1 Set the paths for .android and .AndroidStudio1.4

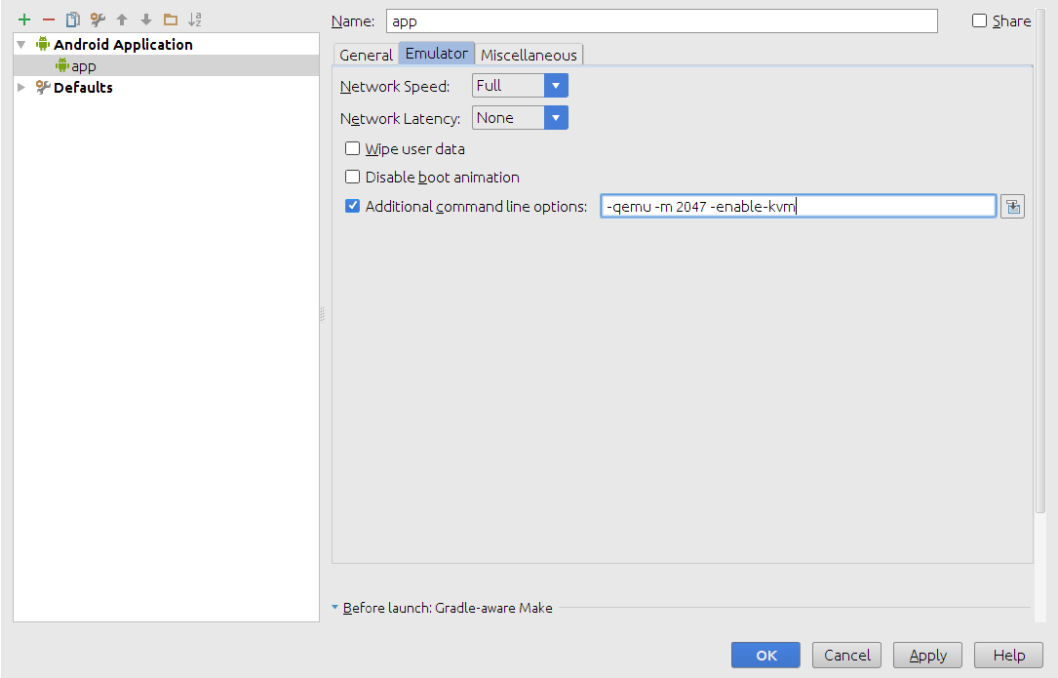
In your home directory, perform the following commands:

```
rm –rf  .android .AndroidStudio1.4
mkdir /home/Local_Data/.android
mkdir /home/Local_Data/.AndroidStudio1.4
ln –s /home/Local_Data/.android .android
ln –s /home/Local_Data/ .AndroidStudio1.4  .AndroidStudio1.4
```

The above will redirect the path from your NFS/homes to the local/home.

2 Setup [10 minutes]

- Open the android studio, /packages/mobserv/ android-studio/studio.sh
- **Set Android location SDK to /packages/mobserv/android-sdk** (cancel if promoted to install an Android SDK)
- Start a new Android studio Project:
  - Application name: MyContactList
  - Company: fr.eurecom
  - Project location: Home/Local_Data/MyContactList
  - Target: phones and tablets API 15
  - Activity: Black Activity
  - Activity name: MainActivity
  - Title: activity_main
  - Finish
- Go the file > project structure
  - Set the java path to /usr/jdk/latest
- Build an Android Virtual Devices:
  - Tools > Android > AVD Manager or Click on AVD manager
  - Create a virtual device
  - Create a New Hardware profile or import the one in android-studio/extra/device.xml
    - Device name: myDevice
    - Screensize: 4.0
    - Resolution : 780x1280
    - RAM: 512
    - Check “Has Hardware Buttons”
    - Check “Has Hardware Keyboard”
    - Uncheck back-facing and front-facing camera
    - Ok
  - System image: marshmallow x86_64, Android 6.0 (with google API)
  - Finish
- In Run > Edit Configurations, Emulator tab, Check Additional command line options and in the input field enter [1]:
  - `-qemu –m 2047 -enable-kvm`



- Run the application
  - Launch emulator : select “myDevice API 23”
  - You should see “HelloWorld”
- Remove the “HelloWorld” form the app

