Report Laboratory Work 2

Elena Morelli

1. During the execution of the request on indicator should show a tringle

```
        case LOADING:
            paint = new Paint();
            loadingIndicator(canvas, paint, width, height);
            break;
```

Creation of a new case in the Indicating View

```
private void loadingIndicator(Canvas canvas, Paint paint, int width, int height){
    paint.setColor(Color.BLUE);
    paint.setStyle(Paint.Style.FILL_AND_STROKE);

    paint.setStrokeWidth(10f);

    Point point1_draw = new Point( x: width/2, y: 0);
    Point point2_draw = new Point( x: 0,height);
    Point point3_draw = new Point(width,height);

    Path path = new Path();
    path.moveTo(point1_draw.x,point1_draw.y);
    path.lineTo(point2_draw.x,point2_draw.y);
    path.lineTo(point3_draw.x,point3_draw.y);
    path.lineTo(point1_draw.x,point1_draw.y);
    path.close();

    canvas.drawPath(path,paint);
}
```

Method used to create the triangle

```
    public interface RequestOperatorListener{
        void success (List<ModelPost> publication);
        void failed (int responseCode);
        void loading();
    }
```

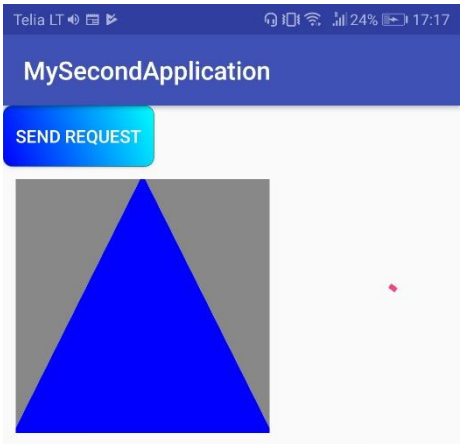
Added the loading declaration in the interface in the RequestOperator class

```
@Override
public void run() {
    super.run();
    loading();
    try{
        List<ModelPost> publication = request();
        if(publication != null){
            success(publication);
        }else{
            failed(responseCode);
        }
    }catch(IOException E){
        failed( code: -1);
    }catch (JSONException e){
        failed( code: -2);
    }
}
```

Call of the loading method inside the run method

```
@Override
public void loading() {
    setIndicatorStatus(IndicatingView.LOADING);
```

Setting the indicator in the Main Activity



2. Work with URL-address of request and the resulting Json should be JSONArray. This array should be made a list of publications. Create a new indicator which should show the number of publications.

```
public List<ModelPost> parsingJsonObject(String response) throws JSONExcepti
    JSONArray array = new JSONArray(response);
    for (int x = 0; x< array.length(); x++ ){
        JSONObject object = array.optJSONObject(x);
        ModelPost post = new ModelPost();

        post.setId(object.optInt( name: "id", fallback: 0));
        post.setUserId(object.optInt( name: "userId", fallback: 0));

        post.setTitle(object.getString( name: "title"));
        post.setBodyText(object.getString( name: "body"));

        list.add(post);
    }
    return list;
}
```

Created a JsonArray. For each jsonObject inside the array a Model Post object is crated.

```
listView = (ListView) findViewById(R.id.list_view);
adapter = new PublicationAdapter( context: this,list);
listView.setAdapter(adapter);
```

Created an adapter class called publicationAdapter

Added all the publications in the list inside the adapter and notify the adapter

```
<shape xmlns:android="http://schemas.android.com/apk/res/android"
    android:shape="oval" >
    <solid android:color="#000" />

</shape>
```

Created the black circle with the list size

```
<TextView
    android:id="@+id/circle"
    android:layout_width="60dp"
    android:layout_height="60dp"
    android:layout_gravity="center"
    android:background="@drawable/circle"
    android:gravity="center"
    android:shadowRadius="10.0"
    android:text="4"
    android:textColor="@android:color/white"
    android:textSize="24sp" />
```

