Introduction to Android Development

Using the Lob API

Karen Kong

Install Android Studio

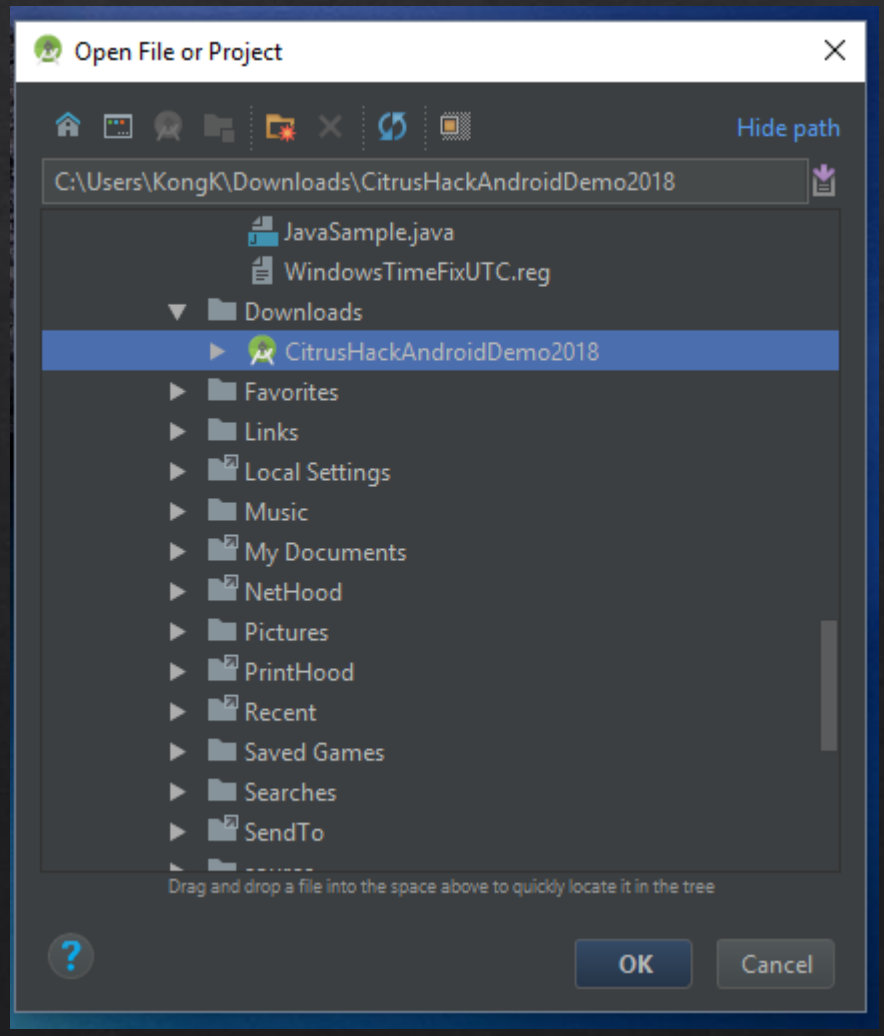
- https://developer.android.com/studio/index.html

Clone/Download the Repository

- https://github.com/kkong006/CitrusHackAndroidDemo2018
- git clone https://github.com/kkong006/CitrusHackAndroidDemo2018.git

