal.get(Calendar.MINUTE);
            sec = cal.get(Calendar.SECOND);

            String d = dd+"/"+mm+"/"+yy;
            String t = hr+":"+min+":"+sec;
            hrs[i]=hr;

            events[i] = crsr.getString(1)+" @ "+d+" @ "+t;  //array with events

            crsr.moveToNext();
        }
        crsr.close();
    }

    //[5] ARRANGE DATA
    /**
     * @Method name: arrange_data
     * @param null
     * @return null
     * @Description: Arranges the retrieved data in a chronological order.
     */
    public void arrange_data()
    {

        for (int i=0; i<hrs.length;i++)
        {
            if(hrs[i]==0){
                evts_chron[0] = events[i];
            }else if(hrs[i]==1){
                evts_chron[1] = events[i];
            }else if(hrs[i]==2){
                evts_chron[2] = events[i];
            }else if(hrs[i]==3){
                evts_chron[3] = events[i];
            }else if(hrs[i]==4){
                evts_chron[4] = events[i];
            }else if(hrs[i]==5){
                evts_chron[5] = events[i];
            }else if(hrs[i]==6){
```

```
                evts_chron[6] = events[i];
            }else if(hrs[i]==7){
                evts_chron[7] = events[i];
            }else if(hrs[i]==8){
                evts_chron[8] = events[i];
            }else if(hrs[i]==9){
                evts_chron[9] = events[i];
            }else if(hrs[i]==10){
                evts_chron[10] = events[i];
            }else if(hrs[i]==11){
                evts_chron[11] = events[i];
            }else if(hrs[i]==12){
                evts_chron[12] = events[i];
            }else if(hrs[i]==13){
                evts_chron[13] = events[i];
            }else if(hrs[i]==14){
                evts_chron[14] = events[i];
            }else if(hrs[i]==15){
                evts_chron[15] = events[i];
            }else if(hrs[i]==16){
                evts_chron[16] = events[i];
            }else if(hrs[i]==17){
                evts_chron[17] = events[i];
            }else if(hrs[i]==18){
                evts_chron[18] = events[i];
            }else if(hrs[i]==19){
                evts_chron[19] = events[i];
            }else if(hrs[i]==20){
                evts_chron[20] = events[i];
            }else if(hrs[i]==21){
                evts_chron[21] = events[i];
            }else if(hrs[i]==22){
                evts_chron[22] = events[i];
            }else if(hrs[i]==23){
                evts_chron[23] = events[i];
            }else if(hrs[i]==24){
                evts_chron[24] = events[i];
            }
        }

        //replace -1 with text and change colour
        for(int i=0;i<evts_chron.length; i++)
        {
            if(evts_chron[i]=="-1"){
                evts_chron[i]="<font color=\"green\">* You can work on the
dissertation.</font>";
                counter+=1;
            }
        }
    }

    //[6] SHOW DATA
    /**
     * @Method name: show_data
     * @param null
     * @return null
     * @Description: Display to screen the retrieved data. In case the slot is
```

```
free, display text in green otherwise display text in white.
     */
    public void show_data()
    {
        txt0.setText(Html.fromHtml(evts_chron[0]), BufferType.SPANNABLE);
        txt1.setText(Html.fromHtml(evts_chron[1]), BufferType.SPANNABLE);
        txt2.setText(Html.fromHtml(evts_chron[2]), BufferType.SPANNABLE);
        txt3.setText(Html.fromHtml(evts_chron[3]), BufferType.SPANNABLE);
        txt4.setText(Html.fromHtml(evts_chron[4]), BufferType.SPANNABLE);
        txt5.setText(Html.fromHtml(evts_chron[5]), BufferType.SPANNABLE);
        txt6.setText(Html.fromHtml(evts_chron[6]), BufferType.SPANNABLE);
        txt7.setText(Html.fromHtml(evts_chron[7]), BufferType.SPANNABLE);
        txt8.setText(Html.fromHtml(evts_chron[8]), BufferType.SPANNABLE);
        txt9.setText(Html.fromHtml(evts_chron[9]), BufferType.SPANNABLE);
        txt10.setText(Html.fromHtml(evts_chron[10]), BufferType.SPANNABLE);
        txt11.setText(Html.fromHtml(evts_chron[11]), BufferType.SPANNABLE);

        txt12.setText(Html.fromHtml(evts_chron[12]), BufferType.SPANNABLE);
        txt13.setText(Html.fromHtml(evts_chron[13]), BufferType.SPANNABLE);
        txt14.setText(Html.fromHtml(evts_chron[14]), BufferType.SPANNABLE);
        txt15.setText(Html.fromHtml(evts_chron[15]), BufferType.SPANNABLE);
        txt16.setText(Html.fromHtml(evts_chron[16]), BufferType.SPANNABLE);
        txt17.setText(Html.fromHtml(evts_chron[17]), BufferType.SPANNABLE);
        txt18.setText(Html.fromHtml(evts_chron[18]), BufferType.SPANNABLE);
        txt19.setText(Html.fromHtml(evts_chron[19]), BufferType.SPANNABLE);
        txt20.setText(Html.fromHtml(evts_chron[20]), BufferType.SPANNABLE);
        txt21.setText(Html.fromHtml(evts_chron[21]), BufferType.SPANNABLE);
        txt22.setText(Html.fromHtml(evts_chron[22]), BufferType.SPANNABLE);
        txt23.setText(Html.fromHtml(evts_chron[23]), BufferType.SPANNABLE);
        txt24.setText(Html.fromHtml(evts_chron[24]), BufferType.SPANNABLE);

    }

   //[7] NOTIFICATION
    /**
     * @Method name: notification
     * @param null
     * @return null
     * @Description: Notify the user about the free time which can be used to
work on the dissertation.
     */
    public void notification()
    {
        int net =  counter -8; // considers 8 hrs of sleep
        c = getApplicationContext();
        CharSequence text = "Considering 8 hours of sleep, you have "+ net +
"hrs to work on the dissertation.";
        int duration = Toast.LENGTH_LONG;

        t= Toast.makeText(c, text, duration);
        t.setGravity(Gravity.CENTER|Gravity.CENTER, 0, 0);
        t.show();
    }
}
```

```
 * Copyright 2013 Matt Joseph

package com.gamification_research;
import com.gamification_research.R;

public class CircularSeekBar extends View {

    /**
     * Used to scale the dp units to pixels
     */
    private final float DPTOPX_SCALE =
getResources().getDisplayMetrics().density;

    /**
     * Minimum touch target size in DP. 48dp is the Android design
recommendation
     */
    private final float MIN_TOUCH_TARGET_DP = 48;

    // Default values
    private static final float DEFAULT_CIRCLE_X_RADIUS = 30f;
    private static final float DEFAULT_CIRCLE_Y_RADIUS = 30f;
    private static final float DEFAULT_POINTER_RADIUS = 7f;
    private static final float DEFAULT_POINTER_HALO_WIDTH = 15f;
    private static final float DEFAULT_POINTER_HALO_BORDER_WIDTH = 15f;
    private static final float DEFAULT_CIRCLE_STROKE_WIDTH = 15f;
    private static final float DEFAULT_START_ANGLE = 270f; // Geometric
(clockwise, relative to 3 o'clock)
    private static final float DEFAULT_END_ANGLE = 270f; // Geometric
(clockwise, relative to 3 o'clock)
    private static final int DEFAULT_MAX = 100;
    private static final int DEFAULT_PROGRESS = 0;
    private static final int DEFAULT_CIRCLE_COLOR = Color.DKGRAY;
    private static final int DEFAULT_CIRCLE_PROGRESS_COLOR = Color.argb(235,
255, 255, 0);
    private static final int DEFAULT_POINTER_COLOR = Color.argb(235, 255, 255,
0);
    private static final int DEFAULT_POINTER_HALO_COLOR = Color.argb(135, 200,
200, 0);
    private static final int DEFAULT_CIRCLE_FILL_COLOR = Color.TRANSPARENT;
    private static final int DEFAULT_POINTER_ALPHA = 135;
    private static final int DEFAULT_POINTER_ALPHA_ONTOUCH = 100;
    private static final boolean DEFAULT_USE_CUSTOM_RADII = false;
    private static final boolean DEFAULT_MAINTAIN_EQUAL_CIRCLE = true;
    private static final boolean DEFAULT_MOVE_OUTSIDE_CIRCLE = false;

    /**
     * {@code Paint} instance used to draw the inactive circle.
     */
    private Paint mCirclePaint;

    /**
     * {@code Paint} instance used to draw the circle fill.
     */
    private Paint mCircleFillPaint;

    /**
```

```
     * {@code Paint} instance used to draw the active circle (represents
progress).
     */
    private Paint mCircleProgressPaint;

    /**
     * {@code Paint} instance used to draw the glow from the active circle.
     */
    private Paint mCircleProgressGlowPaint;

    /**
     * {@code Paint} instance used to draw the center of the pointer.
     * Note: This is broken on 4.0+, as BlurMasks do not work with hardware
acceleration.
     */
    private Paint mPointerPaint;

    /**
     * {@code Paint} instance used to draw the halo of the pointer.
     * Note: The halo is the part that changes transparency.
     */
    private Paint mPointerHaloPaint;

    /**
     * {@code Paint} instance used to draw the border of the pointer, outside
of the halo.
     */
    private Paint mPointerHaloBorderPaint;

    /**
     * The width of the circle (in pixels).
     */
    private float mCircleStrokeWidth;

    /**
     * The X radius of the circle (in pixels).
     */
    private float mCircleXRadius;

    /**
     * The Y radius of the circle (in pixels).
     */
    private float mCircleYRadius;

    /**
     * The radius of the pointer (in pixels).
     */
    private float mPointerRadius;

    /**
     * The width of the pointer halo (in pixels).
     */
    private float mPointerHaloWidth;

    /**
     * The width of the pointer halo border (in pixels).
     */
```

```
    private float mPointerHaloBorderWidth;

    /**
     * Start angle of the CircularSeekBar.
     * Note: If mStartAngle and mEndAngle are set to the same angle, 0.1 is
subtracted
     * from the mEndAngle to make the circle function properly.
     */
    private float mStartAngle;

    /**
     * End angle of the CircularSeekBar.
     * Note: If mStartAngle and mEndAngle are set to the same angle, 0.1 is
subtracted
     * from the mEndAngle to make the circle function properly.
     */
    private float mEndAngle;

    /**
     * {@code RectF} that represents the circle (or ellipse) of the seekbar.
     */
    private RectF mCircleRectF = new RectF();

    /**
     * Holds the color value for {@code mPointerPaint} before the {@code Paint}
instance is created.
     */
    private int mPointerColor = DEFAULT_POINTER_COLOR;

    /**
     * Holds the color value for {@code mPointerHaloPaint} before the {@code
Paint} instance is created.
     */
    private int mPointerHaloColor = DEFAULT_POINTER_HALO_COLOR;

    /**
     * Holds the color value for {@code mCirclePaint} before the {@code Paint}
instance is created.
     */
    private int mCircleColor = DEFAULT_CIRCLE_COLOR;

    /**
     * Holds the color value for {@code mCircleFillPaint} before the {@code
Paint} instance is created.
     */
    private int mCircleFillColor = DEFAULT_CIRCLE_FILL_COLOR;

    /**
     * Holds the color value for {@code mCircleProgressPaint} before the {@code
Paint} instance is created.
     */
    private int mCircleProgressColor = DEFAULT_CIRCLE_PROGRESS_COLOR;

    /**
     * Holds the alpha value for {@code mPointerHaloPaint}.
     */
    private int mPointerAlpha = DEFAULT_POINTER_ALPHA;
```

```
    /**
     * Holds the OnTouch alpha value for {@code mPointerHaloPaint}.
     */
    private int mPointerAlphaOnTouch = DEFAULT_POINTER_ALPHA_ONTOUCH;

    /**
     * Distance (in degrees) that the the circle/semi-circle makes up.
     * This amount represents the max of the circle in degrees.
     */
    private float mTotalCircleDegrees;

    /**
     * Distance (in degrees) that the current progress makes up in the circle.
     */
    private float mProgressDegrees;

    /**
     * {@code Path} used to draw the circle/semi-circle.
     */
    private Path mCirclePath;

    /**
     * {@code Path} used to draw the progress on the circle.
     */
    private Path mCircleProgressPath;

    /**
     * Max value that this CircularSeekBar is representing.
     */
    private int mMax;

    /**
     * Progress value that this CircularSeekBar is representing.
     */
    private int mProgress;

    /**
     * If true, then the user can specify the X and Y radii.
     * If false, then the View itself determines the size of the
CircularSeekBar.
     */
    private boolean mCustomRadii;

    /**
     * Maintain a perfect circle (equal x and y radius), regardless of view or
custom attributes.
     * The smaller of the two radii will always be used in this case.
     * The default is to be a circle and not an ellipse, due to the behavior of
the ellipse.
     */
    private boolean mMaintainEqualCircle;

    /**
     * Once a user has touched the circle, this determines if moving outside
the circle is able
     * to change the position of the pointer (and in turn, the progress).
```

```
     */
    private boolean mMoveOutsideCircle;

    /**
     * Used for when the user moves beyond the start of the circle when moving
counter clockwise.
     * Makes it easier to hit the 0 progress mark.
     */
    private boolean lockAtStart = true;

    /**
     * Used for when the user moves beyond the end of the circle when moving
clockwise.
     * Makes it easier to hit the 100% (max) progress mark.
     */
    private boolean lockAtEnd = false;

    /**
     * When the user is touching the circle on ACTION_DOWN, this is set to
true.
     * Used when touching the CircularSeekBar.
     */
    private boolean mUserIsMovingPointer = false;

    /**
     * Represents the clockwise distance from {@code mStartAngle} to the touch
angle.
     * Used when touching the CircularSeekBar.
     */
    private float cwDistanceFromStart;

    /**
     * Represents the counter-clockwise distance from {@code mStartAngle} to
the touch angle.
     * Used when touching the CircularSeekBar.
     */
    private float ccwDistanceFromStart;

    /**
     * Represents the clockwise distance from {@code mEndAngle} to the touch
angle.
     * Used when touching the CircularSeekBar.
     */
    private float cwDistanceFromEnd;

    /**
     * Represents the counter-clockwise distance from {@code mEndAngle} to the
touch angle.
     * Used when touching the CircularSeekBar.
     * Currently unused, but kept just in case.
     */
    @SuppressWarnings("unused")
    private float ccwDistanceFromEnd;

    /**
     * The previous touch action value for {@code cwDistanceFromStart}.
     * Used when touching the CircularSeekBar.
```

```
     */
    private float lastCWDistanceFromStart;

    /**
     * Represents the clockwise distance from {@code mPointerPosition} to the
touch angle.
     * Used when touching the CircularSeekBar.
     */
    private float cwDistanceFromPointer;

    /**
     * Represents the counter-clockwise distance from {@code mPointerPosition}
to the touch angle.
     * Used when touching the CircularSeekBar.
     */
    private float ccwDistanceFromPointer;

    /**
     * True if the user is moving clockwise around the circle, false if moving
counter-clockwise.
     * Used when touching the CircularSeekBar.
     */
    private boolean mIsMovingCW;

    /**
     * The width of the circle used in the {@code RectF} that is used to draw
it.
     * Based on either the View width or the custom X radius.
     */
    private float mCircleWidth;

    /**
     * The height of the circle used in the {@code RectF} that is used to draw
it.
     * Based on either the View width or the custom Y radius.
     */
    private float mCircleHeight;

    /**
     * Represents the progress mark on the circle, in geometric degrees.
     * This is not provided by the user; it is calculated;
     */
    private float mPointerPosition;

    /**
     * Pointer position in terms of X and Y coordinates.
     */
    private float[] mPointerPositionXY = new float[2];

    /**
     * Listener.
     */
    private OnCircularSeekBarChangeListener mOnCircularSeekBarChangeListener;

    /**
     * Initialize the CircularSeekBar with the attributes from the XML style.
     * Uses the defaults defined at the top of this file when an attribute is
```

```
not specified by the user.
     * @param attrArray TypedArray containing the attributes.
     */
    private void initAttributes(TypedArray attrArray) {
        mCircleXRadius = (float)
(attrArray.getFloat(R.styleable.CircularSeekBar_circle_x_radius,
DEFAULT_CIRCLE_X_RADIUS) * DPTOPX_SCALE);
        mCircleYRadius = (float)
(attrArray.getFloat(R.styleable.CircularSeekBar_circle_y_radius,
DEFAULT_CIRCLE_Y_RADIUS) * DPTOPX_SCALE);
        mPointerRadius = (float)
(attrArray.getFloat(R.styleable.CircularSeekBar_pointer_radius,
DEFAULT_POINTER_RADIUS) * DPTOPX_SCALE);
        mPointerHaloWidth = (float)
(attrArray.getFloat(R.styleable.CircularSeekBar_pointer_halo_width,
DEFAULT_POINTER_HALO_WIDTH) * DPTOPX_SCALE);
        mPointerHaloBorderWidth = (float)
(attrArray.getFloat(R.styleable.CircularSeekBar_pointer_halo_border_width,
DEFAULT_POINTER_HALO_BORDER_WIDTH) * DPTOPX_SCALE);
        mCircleStrokeWidth = (float)
(attrArray.getFloat(R.styleable.CircularSeekBar_circle_stroke_width,
DEFAULT_CIRCLE_STROKE_WIDTH) * DPTOPX_SCALE);

        String tempColor =
attrArray.getString(R.styleable.CircularSeekBar_pointer_color);
        if (tempColor != null) {
            try {
                mPointerColor = Color.parseColor(tempColor);
            } catch (IllegalArgumentException e) {
                mPointerColor = DEFAULT_POINTER_COLOR;
            }
        }

        tempColor =
attrArray.getString(R.styleable.CircularSeekBar_pointer_halo_color);
        if (tempColor != null) {
            try {
                mPointerHaloColor = Color.parseColor(tempColor);
            } catch (IllegalArgumentException e) {
                mPointerHaloColor = DEFAULT_POINTER_HALO_COLOR;
            }
        }

        tempColor =
attrArray.getString(R.styleable.CircularSeekBar_circle_color);
        if (tempColor != null) {
            try {
                mCircleColor = Color.parseColor(tempColor);
            } catch (IllegalArgumentException e) {
                mCircleColor = DEFAULT_CIRCLE_COLOR;
            }
        }

        tempColor =
attrArray.getString(R.styleable.CircularSeekBar_circle_progress_color);
        if (tempColor != null) {
            try {
```