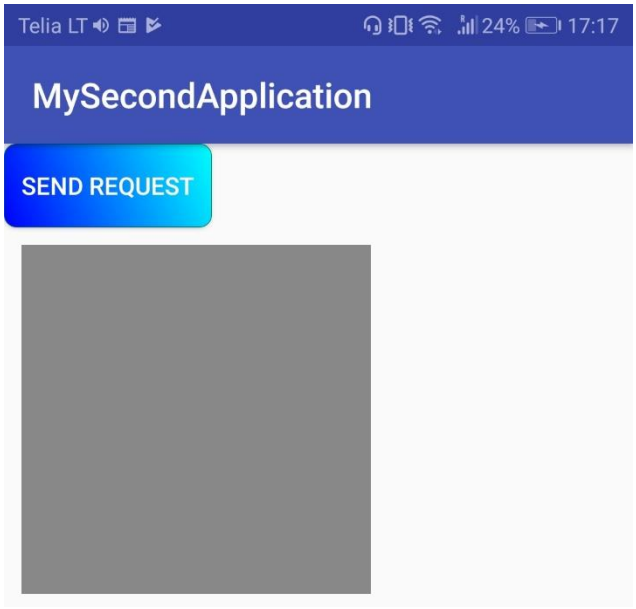
```
public void updatePublication(){
    runOnUiThread(() -> {
            if(publication != null){
                circle.setVisibility(View.VISIBLE);
                circle.setText(String.valueOf(publication.size()));
                list.addAll(publication);
                adapter.notifyDataSetChanged();
                progressBar.setVisibility(View.INVISIBLE);
            } /* else {
                title.setText("");
                bodyText.setText("");
            }*/
    });
}
```

Set the circle visible

Added all the publications to the adapter list

3. Change the design of “Send Request” button so that the button has a gradient colour when it is not pressed

```xml
<?xml version="1.0" encoding="utf-8"?>
<shape
    xmlns:android="http://schemas.android.com/apk/res/android" android:layout_width="match_parent">
    <stroke
        android:width="1px"
        android:color="#006030"/>
    <corners
        android:radius="7dp"/>

    <padding
        android:left="1dp"
        android:right="1dp"
        android:top="1dp"
        android:bottom="1dp" />

    <gradient
        android:startColor="#0000ff"
        android:endColor="#00ffff"
        android:angle="45"/>

</shape>
```

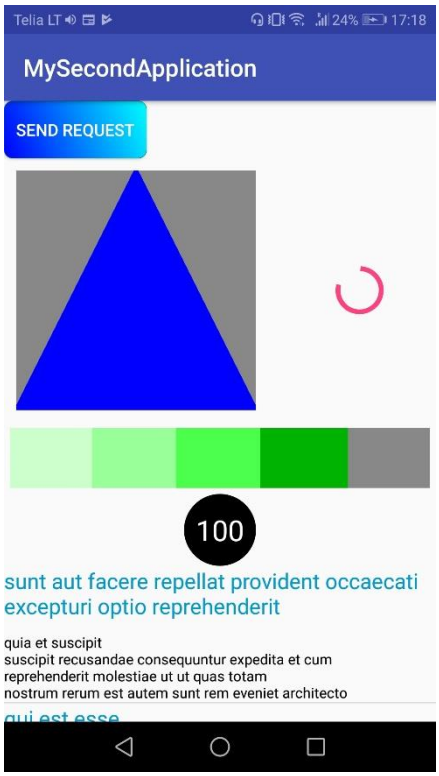


4. Add a progress animation function while the query is executing

```
<ProgressBar
    android:id="@+id/progress_bar"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_gravity="center"
    android:layout_marginLeft="35dp"
    android:indeterminate="false" />
```

```
@Override
public void loading() {
    setIndicatorStatus(IndicatingView.LOADING);
    runOnUiThread(() → { progressBar.setVisibility(View.VISIBLE); });
}
```

```
public void updatePublication(){
    runOnUiThread(() → {
            if(publication != null){
                circle.setVisibility(View.VISIBLE);
                circle.setText(String.valueOf(publication.size()));
                list.addAll(publication);
                adapter.notifyDataSetChanged();
                progressBar.setVisibility(View.INVISIBLE);
            } /* else {
                title.setText("");
                bodyText.setText("");
            }*/
    });
}
```