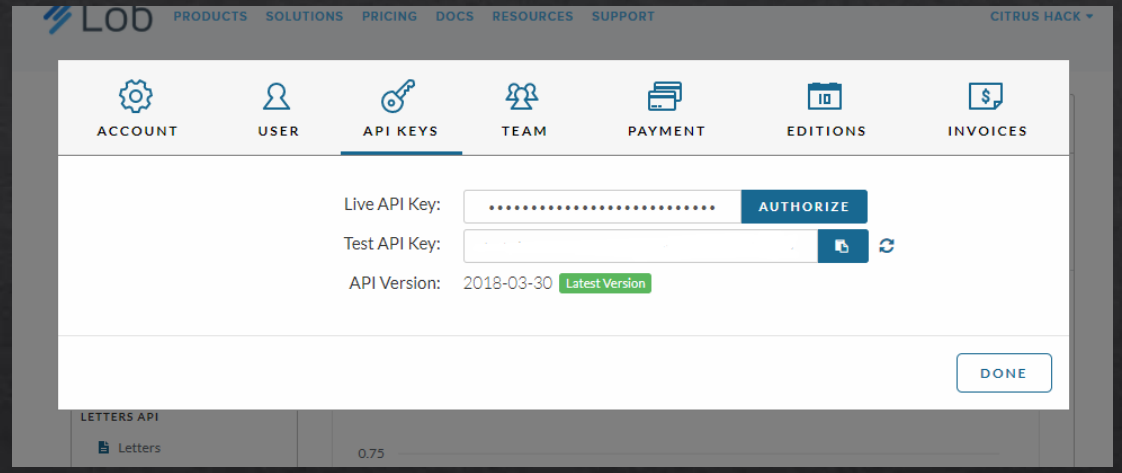
Open the Project in Android Studio

- Open an Existing Android Studio Project
- Select the CitrusHackAndroidDemo2018 folder
- Click “OK”

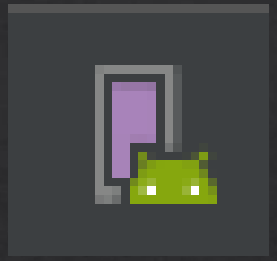


Sign up for a Lob API Account

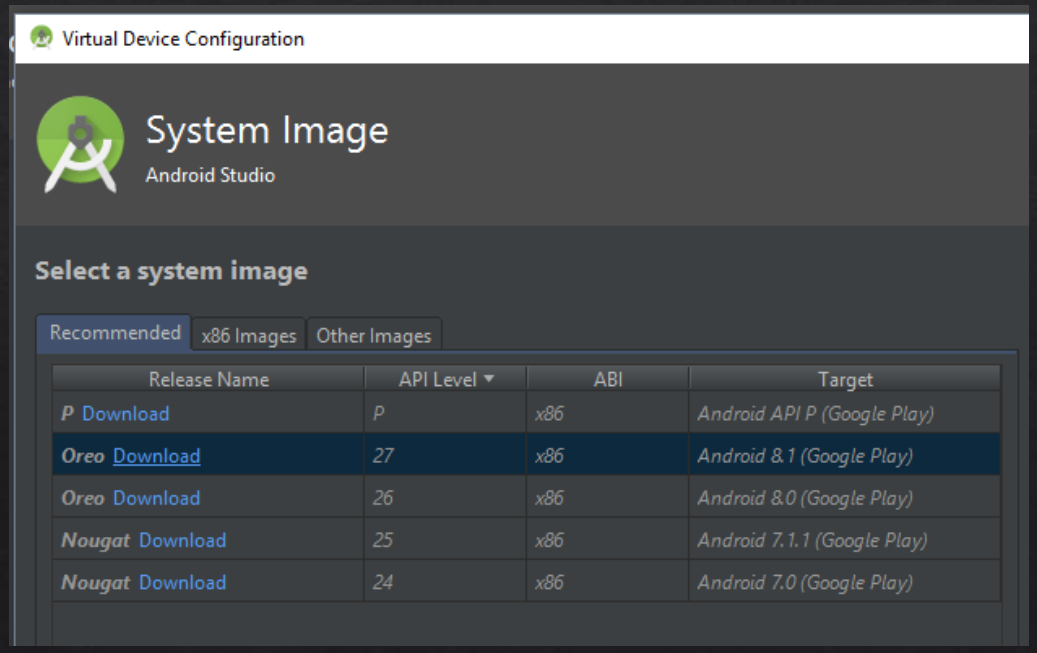
- https://dashboard.lob.com/
- Go to Settings -> API Keys
  - Save the Test API Key
- No credits required for this workshop
- Message Patrick Tumbucon for credits



Create an Emulator



- Tools -> AVD Manager -> Create Virtual Device
- Use recommended settings
- Download Oreo (or any system image from API 21-27)

