HW8:

CONTACT MANAGEMENT APP

Martin Mudenda Bbela

2582912

Overview

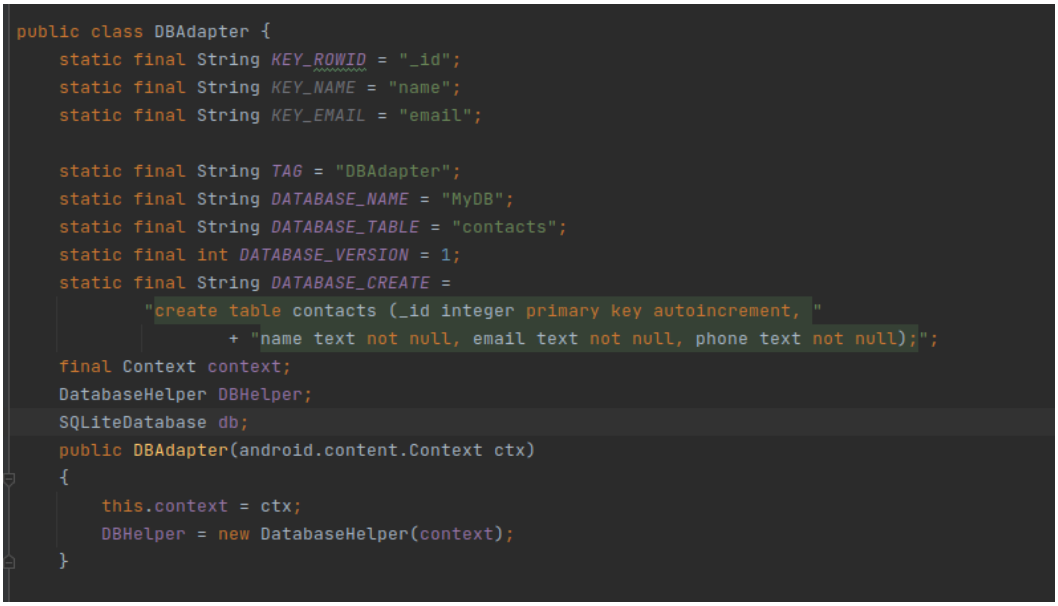
Contact management app version 3:

- Based on contact management app version 2, please use SQLite database to store the collected contact information.
- By using SQLite to store the contacts, your app now can store the contacts persistently, i.e., even if you close your app and restart it, all the contacts will not be lost, and they should still show in the ListView.

METHOD

Since this is a continuation Of Contact Management App 2 I will attach the writeup of that Project at the bottom of this one

I Created a database adapter to facilitate the connection between the Database and the application. The DB will have 1 table called contacts with space for name email and phone. This is the same as the Contacts object.



```
public class DBAdapter {
    static final String KEY_ROWID = "_id";
    static final String KEY_NAME = "name";
    static final String KEY_EMAIL = "email";

    static final String TAG = "DBAdapter";
    static final String DATABASE_NAME = "MyDB";
    static final String DATABASE_TABLE = "contacts";
    static final int DATABASE_VERSION = 1;
    static final String DATABASE_CREATE =
            "create table contacts (_id integer primary key autoincrement, "
                    + "name text not null, email text not null, phone text not null);";
    final Context context;
    DatabaseHelper DBHelper;
    SQLiteDatabase db;
    public DBAdapter(android.content.Context ctx)
    {
        this.context = ctx;
        DBHelper = new DatabaseHelper(context);
    }
```

The adapter has a lot of functionality solely for setting up the DB itself but for the sake of this HW its important to know I created a getAllContacts() function and a insertContact(Contact) function to get information from the database.

```
public long insertContact(String name, String email, String phone) {
    ContentValues initialValues = new ContentValues();
    initialValues.put(ContactConstants.CONTACT_NAME, name);
    initialValues.put(ContactConstants.CONTACT_EMAIL, email);
    initialValues.put(ContactConstants.CONTACT_PHONE,phone );
    return db.insert(DATABASE_TABLE,  nullColumnHack: null, initialValues);
}

public Cursor getAllContacts() {
    return db.query(DATABASE_TABLE, new String[] {KEY_ROWID, ContactConstants.CONTACT_NAME,
            ContactConstants.CONTACT_EMAIL, ContactConstants.CONTACT_PHONE},  selection: null,  selectionArgs: null,  groupBy: null,  having: null,  orderBy: null);
}
```