5. Draw a progress indicator of gradually appearing different colour squares

```
public class ProgressIndicator extends View {
    public static final int NOTEXECUTED = 0;
    public static final int FIRST = 20;
    public static final int SECOND = 40;
    public static final int THIRD = 60;
    public static final int FOURTH = 80;
    public static final int FIFTH = 100;
    private List<Canvas> list = new ArrayList<>();
    int state = NOTEXECUTED;

    public ProgressIndicator(Context context) { super(context); }
    public ProgressIndicator(Context context, @Nullable AttributeSet attrs) {
        super(context, attrs);
    }
    public ProgressIndicator(Context context, @Nullable AttributeSet attrs, int defStyleAttr) {
        super(context, attrs, defStyleAttr);
    }


    protected void onDraw(Canvas canvas) {
        super.onDraw(canvas);
        int width = getWidth()/5;
        int height = getHeight();
        Paint paint;
        switch (state){
            case FIRST:
                paint = new Paint();
                paint.setColor(Color.parseColor( colorString: "#ccffcc"));
                createRectangle(paint, margin: 0,canvas,width,height);
                list.add(canvas);
                break;
```

Created a new class that extends View.

For each switch case a new rectangle is going to be crated as well as the previous ones.

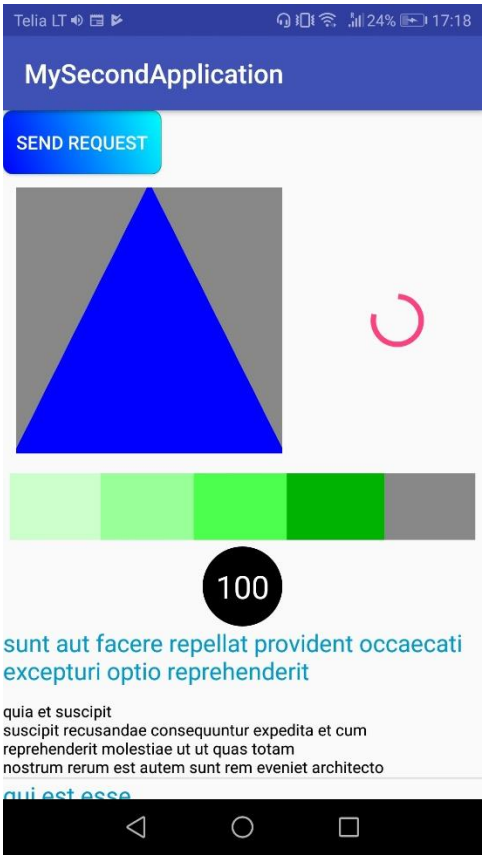
```
            case THIRD:
                paint = new Paint();
                paint.setColor(Color.parseColor( colorString: "#ccffcc"));
                createRectangle(paint, margin: 0,canvas,width,height);
                paint.setColor(Color.parseColor( colorString: "#99ff99"));
                createRectangle(paint,width,canvas, width: width*2,height);
                paint.setColor(Color.parseColor( colorString: "#4dff4d"));
                createRectangle(paint, margin: width*2,canvas, width: width*3,height);
                break;

            case FOURTH:
                paint = new Paint();
                paint.setColor(Color.parseColor( colorString: "#ccffcc"));
                createRectangle(paint, margin: 0,canvas,width,height);
                paint.setColor(Color.parseColor( colorString: "#99ff99"));
                createRectangle(paint,width,canvas, width: width*2,height);
                paint.setColor(Color.parseColor( colorString: "#4dff4d"));
                createRectangle(paint, margin: width*2,canvas, width: width*3,height);
                paint.setColor(Color.parseColor( colorString: "#00b300"));
                createRectangle(paint, margin: width*3,canvas, width: width*4,height);
                break;

            case FIFTH:
                paint = new Paint();
                paint.setColor(Color.parseColor( colorString: "#ccffcc"));
                createRectangle(paint, margin: 0,canvas,width,height);
                paint.setColor(Color.parseColor( colorString: "#99ff99"));
                createRectangle(paint,width,canvas, width: width*2,height);
                paint.setColor(Color.parseColor( colorString: "#4dff4d"));
                createRectangle(paint, margin: width*2,canvas, width: width*3,height);
                paint.setColor(Color.parseColor( colorString: "#00b300"));
                createRectangle(paint, margin: width*3,canvas, width: width*4,height);
                paint.setColor(Color.parseColor( colorString: "#006600"));
                createRectangle(paint, margin: width*4,canvas, width: width*5,height);
                break;
```

```
        case FIFTH:
            paint = new Paint();
            paint.setColor(Color.parseColor( colorString: "#ccffcc"));
            createRectangle(paint, margin: 0,canvas,width,height);
            paint.setColor(Color.parseColor( colorString: "#99ff99"));
            createRectangle(paint,width,canvas, width: width*2,height);
            paint.setColor(Color.parseColor( colorString: "#4dff4d"));
            createRectangle(paint, margin: width*2,canvas, width: width*3,height);
            paint.setColor(Color.parseColor( colorString: "#00b300"));
            createRectangle(paint, margin: width*3,canvas, width: width*4,height);
            paint.setColor(Color.parseColor( colorString: "#006600"));
            createRectangle(paint, margin: width*4,canvas, width: width*5,height);
            break;
        default:
            break;

    }
}

public void createRectangle(Paint paint,int margin,Canvas canvas,int width, int height){
    paint.setStyle(Paint.Style.FILL_AND_STROKE);
    paint.setStrokeWidth(10f);
    canvas.drawRect(margin,  top: 0, width, height, paint);
}

public void setState(int state){this.state = state;}
```

```
View.OnClickListener requestButtonClicked = (v) -> {
        sendRequest();
        if (isTimerRunning){
            timer.cancel();
            timer.purge();
            progressStatus = 20;
            isTimerRunning = false;
        }
        timer = new Timer();
        timer.schedule(new PeriodicTask(), delay: 0);
        isTimerRunning = true;
};

private class PeriodicTask extends TimerTask {
    @Override
    public void run() {
        progIndicator.setState(progressStatus);
        progressStatus += 20;
        progIndicator.invalidate();
        try {
            Thread.sleep( millis: 650);
        } catch(InterruptedException e) {
            e.printStackTrace();
        }
        timer.schedule(new PeriodicTask(), delay: 0);
    }
}
```