3 HTTP request-response: Access the GAE contactlist from Android

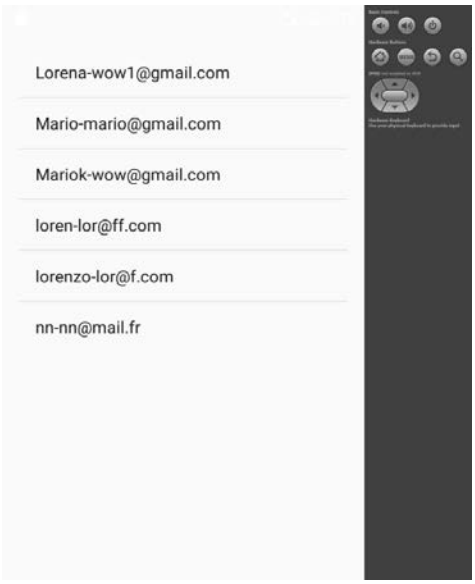
Here, we are building an Android Application capable of getting the contact list from the server, and present it to the user. Further in the lab we are going to add the possibility to modify or delete a contact.

Note: code used here is quick and dirty, and not intended for reusability, maintainability. Exception handling omitted to save space.

All the source code are located the course directory fall_2015/lab_android/lab3/src_code/

3.1 Visualize the information [0h45][SRC CODE task3.1]

In this task we simply visualize the contacts on the server after the application loaded. The result will be the one shown here.



3.1.1 Update the "MainActivity.java" as follows

This is the main view that is presented to the user. It makes use of the loader and the adaptor, and the activity lifecycle. Note the following in the code :

- the application life cycle is used to interact with the adaptor
- the reference to the list view is obtained
- adaptor is set for the a simple list view (**simple_list_item_1**)

```
public class MainActivity extends AppCompatActivity implements LoaderManager.LoaderCallback
s<List<Contact>>, AdapterView.OnItemClickListener {
    private  ListView listView;
    private List <Contact> contacts;


    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);


        setContentView(R.layout.activity_main);
        listView = (ListView) findViewById(R.id.listView1);


    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is present.
        //getMenuInflater().inflate(R.menu.menu_main, menu);
        // return true;
        return false;
    }

    @Override
```

```
    protected void onResume(){
        super.onResume();
        Log.i("main", "onResume");
        getLoaderManager().restartLoader(0,null,this);
    }

    @Override
    public Loader <List<Contact>> onCreateLoader(int id, Bundle args) {

        Log.i("main", "creating loader");
        JsonLoader loader = new JsonLoader(this);
        loader.forceLoad();
        return loader;

    }

    @Override
    public void onLoadFinished(Loader<List<Contact>> arg0, List<Contact> arg1){


        contacts = arg1;
        listView.setAdapter(new ArrayAdapter<Contact>(this, android.R.layout.simple_list_it
em_1,arg1)); //new ContactAdapter(this, R.layout.contact, contacts));


    }
    @Override
    public void onLoaderReset(Loader<List<Contact>> arg0){

    }


    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        // Handle action bar item clicks here. The action bar will
        // automatically handle clicks on the Home/Up button, so long
        // as you specify a parent activity in AndroidManifest.xml.
        int id = item.getItemId();

        //noinspection SimplifiableIfStatement
        if (id == R.id.action_settings) {
            return true;
        }

        return super.onOptionsItemSelected(item);
    }
    @Override
    public void onItemClick(AdapterView<?> parent, View view, int position, long id) {

         Log.i("Main:","someone clicked on item: "+position);

    }
}
```