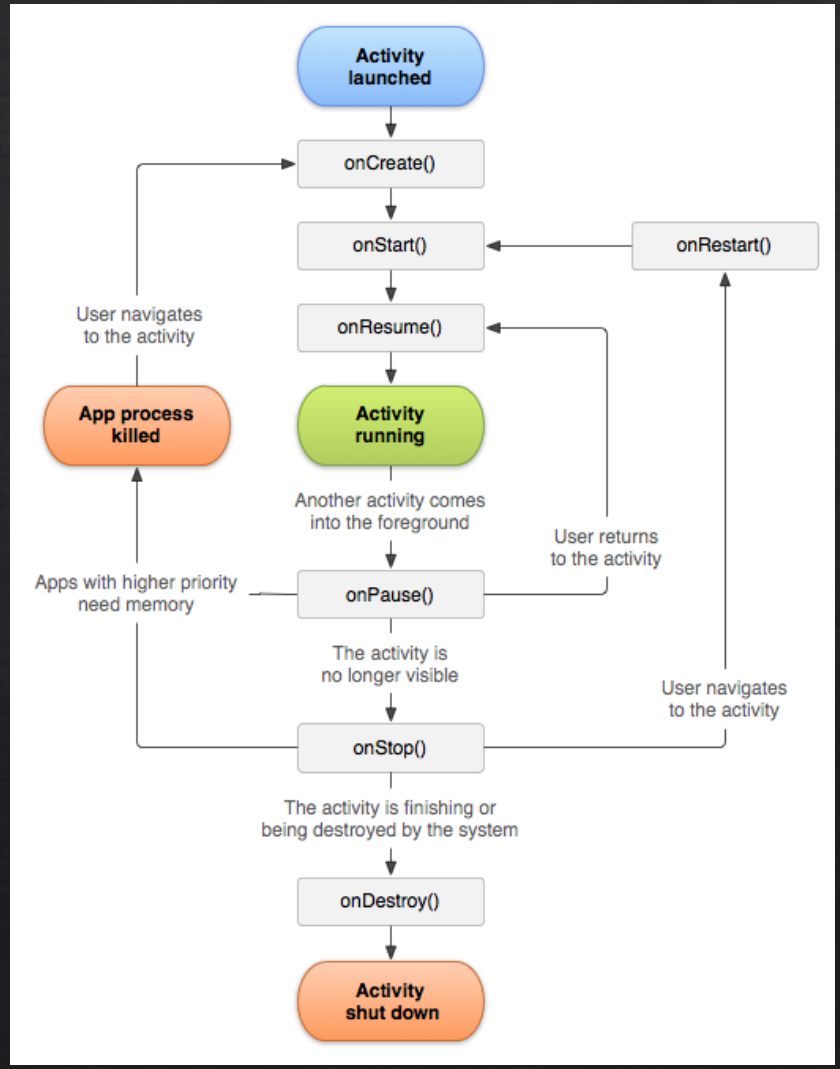


Key Concepts in App Development

- Java/Kotlin/XML
- Activity
- Context
- Layout
- View Elements
- Pagination
- API Client

Activity

- Controller for a page of the app
- Creates a window to put View Elements
- Arranged in a stack
  - Top activity running and others paused
- Activity Lifecycle
  - Active, paused, stopped, killed
- Transition between activities using intents

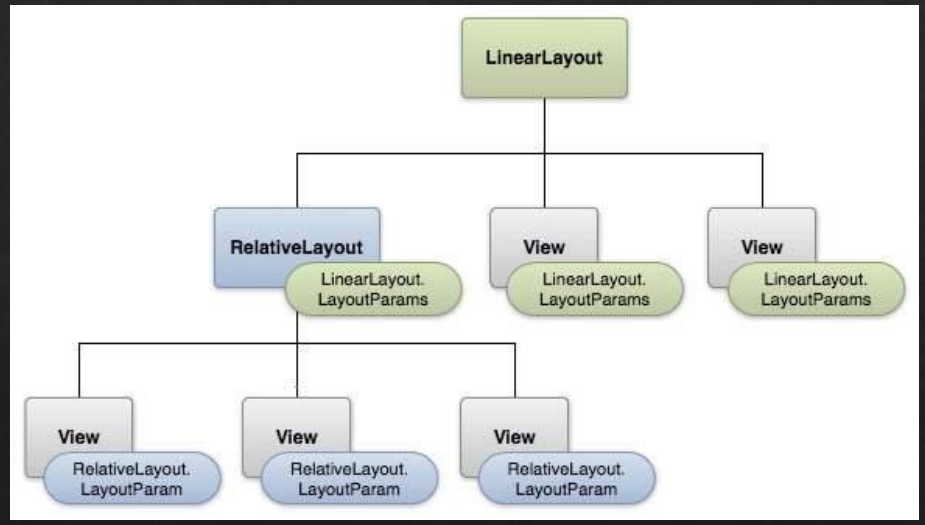


Context

- Global resources
- Access to resources and classes
- In an Activity, access context through the keyword “this”
- In an inner class, getApplicationContext()
- We’ll be passing context from an Activity to access it for this workshop