Start a timer every time the button is pressed.

The time schedule a PeriodicTask class that increments the number of rectangle everytime.



6. Classes

- Indicating view

```
public class IndicatingView extends View {

    public static final int NOTEXECUTED = 0;
    public static final int SUCCESS = 1;
    public static final int FAILED = 2;
    public static final int LOADING = 3;
    public static final int NUMBER = 4;

    int state = NOTEXECUTED;

    public IndicatingView (Context context){super(context);}
    public IndicatingView (Context context, AttributeSet attrs){
super(context,attrs);}
    public IndicatingView (Context context, AttributeSet attrs, int
defStyleAttr){super(context,attrs,defStyleAttr);}

    @Override
    protected void onDraw(Canvas canvas) {
        super.onDraw(canvas);
        int width = getWidth();
        int height = getHeight();
        Paint paint;
        switch (state){
            case SUCCESS:
                paint = new Paint();
                paint.setColor(Color.GREEN);
                paint.setStrokeWidth(20f);

                canvas.drawLine(0,0,width/2,height,paint);
                canvas.drawLine(width/2,height,width,height/2,paint);
                break;
            case LOADING:
                paint = new Paint();
                loadingIndicator(canvas, paint, width, height);
                break;

            case FAILED:
                paint = new Paint();
                paint.setColor(Color.RED);
                paint.setStrokeWidth(20f);
                canvas.drawLine(0,0,width,height,paint);
                canvas.drawLine(0,height,width,0,paint);
                break;
            default:
                 break;

        }
    }

    public  int getState(){return state;}

    public void setState(int state){this.state = state;}

    private void loadingIndicator(Canvas canvas, Paint paint, int
width, int height){
        paint.setColor(Color.BLUE);
        paint.setStyle(Paint.Style.STROKE);
        paint.setColor(Color.YELLOW);
```

```
        paint.setStyle(Paint.Style.FILL);

        paint.setStrokeWidth(10f);

        Point point1_draw = new Point(width/2,0);
        Point point2_draw = new Point(0,height);
        Point point3_draw = new Point(width,height);

        Path path = new Path();
        path.moveTo(point1_draw.x,point1_draw.y);
        path.lineTo(point2_draw.x,point2_draw.y);
        path.lineTo(point3_draw.x,point3_draw.y);
        path.lineTo(point1_draw.x,point1_draw.y);
        path.close();

        canvas.drawPath(path,paint);
    }

}
```