```
                mCircleProgressColor = Color.parseColor(tempColor);
            } catch (IllegalArgumentException e) {
                mCircleProgressColor = DEFAULT_CIRCLE_PROGRESS_COLOR;
            }
        }

        tempColor =
attrArray.getString(R.styleable.CircularSeekBar_circle_fill);
        if (tempColor != null) {
            try {
                mCircleFillColor = Color.parseColor(tempColor);
            } catch (IllegalArgumentException e) {
                mCircleFillColor = DEFAULT_CIRCLE_FILL_COLOR;
            }
        }

        mPointerAlpha = Color.alpha(mPointerHaloColor);

        mPointerAlphaOnTouch =
attrArray.getInt(R.styleable.CircularSeekBar_pointer_alpha_ontouch,
DEFAULT_POINTER_ALPHA_ONTOUCH);
        if (mPointerAlphaOnTouch > 255 || mPointerAlphaOnTouch < 0) {
            mPointerAlphaOnTouch = DEFAULT_POINTER_ALPHA_ONTOUCH;
        }

        mMax = attrArray.getInt(R.styleable.CircularSeekBar_max, DEFAULT_MAX);
        mProgress = attrArray.getInt(R.styleable.CircularSeekBar_progress,
DEFAULT_PROGRESS);
        mCustomRadii =
attrArray.getBoolean(R.styleable.CircularSeekBar_use_custom_radii,
DEFAULT_USE_CUSTOM_RADII);
        mMaintainEqualCircle =
attrArray.getBoolean(R.styleable.CircularSeekBar_maintain_equal_circle,
DEFAULT_MAINTAIN_EQUAL_CIRCLE);
        mMoveOutsideCircle =
attrArray.getBoolean(R.styleable.CircularSeekBar_move_outside_circle,
DEFAULT_MOVE_OUTSIDE_CIRCLE);

        // Modulo 360 right now to avoid constant conversion
        mStartAngle = ((360f +
(attrArray.getFloat((R.styleable.CircularSeekBar_start_angle),
DEFAULT_START_ANGLE) % 360f)) % 360f);
        mEndAngle = ((360f +
(attrArray.getFloat((R.styleable.CircularSeekBar_end_angle), DEFAULT_END_ANGLE)
% 360f)) % 360f);

        if (mStartAngle == mEndAngle) {
            //mStartAngle = mStartAngle + 1f;
            mEndAngle = mEndAngle - .1f;
        }


    }

    /**
     * Initializes the {@code Paint} objects with the appropriate styles.
     */
```

```
    private void initPaints() {
        mCirclePaint = new Paint();
        mCirclePaint.setAntiAlias(true);
        mCirclePaint.setDither(true);
        mCirclePaint.setColor(mCircleColor);
        mCirclePaint.setStrokeWidth(mCircleStrokeWidth);
        mCirclePaint.setStyle(Paint.Style.STROKE);
        mCirclePaint.setStrokeJoin(Paint.Join.ROUND);
        mCirclePaint.setStrokeCap(Paint.Cap.ROUND);

        mCircleFillPaint = new Paint();
        mCircleFillPaint.setAntiAlias(true);
        mCircleFillPaint.setDither(true);
        mCircleFillPaint.setColor(mCircleFillColor);
        mCircleFillPaint.setStyle(Paint.Style.FILL);

        mCircleProgressPaint = new Paint();
        mCircleProgressPaint.setAntiAlias(true);
        mCircleProgressPaint.setDither(true);
        mCircleProgressPaint.setColor(mCircleProgressColor);
        mCircleProgressPaint.setStrokeWidth(mCircleStrokeWidth);
        mCircleProgressPaint.setStyle(Paint.Style.STROKE);
        mCircleProgressPaint.setStrokeJoin(Paint.Join.ROUND);
        mCircleProgressPaint.setStrokeCap(Paint.Cap.ROUND);

        mCircleProgressGlowPaint = new Paint();
        mCircleProgressGlowPaint.set(mCircleProgressPaint);
        mCircleProgressGlowPaint.setMaskFilter(new BlurMaskFilter((5f *
DPTOPX_SCALE), BlurMaskFilter.Blur.NORMAL));

        mPointerPaint = new Paint();
        mPointerPaint.setAntiAlias(true);
        mPointerPaint.setDither(true);
        mPointerPaint.setStyle(Paint.Style.FILL);
        mPointerPaint.setColor(mPointerColor);
        mPointerPaint.setStrokeWidth(mPointerRadius);

        mPointerHaloPaint = new Paint();
        mPointerHaloPaint.set(mPointerPaint);
        mPointerHaloPaint.setColor(mPointerHaloColor);
        mPointerHaloPaint.setAlpha(mPointerAlpha);
        mPointerHaloPaint.setStrokeWidth(mPointerRadius + mPointerHaloWidth);

        mPointerHaloBorderPaint = new Paint();
        mPointerHaloBorderPaint.set(mPointerPaint);
        mPointerHaloBorderPaint.setStrokeWidth(mPointerHaloBorderWidth);
        mPointerHaloBorderPaint.setStyle(Paint.Style.STROKE);

    }

    /**
     * Calculates the total degrees between mStartAngle and mEndAngle, and sets
mTotalCircleDegrees
     * to this value.
     */
    private void calculateTotalDegrees() {
        mTotalCircleDegrees = (360f - (mStartAngle - mEndAngle)) % 360f; //
```

```
Length of the entire circle/arc
        if (mTotalCircleDegrees <= 0f) {
            mTotalCircleDegrees = 360f;
        }
    }

    /**
     * Calculate the degrees that the progress represents. Also called the
sweep angle.
     * Sets mProgressDegrees to that value.
     */
    private void calculateProgressDegrees() {
        mProgressDegrees = mPointerPosition - mStartAngle; // Verified
        mProgressDegrees = (mProgressDegrees < 0 ? 360f + mProgressDegrees :
mProgressDegrees); // Verified
    }

    /**
     * Calculate the pointer position (and the end of the progress arc) in
degrees.
     * Sets mPointerPosition to that value.
     */
    private void calculatePointerAngle() {
        float progressPercent = ((float)mProgress / (float)mMax);
        mPointerPosition = (progressPercent * mTotalCircleDegrees) +
mStartAngle;
        mPointerPosition = mPointerPosition % 360f;
    }

    private void calculatePointerXYPosition() {
        PathMeasure pm = new PathMeasure(mCircleProgressPath, false);
        boolean returnValue = pm.getPosTan(pm.getLength(), mPointerPositionXY,
null);
        if (!returnValue) {
            pm = new PathMeasure(mCirclePath, false);
            returnValue = pm.getPosTan(0, mPointerPositionXY, null);
        }
    }

    /**
     * Initialize the {@code Path} objects with the appropriate values.
     */
    private void initPaths() {
        mCirclePath = new Path();
        mCirclePath.addArc(mCircleRectF, mStartAngle, mTotalCircleDegrees);

        mCircleProgressPath = new Path();
        mCircleProgressPath.addArc(mCircleRectF, mStartAngle,
mProgressDegrees);
    }

    /**
     * Initialize the {@code RectF} objects with the appropriate values.
     */
    private void initRects() {
        mCircleRectF.set(-mCircleWidth, -mCircleHeight, mCircleWidth,
mCircleHeight);
```

```
    }

    @Override
    protected void onDraw(Canvas canvas) {
        super.onDraw(canvas);

        canvas.translate(this.getWidth() / 2, this.getHeight() / 2);

        canvas.drawPath(mCirclePath, mCirclePaint);
        canvas.drawPath(mCircleProgressPath, mCircleProgressGlowPaint);
        canvas.drawPath(mCircleProgressPath, mCircleProgressPaint);

        canvas.drawPath(mCirclePath, mCircleFillPaint);

        canvas.drawCircle(mPointerPositionXY[0], mPointerPositionXY[1],
mPointerRadius + mPointerHaloWidth, mPointerHaloPaint);
        canvas.drawCircle(mPointerPositionXY[0], mPointerPositionXY[1],
mPointerRadius, mPointerPaint);
        if (mUserIsMovingPointer) {
            canvas.drawCircle(mPointerPositionXY[0], mPointerPositionXY[1],
mPointerRadius + mPointerHaloWidth + (mPointerHaloBorderWidth / 2f),
mPointerHaloBorderPaint);
        }
    }

    /**
     * Get the progress of the CircularSeekBar.
     * @return The progress of the CircularSeekBar.
     */
    public int getProgress() {
        int progress = Math.round((float)mMax * mProgressDegrees /
mTotalCircleDegrees);
        return progress;
    }

    /**
     * Set the progress of the CircularSeekBar.
     * If the progress is the same, then any listener will not receive a
onProgressChanged event.
     * @param progress The progress to set the CircularSeekBar to.
     */
    public void setProgress(int progress) {
        if (mProgress != progress) {
            mProgress = progress;
            if (mOnCircularSeekBarChangeListener != null) {
                mOnCircularSeekBarChangeListener.onProgressChanged(this,
progress, false);
            }

            recalculateAll();
            invalidate();
        }
    }

    private void setProgressBasedOnAngle(float angle) {
        mPointerPosition = angle;
        calculateProgressDegrees();
```

```
        mProgress = Math.round((float)mMax * mProgressDegrees /
mTotalCircleDegrees);
    }

    private void recalculateAll() {
        calculateTotalDegrees();
        calculatePointerAngle();
        calculateProgressDegrees();

        initRects();

        initPaths();

        calculatePointerXYPosition();
    }

    @Override
    protected void onMeasure(int widthMeasureSpec, int heightMeasureSpec) {
        int height = getDefaultSize(getSuggestedMinimumHeight(),
heightMeasureSpec);
        int width = getDefaultSize(getSuggestedMinimumWidth(),
widthMeasureSpec);
        if (mMaintainEqualCircle) {
            int min = Math.min(width, height);
            setMeasuredDimension(min, min);
        } else {
            setMeasuredDimension(width, height);
        }

        // Set the circle width and height based on the view for the moment
        mCircleHeight = (float)height / 2f - mCircleStrokeWidth -
mPointerRadius - (mPointerHaloBorderWidth * 1.5f);
        mCircleWidth = (float)width / 2f - mCircleStrokeWidth - mPointerRadius
- (mPointerHaloBorderWidth * 1.5f);

        // If it is not set to use custom
        if (mCustomRadii) {
            // Check to make sure the custom radii are not out of the view. If
they are, just use the view values
            if ((mCircleYRadius - mCircleStrokeWidth - mPointerRadius -
mPointerHaloBorderWidth) < mCircleHeight) {
                mCircleHeight = mCircleYRadius - mCircleStrokeWidth -
mPointerRadius - (mPointerHaloBorderWidth * 1.5f);
            }

            if ((mCircleXRadius - mCircleStrokeWidth - mPointerRadius -
mPointerHaloBorderWidth) < mCircleWidth) {
                mCircleWidth = mCircleXRadius - mCircleStrokeWidth -
mPointerRadius - (mPointerHaloBorderWidth * 1.5f);
            }
        }

        if (mMaintainEqualCircle) { // Applies regardless of how the values
were determined
            float min = Math.min(mCircleHeight, mCircleWidth);
            mCircleHeight = min;
            mCircleWidth = min;
```

```
        }

        recalculateAll();
    }

    @Override
    public boolean onTouchEvent(MotionEvent event) {
        // Convert coordinates to our internal coordinate system
        float x = event.getX() - getWidth() / 2;
        float y = event.getY() - getHeight() / 2;

        // Get the distance from the center of the circle in terms of x and y
        float distanceX = mCircleRectF.centerX() - x;
        float distanceY = mCircleRectF.centerY() - y;

        // Get the distance from the center of the circle in terms of a radius
        float touchEventRadius = (float) Math.sqrt((Math.pow(distanceX, 2) +
Math.pow(distanceY, 2)));

        float minimumTouchTarget = MIN_TOUCH_TARGET_DP * DPTOPX_SCALE; //
Convert minimum touch target into px
        float additionalRadius; // Either uses the minimumTouchTarget size or
larger if the ring/pointer is larger

        if (mCircleStrokeWidth < minimumTouchTarget) { // If the width is less
than the minimumTouchTarget, use the minimumTouchTarget
            additionalRadius = minimumTouchTarget / 2;
        }
        else {
            additionalRadius = mCircleStrokeWidth / 2; // Otherwise use the
width
        }
        float outerRadius = Math.max(mCircleHeight, mCircleWidth) +
additionalRadius; // Max outer radius of the circle, including the
minimumTouchTarget or wheel width
        float innerRadius = Math.min(mCircleHeight, mCircleWidth) -
additionalRadius; // Min inner radius of the circle, including the
minimumTouchTarget or wheel width

        if (mPointerRadius < (minimumTouchTarget / 2)) { // If the pointer
radius is less than the minimumTouchTarget, use the minimumTouchTarget
            additionalRadius = minimumTouchTarget / 2;
        }
        else {
            additionalRadius = mPointerRadius; // Otherwise use the radius
        }

        float touchAngle;
        touchAngle = (float) ((java.lang.Math.atan2(y, x) / Math.PI * 180) %
360); // Verified
        touchAngle = (touchAngle < 0 ? 360 + touchAngle : touchAngle); //
Verified

        cwDistanceFromStart = touchAngle - mStartAngle; // Verified
        cwDistanceFromStart = (cwDistanceFromStart < 0 ? 360f +
cwDistanceFromStart : cwDistanceFromStart); // Verified
        ccwDistanceFromStart = 360f - cwDistanceFromStart; // Verified
```

```
        cwDistanceFromEnd = touchAngle - mEndAngle; // Verified
        cwDistanceFromEnd = (cwDistanceFromEnd < 0 ? 360f + cwDistanceFromEnd :
cwDistanceFromEnd); // Verified
        ccwDistanceFromEnd = 360f - cwDistanceFromEnd; // Verified

        switch (event.getAction()) {
        case MotionEvent.ACTION_DOWN:
            // These are only used for ACTION_DOWN for handling if the pointer
was the part that was touched
            float pointerRadiusDegrees = (float) ((mPointerRadius * 180) /
(Math.PI * Math.max(mCircleHeight, mCircleWidth)));
            cwDistanceFromPointer = touchAngle - mPointerPosition;
            cwDistanceFromPointer = (cwDistanceFromPointer < 0 ? 360f +
cwDistanceFromPointer : cwDistanceFromPointer);
            ccwDistanceFromPointer = 360f - cwDistanceFromPointer;
            // This is for if the first touch is on the actual pointer.
            if (((touchEventRadius >= innerRadius) && (touchEventRadius <=
outerRadius)) && ( (cwDistanceFromPointer <= pointerRadiusDegrees) ||
(ccwDistanceFromPointer <= pointerRadiusDegrees)) ) {
                setProgressBasedOnAngle(mPointerPosition);
                lastCWDistanceFromStart = cwDistanceFromStart;
                mIsMovingCW = true;
                mPointerHaloPaint.setAlpha(mPointerAlphaOnTouch);
                recalculateAll();
                invalidate();
                if (mOnCircularSeekBarChangeListener != null) {
                    mOnCircularSeekBarChangeListener.onProgressChanged(this,
mProgress, true);
                }
                mUserIsMovingPointer = true;
                lockAtEnd = false;
                lockAtStart = false;
            } else if (cwDistanceFromStart > mTotalCircleDegrees) { // If the
user is touching outside of the start AND end
                mUserIsMovingPointer = false;
                return false;
            } else if ((touchEventRadius >= innerRadius) && (touchEventRadius
<= outerRadius)) { // If the user is touching near the circle
                setProgressBasedOnAngle(touchAngle);
                lastCWDistanceFromStart = cwDistanceFromStart;
                mIsMovingCW = true;
                mPointerHaloPaint.setAlpha(mPointerAlphaOnTouch);
                recalculateAll();
                invalidate();
                if (mOnCircularSeekBarChangeListener != null) {
                    mOnCircularSeekBarChangeListener.onProgressChanged(this,
mProgress, true);
                }
                mUserIsMovingPointer = true;
                lockAtEnd = false;
                lockAtStart = false;
            } else { // If the user is not touching near the circle
                mUserIsMovingPointer = false;
                return false;
            }
            break;
```

```
        case MotionEvent.ACTION_MOVE:
            if (mUserIsMovingPointer) {
                if (lastCWDistanceFromStart < cwDistanceFromStart) {
                    if ((cwDistanceFromStart - lastCWDistanceFromStart) > 180f
&& !mIsMovingCW) {
                        lockAtStart = true;
                        lockAtEnd = false;
                    } else {
                        mIsMovingCW = true;
                    }
                } else {
                    if ((lastCWDistanceFromStart - cwDistanceFromStart) > 180f
&& mIsMovingCW) {
                        lockAtEnd = true;
                        lockAtStart = false;
                    } else {
                        mIsMovingCW = false;
                    }
                }

                if (lockAtStart && mIsMovingCW) {
                    lockAtStart = false;
                }
                if (lockAtEnd && !mIsMovingCW) {
                    lockAtEnd = false;
                }
                if (lockAtStart && !mIsMovingCW && (ccwDistanceFromStart > 90))
{
                    lockAtStart = false;
                }
                if (lockAtEnd && mIsMovingCW && (cwDistanceFromEnd > 90)) {
                    lockAtEnd = false;
                }
                // Fix for passing the end of a semi-circle quickly
                if (!lockAtEnd && cwDistanceFromStart > mTotalCircleDegrees &&
mIsMovingCW && lastCWDistanceFromStart < mTotalCircleDegrees) {
                    lockAtEnd = true;
                }

                if (lockAtStart) {
                    // TODO: Add a check if mProgress is already 0, in which
case don't call the listener
                    mProgress = 0;
                    recalculateAll();
                    invalidate();
                    if (mOnCircularSeekBarChangeListener != null) {
                        mOnCircularSeekBarChangeListener.onProgressChanged(this,
mProgress, true);
                    }
                } else if (lockAtEnd) {
                    mProgress = mMax;
                    recalculateAll();
                    invalidate();
                    if (mOnCircularSeekBarChangeListener != null) {
                        mOnCircularSeekBarChangeListener.onProgressChanged(this,
mProgress, true);
                    }
```