Layout

- Holds view elements
  - Views: Buttons, TextViews, RecyclerViews…
  - ViewGroup: layouts
- Declared in XML or instantiated at runtime
  - Each element has attributes
- Identify view elements using the ID attribute



View Elements

- TextView: displays text
- EditText: user enters text
- Button: captures presses
- RecyclerView: shows a list of items, memory efficient
  - ListView
- To access view elements from activity code, bind the view elements to Java variables

Pagination

- Infinite scrolling
- Grabbing a long list of items at once & rendering it into views can be slow
- Split up into pages, sets of n items
- Grab the first page & load it into the views, grab the second page & load it into views, …
- Only load what the user wants to see

API Client

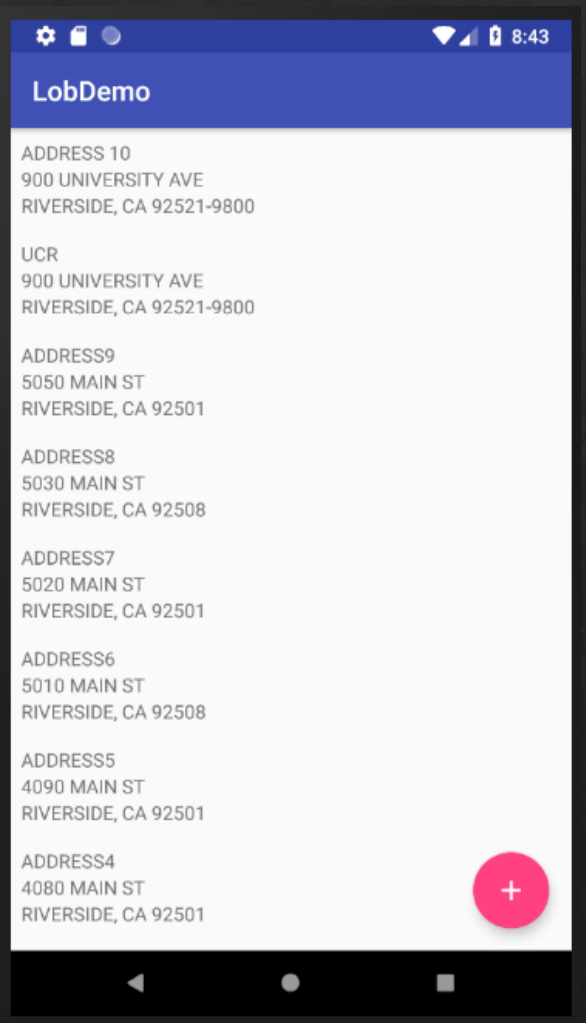
- Used to make API requests
- Authentication
- We'll be sending a request in JSON format a Lob API endpoint
  - Package data in JSON
- We'll receive a response in JSON format
  - Parse JSON to get data