- Main Activity

```
public class MainActivity extends AppCompatActivity implements
RequestOperator.RequestOperatorListener{
    Button sendRequestButton;
    TextView title,bodyText,circle;
    private List<ModelPost> publication = new ArrayList<>();
    private List<ModelPost> list = new ArrayList<>();
    private PublicationAdapter adapter;
    private ListView listView;
    private IndicatingView indicator;
    private ProgressIndicator progIndicator;
    private ProgressBar progressBar;
    private Timer timer;
    private TimerTask task;
    private int progressStatus = 20;
    private boolean isTimerRunning = false;


    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.mainactivitydesign);
        circle = (TextView) findViewById(R.id.circle);
        circle.setVisibility(View.INVISIBLE);

        sendRequestButton = (Button) findViewById(R.id.send_request);
        sendRequestButton.setOnClickListener(requestButtonClicked);
        //title = (TextView) findViewById(R.id.title);
        //bodyText = (TextView) findViewById(R.id.body_text);
        progressBar = (ProgressBar) findViewById(R.id.progress_bar);
        indicator = (IndicatingView)
findViewById(R.id.generated_graphic);
        progIndicator = (ProgressIndicator)
findViewById(R.id.progress_indicator);
        listView = (ListView) findViewById(R.id.list_view);
        adapter = new PublicationAdapter(this,list);
        listView.setAdapter(adapter);
        progressBar.setVisibility(View.INVISIBLE);
    }

    View.OnClickListener requestButtonClicked = new
```

```
View.OnClickListener() {
        @Override
        public void onClick(View v) {
            sendRequest();
            if (isTimerRunning){
                timer.cancel();
                timer.purge();
                progressStatus = 20;
                isTimerRunning = false;
            }
            timer = new Timer();
            timer.schedule(new PeriodicTask(),0);
            isTimerRunning = true;
        }
    };

    private class PeriodicTask extends TimerTask {
        @Override
        public void run() {
            progIndicator.setState(progressStatus);
            progressStatus += 20;
            progIndicator.invalidate();
            try {
                Thread.sleep(700);
            } catch(InterruptedException e) {
                e.printStackTrace();
            }
            timer.schedule(new PeriodicTask(), 0);
        }
    }


    private void sendRequest(){
        RequestOperator ro = new RequestOperator();
        ro.setListener(this);
        ro.start();
    }

    public void updatePublication(){
        runOnUiThread(new Runnable() {
            @Override
            public void run() {
                if(publication != null){
                    circle.setVisibility(View.VISIBLE);

circle.setText(String.valueOf(publication.size()));
                    list.addAll(publication);
                    adapter.notifyDataSetChanged();

                } /* else {
                    title.setText("");
                    bodyText.setText("");
                }*/
            }
        });
    }

    @Override
    public void success(List<ModelPost> publication) {
        this.publication = publication;
        setIndicatorStatus(IndicatingView.SUCCESS);
```

```
        updatePublication();
        progressBar.setVisibility(View.INVISIBLE);
    }

    @Override
    public void failed(int responseCode) {
        this.publication = null;
        setIndicatorStatus(IndicatingView.FAILED);
        updatePublication();
        progressBar.setVisibility(View.INVISIBLE);
    }

    @Override
    public void loading() {
        setIndicatorStatus(IndicatingView.LOADING);
        runOnUiThread(new Runnable() {
            @Override
            public void run() {
                progressBar.setVisibility(View.VISIBLE);
            }
        });
    }


    public void setIndicatorStatus(final int status) {
        runOnUiThread(new Runnable() {
            @Override
            public void run() {
                indicator.setState(status);
                indicator.invalidate();
            }
        });
    }
}
```