```
                } else if ((mMoveOutsideCircle) || (touchEventRadius <=
outerRadius)) {
                    if (!(cwDistanceFromStart > mTotalCircleDegrees)) {
                        setProgressBasedOnAngle(touchAngle);
                    }
                    recalculateAll();
                    invalidate();
                    if (mOnCircularSeekBarChangeListener != null) {
                        mOnCircularSeekBarChangeListener.onProgressChanged(this,
mProgress, true);
                    }
                } else {
                    break;
                }

                lastCWDistanceFromStart = cwDistanceFromStart;
            } else {
                return false;
            }
            break;
        case MotionEvent.ACTION_UP:
            mPointerHaloPaint.setAlpha(mPointerAlpha);
            if (mUserIsMovingPointer) {
                mUserIsMovingPointer = false;
                invalidate();
                if (mOnCircularSeekBarChangeListener != null) {
                    mOnCircularSeekBarChangeListener.onProgressChanged(this,
mProgress, true);
                }
            } else {
                return false;
            }
            break;
        case MotionEvent.ACTION_CANCEL: // Used when the parent view intercepts
touches for things like scrolling
            mPointerHaloPaint.setAlpha(mPointerAlpha);
            mUserIsMovingPointer = false;
            invalidate();
            break;
        }

        if (event.getAction() == MotionEvent.ACTION_MOVE && getParent() !=
null) {
            getParent().requestDisallowInterceptTouchEvent(true);
        }

        return true;
    }

    private void init(AttributeSet attrs, int defStyle) {
        final TypedArray attrArray = getContext().obtainStyledAttributes(attrs,
R.styleable.CircularSeekBar, defStyle, 0);

        initAttributes(attrArray);

        attrArray.recycle();
```

```
        initPaints();
    }

    public CircularSeekBar(Context context) {
        super(context);
        init(null, 0);
    }

    public CircularSeekBar(Context context, AttributeSet attrs) {
        super(context, attrs);
        init(attrs, 0);
    }

    public CircularSeekBar(Context context, AttributeSet attrs, int defStyle) {
        super(context, attrs, defStyle);
        init(attrs, defStyle);
    }

    @Override
    protected Parcelable onSaveInstanceState() {
        Parcelable superState = super.onSaveInstanceState();

        Bundle state = new Bundle();
        state.putParcelable("PARENT", superState);
        state.putInt("MAX", mMax);
        state.putInt("PROGRESS", mProgress);
        state.putInt("mCircleColor", mCircleColor);
        state.putInt("mCircleProgressColor", mCircleProgressColor);
        state.putInt("mPointerColor", mPointerColor);
        state.putInt("mPointerHaloColor", mPointerHaloColor);
        state.putInt("mPointerAlpha", mPointerAlpha);
        state.putInt("mPointerAlphaOnTouch", mPointerAlphaOnTouch);

        return state;
    }

    @Override
    protected void onRestoreInstanceState(Parcelable state) {
        Bundle savedState = (Bundle) state;

        Parcelable superState = savedState.getParcelable("PARENT");
        super.onRestoreInstanceState(superState);

        mMax = savedState.getInt("MAX");
        mProgress = savedState.getInt("PROGRESS");
        mCircleColor = savedState.getInt("mCircleColor");
        mCircleProgressColor = savedState.getInt("mCircleProgressColor");
        mPointerColor = savedState.getInt("mPointerColor");
        mPointerHaloColor = savedState.getInt("mPointerHaloColor");
        mPointerAlpha = savedState.getInt("mPointerAlpha");
        mPointerAlphaOnTouch = savedState.getInt("mPointerAlphaOnTouch");

        initPaints();

        recalculateAll();
    }
```

```
    public void setOnSeekBarChangeListener(OnCircularSeekBarChangeListener l) {
        mOnCircularSeekBarChangeListener = l;
    }

    public interface OnCircularSeekBarChangeListener {

        public abstract void onProgressChanged(CircularSeekBar circularSeekBar,
int progress, boolean fromUser);
    }

    /**
     * Sets the circle color.
     * @param color the color of the circle
     */
    public void setCircleColor(int color) {
        mCircleColor = color;
        mCirclePaint.setColor(mCircleColor);
        invalidate();
    }

    /**
     * Gets the circle color.
     * @return An integer color value for the circle
     */
    public int getCircleColor() {
        return mCircleColor;
    }

    /**
     * Sets the circle progress color.
     * @param color the color of the circle progress
     */
    public void setCircleProgressColor(int color) {
        mCircleProgressColor = color;
        mCircleProgressPaint.setColor(mCircleProgressColor);
        invalidate();
    }

    /**
     * Gets the circle progress color.
     * @return An integer color value for the circle progress
     */
    public int getCircleProgressColor() {
        return mCircleProgressColor;
    }

    /**
     * Sets the pointer color.
     * @param color the color of the pointer
     */
    public void setPointerColor(int color) {
        mPointerColor = color;
        mPointerPaint.setColor(mPointerColor);
        invalidate();
    }
```

```java
/**
 * Gets the pointer color.
 * @return An integer color value for the pointer
 */
public int getPointerColor() {
    return mPointerColor;
}

/**
 * Sets the pointer halo color.
 * @param color the color of the pointer halo
 */
public void setPointerHaloColor(int color) {
    mPointerHaloColor = color;
    mPointerHaloPaint.setColor(mPointerHaloColor);
    invalidate();
}

/**
 * Gets the pointer halo color.
 * @return An integer color value for the pointer halo
 */
public int getPointerHaloColor() {
    return mPointerHaloColor;
}

/**
 * Sets the pointer alpha.
 * @param alpha the alpha of the pointer
 */
public void setPointerAlpha(int alpha) {
    if (alpha >=0 && alpha <= 255) {
        mPointerAlpha = alpha;
        mPointerHaloPaint.setAlpha(mPointerAlpha);
        invalidate();
    }
}

/**
 * Gets the pointer alpha value.
 * @return An integer alpha value for the pointer (0..255)
 */
public int getPointerAlpha() {
    return mPointerAlpha;
}

/**
 * Sets the pointer alpha when touched.
 * @param alpha the alpha of the pointer (0..255) when touched
 */
public void setPointerAlphaOnTouch(int alpha) {
    if (alpha >=0 && alpha <= 255) {
        mPointerAlphaOnTouch = alpha;
    }
}

/**
```

```
     * Gets the pointer alpha value when touched.
     * @return An integer alpha value for the pointer (0..255) when touched
     */
    public int getPointerAlphaOnTouch() {
        return mPointerAlphaOnTouch;
    }

    /**
     * Sets the circle fill color.
     * @param color the color of the circle fill
     */
    public void setCircleFillColor(int color) {
        mCircleFillColor = color;
        mCircleFillPaint.setColor(mCircleFillColor);
        invalidate();
    }

    /**
     * Gets the circle fill color.
     * @return An integer color value for the circle fill
     */
    public int getCircleFillColor() {
        return mCircleFillColor;
    }

    /**
     * Set the max of the CircularSeekBar.
     * If the new max is less than the current progress, then the progress will
be set to zero.
     * If the progress is changed as a result, then any listener will receive a
onProgressChanged event.
     * @param max The new max for the CircularSeekBar.
     */
    public void setMax(int max) {
        if (!(max <= 0)) { // Check to make sure it's greater than zero
            if (max <= mProgress) {
                mProgress = 0; // If the new max is less than current progress,
set progress to zero
                if (mOnCircularSeekBarChangeListener != null) {
                    mOnCircularSeekBarChangeListener.onProgressChanged(this,
mProgress, false);
                }
            }
            mMax = max;

            recalculateAll();
            invalidate();
        }
    }

    /**
     * Get the current max of the CircularSeekBar.
     * @return Synchronized integer value of the max.
     */
    public synchronized int getMax() {
        return mMax;
    }
```

```
}
```

```java
package com.gamification_research;

import android.content.ContentValues;

/**
 * @author Jonathan Cassar
 * @Class Name: Data
 * @Description: Provides methods to read and write in the SQLlite database.
 */
public class Data
{
    //DATABASE NAME
    private static final String DATABASE_NAME = "db_yprototaype";

    //DATABASE TABLE
    public static final String TBL_USERS = "tbl_users";

    //DATABASE TABLE HEADERS
    public static final String KEY_NAME = "user_name";
    public static final String KEY_DAILY_PTS = "daily_pts";
    public static final String KEY_START_DATE = "start_date";
    public static final String KEY_LEVEL = "level";
    public static final String KEY_DISSERTATION_LENGTH = "dissertation_length";
    public static final String KEY_STATE = "state";
    public static final String KEY_INTRO_STATE = "intro_state";
    public static final String KEY_LITREVIEW_STATE = "litreview_state";
    public static final String KEY_METOD_STATE = "metod_state";
    public static final String KEY_RESUL_STATE = "result_state";
    public static final String KEY_CONCL_STATE = "concl_state";
    public static final String KEY_EVALUATION_STATE = "evaluation_state";
    public static final String KEY_TOTAL_POINTS = "total_pts";


    //DATABASE METADATA - VERSION
    private static final int DATABASE_VERSION = 1;

    //VARIABLES
    private dbHelper ypr_helper;
    private final Context ypr_context;
    private SQLiteDatabase ypr_my_database;


    //[1]CONSTRUCTOR
    /**
     * @Method name: Data
     * @param Context
     * @return null
     * @Description: Class constructor.
     */
    public Data(Context c)
    {
        this.ypr_context = c;
    }

    //[2]DATABASE HELPER
    /**
     * @Method name: dbHelper
```

```
     * @param null
     * @return null
     * @Extends SQLiteOpenHelper
     * @Description: Inner class which creates a database helper.
     */
    public static class dbHelper extends SQLiteOpenHelper
    {
        //CONSTRUCTOR - dbHelper class
        public dbHelper(Context context)
        {
            super(context, DATABASE_NAME, null, DATABASE_VERSION);
        }

        //CREATE SQL TABLE
        /**
         * @Method name: onCreate
         * @param Datatbase
         * @return null
         * @Description: TSQL to create the database
         */
        public void onCreate(SQLiteDatabase db)
        {
            db.execSQL("CREATE TABLE "+TBL_USERS +" (" +
                    KEY_NAME + " TEXT PRIMARY KEY NOT NULL, "+
                    KEY_DAILY_PTS + " INTEGER, " +
                    KEY_START_DATE + " INTEGER, " +
                    KEY_LEVEL + " INTEGER, " +
                    KEY_DISSERTATION_LENGTH + " INTEGER, " +
                    KEY_STATE + " TEXT, " +
                    KEY_INTRO_STATE + " TEXT, " +
                    KEY_LITREVIEW_STATE + " TEXT, " +
                    KEY_METOD_STATE  + " TEXT, " +
                    KEY_RESUL_STATE + " TEXT, " +
                    KEY_CONCL_STATE + " TEXT, " +
                    KEY_EVALUATION_STATE + " TEXT, "+
                    KEY_TOTAL_POINTS + " INTEGER);"
                    );
        }

        //UPGRATE SQL TABLE
        public void onUpgrade(SQLiteDatabase db, int oldVersion, int
newVersion)
        {
            db.execSQL("DROP TABLE IF EXIST "+ TBL_USERS);
            onCreate(db);
        }
    }

    //[3] OPEN DATABASE
    /**
     * @Method name: open
     * @param null
     * @return null
     * @Description: Open the database.
     */
    public Data open()
    {
```

```
        ypr_helper = new dbHelper(ypr_context);
        ypr_my_database =  ypr_helper.getWritableDatabase();
        return this;
    }

    //[4] CLOSE DATABASE
    /**
     * @Method name: close
     * @param null
     * @return null
     * @Description: Closes the database.
     */
    public void close()
    {
        ypr_helper.close();
    }

    //[5] READ - USER
    /**
     * @Method name: read_user
     * @param String
     * @return String
     * @Description: Reads the user from the database.
     */
    public String read_user(String user)
    {
        String loged_in ="";

        String [] col  =  new String[]{KEY_NAME};
        Cursor c  = ypr_my_database.query(TBL_USERS,col, null, null, null,
null, null);

        int iUser = c.getColumnIndex(KEY_NAME);

        for(c.moveToFirst(); !c.isAfterLast();c.moveToNext())
        {
            if(c.getString(iUser).equals(user))
            {
                loged_in = c.getString(iUser);
            }
        }
        return loged_in;
    }

    //[6] READ - TOTAL POINTS
    /**
     * @Method name: read_total_pts
     * @param String
     * @return String
     * @Description: Reads the total points.
     */
    public String read_total_pts(String user)
    {
        String totalpts="";

        String [] col  =  new String[]{KEY_NAME,KEY_TOTAL_POINTS};
        Cursor c  = ypr_my_database.query(TBL_USERS,col, null, null, null,
```

```
null, null);

        int iUser = c.getColumnIndex(KEY_NAME);
        int iTotalPts = c.getColumnIndex(KEY_TOTAL_POINTS);

        for(c.moveToFirst(); !c.isAfterLast();c.moveToNext())
        {
            if(c.getString(iUser).equals(user))
            {
                totalpts = c.getString(iTotalPts);
            }
        }
        return totalpts;
    }

    //[7] READ - DAILY POINTS
    /**
     * @Method name: read_daily_pts
     * @param String
     * @return String
     * @Description: Reads the daily points.
     */
    public String read_daily_pts(String user)
    {
        String dailypts="";

        String [] col  =  new String[]{KEY_NAME, KEY_DAILY_PTS};
        Cursor c  = ypr_my_database.query(TBL_USERS,col, null, null, null,
null, null);

        int iUser = c.getColumnIndex(KEY_NAME);
        int iDailyPts = c.getColumnIndex(KEY_DAILY_PTS);

        for(c.moveToFirst(); !c.isAfterLast();c.moveToNext())
        {
            if(c.getString(iUser).equals(user))
            {
                dailypts = c.getString(iDailyPts);
            }
        }
        return dailypts;
    }

    //[8] WRITE - USER
    /**
     * @Method name: write_user
     * @param String
     * @return null
     * @Description: Reads a user to the database.
     */
    public void write_user(String name)
    {
        ContentValues cv =  new ContentValues();
        cv.put(KEY_NAME, name);
        ypr_my_database.insert(TBL_USERS, null, cv);
    }
```

```
    //[9] WRITE - DAILY POINTS TO DATABASE
    /**
     * @Method name: write_daily
     * @param String
     * @param String
     * @return null
     * @Description: Write daily points to database.
     */
    public void write_daily(String user, String pt_days)
    {
        ContentValues cv =  new ContentValues();
        String where = "user_name=?";
        String[]where_args = new String[]{user};
        cv.put(KEY_DAILY_PTS,pt_days);
        ypr_my_database.update(TBL_USERS,cv,where,where_args);
    }

    //[10] WRITE - TOTAL POINTS TO DATABASES
    /**
     * @Method name: write_total
     * @param String
     * @param String
     * @return null
     * @Description: Write total points to database.
     */
    public void write_total(String user, String pt_total)
    {
        ContentValues cv =  new ContentValues();
        String where = "user_name=?";
        String[]where_args = new String[]{user};
        cv.put(KEY_TOTAL_POINTS,pt_total);
        ypr_my_database.update(TBL_USERS,cv,where,where_args);
    }

    //[11] DELETE ALL DATA
    /**
     * @Method name: delete_alla
     * @param null
     * @return null
     * @Description: Deletes all data from database
     */
    public void delete_all()
    {
        ypr_my_database.delete(TBL_USERS, null, null);
    }

    //[12] DELETE DAILY POINTS
    /**
     * @Method name: reset_daily
     * @param null
     * @return null
     * @Description: reset all daily data from database by write all fields
with 0
     */
    public void reset_daily()
    {
       ContentValues cv =  new ContentValues();
```

```
        cv.put(KEY_DAILY_PTS, "0");
        ypr_my_database.update(TBL_USERS, cv, null, null);
    }

    //[13] READ - LEVEL
    /**
     * @Method name: read_level
     * @param String
     * @return String
     * @Description: Reads the current level reached by the user from the
database.
     */
    public String read_level(String user)
    {
        String level="";

        String [] col  =  new String[]{KEY_NAME, KEY_LEVEL};
        Cursor c  = ypr_my_database.query(TBL_USERS,col, null, null, null,
null, null);

        int iUser = c.getColumnIndex(KEY_NAME);
        int iDailyPts = c.getColumnIndex(KEY_LEVEL);

        for(c.moveToFirst(); !c.isAfterLast();c.moveToNext())
        {
            if(c.getString(iUser).equals(user))
            {
                level = c.getString(iDailyPts);
            }
        }
        return level;

    }

    //[14] WRITE - LEVEL
    /**
     * @Method name: write_level
     * @param String
     * @param String
     * @return null
     * @Description: Write level reached by user to database.
     */
    public void write_level(String user, String level)
    {
        ContentValues cv =  new ContentValues();
        String where = "user_name=?";
        String[]where_args = new String[]{user};
        cv.put(KEY_LEVEL,level);
        ypr_my_database.update(TBL_USERS,cv,where,where_args);
    }

    //[15] WRITE - STATE
    /**
     * @Method name: write_total
     * @param String
     * @param String
     * @return null
```

```
     * @Description: Write total points to database.
     */
    public void btn_write_state(String user, String state)
    {
        ContentValues cv =  new ContentValues();
        String where = "user_name=?";
        String[]where_args = new String[]{user};
        cv.put(KEY_STATE,state);
        ypr_my_database.update(TBL_USERS,cv,where,where_args);
    }

    //[16] READ - STATE
    /**
     * @Method name: btn_read_state
     * @param String
     * @return null
     * @Description: Read the state of the button.
     */
    public String btn_read_state(String user)
    {
        String state="";