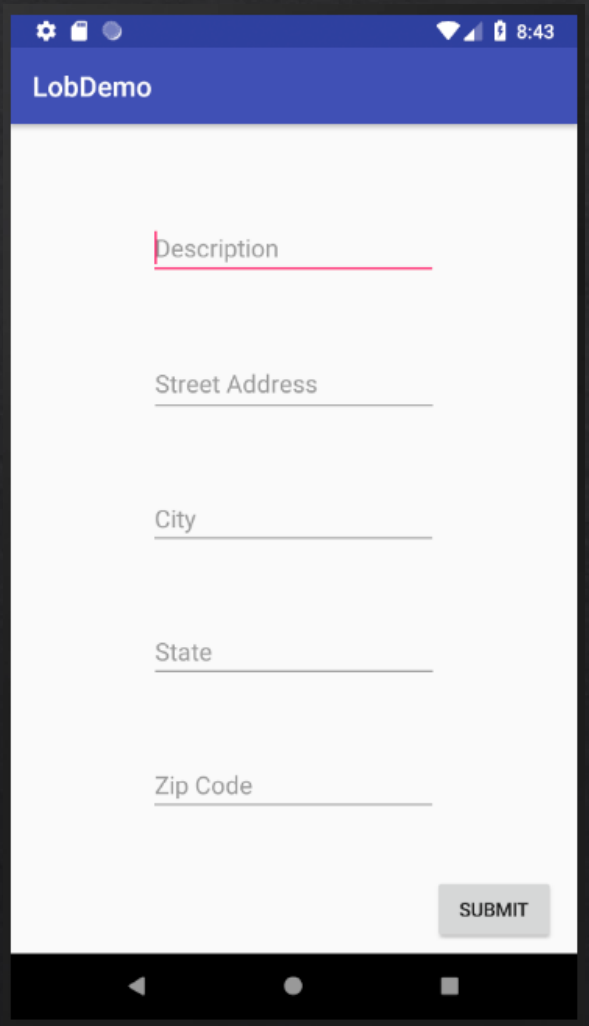
App Overview

- Address Book
- Using Lob API to add, save, and get address book (https://lob.com/docs)
- Java for activities, XML for layouts
- Skeleton code has layouts, activities, and API client outline set up
- Butterknife for view binding (http://jakewharton.github.io/butterknife)
- Android Asynchronous Http Client (http://loopj.com/android-async-http)
- GSON to serialize/deserialize JSON data (http://www.javadoc.io/doc/com.google.code.gson/gson/2.8.2)
- Project Structure

Lob API Calls

- Create an address
- List all addresses
- sync/LobDemoClient.java





Address Object

- https://lob.com/docs#addresses
- Address.java
- Important Attributes
  - name
  - address_line1
  - address_line2
  - address_city
  - address_state
  - address_zip


```
{
  "id": "adr_d3489cd64c791ab5",
  "description": "Harry - Office",
  "name": "HARRY ZHANG",
  "company": "LOB",
  "phone": "5555555555",
  "email": "harry@lob.com",
  "address_line1": "185 BERRY ST STE 6100",
  "address_line2": "",
  "address_city": "SAN FRANCISCO",
  "address_state": "CA",
  "address_zip": "94107-1741",
  "address_country": "UNITED STATES",
  "metadata": {},
  "date_created": "2017-09-05T17:47:53.767Z",
  "date_modified": "2017-09-05T17:47:53.767Z",
  "object": "address"
}
```

Create an Address

- https://lob.com/docs#addresses_create

```
public void createAddress(String description, String address, String city,
                          String state, String zip, Context context, JsonHttpResponseHandler handler) {
    AsyncHttpClient client = new AsyncHttpClient();
    client.setBasicAuth(context.getString(R.string.test_key), context.getString(R.string.password));
    RequestParams params = new RequestParams();
    params.put("name", description);
    params.put("address_line1", address);
    params.put("address_city", city);
    params.put("address_state", state);
    params.put("address_zip", zip);
    client.post( url: "https://api.lob.com/v1/addresses", params, handler);
}
```

List All Addresses

- https://lob.com/docs#addresses_list

```
public void getAddresses(int offset, int limit, Context context, JsonHttpResponseHandler handler) {
    AsyncHttpClient client = new AsyncHttpClient();
    client.setBasicAuth(context.getString(R.string.test_key), context.getString(R.string.password));
    RequestParams params = new RequestParams();
    params.put("offset", offset);
    params.put("limit", limit);
    client.get( url: "https://api.lob.com/v1/addresses", params, handler);
}
```

Add Address Activity

- When the user presses the button
  - Get the text for the address the user typed
  - Use the API client to issue a POST request to the create an address endpoint
  - Handle a response from the API

AddAddressActivity.java

- Bind the views to variables

```
@BindView(R.id.etDescription) EditText mEtDescription;
@BindView(R.id.etAddress) EditText mEtAddress;
@BindView(R.id.etCity)  EditText mEtCity;
@BindView(R.id.etState) EditText mEtState;
@BindView(R.id.etZip) EditText mEtZip;
```

AddAddressActivity.java

- Add the OnClick listener to the button

```
@OnClick(R.id.btSubmit)
void onClickSubmit() {
    String description = mEtDescription.getText().toString();
    String address = mEtAddress.getText().toString();
    String city = mEtCity.getText().toString();
    String state = mEtState.getText().toString();
    String zip = mEtZip.getText().toString();
    LobDemoApp.getLobDemoClient().createAddress(description, address, city, state, zip,
            context: this, new JsonHttpResponseHandler() {
        @Override
        public void onSuccess(int statusCode, Header[] headers, JSONObject response) {
            finish();
        }
    });
}
```