- Model Post

```
public class ModelPost {
    int id;
    int userId;
    String title;
    String bodyText;

    public ModelPost(){}

    public ModelPost(int id, int userId, String title, String
bodyText) {
        this.id = id;
        this.userId = userId;
        this.title = title;
        this.bodyText = bodyText;
    }

    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }

    public int getUserId() {
```

```
        return userId;
    }

    public void setUserId(int userId) {
        this.userId = userId;
    }

    public String getTitle() {
        return title;
    }

    public void setTitle(String title) {
        this.title = title;
    }

    public String getBodyText() {
        return bodyText;
    }

    public void setBodyText(String bodyText) {
        this.bodyText = bodyText;
    }
}
```

- Progress Indicator

```
public class ProgressIndicator extends View {
    public static final int NOTEXECUTED = 0;
    public static final int FIRST = 20;
    public static final int SECOND = 40;
    public static final int THIRD = 60;
    public static final int FOURTH = 80;
    public static final int FIFTH = 100;
    private List<Canvas> list = new ArrayList<>();
    int state = NOTEXECUTED;

    public ProgressIndicator(Context context) {
        super(context);
    }
    public ProgressIndicator(Context context, @Nullable AttributeSet
attrs) {
        super(context, attrs);
    }
    public ProgressIndicator(Context context, @Nullable AttributeSet
attrs, int defStyleAttr) {
        super(context, attrs, defStyleAttr);
    }


    protected void onDraw(Canvas canvas) {
        super.onDraw(canvas);
        int width = getWidth()/5;
        int height = getHeight();
        Paint paint;
        switch (state){
            case FIRST:
                paint = new Paint();
                paint.setColor(Color.parseColor("#ccffcc"));
                createRectangle(paint,0,canvas,width,height);
                list.add(canvas);
                break;
```

```
            case SECOND:
                paint = new Paint();
                paint.setColor(Color.parseColor("#ccffcc"));
                createRectangle(paint,0,canvas,width,height);
                paint.setColor(Color.parseColor("#99ff99"));
                createRectangle(paint,width,canvas,width*2,height);
                break;

            case THIRD:
                paint = new Paint();
                paint.setColor(Color.parseColor("#ccffcc"));
                createRectangle(paint,0,canvas,width,height);
                paint.setColor(Color.parseColor("#99ff99"));
                createRectangle(paint,width,canvas,width*2,height);
                paint.setColor(Color.parseColor("#4dff4d"));
                createRectangle(paint,width*2,canvas,width*3,height);
                break;

            case FOURTH:
                paint = new Paint();
                paint.setColor(Color.parseColor("#ccffcc"));
                createRectangle(paint,0,canvas,width,height);
                paint.setColor(Color.parseColor("#99ff99"));
                createRectangle(paint,width,canvas,width*2,height);
                paint.setColor(Color.parseColor("#4dff4d"));
                createRectangle(paint,width*2,canvas,width*3,height);
                paint.setColor(Color.parseColor("#00b300"));
                createRectangle(paint,width*3,canvas,width*4,height);
                break;

            case FIFTH:
                paint = new Paint();
                paint.setColor(Color.parseColor("#ccffcc"));
                createRectangle(paint,0,canvas,width,height);
                paint.setColor(Color.parseColor("#99ff99"));
                createRectangle(paint,width,canvas,width*2,height);
                paint.setColor(Color.parseColor("#4dff4d"));
                createRectangle(paint,width*2,canvas,width*3,height);
                paint.setColor(Color.parseColor("#00b300"));
                createRectangle(paint,width*3,canvas,width*4,height);
                paint.setColor(Color.parseColor("#006600"));
                createRectangle(paint,width*4,canvas,width*5,height);
                break;
            default:
                break;

        }
    }

    public void createRectangle(Paint paint,int margin,Canvas
canvas,int width, int height){
        paint.setStyle(Paint.Style.FILL_AND_STROKE);
        paint.setStrokeWidth(10f);
        canvas.drawRect(margin, 0, width, height, paint);
    }

    public void setState(int state){this.state = state;}

}
```