        String [] col  =  new String[]{KEY_NAME, KEY_STATE};
        Cursor c  = ypr_my_database.query(TBL_USERS,col, null, null, null,
null, null);

        int iUser = c.getColumnIndex(KEY_NAME);
        int iState = c.getColumnIndex(KEY_STATE);

        for(c.moveToFirst(); !c.isAfterLast();c.moveToNext())
        {
            if(c.getString(iUser).equals(user))
            {
                state = c.getString(iState);
            }
        }
        return state;
    }

    //[17] READ - INTRODUCTION STATE
    /**
     * @Method name: read_intro_state
     * @param String
     * @return String
     * @Description: Read the state of the seek Bar (introduction).
     */
    public String read_intro_state(String user)
    {
        String progress_state="";

        String [] col  =  new String[]{KEY_NAME, KEY_INTRO_STATE};
        Cursor c  = ypr_my_database.query(TBL_USERS,col, null, null, null,
null, null);

        int iUser = c.getColumnIndex(KEY_NAME);
        int iProgressState = c.getColumnIndex(KEY_INTRO_STATE);
```

```
        for(c.moveToFirst(); !c.isAfterLast();c.moveToNext())
        {
            if(c.getString(iUser).equals(user))
            {
                progress_state = c.getString(iProgressState);
            }
        }
        return progress_state;
    }

    //[18] WRITE - PROGRESS STATE
    /**
     * @Method name: write_intro_state
     * @param String
     * @param String
     * @Description: Write the state of the seek Bar (introduction).
     */
    public void write_intro_state(String user, String state)
    {
        ContentValues cv =  new ContentValues();
        String where = "user_name=?";
        String[]where_args = new String[]{user};
        cv.put(KEY_INTRO_STATE,state);
        ypr_my_database.update(TBL_USERS,cv,where,where_args);
    }

    //[19] READ - LITERATURE REVIEW STATE
    /**
     * @Method name: read_litreview_state
     * @param String
     * @return String
     * @Description: Read the state of the seek Bar (Lit Review).
     */
    public String read_litreview_state (String user)
    {
        String progress_state="";

        String [] col  =  new String[]{KEY_NAME, KEY_LITREVIEW_STATE};
        Cursor c  = ypr_my_database.query(TBL_USERS,col, null, null, null,
null, null);

        int iUser = c.getColumnIndex(KEY_NAME);
        int iProgressState = c.getColumnIndex(KEY_LITREVIEW_STATE);

        for(c.moveToFirst(); !c.isAfterLast();c.moveToNext())
        {
            if(c.getString(iUser).equals(user))
            {
                progress_state = c.getString(iProgressState);
            }
        }
        return progress_state;
    }

    //[20] WRITE - LIT REVIEW STATE
    /**
     * @Method name: write_literview_state
```

```
     * @param String
     * @param String
     * @Description: Write the state of the seek Bar (Literature review).
     */
    public void write_literview_state (String user, String state)
    {
        ContentValues cv =  new ContentValues();
        String where = "user_name=?";
        String[]where_args = new String[]{user};
        cv.put(KEY_LITREVIEW_STATE,state);
        ypr_my_database.update(TBL_USERS,cv,where,where_args);
    }

    //[21] READ - METOD STATE
    /**
     * @Method name: read_metod_state
     * @param String
     * @return String
     * @Description: Read the state of the seek Bar (Methodology).
     */
    public String read_metod_state(String user)
    {
        String progress_state="";

        String [] col  =  new String[]{KEY_NAME, KEY_METOD_STATE};
        Cursor c  = ypr_my_database.query(TBL_USERS,col, null, null, null,
null, null);

        int iUser = c.getColumnIndex(KEY_NAME);
        int iProgressState = c.getColumnIndex(KEY_METOD_STATE);

        for(c.moveToFirst(); !c.isAfterLast();c.moveToNext())
        {
            if(c.getString(iUser).equals(user))
            {
                progress_state = c.getString(iProgressState);
            }
        }
        return progress_state;
    }

    //[22] WRITE - METOD STATE
    /**
     * @Method name: write_metod_state
     * @param String
     * @param String
     * @Description: Write the state of the seek Bar (Methodology).
     */
    public void write_metod_state (String user, String state)
    {
        ContentValues cv =  new ContentValues();
        String where = "user_name=?";
        String[]where_args = new String[]{user};
        cv.put(KEY_METOD_STATE,state);
        ypr_my_database.update(TBL_USERS,cv,where,where_args);
    }
```

```
    //[23] READ - RESULT STATE
    /**
     * @Method name: read_result_state
     * @param String
     * @return String
     * @Description: Read the state of the seek Bar (Result).
     */
    public String read_result_state(String user)
    {
        String progress_state="";

        String [] col  =  new String[]{KEY_NAME, KEY_RESUL_STATE};
        Cursor c  = ypr_my_database.query(TBL_USERS,col, null, null, null,
null, null);

        int iUser = c.getColumnIndex(KEY_NAME);
        int iProgressState = c.getColumnIndex(KEY_RESUL_STATE);

        for(c.moveToFirst(); !c.isAfterLast();c.moveToNext())
        {
            if(c.getString(iUser).equals(user))
            {
                progress_state = c.getString(iProgressState);
            }
        }
        return progress_state;
    }

    //[24] WRITE - RESULT STATE
    /**
     * @Method name: write_res_state
     * @param String
     * @param String
     * @Description: Write the state of the seek Bar (Results).
     */
    public void write_res_state (String user, String state)
    {
        ContentValues cv =  new ContentValues();
        String where = "user_name=?";
        String[]where_args = new String[]{user};
        cv.put(KEY_RESUL_STATE,state);
        ypr_my_database.update(TBL_USERS,cv,where,where_args);
    }

    //[26] READ - CONCLUSION STATE
    /**
     * @Method name: read_concl_state
     * @param String
     * @return String
     * @Description: Read the state of the seek Bar (Conclusion).
     */
    public String read_concl_state(String user)
    {
        String progress_state="";

        String [] col  =  new String[]{KEY_NAME, KEY_CONCL_STATE};
        Cursor c  = ypr_my_database.query(TBL_USERS,col, null, null, null,
```

```
null, null);

        int iUser = c.getColumnIndex(KEY_NAME);
        int iProgressState = c.getColumnIndex(KEY_CONCL_STATE);

        for(c.moveToFirst(); !c.isAfterLast();c.moveToNext())
        {
            if(c.getString(iUser).equals(user))
            {
                progress_state = c.getString(iProgressState);
            }
        }
        return progress_state;
    }

    //[27] WRITE - CONCLUSION STATE
    /**
     * @Method name: write_conc_state
     * @param String
     * @param String
     * @Description: Write the state of the seek Bar (Conclusion).
     */
    public void write_conc_state(String user, String state)
    {
        ContentValues cv =  new ContentValues();
        String where = "user_name=?";
        String[]where_args = new String[]{user};
        cv.put(KEY_CONCL_STATE,state);
        ypr_my_database.update(TBL_USERS,cv,where,where_args);
    }

    //[28] WRITE - EVALUATION STATE
    /**
     * @Method name: write_evaluation_state
     * @param String
     * @param String
     * @Description: Write the evaluation of the seek Bar
     */
    public void write_evaluation_state(String user, String state)
    {
        ContentValues cv =  new ContentValues();
        String where = "user_name=?";
        String[]where_args = new String[]{user};
        cv.put(KEY_EVALUATION_STATE,state);
        ypr_my_database.update(TBL_USERS,cv,where,where_args);
    }

    //[29] READ - EVALUATION STATE
    /**
     * @Method name: read_evaluation_state
     * @param String
     * @return String
     * @Description: Read the evaluation of the seek Bar
     */
    public String read_evaluation_state(String user)
    {
        String evaluation_state="";
```

```
        String [] col  =  new String[]{KEY_NAME, KEY_EVALUATION_STATE};
        Cursor c  = ypr_my_database.query(TBL_USERS,col, null, null, null,
null, null);

        int iUser = c.getColumnIndex(KEY_NAME);
        int iProgressState = c.getColumnIndex(KEY_EVALUATION_STATE);

        for(c.moveToFirst(); !c.isAfterLast();c.moveToNext())
        {
            if(c.getString(iUser).equals(user))
            {
                evaluation_state = c.getString(iProgressState);
            }
        }
        return evaluation_state;
    }

}
```

```java
package com.gamification_research;
import android.app.Activity;


/**
 * @author Jonathan Cassar
 * @Class Name: Evaluation_screen
 * @Description: Provides methods for the Evaluations screen
 */
public class Evaluation_screen extends Activity implements
OnCircularSeekBarChangeListener, OnClickListener
{
    //VARIABLES
    CircularSeekBar sb;
    Data d;
    ImageView btn_help;
    TextView sample, comp_surveys, message,pts_day;
    SharedPreferences getData;
    String user, completed, total_points, daily_points;
    int surveys,intro,stop, tot_pts, day_pts, progress, counter;
    int base_pts = 2;
    double percent;


    //[1] ONCREATE
    /**
     * @Method name: onCreate
     * @param Bundle
     * @return null
     * @Description: Class constructor.
     */
    protected void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.evaluation_screen);
        getData =
PreferenceManager.getDefaultSharedPreferences(getBaseContext());

        //References
        sb = (CircularSeekBar) findViewById(R.id.circularSeekBar1);
        sample= (TextView)findViewById(R.id.txt_svr_evaluation);
        comp_surveys= (TextView)findViewById(R.id.txt_prog_evaluation);
        message =  (TextView)findViewById(R.id.txt_hints_evaluation);
        pts_day = (TextView)findViewById(R.id.txt_tag_pts_eval);
        btn_help = (ImageView)findViewById(R.id.btn_help_eval);

        //Listeners
        sb.setOnSeekBarChangeListener(this);
        btn_help.setOnClickListener(this);

        //Shared Preferences
        user = getData.getString("user_name", "user");

        //Methods
        get_data();
        set_data();
        popup();
```

```
    }

    //[2] GET DATA
    /**
     * @Method name: get_data
     * @param  null
     * @return null
     * @Description: Get data from various sources to be used within the class.
     */
    public void get_data()
    {
      getData =
PreferenceManager.getDefaultSharedPreferences(getBaseContext());
      d= new Data(this);
      d.open();
      user = d.read_user(getData.getString("user_name", "User"));
      total_points = d.read_total_pts(user);
      daily_points = d.read_daily_pts(user);
      try{progress = Integer.parseInt(d.read_evaluation_state(user));
          sb.setProgress(progress);}catch(Exception e){} //read data from
database
      d.close();
    }

    //[3] SET DATA
    /**
     * @Method name: set_data
     * @param null
     * @return null
     * @Description: Set the variables with the retrieved data before.
     */
    public void set_data()
    {
        //set number of surveys to be done
        getData =
PreferenceManager.getDefaultSharedPreferences(getBaseContext());
        sample.setText(getData.getString("surveys", "s"));

        //set maximum progress bar
        surveys  = Integer.parseInt(getData.getString("surveys", "s"));
        sb.setMax(surveys);

        //Caculate percentage of completeness
        percent = (float)progress/(float)surveys;

        //Set initial hint
        message(percent);

        //Set daily points earned from surveys only
        pts_day.setText("0");

        //set Color
        setColor();

        //set progress text
        try{
            d.open();
```

```
            comp_surveys.setText(Integer.toString(sb.getProgress()));
            d.close();
        }catch(Exception e){}

    }

    //[4] ON PROGRESS
    /**
     * @Method name: onProgressChanged
     * @param Circular seekbar, Integer, boolean
     * @return null
     * @Description: On seek bar progress, update text and save state to
database.
     */
    public void onProgressChanged(CircularSeekBar sb, int progress, boolean
fromUser)
    {
        switch(sb.getId())
        {
        case R.id.circularSeekBar1:
            intro = progress;
            setColor();    //set col
            comp_surveys.setText(""+progress);
            counter+=base_pts; //needed to keep record of daily points earned
only from surveys
            pts_day.setText(""+counter);
            save_state(sb.getProgress());
            earn_points();
            break;
        }
    }

    //[5] SAVE STATE
    /**
     * @Method name: save_state
     * @param Integer
     * @return null
     * @Description: Save seekbar state to database.
     */
    public void save_state(int state)
    {
        try{
            d.open();
            d.write_evaluation_state(user,Integer.toString(state));
            d.close();
        }catch(Exception e){}
    }

    //[6] ON STOP
    /**
     * @Method name: onStop
     * @param null
     * @return null
     * @Description: On activity stop save the seekbar state to the database.
     */
    public void onStop()
    {
```

```
    super.onStop();
    earn_points();
    update_points();
    save_points();
    save_state(sb.getProgress());
}

//[7] POPUP
/**
 * Method name: popup
 * @param null
 * @return null
 * @Description: Pop a message to instruct user how to operate.
 */
public void popup()
{
        Context c = getApplicationContext();
        CharSequence text = " Rotate wheel to mark survey progress.";
        int duration = Toast.LENGTH_SHORT;

        Toast t= Toast.makeText(c, text, duration);
        t.setGravity(Gravity.CENTER|Gravity.CENTER, 0, 0);
        t.show();
}

//[8] EARN POINTS
/**
 * Method name: earn_points
 * @param null
 * @return null
 * @Description: earn points with every survey completed
 */
public void earn_points()
{
    day_pts+=base_pts;
    tot_pts+=base_pts;
}

//[9] UPDATE POINTS
/**
 * Method name: update_points
 * @param null
 * @return null
 * @Description: update points
 */
public void update_points()
{
    int total = 0;
    int day = 0;

    //convert retrieved data from database to string
    try{total = Integer.parseInt(total_points);
    day = Integer.parseInt(daily_points);
    }catch(Exception e){};

    //Add points
    day+=day_pts;
```

```
        total+=tot_pts;

        //convert back to string
        try{
            daily_points= Integer.toString(day);
            total_points = Integer.toString(total);
        }catch(Exception e){}
    }

    //[10] SAVE POINTS
    /**
     * Method name: save_points
     * @param null
     * @return null
     * @Description: save points to database
     */
    public void save_points()
    {
        d = new Data(this);
        d.open();
        d.write_daily(user,daily_points);
        d.write_total(user,total_points);
        d.close();
    }

    //[11] SET HINT
        /**
         * Method name: set hint
         * @param null
         * @return null
         * @Description: Dsiplays hint messages to the user
         */
    public void message (double percent)
    {
        if(percent>0.0  & percent<=0.2){message.setText(" Try and ask your
peers!");}
        if(percent>=0.2 & percent<=0.4){message.setText(" Why not include your
friends?");}
        if(percent>=0.4 & percent<=0.6){message.setText(" Time to do some
chasing");}
        if(percent>=0.6 & percent<=0.8){message.setText(" Any luck with the
phone?");}
        if(percent>=0.8){message.setText(" Have you considered collecting data
from social media?");}
    }

    //[12] SET COLOR
        /**
         * Method name: set_color
         * @param null
         * @return null
         * @Description: sets the color of the progress wheel
         */
    public void setColor()
    {
        percent = (float)intro/(float)surveys;
        if(percent>0.0  &
```

```
percent<0.2){comp_surveys.setTextColor(Color.parseColor("#330000"));sb.setCircl
eProgressColor(Color.parseColor("#330000"));}
        if(percent>=0.2 &
percent<0.4){comp_surveys.setTextColor(Color.parseColor("#FF0000"));sb.setCircl
eProgressColor(Color.parseColor("#FF0000"));}
        if(percent>=0.4 &
percent<0.6){comp_surveys.setTextColor(Color.parseColor("#FF6600"));sb.setCircl
eProgressColor(Color.parseColor("#FF6600"));}
        if(percent>=0.6 &
percent<0.8){comp_surveys.setTextColor(Color.parseColor("#FFEE00"));sb.setCircl
eProgressColor(Color.parseColor("#FFEE00"));}
        if(percent>=0.8){comp_surveys.setTextColor(Color.parseColor("#33CC00"))
;sb.setCircleProgressColor(Color.parseColor("#33CC00"));}
    }

    //[13] SET ON CLICK METHOD
    /**
     * Method name: onClick
     * @param null
     * @return null
     * @Description: set on click methods
     */
    public void onClick(View arg0)
    {
        switch(arg0.getId())
        {
        case R.id.btn_help_eval:
            try{message(arg0);}catch(Exception e){e.printStackTrace();};
            break;
        }

    }

    //[14] SET HELP
    /**
     * Method name: message
     * @param View
     * @return null
     * @Description: Set the help
     */
    public void message(View view)
    {
        AlertDialog d = new AlertDialog.Builder(this).create();
        d.setTitle("Evaluation Progress");
        d.setMessage(
                    "Rotate the Wheel clockwise to mark the survey progress."+
                    " In the process, earn two points per each survey
completed."+
                    " Don't forget to check the hints at the bottom of the
screen");

        d.setButton("OK", new DialogInterface.OnClickListener()
        {
            public void onClick(DialogInterface dialog, int which)
            {
                //REMAIN AT LOGIN SCREEN
            }
```

```
        });
        d.setIcon(R.drawable.help);
        d.show();
    }
}
```

```java
package com.gamification_research;

import java.text.SimpleDateFormat;

/**
 * @author Jonathan Cassar
 * @Class Name: Main_screen
 * @Description: The main activity of the application which is loaded first
when the user clicks the icon
 * @TargetApi(Build.VERSION_CODES.HONEYCOMB)
 */
public class Main_screen extends Activity implements OnClickListener
{
    //VARIABLES
    Button btn_papers, btn_progress, btn_evaluation, btn_calendar, btn_next;
    ImageSwitcher is;
    TextView usr, tot_pts, dly_pts, elpsd_dys, lvl, perCent;
    ProgressBar pb;
    Data d;
    SharedPreferences getData;

    String user, total_points, daily_points, elapsed_days, per_cent;
    int lvl_counter = 0;
    int index = -1; //Image slider - keep current index of image id array
    int array_length = 6;


    //[1] ON CREATE METHOD
    /**
     * @Method name: onCreate
     * @param Bundle savedInstances
     * @return null
     * @Description: On Create method.
     */
    protected void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main_screen);