Main Activity

- Set up the RecyclerView
- Get the addresses from the Lob API and load them into the RecyclerView
- Implement pagination for infinite scrolling if the user has a lot of addresses
  - Initial loading
  - Load more pages when needed

MainActivity.java

- Add OnClick listener to the Add button that transitions to AddAddressActivity.java

```
@OnClick(R.id.fabAdd)
void onClickAdd() {
    Intent addAddressIntent = new Intent( packageContext: this, AddAddressActivity.class);
    startActivity(addAddressIntent);
}
```

MainActivity.java Class Variables

```
// TODO: Define a page size
private static final int PAGE_SIZE = 25;

// TODO: Bind the recyclerview to a member variable
@BindView(R.id.rvAddresses) RecyclerView mRvAddresses;

// TODO: Declare the adapter for the recyclerview
private AddressAdapter mAdapter;

// TODO: Declare the layout manager for the recyclerview
private LinearLayoutManager mLayoutManager;

// TODO: Declare booleans isLoading and isLastPage for pagination
private boolean isLoading;
private boolean isLastPage;
```

MainActivity.java OnCreate()

```
// TODO: Create the adapter
mAdapter = new AddressAdapter( context: this, new ArrayList<Address>());

// TODO: Attach the adapter to the recyclerview
mRvAddresses.setAdapter(mAdapter);

// TODO: Set the layout manager
mLayoutManager = new LinearLayoutManager( context: this);
mRvAddresses.setLayoutManager(mLayoutManager);
```

MainActivity.java Pagination

- Add an OnScrollListener to the RecyclerView in OnCreate()

```
mRvAddresses.addOnScrollListener(new RecyclerView.OnScrollListener() {
    @Override
    public void onScrollStateChanged(RecyclerView recyclerView, int newState) {
        super.onScrollStateChanged(recyclerView, newState);
    }

    @Override
    public void onScrolled(RecyclerView recyclerView, int dx, int dy) {
        super.onScrolled(recyclerView, dx, dy);

        // Get the relevant item counts
        int visibleItemCount = mLayoutManager.getChildCount();
        int totalItemCount = mLayoutManager.getItemCount();
        int firstVisibleItem = mLayoutManager.findFirstVisibleItemPosition();

        // If there isn't a request at the moment and we haven't hit the api limit
        if(!isLoading && !isLastPage) {
            // If the user scrolled past the last item, fetch the next page
            if((visibleItemCount + firstVisibleItem) >= totalItemCount
                    && firstVisibleItem >= 0
                    && totalItemCount >= PAGE_SIZE) {
                loadMoreAddresses();
            }
        }
    }
```

MainActivity.java Pagination

```
private void loadAddresses() {
    // Reset the loading state and current page
    isLoading = false;
    isLastPage = false;
    mAdapter.clearAll();

    // Fetch items starting from the first
    loadMoreAddresses();
}
```

MainActivity.java Pagination

```
private void loadMoreAddresses() {
    if(!isLastPage) {
        isLoading = true;
        LobDemoApp.getLobDemoClient().getAddresses(mLayoutManager.getItemCount(), PAGE_SIZE,  context: this, new JsonHttpResponseHandler() {
            @Override
            public void onSuccess(int statusCode, Header[] headers, JSONObject response) {
                // We've received a response and are no longer loading
                isLoading = false;

                // Parse the response
                parseAddresses(response);
            }
        });
    }
}
```

MainActivity.java Pagination

```
private void parseAddresses(JSONObject addresses) {
    try {
        // Cast the response to addresses
        Gson gson = new Gson();
        Type type = new TypeToken<List<Address>>(){}.getType();
        List<Address> newAddresses = gson.fromJson(addresses.getJSONArray( name: "data").toString(), type);
        // Add the new addresses to the adapter
        mAdapter.addAll(newAddresses);
        // If we don't get a full page, there are no results left
        if(newAddresses.size() < PAGE_SIZE) {
            isLastPage = true;
        }
    } catch (JSONException e) {
        e.printStackTrace();
    }
}
```

Extra: UI Tweaks

- Color Theme
- View element attributes
- Layouts