3.1.2 Create the "JsonLoader.java" class

Create the JsonLoader.java class and add the following code to it. This class will asynchronously get the results in JSON format from the server, retrieve the data, and finally populate the data model "Contact"

```
public class JsonLoader extends AsyncTaskLoader <List<Contact>>{
    public  JsonLoader(Context context){
        super(context);
        Log.i("main","loader created");
    }

    @Override
    public List<Contact> loadInBackground() {
        Log.i("main", "loader in Background");
        URL page = null;
        try {
            page = new URL("YOUR_GAE_URL?respType=json");
        } catch (MalformedURLException e) {
            e.printStackTrace();
        }
        StringBuffer text = new StringBuffer();
        HttpURLConnection conn = null;
        try {
            conn = (HttpURLConnection) page.openConnection();

        } catch (IOException e) {
            e.printStackTrace();
        }
        try {
            conn.connect();
        } catch (IOException e) {
            e.printStackTrace();
        }
        InputStreamReader in = null;
        String jsonString=null;
        try {
            in = new InputStreamReader((InputStream) conn.getContent(),"UTF-8");
            Reader reader = in;
            jsonString = readAll(in);

        } catch (IOException e) {
            e.printStackTrace();
        }
        if (jsonString!=null){
            List<Contact> result;
            try {
                JSONArray listOfContact = new JSONArray(jsonString);
                int len = listOfContact.length();
                result = new ArrayList<Contact>(len);
                Log.i("main","json string length is "+len);
                for(int i=0;i<len;i++) {
                    result.add(new Contact(listOfContact.getJSONObject(i)));
                }
            }catch (JSONException e){
                //TODO handle
                e.printStackTrace();
                return null;


            }
```

```
            return result;

        }else {

              return null;
        }
    }

    private Bitmap loadImage(String imgLink)  {
                //To be implemented

}

    private String readAll(Reader reader) throws IOException {
        StringBuilder builder = new StringBuilder(4096);
        for (CharBuffer buf = CharBuffer.allocate(512); (reader.read(buf)) > -1; buf
                .clear()) {
            builder.append(buf.flip());
        }
        return builder.toString();
    }

}
```

3.1.3 Add the data model class naming it “Contact.java”

This is the class that implements the data model for the contacts.

```
public class Contact implements Serializable {

    private final String id;
    private final String name;
    private final String email;
    private final String phone;

    public Contact(JSONObject jsonObject) throws JSONException {

        id = jsonObject.getString("id");
        phone = jsonObject.getString("phone");
        email = jsonObject.getString("email");
        name = jsonObject.getString("name");

    }
    @Override
    public String toString(){
        return String.format("%s-%s",name,email);
    }

    public String getId() {
        return id;
    }

    public String getName() {
        return name;
    }

    public String getEmail() {
```

```
        return email;
    }

    public String getPhone() {
        return phone;
    }
}

        return builder.toString();
    }

}
```

3.1.4 Add a listview to the activity_main.xml layout

Now we simply update the activity main xml in order to include the listview which we would like to see

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/LinearLayout1"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context=".MainActivity" >

    <ListView
        android:id="@+id/listView1"
        android:layout_width="match_parent"
        android:layout_height="wrap_content" >
    </ListView>

</LinearLayout>
```

3.1.5 Run and test the code

Use your previously created emulator, and run the app, and check if the contactlist is present. Make sure that your GAE app is running and contains data.

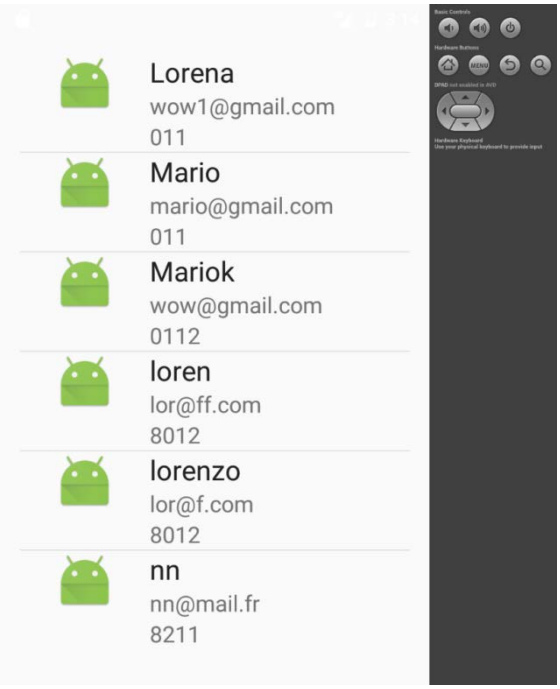
Questons:

- **Why the applications does not work?**
- **What is a loader and what is and adaptor?**
- **When the method "`public List<Contact> loadInBackground(){}`" is called ?**
- **Main activity is implementing "`AdapterView.OnItemClickListener`" in particular the method `onItemClick` is already able to show which contact has been clicked in the list. Why the program is not able to detect when a user clicks over an Item? What is a listener?**

- **To test: change the contact list on the server, and check if they are updated on the client.**
  a. **Use chrome REST client you used before.**

3.2 Adding a new list adapter [30 minutes][SRC CODE task3.2]

We will now make the application "prettier" adding a dedicated contact (array adapter) viewer for the list. At the end of this task the result should be something similar to this:



3.2.1 Create the new layout file "contact.xml"

After created the file copy this body in its text panel and locate it in layout directory.

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

    <!--  -->

    <ImageView
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:id="@+id/user_pic"
        android:layout_weight="1"
        android:src="@android:mipmap/sym_def_app_icon" />

    <LinearLayout
        android:orientation="vertical"
        android:layout_width="fill_parent"
```

```
        android:layout_height="wrap_content"
        android:layout_weight="0.5">

        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:textAppearance="?android:attr/textAppearanceLarge"
            android:text="Test Testing"
            android:id="@+id/input_name" />

        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:textAppearance="?android:attr/textAppearanceMedium"
            android:text="test@test.com"
            android:id="@+id/input_email" />

        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:textAppearance="?android:attr/textAppearanceMedium"
            android:text="+33 9994685785"
            android:id="@+id/input_number" />
    </LinearLayout>
</LinearLayout>
```