        //Get Reference
        btn_papers = (Button)findViewById(R.id.btn_papers_main);
        btn_progress = (Button)findViewById(R.id.btn_progress_main);
        btn_evaluation = (Button)findViewById(R.id.btn_evaluation_main);
        btn_calendar = (Button)findViewById(R.id.btn_calendar_main);
        btn_next = (Button)findViewById(R.id.btn_next_main);
        is = (ImageSwitcher)findViewById(R.id.imgSwitcher_main);
        usr = (TextView)findViewById(R.id.txt_user_main);
        tot_pts = (TextView)findViewById(R.id.txt_tpts_main);
        dly_pts = (TextView)findViewById(R.id.txt_dpts_main);
        elpsd_dys = (TextView)findViewById(R.id.txt_eps_main);
        lvl =  (TextView)findViewById(R.id.txt_lvl_main);
        pb = (ProgressBar)findViewById(R.id.prog_bar_progress_screen);
        perCent = (TextView)findViewById(R.id.txt_percebt_main);

        //Listeners
        btn_papers.setOnClickListener(this);
        btn_progress.setOnClickListener(this);
```

```
        btn_evaluation.setOnClickListener(this);
        btn_calendar.setOnClickListener(this);
        btn_next.setOnClickListener(this);

        //Methods
        get_data();
        set_data();
        determine_level();
        image_slider();
        popup();
        progress_bar();

        //reset data
        reset_data();
        reset_daily_pts();
        notification();
    }

    //[2] MENU INFLATOR
    /**
     * @Method name: onCreateOptionsMenue
     * @param menue
     * @return boolean
     * @Description: Create a menu for the activity.
     */
    public boolean onCreateOptionsMenu(android.view.Menu menu)
    {
        super.onCreateOptionsMenu(menu);
        MenuInflater mi = getMenuInflater();
        mi.inflate(R.menu.main_screen,menu);
        progress_bar();
        return true;
    }

    //[3] MENU SELECTED OPTIONS
    /**
     * @Method name: onOptionsItemSelect
     * @param Meneu Item
     * @return boolean
     * @Description: Loads the Menu.
     */
    public boolean onOptionsItemSelected(MenuItem item)
    {
        switch(item.getItemId())
        {
        case R.id.action_settings:
            startActivity(new Intent(this,Settings_screen.class));
        }
        return false;
    }

    //[4] CLICK HANDLER
    /**
     * @Method name: onClick
     * @param View
     * @return null
     * @Description: Handles the buttons click events.
```

```
     */
    public void onClick(View arg0)
    {
        switch(arg0.getId())
        {
        case R.id.btn_papers_main:
            save_data_papers();
            startActivity(new Intent(this,Papers_screen.class));
            break;
        case R.id.btn_progress_main:
            startActivity(new Intent(this,Progress_screen.class));
            break;
        case R.id.btn_evaluation_main:
            startActivity(new Intent(this,Evaluation_screen.class));
            break;
        case R.id.btn_calendar_main:
            startActivity(new Intent(this,Calendar_screen.class));
            break;
        case R.id.btn_next_main:
            index++;
            lvl_counter = index+1;
            try {lvl.setText(""+lvl_counter);}catch(Exception e){}

            if(index==array_length) //reset when index reaches max
                index=0;
            lvl_counter=0;
            is.setImageResource(determine_badge(index));
            break;
        }
    }

    //[5] IMAGE SLIDER
    /**
     * Method name: image_slider
     * @param null
     * @return null
     * @Description: Create an image slider through the use of Image View and
Animation methods. The
     * slider can be loaded with images and slide from left to right.
     */
    public void image_slider()
    {
        is.setFactory(new ViewFactory()
        {
            public View makeView()
            {
                // Create a new ImageView set it's properties
                ImageView iv = new ImageView(getApplicationContext());
                iv.setScaleType(ImageView.ScaleType.FIT_CENTER);
                iv.setLayoutParams(new
ImageSwitcher.LayoutParams(LayoutParams.FILL_PARENT,LayoutParams.FILL_PARENT));
                return iv;
            }
        });

        //Declare animations
        Animation in =
```

```
AnimationUtils.loadAnimation(this,android.R.anim.slide_in_left);
        Animation out =
AnimationUtils.loadAnimation(this,android.R.anim.slide_out_right);

        //Set Animations
        is.setInAnimation(in);
        is.setOutAnimation(out);

    }

    //[6] POPUP
    /**
     * Method name: popup
     * @param null
     * @return null
     * @Description: Pop a message when prompted. The message can be set to
have a short or long duration.
     */
    public void popup()
    {
        if(user=="User"){
            Context c = getApplicationContext();
            CharSequence text = "Go to settings and enter details.";
            int duration = Toast.LENGTH_SHORT;

            Toast t= Toast.makeText(c, text, duration);
            t.setGravity(Gravity.CENTER|Gravity.CENTER, 0, 0);
            t.show();
        }
    }

    //[7] GET DATA
    /**
     * @Method name: get_data
     * @param null
     * @return null
     * @Description: Loads the required data from the SQLite database and from
the Shared Preferences.
     */
    public void get_data()
    {
        getData =
PreferenceManager.getDefaultSharedPreferences(getBaseContext());
        user = get_user();
        d = new Data(this);
        d.open();
        total_points = d.read_total_pts(user);
        daily_points = d.read_daily_pts(user);
        d.close();
        elapsed_days = days_elapsed(); //determine elapsed days
        per_cent = percent();
    }

    //[8] SET DATA
    /**
     * @Method name: set_data
     * @param null
```

```
     * @return null
     * @Description: Sets the data so that is can be used by methods within the
Main activity.
     */
    public void set_data()
    {
        usr.setText(user);
        tot_pts.setText(total_points);
        dly_pts.setText(daily_points);
        elpsd_dys.setText(elapsed_days);
        perCent.setText(per_cent);
    }

    //[9] GET USER NAME
    /**
     * @Method name: get_user
     * @param null
     * @return user
     * @Description: Loads the user name from the Shared Preferences and writes
the user to the SQLITE database.
     */
    public String get_user()
    {
        String user="";
        getData =
PreferenceManager.getDefaultSharedPreferences(getBaseContext());
        user = getData.getString("user_name", "User");
        d =  new Data(this);
        d.open();
        d.write_user(user);
        d.close();
        return user;
    }

    //[10] GET DAYS ELAPSED
    /**
     * @Method name: get_elapsed
     * @param null
     * @return elapsed days
     * @Description: Calculates the days elapsed from a particular date chosen
and inputed by the user in the application.
     */
    @SuppressLint("SimpleDateFormat")
    public String days_elapsed()
    {
        getData =
PreferenceManager.getDefaultSharedPreferences(getBaseContext());
        String elapsed="";
        int diff =0;

        SimpleDateFormat dfDate  = new SimpleDateFormat("dd/MM/yyyy");

        java.util.Date start = null;
        java.util.Date today = null;
        Calendar cal_today = Calendar.getInstance();
        Calendar cal_start =
DatePreference.getDateFor(PreferenceManager.getDefaultSharedPreferences(this),"
```

```
ob");
        try {
            start = dfDate.parse(dfDate.format(cal_start.getTime()));
            today = dfDate.parse(dfDate.format(cal_today.getTime()));
            diff = (int) ((start.getTime() - today.getTime())/ (1000 * 60 * 60
* 24))*-1; //-1 to eliminate -ve
            elapsed = Integer.toString(diff);

        } catch (java.text.ParseException e){
            e.printStackTrace();
        }
        return elapsed;
    }

    //[11] DETERMINE LEVEL
    /**
     * @Method name: determine_level
     * @param View
     * @return level
     * @Description: Determine the level reached by the user according to the
total points earned.
     */
    public int determine_level()
    {
        int level = 0;
        int pts = -1;
        try{pts = Integer.parseInt(total_points);}catch(Exception e){};

        if(pts>100 & pts<=200){
            level = 1;
try{d.open();d.write_level(user,Integer.toString(level));d.close();}catch(Excep
tion e){}
        }else if(pts>200 & pts<=300){
            level = 2;
try{d.open();d.write_level(user,Integer.toString(level));d.close();}catch(Excep
tion e){}
        }else if(pts>300 & pts<=400){
            level =3;
try{d.open();d.write_level(user,Integer.toString(level));d.close();}catch(Excep
tion e){}
        }else if(pts>400 & pts<=500){
            level = 4;
try{d.open();d.write_level(user,Integer.toString(level));d.close();}catch(Excep
tion e){}
        }else if(pts>500 & pts<=600){
            level = 5;
try{d.open();d.write_level(user,Integer.toString(level));d.close();}catch(Excep
tion e){}
        }else if(pts>600){
            level = 6;
try{d.open();d.write_level(user,Integer.toString(level));d.close();}catch(Excep
tion e){}
        }
        return level;
    }

    //[12] DETERMINE BADGE
```

```
    /**
     * @Method name: determine_badge
     * @param int
     * @return array
     * @Description: Determine the badges earned by the user according to the
level reached and total points earned.
     */
    public int determine_badge(int index)
    {
        int image[] =
{R.drawable.unknown,R.drawable.unknown,R.drawable.unknown,R.drawable.unknown,R.d
rawable.unknown,R.drawable.unknown};

        switch(determine_level()){
        case 1:
            image[0] = R.drawable.novice;
            break;
        case 2:
            image[0] = R.drawable.novice; image[1] = R.drawable.bookworm;
            break;
        case 3:
            image[0] = R.drawable.novice; image[1] = R.drawable.bookworm;
image[2] = R.drawable.junior_researcher;
            break;
        case 4:
            image[0] = R.drawable.novice; image[1] = R.drawable.bookworm;
image[2] = R.drawable.junior_researcher; image[3] =
R.drawable.master_researcher;
            break;
        case 5:
            image[0] = R.drawable.novice; image[1] = R.drawable.bookworm;
image[2] = R.drawable.junior_researcher; image[3] =
R.drawable.master_researcher; image[4] = R.drawable.achademic;
            break;
        case 6:
            image[0] = R.drawable.novice; image[1] = R.drawable.bookworm;
image[2] = R.drawable.junior_researcher; image[3] =
R.drawable.master_researcher; image[4] = R.drawable.achademic; image[5] =
R.drawable.guru;
            break;
        }
        return image[index];
    }

    //[13] DETERMINE PERCENTAGE
    /**
     * @Method name: percentage
     * @param null
     * @return null
     * @Description: Determine the dissertation percentage completed according
to the total points earned.
     */
    public String percent()
    {
        //Set percentage
        float percent =
(float)Math.round((determine_level()*100)/6f);//calculate progress according to
```

```
the level reached
        String per_cent = Float.toString(percent);
        return per_cent;
    }

    //[14] DETERMINE PROGRESS BAR
    /**
     * @Method name: progress_bar
     * @param null
     * @return null
     * @Description: Determine the level of the progress bar according to the
total points earned.
     */
    public void progress_bar()
    {     try{
        d.open();
        pb.setProgress(Integer.parseInt(d.read_level(user))*100);
        d.close();}catch(Exception e){};
    }

    //[15] RESET DATA
    /**
     * @Method name: reset_data
     * @param null
     * @return null
     * @Description: Reset all data saved by the user to default.
     */
    public void reset_data()
    {
        getData =
PreferenceManager.getDefaultSharedPreferences(getBaseContext());
        boolean reset = getData.getBoolean("reset", false);

        if(reset){
            getData.edit().clear().commit();
            d.open();
            d.delete_all();
            d.close();
        }
    }

    //[16] RESET DAILY POINTS
    /**
     * @Method name: reset_daily_pts
     * @param null
     * @return null
     * @Description: Reset daily points to zero.
     */
    public void reset_daily_pts()
    {
        getData =
PreferenceManager.getDefaultSharedPreferences(getBaseContext());
        boolean reset_daily = getData.getBoolean("reset_daily", false);

        if(reset_daily)
        {
            try{
```

```
                d.open();
                d.reset_daily();
                d.close();
            }catch(Exception e){}
        }
    }

    //[17] SEND NOTIFICATIONS
    /**
     * @Method name: notification
     * @param null
     * @return null
     * @Description: Send notifications to user every 6 hours.
     */
    public void notification()
    {
        getData =
PreferenceManager.getDefaultSharedPreferences(getBaseContext());
        boolean notification = getData.getBoolean("Notifications", false);

        if(notification){
            Calendar cal = Calendar.getInstance();

            cal.set(Calendar.HOUR_OF_DAY,8);
            cal.set(Calendar.MINUTE,0);
            cal.set(Calendar.SECOND, 0);
            cal.set(Calendar.MILLISECOND,0);
            long start = cal.getTimeInMillis();

            PendingIntent pi = PendingIntent.getService(this, 0 , new
Intent(this,Notification.class),PendingIntent.FLAG_UPDATE_CURRENT);
            AlarmManager am = (AlarmManager)
this.getSystemService(Context.ALARM_SERVICE);
            am.setRepeating(AlarmManager.RTC_WAKEUP,start,21600000, pi);

        }
    }

    //[18] SAVE DATA FOR PAPERS
    /**
     * @Method name: save_data_papers
     * @param null
     * @return null
     * @Description: Save paper's button original state. This saved data will
be accessed by the Papers Activity.
     */
    public void save_data_papers()
    {
        String data ="000000000000000000000000";
        d.open();
        d.btn_write_state(data, user);
        d.close();
    }
}
```

```java
package com.gamification_research;
import com.gamification_research.R;

/**
 * @author Jonathan Cassar
 * @Class Name: Notifications
 * @extends: Services
 * @Description: Uses the service interface to send notifications at various
intervals. This class also caters
 * for mobile vibration and light beeping
 * @TargetApi(Build.VERSION_CODES.HONEYCOMB)
 */
public class Notification extends Service
{
    //VARIABLES
    Intent i,j;
    PendingIntent pi;
    NotificationManager nm;
    NotificationCompat.Builder nbldr;
    Uri s;

    //[1] ON BIND
    /**
     * @Method name: onBind
     * @param Integer
     * @return null
     */
    public IBinder onBind(Intent arg0) {

        return null;
    }

    //[2]ON CREATE
    /**
     * @Method name: onCreate
     * @param null
     * @return null
     * @Override: Constructor override
     */
    public void onCreate()  {

        super.onCreate();
        displayNotification();
    }

    //[3]ON START
    /**
     * @Method name: onStart
     * @param Intent
     * @param int
     * @return null
     * @SuppressWarnings("deprecation")
     * @Description: On Start call the notifications manager
     */

    public void onStart(Intent intent, int startId) {
        super.onStart(intent, startId);
```

```
        displayNotification();
    }

    //[4] ON DESTROY
    /**
     * @Method name: onDestroy
     * @param null
     * @return null
     */
    public void onDestroy()  {

        super.onDestroy();
    }

    //[5] NOTIFICATION
    /**
     * @Method name: onDisplay
     * @param null
     * @return null
     * @Description:  Notify the user through a notification every 6 hours.
With every notification, the
     * mobile is allowed to beep, flash light and vibration.
     */
    public void displayNotification()
    {


        long [] vibrate = {500,500,500,500,500,500,500,500,500};
        i = new Intent(this, Main_screen.class);
        pi = PendingIntent.getActivity(this, 1, i,
PendingIntent.FLAG_UPDATE_CURRENT);
        nm = (NotificationManager)
this.getSystemService(Context.NOTIFICATION_SERVICE);
        s = RingtoneManager.getDefaultUri(RingtoneManager.TYPE_NOTIFICATION);

        nbldr = new NotificationCompat.Builder(this);
        nbldr.setContentIntent(pi);
        nbldr.setAutoCancel(true);
        nbldr.setSmallIcon(R.drawable.star);
        nbldr.setContentTitle("GResearch");
        nbldr.setContentText("Stay Focused. Work on dissertation.");
        nbldr.setLights(Color.MAGENTA, 500, 500);     //beep light
        nbldr.setVibrate(vibrate);   //vibrate device
        nbldr.setSound(s);           //play sound
        nm.notify(1, nbldr.build());

    }

}
```

```java
package com.gamification_research;

import java.io.BufferedReader;

/**
 * @author Jonathan Cassar
 * @Class Name: Papers Screen
 * @Description: This class is accountable for the papers screen, handling of buttons, remembering of button state and progression
 * @TargetApi(Build.VERSION_CODES.HONEYCOMB)
 */
public class Papers_screen extends Activity implements OnClickListener
{
    //VARIABLES
    Button btn01,btn02,btn03,btn04,
           btn05,btn06,btn07,btn08,
           btn09,btn10,btn11,btn12,
           btn13,btn14,btn15,btn16,
           btn17,btn18,btn19,btn20,
           btn21,btn22,btn23,btn24,
           btn25;

    Data d;
    InputStream is;
    SharedPreferences getData;
    String user, total_points, daily_points, state, ppr_abstract;
    int points = 15;
    int btnId;
    ImageView btn_help;
    boolean btn01_clicked,btn02_clicked,btn03_clicked,btn04_clicked,
            btn05_clicked,btn06_clicked,btn07_clicked,btn08_clicked,
            btn09_clicked,btn10_clicked,btn11_clicked,btn12_clicked,
            btn13_clicked,btn14_clicked,btn15_clicked,btn16_clicked,
            btn17_clicked,btn18_clicked,btn19_clicked,btn20_clicked,
            btn21_clicked,btn22_clicked,btn23_clicked,btn24_clicked,
            btn25_clicked;



    //[1] ON CREATE METHOD
    /**
     * @Method name: onCreate
     * @param Bundle savedInstanceState
     * @return null
     * @Description: On create method.
     */
    protected void onCreate(Bundle savedInstanceState)
    {
        // TODO Auto-generated method stub
        super.onCreate(savedInstanceState);
        setContentView(R.layout.papers_screen);

        //Get Reference
        btn01 = (Button)findViewById(R.id.btn01_paper);
        btn02 = (Button)findViewById(R.id.btn02_paper);
        btn03 = (Button)findViewById(R.id.btn03_paper);
        btn04 = (Button)findViewById(R.id.btn04_paper);
```