- Publication Adapter

```
public class PublicationAdapter extends ArrayAdapter<ModelPost>{
    private TextView title;
    private TextView body;

    public PublicationAdapter(@NonNull Context context, List<ModelPost>
publications) {
        super(context, R.layout.item, publications);
    }

    @NonNull
    @Override
    public View getView(int position, @Nullable View convertView,
@NonNull ViewGroup parent) {
        View v = convertView;
        if (v == null){
            LayoutInflater inflater = (LayoutInflater)
getContext().getSystemService(Context.LAYOUT_INFLATER_SERVICE);
            v = inflater.inflate(R.layout.item,null);
        }

        final ModelPost item = getItem(position);

        title = v.findViewById(R.id.title);
        body = v.findViewById(R.id.body_text);

        title.setText(item.getTitle());
        body.setText(item.getBodyText());
        return v;
    }
}
```

- Request Operator

```
public class RequestOperator extends Thread {

    public interface RequestOperatorListener{
        void success (List<ModelPost> publication);
        void failed (int responseCode);
        void loading();
    }

    private RequestOperatorListener listener;
    private int responseCode;
    private List <ModelPost> list;

    public void setListener (RequestOperatorListener listener){
        this.listener = listener;
        this.list = new ArrayList<>();
    }

    @Override
    public void run() {
        super.run();
        loading();
        try{
            List<ModelPost> publication = request();
            if(publication != null){
                success(publication);
            }else{
                failed(responseCode);
```

```
            }
        }catch(IOException E){
            failed(-1);
        }catch (JSONException e){
            failed(-2);
        }
    }

    private List<ModelPost> request() throws IOException, JSONException
{
        //URL address
        URL obj = new URL("http://jsonplaceholder.typicode.com/posts");

        HttpURLConnection con = (HttpURLConnection)
obj.openConnection();

        con.setRequestMethod("GET");

        con.setRequestProperty("Content-Type","application/json");

        responseCode = con.getResponseCode();

        System.out.println("Response Code" + responseCode);

        InputStreamReader streamReader;

        if(responseCode == 200){
            streamReader = new InputStreamReader(con.getInputStream());
        } else {
            streamReader = new InputStreamReader(con.getErrorStream());
        }

        BufferedReader in = new BufferedReader(streamReader);
        String inputLine;
        StringBuffer response = new StringBuffer();
        try {
            while ((inputLine = in.readLine()) != null) {
                response.append(inputLine);
                sleep(4);
            }
            in.close();
        }catch (InterruptedException e){
            e.printStackTrace();
        }

       // System.out.println(response.toString());

        if(responseCode == 200){
            return parsingJsonObject(response.toString());
        } else{
            return null;
        }
    }

    public List<ModelPost> parsingJsonObject(String response) throws
JSONException {
        JSONArray array = new JSONArray(response);
        for (int x = 0; x< array.length(); x++ ){
            JSONObject object = array.optJSONObject(x);
            ModelPost post = new ModelPost();
```

```
            post.setId(object.optInt("id",0));
            post.setUserId(object.optInt("userId",0));

            post.setTitle(object.getString("title"));
            post.setBodyText(object.getString("body"));

            list.add(post);
        }
        return list;
    }

    private void failed(int code){
        if(listener != null){
            listener.failed(code);
        }
    }

    private void success(List<ModelPost> publication){
        if(listener != null){
            listener.success(publication);
        }
    }

    private void loading(){
        if (listener != null){
            listener.loading();
        }
    }

}
```