3.2.2 Create "ContactAdapter.java"

This class will be an extension of the arrayAdapter and will define which information need to be inserted in the new contact view we are going to use through the created xml file.

```
public class ContactAdapter extends ArrayAdapter<Contact> {

    private final Context context;
    private final int id;
    private final List<Contact> contacts;
    private Contact contact;

    public ContactAdapter(Context context, int id, List<Contact> contacts)
    {
        super(context,id, contacts);
        this.context = context;
        this.id = id;
        this.contacts = contacts;
    }

    @SuppressLint("ViewHolder")
    @Override
    public View getView(int position, View convertView, ViewGroup parent)
    {
        LayoutInflater inflater = (LayoutInflater) context.getSystemService(Context.LAYOUT_
INFLATER_SERVICE);
        View rowView = inflater.inflate(id, parent, false);

        contact = contacts.get(position);
        TextView contactName= (TextView) rowView.findViewById(R.id.input_name);
```

```
        TextView contactEmail= (TextView) rowView.findViewById(R.id.input_email);
        TextView contactNumber= (TextView) rowView.findViewById(R.id.input_number);

        contactName.setText(contact.getName());
        contactEmail.setText(contact.getEmail());
        contactNumber.setText(contact.getPhone());

        return rowView;
    }

}
```

3.2.3 Modify "Main_activity.java" to use the new adapter

```
@Override
public void onLoadFinished(Loader<List<Contact>> arg0, List<Contact> arg1){
    contacts = arg1;
    ContactAdapter Ca; Ca = new ContactAdapter(this,R.layout.contact, contacts);
    listView.setAdapter(Ca); //new ContactAdapter(this, R.layout.contact, contacts));
    //Any additional code you might already have added for previous tasks
}
```

3.2.4 Run the code

Simply verify that the code is running normally.

4 Modify and delete a contact [1h00][SRC CODE task 4]

In this task we are going to give to the app user the possibility, by selecting a contact, to modify it and delete it. MAKE SURE YOUR GOOGLE APP ENGINE APIS ARE WORKING PROPERLY.

4.1 Create the new blank activity "ModifyContactActivity"

This activity will help the user to access to the contact details and, through some text fields to modify them. It will contain also a modify and delete button as show in the picture.