```
btn05 = (Button)findViewById(R.id.btn05_paper);
btn06 = (Button)findViewById(R.id.btn06_paper);
btn07 = (Button)findViewById(R.id.btn07_paper);
btn08 = (Button)findViewById(R.id.btn08_paper);

btn09 = (Button)findViewById(R.id.btn09_paper);
btn10 = (Button)findViewById(R.id.btn10_paper);
btn11 = (Button)findViewById(R.id.btn11_paper);
btn12 = (Button)findViewById(R.id.btn12_paper);

btn13 = (Button)findViewById(R.id.btn13_paper);
btn14 = (Button)findViewById(R.id.btn14_paper);
btn15 = (Button)findViewById(R.id.btn15_paper);
btn16 = (Button)findViewById(R.id.btn16_paper);

btn17 = (Button)findViewById(R.id.btn17_paper);
btn18 = (Button)findViewById(R.id.btn18_paper);
btn19 = (Button)findViewById(R.id.btn19_paper);
btn20 = (Button)findViewById(R.id.btn20_paper);

btn21 = (Button)findViewById(R.id.btn21_paper);
btn22 = (Button)findViewById(R.id.btn22_paper);
btn23 = (Button)findViewById(R.id.btn23_paper);
btn24 = (Button)findViewById(R.id.btn24_paper);

btn25 = (Button)findViewById(R.id.btn25_paper);

btn_help = (ImageView)findViewById(R.id.btn_help_papers);

//Listeners
btn01.setOnClickListener(this);
btn02.setOnClickListener(this);
btn03.setOnClickListener(this);
btn04.setOnClickListener(this);

btn05.setOnClickListener(this);
btn06.setOnClickListener(this);
btn07.setOnClickListener(this);
btn08.setOnClickListener(this);

btn09.setOnClickListener(this);
btn10.setOnClickListener(this);
btn11.setOnClickListener(this);
btn12.setOnClickListener(this);

btn13.setOnClickListener(this);
btn14.setOnClickListener(this);
btn15.setOnClickListener(this);
btn16.setOnClickListener(this);

btn17.setOnClickListener(this);
btn18.setOnClickListener(this);
btn19.setOnClickListener(this);
btn20.setOnClickListener(this);

btn21.setOnClickListener(this);
```

```
        btn22.setOnClickListener(this);
        btn23.setOnClickListener(this);
        btn24.setOnClickListener(this);

        btn25.setOnClickListener(this);

        btn_help.setOnClickListener(this);

        //set button sates
        btn01_clicked = btn02_clicked = btn03_clicked = btn04_clicked =
        btn05_clicked = btn06_clicked = btn07_clicked = btn08_clicked =
        btn09_clicked = btn10_clicked = btn11_clicked = btn12_clicked =
        btn13_clicked = btn14_clicked = btn15_clicked = btn16_clicked =
        btn17_clicked = btn18_clicked = btn19_clicked = btn20_clicked =
        btn21_clicked = btn22_clicked = btn23_clicked = btn24_clicked =
        btn25_clicked = false;

        //Methods
        get_data();
        read_state();
    }

    //[2] CLICK HANDLER
    /**
     * @Method name: onClick
     * @param View
     * @return null
     * @Description: Handles all the buttons clicks and register with the
database the button event.
     */
    public void onClick(View arg0)
    {
        switch(arg0.getId())
        {
        case R.id.btn01_paper:
            if(btn01_clicked==false){btnId =1; message(arg0); btn01_clicked =
true;}else{btnId =-1; message_undo(arg0);btn01_clicked = false;}
            break;
        case R.id.btn02_paper:
            if(btn02_clicked==false){btnId =2; message(arg0); btn02_clicked =
true;}else{btnId =-2; message_undo(arg0);btn02_clicked = false;}
            break;
        case R.id.btn03_paper:
            if(btn03_clicked==false){btnId =3; message(arg0); btn03_clicked =
true;}else{btnId =-3; message_undo(arg0);btn03_clicked = false;}
            break;
        case R.id.btn04_paper:
            if(btn04_clicked==false){btnId =4; message(arg0); btn04_clicked =
true;}else{btnId =-4; message_undo(arg0);btn04_clicked = false;}
            break;
        case R.id.btn05_paper:
            if(btn05_clicked==false){btnId =5; message(arg0); btn05_clicked =
true;}else{btnId =-5; message_undo(arg0);btn05_clicked = false;}
            break;
        case R.id.btn06_paper:
            if(!btn06_clicked){btnId =6; message(arg0); btn06_clicked =
true;}else{btnId =-6; message_undo(arg0);btn06_clicked = false;}
```

```
            break;
        case R.id.btn07_paper:
            if(!btn07_clicked){btnId =7; message(arg0); btn07_clicked =
true;}else{btnId =-7; message_undo(arg0);btn07_clicked = false;}
            break;
        case R.id.btn08_paper:
            if(!btn08_clicked){btnId =8; message(arg0); btn08_clicked =
true;}else{btnId =-8; message_undo(arg0);btn08_clicked = false;}
            break;

        case R.id.btn09_paper:
            if(!btn09_clicked){btnId =9; message(arg0); btn09_clicked =
true;}else{btnId =-9; message_undo(arg0);btn09_clicked = false;}
            break;
        case R.id.btn10_paper:
            if(!btn10_clicked){btnId =10; message(arg0); btn10_clicked =
true;}else{btnId =-10; message_undo(arg0);btn10_clicked = false;}
            break;
        case R.id.btn11_paper:
            if(!btn11_clicked){btnId =11; message(arg0); btn11_clicked =
true;}else{btnId =-11; message_undo(arg0);btn11_clicked = false;}
            break;
        case R.id.btn12_paper:
            if(!btn12_clicked){btnId =12; message(arg0); btn12_clicked =
true;}else{btnId =-12; message_undo(arg0);btn12_clicked = false;}
            break;

        case R.id.btn13_paper:
            if(!btn13_clicked){btnId =13; message(arg0); btn13_clicked =
true;}else{btnId =-13; message_undo(arg0);btn13_clicked = false;}
            break;
        case R.id.btn14_paper:
            if(!btn14_clicked){btnId =14; message(arg0); btn14_clicked =
true;}else{btnId =-14; message_undo(arg0);btn14_clicked = false;}
            break;
        case R.id.btn15_paper:
            if(!btn15_clicked){btnId =15; message(arg0); btn15_clicked =
true;}else{btnId =-15; message_undo(arg0);btn15_clicked = false;}
            break;
        case R.id.btn16_paper:
            if(!btn16_clicked){btnId =16; message(arg0); btn16_clicked =
true;}else{btnId =-16; message_undo(arg0);btn16_clicked = false;}
            break;

        case R.id.btn17_paper:
            if(!btn17_clicked){btnId =17; message(arg0); btn17_clicked =
true;}else{btnId =-17; message_undo(arg0);btn17_clicked = false;}
            break;
        case R.id.btn18_paper:
            if(!btn18_clicked){btnId =18; message(arg0); btn18_clicked =
true;}else{btnId =-18; message_undo(arg0);btn18_clicked = false;}
            break;
        case R.id.btn19_paper:
            if(!btn19_clicked){btnId =19; message(arg0); btn19_clicked =
true;}else{btnId =-19; message_undo(arg0);btn19_clicked = false;}
            break;
```

```
        case R.id.btn20_paper:
            if(!btn20_clicked){btnId =20; message(arg0); btn20_clicked =
true;}else{btnId =-20; message_undo(arg0);btn20_clicked = false;}
            break;

        case R.id.btn21_paper:
            if(!btn21_clicked){btnId =21; message(arg0); btn21_clicked =
true;}else{btnId =-21; message_undo(arg0);btn21_clicked = false;}
            break;

        case R.id.btn22_paper:
            if(!btn22_clicked){btnId =22; message(arg0); btn22_clicked =
true;}else{btnId =-22; message_undo(arg0);btn22_clicked = false;}
            break;

        case R.id.btn23_paper:
            if(!btn23_clicked){btnId =23; message(arg0); btn23_clicked =
true;}else{btnId =-23; message_undo(arg0);btn23_clicked = false;}
            break;

        case R.id.btn24_paper:
            if(!btn24_clicked){btnId =24; message(arg0); btn24_clicked =
true;}else{btnId =-24; message_undo(arg0);btn24_clicked = false;}
            break;

        case R.id.btn25_paper:
            if(!btn25_clicked){btnId =25; message(arg0); btn25_clicked =
true;}else{btnId =-25; message_undo(arg0);btn25_clicked = false;}
            break;
        case R.id.btn_help_papers:
            try{help_msg(arg0);}catch(Exception e){e.printStackTrace();};
            break;
        }
    }

    //[3] GET DATA
    /**
     * @Method name: get_data
     * @param null
     * @return: null
     * @Description: Get all the data required by the methods within this
class.
     */
    public void get_data()
    {
        getData =
PreferenceManager.getDefaultSharedPreferences(getBaseContext());
        d = new Data(this);
        d.open();
        user = d.read_user(getData.getString("user_name", "User"));
        total_points = d.read_total_pts(user);
        daily_points = d.read_daily_pts(user);
        d.close();
    }


    //[4] POP UP NOTIFICATION OF POINTS EARNRED
```

```
    /**
     * @Method name: pop
     * @param null
     * @return null
     * @Description: Pop a message to notify the user that 10 points were
earned.
     */
    public void popup()
    {
        Context c = getApplicationContext();
        CharSequence text = "You have earned "+points+" points";
        int duration = Toast.LENGTH_SHORT;

        Toast t= Toast.makeText(c, text, duration);
        t.setGravity(Gravity.TOP|Gravity.TOP,0,100);
        t.show();
    }

    //[5A] UPDATE POINTS
    /**
     * @Method name: update_points
     * @param null
     * @return integer
     * @Description: Update all the points and store.
     */
    public int update_points()
    {
        int total = 0;
        int day = 0;

        //convert retrieved data from database to string
        try{total = Integer.parseInt(total_points);
            day = Integer.parseInt(daily_points);
            }catch(Exception e){};

        //Add points
        day+=points;
        total+=points;

        //convert back to string
        try{
            daily_points= Integer.toString(day);
            total_points = Integer.toString(total);
        }catch(Exception e){}

        return points;
    }

    //[5B] DECREMENT POINTS
    /**
     * @Method name: decrement_points
     * @param null
     * @return integer
     * @Description: Decrement all the points and store.
     */
    public int decrement_points()
    {
```

```
        int total = 0;
        int day = 0;

        //convert retrieved data from database to string
        try{total = Integer.parseInt(total_points);
            day = Integer.parseInt(daily_points);
            }catch(Exception e){};

        //Add points
        day-=points;
        total-=points;

        //convert back to string
        try{
            daily_points= Integer.toString(day);
            total_points = Integer.toString(total);
        }catch(Exception e){}

        return points;
    }

    //[6] SAVE POINTS TO DATABASE
    /**
     * @Method name: save_pts_database
     * @param null
     * @return null
     * @Description: Save points to database.
     */
    public void save_pts_database()
    {
        d = new Data(this);
        d.open();
        d.write_daily(user,daily_points);
        d.write_total(user,total_points);
        d.close();
    }

    //[7] SAVE STATE OF BUTTONS
    /**
     * @Method name: save_state
     * @param String
     * @return null
     * @Description: Check the button state and save the sate to the database.
     */
    public void save_state(String name)
    {
         String state ="";
         String all_state="";
         String [] btns_state = new String[25];
         char [] chars;

         try{
             d.open();
             state = d.btn_read_state(user); //(a)bring state as stored in Main
screen i.e all zeros
             d.close();
             chars = state.toCharArray(); //(b)convert string to characters
```

```
            for(int i=0;i<chars.length;i++)
            {
                btns_state[i] = Character.toString(chars[i]); //(c)save
characters in array
            }
         }catch(Exception e){}

        if(name =="btn01"){btns_state[0]="1";} //(d)if button is clicked update
array
        if(name =="btn02"){btns_state[1]="1";}
        if(name =="btn03"){btns_state[2]="1";}
        if(name =="btn04"){btns_state[3]="1";}

        if(name =="btn05"){btns_state[4]="1";}
        if(name =="btn06"){btns_state[5]="1";}
        if(name =="btn07"){btns_state[6]="1";}
        if(name =="btn08"){btns_state[7]="1";}

        if(name =="btn09"){btns_state[8]="1";}
        if(name =="btn10"){btns_state[9]="1";}
        if(name =="btn11"){btns_state[10]="1";}
        if(name =="btn12"){btns_state[11]="1";}

        if(name =="btn13"){btns_state[12]="1";}
        if(name =="btn14"){btns_state[13]="1";}
        if(name =="btn15"){btns_state[14]="1";}
        if(name =="btn16"){btns_state[15]="1";}

        if(name =="btn17"){btns_state[16]="1";}
        if(name =="btn18"){btns_state[17]="1";}
        if(name =="btn19"){btns_state[18]="1";}
        if(name =="btn20"){btns_state[19]="1";}

        if(name =="btn21"){btns_state[20]="1";}
        if(name =="btn22"){btns_state[21]="1";}
        if(name =="btn23"){btns_state[22]="1";}
        if(name =="btn24"){btns_state[23]="1";}

        if(name =="btn25"){btns_state[24]="1";}


        //(e) convert back to string
        for(int i=0; i<btns_state.length;i++)
        {
            all_state = all_state+btns_state[i];
        }

        //(f) save updates string to database
        try{
            d.open();
            d.btn_write_state(user,all_state);
            d.close();
        }catch(Exception e){}
    }

    //[8] READ STATE OF BUTTONS
```

```
    /**
     * @Method name: read_state
     * @param null
     * @return null
     * @Description: Read the state of the button before draw it to screen.
     */
    public void read_state()
    {
        try{
            d.open();
            String state = d.btn_read_state(user);
            char[] individual_state = state.toCharArray();

            for(int i=0; i<individual_state.length;i++)
            {
                if(individual_state[0]==
'1'){btn01.setBackgroundResource(R.drawable.check);btn01_clicked = true;}
                if(individual_state[1]==
'1'){btn02.setBackgroundResource(R.drawable.check);btn02_clicked = true;}
                if(individual_state[2]==
'1'){btn03.setBackgroundResource(R.drawable.check);btn03_clicked = true;}
                if(individual_state[3]==
'1'){btn04.setBackgroundResource(R.drawable.check);btn04_clicked = true;}

                if(individual_state[4]==
'1'){btn05.setBackgroundResource(R.drawable.check);btn05_clicked = true;
                if(individual_state[5]==
'1'){btn06.setBackgroundResource(R.drawable.check);btn06_clicked = true;}
                if(individual_state[6]==
'1'){btn07.setBackgroundResource(R.drawable.check);btn07_clicked = true;}
                if(individual_state[7]==
'1'){btn08.setBackgroundResource(R.drawable.check);btn08_clicked = true;}

                if(individual_state[8]==
'1'){btn09.setBackgroundResource(R.drawable.check);btn09_clicked = true;}
                if(individual_state[9]==
'1'){btn10.setBackgroundResource(R.drawable.check);btn10_clicked = true;}
                if(individual_state[10]==
'1'){btn11.setBackgroundResource(R.drawable.check);btn11_clicked = true;}
                if(individual_state[11]==
'1'){btn12.setBackgroundResource(R.drawable.check);btn12_clicked = true;}

                if(individual_state[12]==
'1'){btn13.setBackgroundResource(R.drawable.check);btn13_clicked = true;}
                if(individual_state[13]==
'1'){btn14.setBackgroundResource(R.drawable.check);btn14_clicked = true;}
                if(individual_state[14]==
'1'){btn15.setBackgroundResource(R.drawable.check);btn15_clicked = true;}
                if(individual_state[15]==
'1'){btn16.setBackgroundResource(R.drawable.check);btn16_clicked = true;}

                if(individual_state[16]==
'1'){btn17.setBackgroundResource(R.drawable.check);btn17_clicked = true;}
                if(individual_state[17]==
'1'){btn18.setBackgroundResource(R.drawable.check);btn18_clicked = true;}
                if(individual_state[18]==
'1'){btn19.setBackgroundResource(R.drawable.check);btn19_clicked = true;}
```

```
                if(individual_state[19]==
'1'){btn20.setBackgroundResource(R.drawable.check);btn20_clicked = true;}

                if(individual_state[20]==
'1'){btn21.setBackgroundResource(R.drawable.check);btn21_clicked = true;}
                if(individual_state[21]==
'1'){btn22.setBackgroundResource(R.drawable.check);btn22_clicked = true;}
                if(individual_state[22]==
'1'){btn23.setBackgroundResource(R.drawable.check);btn23_clicked = true;}
                if(individual_state[23]==
'1'){btn24.setBackgroundResource(R.drawable.check);btn24_clicked = true;}

                if(individual_state[24]==
'1'){btn25.setBackgroundResource(R.drawable.check);btn25_clicked = true;}

            }
            d.close();
        }}catch(Exception e){}
    }

    //[9] SET HELP
    /**
     * Method name: message
     * @param View
     * @return null
     * @Description: Ouput the abstract of the paper
     */
    public void message(View view)
    {
        AlertDialog d = new AlertDialog.Builder(this).create();
        d.setTitle("Abstract");
        d.setMessage(paper_abstract(btnId));

        d.setButton("READ", new DialogInterface.OnClickListener()
        {
            public void onClick(DialogInterface dialog, int which)
            {
                if (btnId==1){popup();
update_points();save_pts_database();save_state("btn01");btn01.setBackgroundReso
urce(R.drawable.check);}
                if (btnId==2){popup();
update_points();save_pts_database();save_state("btn02");btn02.setBackgroundReso
urce(R.drawable.check);}
                if (btnId==3){popup();
update_points();save_pts_database();save_state("btn03");btn03.setBackgroundReso
urce(R.drawable.check);}
                if (btnId==4){popup();
update_points();save_pts_database();save_state("btn04");btn04.setBackgroundReso
urce(R.drawable.check);}

                if (btnId==5){popup();
update_points();save_pts_database();save_state("btn05");btn05.setBackgroundReso
urce(R.drawable.check);}
                if (btnId==6){popup();
update_points();save_pts_database();save_state("btn06");btn06.setBackgroundReso
urce(R.drawable.check);}
                if (btnId==7){popup();
```

```
update_points();save_pts_database();save_state("btn07");btn07.setBackgroundReso
urce(R.drawable.check);}
                if (btnId==8){popup();
update_points();save_pts_database();save_state("btn08");btn08.setBackgroundReso
urce(R.drawable.check);}

                if (btnId==9) {popup();
update_points();save_pts_database();save_state("btn09");btn09.setBackgroundReso
urce(R.drawable.check);}
                if (btnId==10){popup();
update_points();save_pts_database();save_state("btn10");btn10.setBackgroundReso
urce(R.drawable.check);}
                if (btnId==11){popup();
update_points();save_pts_database();save_state("btn11");btn11.setBackgroundReso
urce(R.drawable.check);}
                if (btnId==12){popup();
update_points();save_pts_database();save_state("btn12");btn12.setBackgroundReso
urce(R.drawable.check);}

                if (btnId==13){popup();
update_points();save_pts_database();save_state("btn13");btn13.setBackgroundReso
urce(R.drawable.check);}
                if (btnId==14){popup();
update_points();save_pts_database();save_state("btn14");btn14.setBackgroundReso
urce(R.drawable.check);}
                if (btnId==15){popup();
update_points();save_pts_database();save_state("btn15");btn15.setBackgroundReso
urce(R.drawable.check);}
                if (btnId==16){popup();
update_points();save_pts_database();save_state("btn16");btn16.setBackgroundReso
urce(R.drawable.check);}

                if (btnId==17){popup();
update_points();save_pts_database();save_state("btn17");btn17.setBackgroundReso
urce(R.drawable.check);}
                if (btnId==18){popup();
update_points();save_pts_database();save_state("btn18");btn18.setBackgroundReso
urce(R.drawable.check);}
                if (btnId==19){popup();
update_points();save_pts_database();save_state("btn19");btn19.setBackgroundReso
urce(R.drawable.check);}
                if (btnId==20){popup();
update_points();save_pts_database();save_state("btn20");btn20.setBackgroundReso
urce(R.drawable.check);}

                if (btnId==21){popup();
update_points();save_pts_database();save_state("btn21");btn21.setBackgroundReso
urce(R.drawable.check);}
                if (btnId==22){popup();
update_points();save_pts_database();save_state("btn22");btn22.setBackgroundReso
urce(R.drawable.check);}
                if (btnId==23){popup();
update_points();save_pts_database();save_state("btn23");btn23.setBackgroundReso
urce(R.drawable.check);}
                if (btnId==24){popup();
update_points();save_pts_database();save_state("btn24");btn24.setBackgroundReso
urce(R.drawable.check);}
```

```
                if (btnId==25){popup();
update_points();save_pts_database();save_state("btn25");btn25.setBackgroundReso
urce(R.drawable.check);}
            }
        });
        d.setIcon(R.drawable.note);
        d.show();
    }

    //[10] MESSAGE UNDO
    /**
     * Method name: Message Undo
     * @param View
     * @return null
     * @Description: Undo decision
     */
    @SuppressWarnings("deprecation")
    public void message_undo(View view)
    {
        AlertDialog d = new AlertDialog.Builder(this).create();
        d.setTitle("Cancel");
        d.setMessage("Are you sure you want to cancel?");

        d.setButton("Yes", new DialogInterface.OnClickListener()
        {
            @SuppressLint("NewApi")
            public void onClick(DialogInterface dialog, int which)
            {
                if(btnId==-1){undo_state("btn01");decrement_points();
save_pts_database();btn01.setBackgroundDrawable(getResources().getDrawable(R.dr
awable.btn_design_whisky));}
                if(btnId==-2){undo_state("btn02");decrement_points();
save_pts_database();btn02.setBackgroundDrawable(getResources().getDrawable(R.dr
awable.btn_design_whisky));}
                if(btnId==-3){undo_state("btn03");decrement_points();
save_pts_database();btn03.setBackgroundDrawable(getResources().getDrawable(R.dr
awable.btn_design_whisky));}
                if(btnId==-4){undo_state("btn04");decrement_points();
save_pts_database();btn04.setBackgroundDrawable(getResources().getDrawable(R.dr
awable.btn_design_whisky));}

                if(btnId==-5){undo_state("btn05");decrement_points();
save_pts_database();btn05.setBackgroundDrawable(getResources().getDrawable(R.dr
awable.btn_design_whisky));}
                if(btnId==-6){undo_state("btn06");decrement_points();
save_pts_database();btn06.setBackgroundDrawable(getResources().getDrawable(R.dr
awable.btn_design_whisky));}
                if(btnId==-7){undo_state("btn07");decrement_points();
save_pts_database();btn07.setBackgroundDrawable(getResources().getDrawable(R.dr
awable.btn_design_whisky));}
                if(btnId==-8){undo_state("btn08");decrement_points();
save_pts_database();btn08.setBackgroundDrawable(getResources().getDrawable(R.dr
awable.btn_design_whisky));}

                if(btnId==-9){undo_state("btn09");decrement_points();
save_pts_database();btn09.setBackgroundDrawable(getResources().getDrawable(R.dr
```

```
awable.btn_design_whisky));}
                if(btnId==-10){undo_state("btn10");decrement_points();
save_pts_database();btn10.setBackgroundDrawable(getResources().getDrawable(R.dr
awable.btn_design_whisky));}
                if(btnId==-11){undo_state("btn11");decrement_points();
save_pts_database();btn11.setBackgroundDrawable(getResources().getDrawable(R.dr
awable.btn_design_whisky));}
                if(btnId==-12){undo_state("btn12");decrement_points();
save_pts_database();btn12.setBackgroundDrawable(getResources().getDrawable(R.dr
awable.btn_design_whisky));}

                if(btnId==-13){undo_state("btn13");decrement_points();
save_pts_database();btn13.setBackgroundDrawable(getResources().getDrawable(R.dr
awable.btn_design_whisky));}
                if(btnId==-14){undo_state("btn14");decrement_points();
save_pts_database();btn14.setBackgroundDrawable(getResources().getDrawable(R.dr
awable.btn_design_whisky));}
                if(btnId==-15){undo_state("btn15");decrement_points();
save_pts_database();btn15.setBackgroundDrawable(getResources().getDrawable(R.dr
awable.btn_design_whisky));}
                if(btnId==-16){undo_state("btn16");decrement_points();
save_pts_database();btn16.setBackgroundDrawable(getResources().getDrawable(R.dr
awable.btn_design_whisky));}

                if(btnId==-17){undo_state("btn17");decrement_points();
save_pts_database();btn17.setBackgroundDrawable(getResources().getDrawable(R.dr
awable.btn_design_whisky));}
                if(btnId==-18){undo_state("btn18");decrement_points();
save_pts_database();btn18.setBackgroundDrawable(getResources().getDrawable(R.dr
awable.btn_design_whisky));}
                if(btnId==-19){undo_state("btn19");decrement_points();
save_pts_database();btn19.setBackgroundDrawable(getResources().getDrawable(R.dr
awable.btn_design_whisky));}
                if(btnId==-20){undo_state("btn20");decrement_points();
save_pts_database();btn20.setBackgroundDrawable(getResources().getDrawable(R.dr
awable.btn_design_whisky));}

                if(btnId==-21){undo_state("btn21");decrement_points();
save_pts_database();btn21.setBackgroundDrawable(getResources().getDrawable(R.dr
awable.btn_design_whisky));}
                if(btnId==-22){undo_state("btn22");decrement_points();
save_pts_database();btn22.setBackgroundDrawable(getResources().getDrawable(R.dr
awable.btn_design_whisky));}
                if(btnId==-23){undo_state("btn23");decrement_points();
save_pts_database();btn23.setBackgroundDrawable(getResources().getDrawable(R.dr
awable.btn_design_whisky));}
                if(btnId==-24){undo_state("btn24");decrement_points();
save_pts_database();btn24.setBackgroundDrawable(getResources().getDrawable(R.dr
awable.btn_design_whisky));}

                if(btnId==-25){undo_state("btn25");decrement_points();
save_pts_database();btn25.setBackgroundDrawable(getResources().getDrawable(R.dr
awable.btn_design_whisky));}
                    }
            });
            d.setIcon(R.drawable.exclamation);
            d.show();
```

```
    }


    //[11] CHANGE STATE OF BUTTON FROM DATABASE
    /**
     * Method name: undo_state
     * @param View
     * @return null
     * @Description: Change state of button
     */
    public void undo_state(String name)
    {
         String state ="";
         String all_state="";
         String [] btns_state = new String[25];
         char [] chars;

         try{
             d.open();
             state = d.btn_read_state(user); //(a)bring state as stored in Main
screen i.e all zeros
             d.close();
             chars = state.toCharArray(); //(b)convert string to characters

             for(int i=0;i<chars.length;i++)
             {
                 btns_state[i] = Character.toString(chars[i]); //(c)save
characters in array
             }
         }catch(Exception e){}

        if(name =="btn01"){btns_state[0]="0";} //(d)if button is clicked update
array to pref state
        if(name =="btn02"){btns_state[1]="0";}
        if(name =="btn03"){btns_state[2]="0";}
        if(name =="btn04"){btns_state[3]="0";}

        if(name =="btn05"){btns_state[4]="0";}
        if(name =="btn06"){btns_state[5]="0";}
        if(name =="btn07"){btns_state[6]="0";}
        if(name =="btn08"){btns_state[7]="0";}

        if(name =="btn09"){btns_state[8]="0";}
        if(name =="btn10"){btns_state[9]="0";}
        if(name =="btn11"){btns_state[10]="0";}
        if(name =="btn12"){btns_state[11]="0";}

        if(name =="btn13"){btns_state[12]="0";}
        if(name =="btn14"){btns_state[13]="0";}
        if(name =="btn15"){btns_state[14]="0";}
        if(name =="btn16"){btns_state[15]="0";}

        if(name =="btn17"){btns_state[16]="0";}
        if(name =="btn18"){btns_state[17]="0";}
        if(name =="btn19"){btns_state[18]="0";}
        if(name =="btn20"){btns_state[19]="0";}
```

```
    if(name =="btn21"){btns_state[20]="0";}
    if(name =="btn22"){btns_state[21]="0";}
    if(name =="btn23"){btns_state[22]="0";}
    if(name =="btn24"){btns_state[23]="0";}

    if(name =="btn25"){btns_state[24]="1";}


    //(e) convert back to string
    for(int i=0; i<btns_state.length;i++)
    {
        all_state = all_state+btns_state[i];
    }

    //(f) save updates string to database
    try{
        d.open();
        d.btn_write_state(user,all_state);
        d.close();
    }catch(Exception e){}
}

//[12] PAPER ABSTRACTS
/**
 * Method name: paper_abstract
 * @param View
 * @return null
 * @Description: Abstracts of papers
 */
public String paper_abstract(int btnId)
{
    if(btnId ==1){ppr_abstract =  readText(btnId);}
    if(btnId ==2){ppr_abstract =  readText(btnId);}
    if(btnId ==3){ppr_abstract =  readText(btnId);}
    if(btnId ==4){ppr_abstract =  readText(btnId);}

    if(btnId ==5){ppr_abstract =  readText(btnId);}
    if(btnId ==6){ppr_abstract =  readText(btnId);}
    if(btnId ==7){ppr_abstract =  readText(btnId);}
    if(btnId ==8){ppr_abstract =  readText(btnId);}

    if(btnId ==9){ppr_abstract =  readText(btnId);}
    if(btnId ==10){ppr_abstract =  readText(btnId);}
    if(btnId ==11){ppr_abstract =  readText(btnId);}
    if(btnId ==12){ppr_abstract =  readText(btnId);}

    if(btnId ==13){ppr_abstract =  readText(btnId);}
    if(btnId ==14){ppr_abstract =  readText(btnId);}
    if(btnId ==15){ppr_abstract =  readText(btnId);}
    if(btnId ==16){ppr_abstract =  readText(btnId);}

    if(btnId ==17){ppr_abstract =  readText(btnId);}
    if(btnId ==18){ppr_abstract =  readText(btnId);}
    if(btnId ==19){ppr_abstract =  readText(btnId);}
    if(btnId ==20){ppr_abstract =  readText(btnId);}

    if(btnId ==21){ppr_abstract =  readText(btnId);}
```

```
        if(btnId ==22){ppr_abstract =  readText(btnId);}
        if(btnId ==23){ppr_abstract =  readText(btnId);}
        if(btnId ==24){ppr_abstract =  readText(btnId);}

        if(btnId ==25){ppr_abstract =  readText(btnId);}

        return ppr_abstract;
    }

    //[13] SET HELP
    /**
     * Method name: message
     * @param View
     * @return null
     * @Description: Set the help
     */
    public void help_msg(View view)
    {
        AlertDialog d = new AlertDialog.Builder(this).create();
        d.setTitle("Progress Papers");
        d.setMessage(
                    "Click on the Buttons to discover paper's authors and
abstract."+
                    " Read papers and earn 15 points with every read."+
                    " Clicked the wrong paper? No problem. Just re-click to
undo your decision.");

        d.setButton("OK", new DialogInterface.OnClickListener()
        {
            public void onClick(DialogInterface dialog, int which)
            {
                //REMAIN AT LOGIN SCREEN
            }
        });
        d.setIcon(R.drawable.help);
        d.show();
    }

    public String readText(int btnId)
    {
        StringBuilder contents = new StringBuilder();
        String sep = System.getProperty("line.separator");

        try {
            if(btnId==1){is =
getResources().openRawResource(R.raw.abstract01);}
            if(btnId==2){is =
getResources().openRawResource(R.raw.abstract02);}
            if(btnId==3){is =
getResources().openRawResource(R.raw.abstract03);}
            if(btnId==4){is =
getResources().openRawResource(R.raw.abstract04);}

            if(btnId==5){is =
getResources().openRawResource(R.raw.abstract05);}
            if(btnId==6){is =
getResources().openRawResource(R.raw.abstract06);}
```

```
            if(btnId==7){is =
getResources().openRawResource(R.raw.abstract07);}
            if(btnId==8){is =
getResources().openRawResource(R.raw.abstract08);}

            if(btnId==9){is =
getResources().openRawResource(R.raw.abstract09);}
            if(btnId==10){is =
getResources().openRawResource(R.raw.abstract10);}
            if(btnId==11){is =
getResources().openRawResource(R.raw.abstract11);}
            if(btnId==12){is =
getResources().openRawResource(R.raw.abstract12);}

            if(btnId==13){is =
getResources().openRawResource(R.raw.abstract13);}
            if(btnId==14){is =
getResources().openRawResource(R.raw.abstract14);}
            if(btnId==15){is =
getResources().openRawResource(R.raw.abstract15);}
            if(btnId==16){is =
getResources().openRawResource(R.raw.abstract16);}

            if(btnId==17){is =
getResources().openRawResource(R.raw.abstract17);}
            if(btnId==18){is =
getResources().openRawResource(R.raw.abstract18);}
            if(btnId==19){is =
getResources().openRawResource(R.raw.abstract19);}
            if(btnId==20){is =
getResources().openRawResource(R.raw.abstract20);}

            if(btnId==21){is =
getResources().openRawResource(R.raw.abstract21);}
            if(btnId==22){is =
getResources().openRawResource(R.raw.abstract22);}
            if(btnId==23){is =
getResources().openRawResource(R.raw.abstract23);}
            if(btnId==24){is =
getResources().openRawResource(R.raw.abstract24);}
            if(btnId==25){is =
getResources().openRawResource(R.raw.abstract25);}

            BufferedReader input =  new BufferedReader(new
InputStreamReader(is), 1024*8);
            try {
                String line = null;
                while (( line = input.readLine()) != null){
                    contents.append(line);
                    contents.append(sep);
                }
            }catch(Exception e){}
        }
        catch (Exception e){}

        return contents.toString();
    }
```

```
}
```

```
package com.gamification_research;

import com.gamification_research.R;

/**
 * @author Jonathan Cassar
 * @Class Name: Progress_screen
 * @Description: This class is accountable for the Progress screen activity. It
deals with saving, retrieving and
 * displaying points earned by the user.
 */
public class Progress_screen extends Activity implements
OnSeekBarChangeListener, OnClickListener
{
    //VARIABLES
    TextView cnt_intro, cnt_litreview, cnt_metod, cnt_resul, cnt_concl,
             wrd_intro, wrd_litreview, wrd_metod, wrd_resul, wrd_concl;

    SeekBar bar_intro, bar_litreview, bar_metod, bar_result, bar_concl;

    ImageView img_intro_red, img_intro_yel, img_intro_grn;
    ImageView img_lit_red, img_lit_yel, img_lit_grn;
    ImageView img_met_red, img_met_yel, img_met_grn;
    ImageView img_res_red, img_res_yel, img_res_grn;
    ImageView img_con_red, img_con_yel, img_con_grn;
    ImageView btn_help;

    SharedPreferences getData;
    Data d;
    String user, total_points, daily_points;
    int words, intro, litreview, metod, result, concl;
    float per_intro, per_lit, per_met, per_res, per_con;
    int start, stop;


    //[1] ON CREATE METHOD
    /**
     * @Method name: onCreate
     * @param Bundle savedInstances
     * @return null
     * @Description: On Create method.
     */
    protected void onCreate(Bundle savedInstanceState)
    {
        // TODO Auto-generated method stub
        super.onCreate(savedInstanceState);
        setContentView(R.layout.progress_screen);
        getData =
PreferenceManager.getDefaultSharedPreferences(getBaseContext());

        //Get References

        //references - word counter
        cnt_intro =(TextView)findViewById(R.id.txt_intro_count_progress);
        cnt_litreview =(TextView)findViewById(R.id.txt_lit_count_progress);
        cnt_metod =(TextView)findViewById(R.id.txt_met_count_progress);
        cnt_resul =(TextView)findViewById(R.id.txt_res_count_progress);
```

```
        cnt_concl =(TextView)findViewById(R.id.txt_conc_count_progress);