4.1.1 Fill the body of the "`ModifyContactActivity.java`" with this simple code.

```
public class ModifyContactActivity extends AppCompatActivity implements android.view.View.O
nClickListener {
    private Contact contact;
    private EditText name;
    private EditText email;
    private EditText number;
    private EditText url;
    private Button modify;
    private  Button delete;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
```

```
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_modify_contact);
        Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);
        setSupportActionBar(toolbar);
        Intent intent= getIntent();
        contact = (Contact) intent.getSerializableExtra("contact");
        fillForm();
        modify = (Button) findViewById(R.id.modify);
        delete = (Button) findViewById(R.id.delete);
        modify.setOnClickListener(this);
        delete.setOnClickListener(this);
        FloatingActionButton fab = (FloatingActionButton) findViewById(R.id.fab);
        fab.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                Snackbar.make(view, "Replace with your own action", Snackbar.LENGTH_LONG)
                        .setAction("Action", null).show();
            }
        });
    }

    private void fillForm(){
        name = (EditText) findViewById(R.id.name);
        email = (EditText) findViewById(R.id.email);
        number = (EditText) findViewById(R.id.number);
        url = (EditText) findViewById(R.id.url);

        name.setText(contact.getName());
        email.setText(contact.getEmail());
        number.setText(contact.getPhone());
        url.setText("None");

        name.setImeOptions(EditorInfo.IME_ACTION_DONE);
        email.setImeOptions(EditorInfo.IME_ACTION_DONE);
        number.setImeOptions(EditorInfo.IME_ACTION_DONE);
        url.setImeOptions(EditorInfo.IME_ACTION_DONE);
    }

    @Override
    public void onClick(View v){
        Handler handler= new Handler();
        if (v.getId()==R.id.modify){
            Log.i("main", "I clicked modify");
            Map<String,String> data = new HashMap<String,String>();
            data.put("name", name.getText().toString());
            data.put("email",email.getText().toString());
            data.put("phone",number.getText().toString());
            ResponseHandler rh = new ResponseHandler(this,"save?id=" + contact.getId(), "PO
ST", handler, this, data);
            new ModifyTask().execute(rh);

        }
        else{
            ResponseHandler rh = new ResponseHandler(this,"delete?id=" + contact.getId(), "
GET", handler, this, null);
            new ModifyTask().execute(rh);
            Log.i("main", "I clicked delete");
        }
```

```
    }
}
```

4.1.2 Define the view through the "content_modify_contact.xml"

As you can notice this file is automatically created together with the new activity, is sufficient to change its code with this.

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

<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context="fr.eurecom.mycontactlist.ModifyContactActivity" >

    <TextView
        android:id="@+id/title"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textSize="30sp"
        android:layout_alignParentTop="true"
        android:layout_centerHorizontal="true"
        android:text="Modify a contact" />

    <EditText
        android:id="@+id/name"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentLeft="true"
        android:layout_alignParentRight="true"
        android:layout_below="@+id/title"
        android:ems="10"
        android:hint="Name"
        android:inputType="textPersonName" >

        <requestFocus />
    </EditText>

    <EditText
        android:id="@+id/email"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentLeft="true"
        android:layout_alignParentRight="true"
        android:layout_below="@+id/name"
        android:ems="10"
        android:hint="Email"
        android:inputType="textEmailAddress" />

    <EditText
        android:id="@+id/number"
```

```
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentLeft="true"
        android:layout_alignParentRight="true"
        android:layout_below="@+id/email"
        android:ems="10"
        android:hint="Phone Number"
        android:inputType="phone" />

    <EditText
        android:id="@+id/url"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignLeft="@+id/number"
        android:layout_alignRight="@+id/number"
        android:layout_below="@+id/number"
        android:ems="10"
        android:hint="Image's URL"
        android:inputType="textPersonName" />

    <Button
        android:id="@+id/delete"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignBaseline="@+id/modify"
        android:layout_alignBottom="@+id/modify"
        android:layout_toRightOf="@+id/modify"
        android:text="Delete" />

    <Button
        android:id="@+id/modify"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentLeft="true"
        android:layout_below="@+id/url"
        android:layout_margin="20sp"
        android:text="Modify" />

</RelativeLayout
```

4.2 Create a new Class "ResponseHandler.java"