        //references - words set by user
        wrd_intro =(TextView)findViewById(R.id.txt_intro_progress);
        wrd_litreview =(TextView)findViewById(R.id.txt_lit_progress);
        wrd_metod =(TextView)findViewById(R.id.txt_met_progress);
        wrd_resul =(TextView)findViewById(R.id.txt_res_progress);
        wrd_concl =(TextView)findViewById(R.id.txt_conc_progress);

        //reference - seek bars
        bar_intro = (SeekBar)findViewById(R.id.bar_int_progress);
        bar_litreview = (SeekBar)findViewById(R.id.bar_liter_progress);
        bar_metod = (SeekBar)findViewById(R.id.bar_met_progress);
        bar_result = (SeekBar)findViewById(R.id.bar_res_progress);
        bar_concl = (SeekBar)findViewById(R.id.bar_conc_progress);

        //reference - images
        img_intro_red =  (ImageView)findViewById(R.id.img_red_intro);
        img_intro_yel = (ImageView)findViewById(R.id.img_yellow_intro);
        img_intro_grn = (ImageView)findViewById(R.id.img_green_intro);

        img_lit_red =  (ImageView)findViewById(R.id.img_red_lit);
        img_lit_yel = (ImageView)findViewById(R.id.img_yellow_lit);
        img_lit_grn = (ImageView)findViewById(R.id.img_green_lit);

        img_met_red =  (ImageView)findViewById(R.id.img_red_met);
        img_met_yel = (ImageView)findViewById(R.id.img_yellow_met);
        img_met_grn = (ImageView)findViewById(R.id.img_green_met);

        img_res_red =  (ImageView)findViewById(R.id.img_red_res);
        img_res_yel = (ImageView)findViewById(R.id.img_yellow_res);
        img_res_grn = (ImageView)findViewById(R.id.img_green_res);

        img_con_red =  (ImageView)findViewById(R.id.img_red_con);
        img_con_yel = (ImageView)findViewById(R.id.img_yellow_con);
        img_con_grn = (ImageView)findViewById(R.id.img_green_con);

        btn_help = (ImageView)findViewById(R.id.btn_help_prog);

        //Listeners
        bar_intro.setOnSeekBarChangeListener(this);
        bar_litreview.setOnSeekBarChangeListener(this);
        bar_metod.setOnSeekBarChangeListener(this);
        bar_result.setOnSeekBarChangeListener(this);
        bar_concl.setOnSeekBarChangeListener(this);
        btn_help.setOnClickListener(this);


        //Shared preferences
        user = getData.getString("user_name", "user");

        //methods
        get_data();
        set_data();

    }
```

```
    //[2] GET DATA
    /**
     * @Method name: get_data
     * @param null
     * @return null
     * @Description: Get the data which is to be used by the Progress Activity.
     */
    public void get_data()
    {
        //get data from parameters entered by user
        getData =
PreferenceManager.getDefaultSharedPreferences(getBaseContext());
        words = Integer.parseInt(getData.getString("words", "w"));
        per_intro = (float)Float.parseFloat(getData.getString("intro_length",
"i"))/(float)100;    //divide by 100 to get percent
        per_lit =
(float)Float.parseFloat(getData.getString("lit_review_length", "ir"))/
(float)100;
        per_met =
(float)Float.parseFloat(getData.getString("methodology_length", "m"))/
(float)100;
        per_res = (float)Float.parseFloat(getData.getString("results_length",
"rl"))/(float)100;
        per_con =
(float)Float.parseFloat(getData.getString("conclusion_length", "cl"))/
(float)100;

        d = new Data(this);
        d.open();
        user = d.read_user(getData.getString("user_name", "User"));
        total_points = d.read_total_pts(user);
        daily_points = d.read_daily_pts(user);
        d.close();
    }

    //[3] SET DATA
    /**
     * @Method name: set_data
     * @param null
     * @return null
     * @Description: Set the data which is to be used by the Progress Activity.
     */
    public void set_data()
    {
        cnt_intro.setText("0");
        cnt_litreview.setText("0");
        cnt_metod.setText("0");
        cnt_resul.setText("0");
        cnt_concl.setText("0");

        //distribute the length of the dissertation
        wrd_intro.setText(Integer.toString(Math.round(words*per_intro)));
        wrd_litreview.setText(Integer.toString(Math.round(words*per_lit)));
        wrd_metod.setText(Integer.toString(Math.round(words*per_met)));
        wrd_resul.setText(Integer.toString(Math.round(words*per_res)));
        wrd_concl.setText(Integer.toString(Math.round(words*per_con)));
```

```
        //set seek bar max value
        bar_intro.setMax(Math.round(words*per_intro));
        bar_litreview.setMax(Math.round(words*per_lit));
        bar_metod.setMax(Math.round(words*per_met));
        bar_result.setMax(Math.round(words*per_res));
        bar_concl.setMax(Math.round(words*per_con));

        //set progress bars
        d.open();
        try{bar_intro.setProgress(Integer.parseInt(d.read_intro_state(user)));}c
atch(Exception e){}
        try{bar_litreview.setProgress(Integer.parseInt(d.read_litreview_state(u
ser)));}catch(Exception e){}
        try{bar_metod.setProgress(Integer.parseInt(d.read_metod_state(user)));}c
atch(Exception e){}
        try{bar_result.setProgress(Integer.parseInt(d.read_result_state(user)))
}catch(Exception e){}
        try{bar_concl.setProgress(Integer.parseInt(d.read_concl_state(user)));}c
atch(Exception e){}
    }

    //[4] PROGRESS BAR HANDLER
    /**
     * @Method name: onProgressChanged
     * @param seek bar, integer, boolean
     * @return null
     * @Description: Handles the Seek Bar changes and updates the number of
words
     */
    public void onProgressChanged(SeekBar bar, int progress, boolean fromUser)
{

        switch(bar.getId())
        {
        case R.id.bar_int_progress:
            cnt_intro.setText(""+progress); //show words
            intro = progress;
            check_images();
            break;
        case R.id.bar_liter_progress:
            cnt_litreview.setText(""+progress);
            litreview = progress;
            check_images();
            break;
        case R.id.bar_met_progress:
            cnt_metod.setText(""+progress);
            metod = progress;
            check_images();
            break;
        case R.id.bar_res_progress:
            cnt_resul.setText(""+progress);
            result = progress;
            check_images();
            break;
        case R.id.bar_conc_progress:
            cnt_concl.setText(""+progress);
            concl = progress;
```

```
            check_images();
            break;
        }
    }

    //[5] ON START HANDLER
    /**
     * @Method name: onStartTrackingTouch
     * @param seek bar,
     * @return null
     * @Description: On start interacting with the seek button get the data
from the database.
     */
    public void onStartTrackingTouch(SeekBar bar)
    {
    }

    //[6] ON STOP HANDLER
    /**
     * @Method name: onStopTrackingTouch
     * @param seek bar
     * @return null
     * @Description: According to the state of the seek bar earn points and
save points.
     */
    public void onStopTrackingTouch(SeekBar bar)
    {
        switch(bar.getId())
        {
        case R.id.bar_int_progress:
            stop = intro;
            earn_points();
            save_pts_database();
            save_intro_sate(stop);
            break;
        case R.id.bar_liter_progress:
            stop = litreview ;
            earn_points();
            save_pts_database();
            save_lit_sate(stop);
            break;
        case R.id.bar_met_progress:
            stop = metod;
            earn_points();
            save_pts_database();
            save_metod_sate(stop);
            break;
        case R.id.bar_res_progress:
            stop = result;
            earn_points();
            save_pts_database();
            save_res_sate(stop);
            break;
        case R.id.bar_conc_progress:
            stop = concl;
            earn_points();
            save_pts_database();
```

```
            save_conc_sate(stop);
            break;
        }
    }

    //[7] CHECK IMAGES
    /**
     * @Method name: check_images
     * @param null
     * @return null
     * @Description: Check if the seek bar has reached max and if so display
image.
     */
    public void check_images()
    {
        //Image for Introduction
        if(intro>(int)Math.round((words*per_intro)*0.35)){img_intro_red.setVisi
bility(View.INVISIBLE);}
        if(intro>(int)Math.round((words*per_intro)*0.80)){img_intro_yel.setVisi
bility(View.INVISIBLE);}

        //Image for Lit Review
        if(litreview>Math.round((words*per_lit)*0.35)){img_lit_red.setVisibilit
y(View.INVISIBLE);}
        if(litreview>Math.round((words*per_lit)*0.80)){img_lit_yel.setVisibilit
y(View.INVISIBLE);}

        //Image for Methodology
        if(metod>Math.round((words*per_met)*0.35)){img_met_red.setVisibility(Vi
ew.INVISIBLE);}
        if(metod>Math.round((words*per_met)*0.80)){img_met_yel.setVisibility(Vi
ew.INVISIBLE);}

        //Image for Result
        if(result>Math.round((words*per_res)*0.35)){img_res_red.setVisibility(V
iew.INVISIBLE);}
        if(result>Math.round((words*per_res)*0.80)){img_res_yel.setVisibility(V
iew.INVISIBLE);}

        //Image for Conclusion
        if(concl>Math.round((words*per_con)*0.35)){img_con_red.setVisibility(Vi
ew.INVISIBLE);}
        if(concl>Math.round((words*per_con)*0.80)){img_con_yel.setVisibility(Vi
ew.INVISIBLE);}

    }

    //[8] EARN POINTS
    /**
     * @Method name: earn_points
     * @param null
     * @return null
     * @Description: Earn points.
     */
    public void earn_points()
    {
        //Introduction pts
```

```
        if(intro>Math.round((words*per_intro)*0.35) &
intro<Math.round((words*per_intro)*0.80)){update_points(10);
        }else{if (intro>=Math.round((words*per_intro)*0.80) &
intro<Math.round((words*per_intro))){update_points(40);
        }else{if (intro ==
Math.round((words*per_intro))){update_points(150);}}}

        //Lit Review pts
        if(litreview>Math.round((words*per_lit)*0.35) &
litreview<Math.round((words*per_lit)*0.80)){update_points(10);
        }else{if (litreview>=Math.round((words*per_lit)*0.80) &
litreview<Math.round((words*per_lit))){update_points(40);
        }else{if (litreview ==
Math.round((words*per_lit))){update_points(150);}}}

        //Methodology pts
        if(metod>Math.round((words*per_met)*0.35) &
metod<Math.round((words*per_met)*0.80)){update_points(10);
        }else{if (metod>=Math.round((words*per_met)*0.80) &
metod<Math.round((words*per_met))){update_points(40);
        }else{if (metod == Math.round((words*per_met))){update_points(150);}}}

        //Results pts
        if(result>Math.round((words*per_res)*0.35) &
result<Math.round((words*per_res)*0.80)){update_points(10);
        }else{if (result>=Math.round((words*per_res)*0.80) &
result<Math.round((words*per_res))){update_points(40);
        }else{if (result == Math.round((words*per_res))){update_points(150);}}}

        //Conclusion pts
        if(concl>Math.round((words*per_con)*0.35) &
concl<Math.round((words*per_con)*0.80)){update_points(10);
        }else{if (concl>=Math.round((words*per_con)*0.80) &
concl<Math.round((words*per_con))){update_points(40);
        }else{if (concl == Math.round((words*per_con))){update_points(150);}}}

    }

    //[9] UPDATE POINTS
    /**
     * @Method name: update_points
     * @param points
     * @return null
     * @Description: Update accumulated points with earned ones.
     */
    public void update_points(int points)
    {
        int total = 0;
        int day = 0;

        //convert retrieved data from database to string
        try{total = Integer.parseInt(total_points);
        day = Integer.parseInt(daily_points);
        }catch(Exception e){};

        //Add points
        day+=points;
```

```
    total+=points;

    //convert back to string
    try{
        daily_points= Integer.toString(day);
        total_points = Integer.toString(total);
    }catch(Exception e){}

}

//[10] SAVE POINTS TO DATABASE
/**
 * @Method name: save_pts_database
 * @param null
 * @return null
 * @Description: Save points to database.
 */
public void save_pts_database()
{
    d = new Data(this);
    d.open();
    d.write_daily(user,daily_points);
    d.write_total(user,total_points);
    d.close();
}

//[11A] SAVE INTRO STATE
/**
 * @Method name: save_intro_state
 * @param null
 * @return null
 * @Description: Call database method to save introduction state.
 */
public void save_intro_sate(int state)
{
    try{
        d.open();
        d.write_intro_state(user,Integer.toString(state));
        d.close();
    }catch(Exception e){}
}

//[11B] SAVE LITERATURE REVIEW STATE
/**
 * @Method name: save_lit_state
 * @param null
 * @return null
 * @Description: Call database method to save lit review state.
 */
public void save_lit_sate(int state)
{
    try{
        d.open();
        d.write_literview_state(user,Integer.toString(state));
        d.close();
    }catch(Exception e){}
}
```

```
//[11C] SAVE METHODOLOGY STATE
/**
 * @Method name: save_metod_state
 * @param null
 * @return null
 * @Description: Call database method to save methodology state.
 */
public void save_metod_sate(int state)
{
    try{
        d.open();
        d.write_metod_state(user,Integer.toString(state));
        d.close();
    }catch(Exception e){}
}

//[11D] SAVE RESULTS STATE
/**
 * @Method name: save_res_state
 * @param null
 * @return null
 * @Description: Call database method to save results state.
 */
public void save_res_sate(int state)
{
    try{
        d.open();
        d.write_res_state(user,Integer.toString(state));
        d.close();
    }catch(Exception e){}
}

//[11E] SAVE CONCLUSION STATE
/**
 * @Method name: save_conc_state
 * @param null
 * @return null
 * @Description: Call database method to save conclusion state.
 */
public void save_conc_sate(int state)
{
    try{
        d.open();
        d.write_conc_state(user,Integer.toString(state));
        d.close();
    }catch(Exception e){}
}

//[12] ON CLICK VIEW
/**
 *
 */
public void onClick(View arg0)
{
    switch(arg0.getId())
    {
```

```
        case R.id.btn_help_prog:
            try{message(arg0);}catch(Exception e){e.printStackTrace();};
            break;
        }

    }

    //[13] HELP MESSAGE
    /**
     * Method name: message
     * @param View
     * @return null
     * @Description: Set the help
     */
    public void message(View view)
    {
        AlertDialog d = new AlertDialog.Builder(this).create();
        d.setTitle("Progress");
        d.setMessage(
                    "Slide the sliders to the right."+
                    " In the process, earn 10, 40 and 150 points if you
complete more than 35%, 80% and 100% of the chapter respectively."+
                    " Write, and turn all to green.");

        d.setButton("OK", new DialogInterface.OnClickListener()
        {
            public void onClick(DialogInterface dialog, int which)
            {
                //REMAIN AT LOGIN SCREEN
            }
        });
        d.setIcon(R.drawable.help);
        d.show();
    }
}
```

```java
package com.gamification_research;

import com.gamification_research.R;

@SuppressLint("NewApi")
public class Settings_screen extends PreferenceActivity
{
    Data d;
    SharedPreferences getData;

    @SuppressWarnings("deprecation")
    public void onCreate(Bundle savedInstanceState)
    {
        // TODO Auto-generated method stub
        super.onCreate(savedInstanceState);
        addPreferencesFromResource(R.xml.preferences);
    }
}
```

```java
 * Copyright 2011 Google Inc.

package com.gamification_research;

import android.content.Context;

/**
 * Custom view that draws a vertical time "ruler" representing the
chronological
 * progression of a single day. Usually shown along with {@link BlockView}
 * instances to give a spatial sense of time.
 */
public class Time_Ruler extends View {

    private int mHeaderWidth = 70;
    private int mHourHeight = 45;
    private int mLabelTextSize = 20;
    private int mLabelPaddingLeft = 5;
    private int mLabelColor = Color.BLACK;
    private int mDividerColor = Color.LTGRAY;
    private int mStartHour = 0;
    private int mEndHour = 24;
    private Paint mDividerPaint = new Paint();
    private Paint mLabelPaint = new Paint();

    //CONSTRUCTOR 1
    public Time_Ruler(Context context)
    {
        this(context, null);
    }

    //CONSTRUCTOR 2
    public Time_Ruler(Context context, AttributeSet attrs)
    {
        this(context, attrs, 0);
    }

    //CONSTROCTOR 3
    public Time_Ruler(Context context, AttributeSet attrs, int defStyle)
    {
        super(context, attrs, defStyle);
    }

    //DRAW
    @Override
    protected synchronized void onDraw(Canvas canvas) {
        super.onDraw(canvas);

        final int hourHeight = mHourHeight;

        final Paint dividerPaint = mDividerPaint;
        dividerPaint.setColor(mDividerColor);
        dividerPaint.setStyle(Style.FILL);

        final Paint labelPaint = mLabelPaint;
        labelPaint.setColor(mLabelColor);
        labelPaint.setTextSize(mLabelTextSize);
```

```
        labelPaint.setTypeface(Typeface.DEFAULT_BOLD);
        labelPaint.setAntiAlias(true);

        final FontMetricsInt metrics = labelPaint.getFontMetricsInt();
        final int labelHeight = Math.abs(metrics.ascent);
        final int labelOffset = labelHeight + mLabelPaddingLeft;

        final int right = getRight();

        // Walk left side of canvas drawing time stamps
        final int hours = mEndHour - mStartHour;
        for (int i = 0; i <= hours; i++) {
            final int dividerY = hourHeight * i;
            final int nextDividerY = hourHeight * (i + 1);
            canvas.drawLine(0, dividerY, right, dividerY, dividerPaint);

            // draw text title for time stamp
            canvas.drawRect(0, dividerY, mHeaderWidth, nextDividerY,
dividerPaint);


            // 24-hour mode when set in framework.
            final int hour = mStartHour + i;
            String label;
            if (hour == 0) {
                label = "12am";
            } else if (hour <= 11) {
                label = hour + "am";
            } else if (hour == 12) {
                label = "12pm";
            } else {
                label = (hour - 12) + "pm";
            }

            final float labelWidth = labelPaint.measureText(label);

            canvas.drawText(label, 0, label.length(), mHeaderWidth - labelWidth
                    - mLabelPaddingLeft, dividerY + labelOffset, labelPaint);
        }
    }
}
```