```
public class ResponseHandler {

    private final String url = "THE URL TO YOUR GAE";

    private Context context;
    private String request;
    private String format;
    private Handler handler;
    private Activity activity;
    private Map<String,String> data;
    private int statusCode;

    public Activity getActivity() {
        return activity;
    }
```

```
    public ResponseHandler(Context context, String request, String format, Handler handler,
Activity activity, Map<String,String> data)
    {
        this.context = context;
        this.request = url + request;
        this.format = format;
        this.handler = handler;
        this.activity = activity;
        this.data = data;
    }

    public void finish()
    {
        handler.post(new Runnable() {
            public void run() {
                activity.finish();
            }
        });
    }

    private void popUp(String message)
    {
        Toast.makeText(context, message, Toast.LENGTH_SHORT).show();
    }

    public String start()
    {
        String out = null;
        if(format.compareTo("POST") == 0)
            try {
                out = postData();
                Log.i("Response==", out);
            } catch (IOException e) {
                e.printStackTrace();
            }
        else{
            try {
                deleteData();
            }catch (Exception e){


            }
        }
        return out;
    }

    public void respond(String response)
    {
        /* To be implemented
        * If the statusCode is ok?
        * Notify the user about the successful operation
        * If not simply state that a problem has occurred
        * Remember to conclude the activity
        * */
    }

    private String convertParameters(Map <String,String> data){
```

```
        String result = "";
        for (Map.Entry<String,String> e: data.entrySet()){
            result += e.getKey()+"="+e.getValue()+"&";
        }
        return result.substring(0,result.length()-1);
    }

    public String postData() throws IOException {
        URL urlToRequest = new URL(request);
        HttpURLConnection urlConnection = null;
        String postParameters = convertParameters(data);
        Log.d(ResponseHandler.class.toString(), postParameters);
        urlConnection = (HttpURLConnection) urlToRequest.openConnection();
        urlConnection.setDoOutput(true);
        urlConnection.setRequestMethod("POST");
        urlConnection.setRequestProperty("Content-Type",
                "application/x-www-form-urlencoded");
        urlConnection.setFixedLengthStreamingMode(postParameters.getBytes().length);
        PrintWriter out = new PrintWriter(urlConnection.getOutputStream());
        out.print(postParameters);
        out.close();
        return urlConnection.getResponseMessage();
    }

    private void deleteData() throws MalformedURLException, IOException {
        Log.i("Request:",request);
        URL page = new URL(request);
        HttpURLConnection conn = null;
        conn = (HttpURLConnection) page.openConnection();
        conn.connect();
        InputStreamReader in = null;
        String message = conn.getResponseMessage();
        return;
    }
}
```

4.3 Create the file "ModifyTask.java"

This will help the server communication operations to be run asynchronously, use the following code:

```
public class ModifyTask extends AsyncTask<ResponseHandler, Void, String> {

    private ResponseHandler rh;

    @Override
    protected String doInBackground(ResponseHandler... rhs) {
        rh = rhs[0];
        return rh.start();
    }
    @Override
    protected void onPostExecute(String response) {
        rh.respond(response);
    }
}
```

4.4 Questions and part to be completed

Now the code should not complain about any error but the new activity is never used. Please add the complete the code in the following way:

- **In "`onItemClick`" in "`MainActivity.java`" start properly the ModifyContactActivity. After that the code should run correctly hopefully without crashing;**
- **Complete "`Respond`" in `ResponseHandler`.java (some indications in the method comments), by concluding correctly the activity after a modify/delete the application should automatically return to the listview;**
  - **The handler is associated with which concept?**
- **Add a user picture**
  - **If the API provides correctly the link of an image for each user simply retrieve it from the JSON object, if not, you can use a link to an image which can be the same for all contact just to test the functionality**
  - **Adjust the contact class constructor to accept a bitmap image as additional parameter**
  - **Implement the method loadImage in JsonLoader : in this method load the image as bitmap**
  - **When creating a contact in JSONLoader add the bitmap image retrieved through loadImage.**
  - **In contact adapter retrieve the Imageview and load the contact image in it**
- **Make the URL to you GAE contactlist server a setting in your android application.**
  - **For test and comparison, you may use http://hellomoongae.appspot.com/contactlist?respType=json**

5 Reference

[1] https://software.intel.com/en-us/blogs/2012/03/12/how-to-start-intel-hardware-assisted-virtualization-hypervisor-on-linux-to-speed-up-intel-android-x86-emulator
[2] http://developer.android.com/training/basics/firstapp/index.html
[3] http://developer.android.com/guide/components/loaders.html
[4] http://developer.android.com/sdk/installing/studio.html
[5] http://developer.android.com/reference/android/os/Handler.html