Main Activity

To enhance the previous app I added a Database Manager to the Main activity. Now the Database itself will be queried to get the contacts for the application. When a contact is added to the Database I simply run updateContact() which will querry the database and fill the contactNames and contact HashMap then populate the list view as it did before.

```
private void updateContacts() {

    contactNames.clear();
    contacts.clear();

    db.open();
    Cursor c = db.getAllContacts();

    Log.d(this.getLocalClassName(),  msg: "updateContacts: "+ DatabaseUtils.dumpCursorToString(c));

    if(c.moveToFirst()){
        do {
            Contact contact = new Contact();
            Log.d(this.getLocalClassName(),  msg: "updateContacts: "+ DatabaseUtils.dumpCurrentRowToString(c));
            contact.setName(c.getString( columnIndex: 1));
            contact.setEmail(c.getString( columnIndex: 2));
            contact.setPhone(c.getString( columnIndex: 3));
            contactNames.add(contact.getName());
            contacts.put(contact.getName(), contact);

        }while(c.moveToNext());
    }
    db.close();

    contactsAdapter.notifyDataSetChanged();

}
```

Result

The application works exactly the same as the previous iteration of this application except the data persists and will stay available to the user even after closing and opening up the application again

CONTACT MANAGEMENT APP 2 WRITEUP

Overview:

Contact management app version 2

- Like version 1, the layout for the main activity should have a button (at the bottom of the layout) for users to add a new contact, and the new contact information should be passed back to the main activity on exit of that activity
- The created contacts should be displayed as a ListView in the main activity
- In the ListView, each contact's full name should be displayed
- In the main activity, a user could click any row of the contact, a third activity would open to display the contact details

Method:

ContactConstants

This is a simple class that will hold constants that will be used to refrence values that are passed through intents .

```java
public class ContactConstants {
    public static final String CONTACT_NAME = "contact_name";
    public static final String CONTACT_EMAIL = "contact_email";
    public static final String CONTACT_PHONE = "contact_phone";
}
```

Contact

A custom object known as contact has been created to house all the information about a specific contact. This will make it easier to handle transferring, creating and showing contacts.

```java
public class Contact {
    private String name, phone, email;

    public String getName() { return name; }

    public void setName(String name) { this.name = name; }

    public String getPhone() { return phone; }

    public void setPhone(String phone) { this.phone = phone; }

    public String getEmail() { return email; }

    public void setEmail(String email) { this.email = email; }
}
```

CreateContactActivity

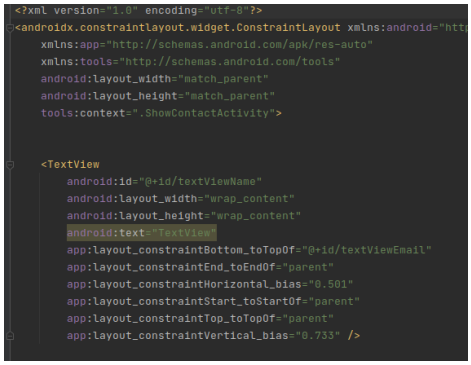
Create Contact has a Linear Layout with each Edit Texts to add a name email and Phone. The set Contact button will will return these values to the Main activity activity

```xml
<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:layout_weight="1"
    android:orientation="horizontal">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:text="Name" />

    <EditText
        android:id="@+id/editTextTextPersonName"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_weight="3"
        android:ems="10"
        android:hint="Name"
        android:inputType="textPersonName" />
</LinearLayout>
```

```java
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_create_contact);
    btnFinish       =  findViewById(R.id.buttonFinish);
    etPersonName    = findViewById(R.id.editTextTextPersonName);
    etEmailAddress  = findViewById(R.id.editTextTextEmailAddress);
    etPhone         = findViewById(R.id.editTextPhone);

    btnFinish.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            Intent returnContact =  new Intent();
            returnContact.putExtra(ContactConstants.CONTACT_NAME, etPersonName.getText().toString());
            returnContact.putExtra(ContactConstants.CONTACT_EMAIL, etEmailAddress.getText().toString());
            returnContact.putExtra(ContactConstants.CONTACT_PHONE, etPhone.getText().toString());
            setResult(RESULT_OK, returnContact);
            finish();
        }
    });
}
```

ShowContactActivity

Similar to the previous Contact Manager App, it will contain three TextViews that will be wired to show the values of the Contact Information sent to it through an intent.



```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="htt
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".ShowContactActivity">


    <TextView
        android:id="@+id/textViewName"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="TextView"
        app:layout_constraintBottom_toTopOf="@+id/textViewEmail"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.501"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.733" />
```

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

    Intent contactI

    btnReturn = fin
    tvName
    tvPhone             = findViewById(R.id.textViewPhone);
    tvEmail             = findViewById(R.id.textViewEmail);

    tvName.setText(contactInformation.getStringExtra(ContactConstants.CONTACT_NAME));
    tvPhone.setText(contactInformation.getStringExtra(ContactConstants.CONTACT_PHONE));
    tvEmail.setText(contactInformation.getStringExtra(ContactConstants.CONTACT_EMAIL));

    btnReturn.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) { finish(); }
    });
```

```
Bundle savedInstanceState

savedInstanceState – If the activity is being re-initialized after previously being shut down then this Bundle contains the data it most recently supplied in onSaveInstanceState.
Note: Otherwise it is null.
```

Main Activity

The main activity layout is a List view Within a Linear Layout and an add contact button Below it.

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:gravity="center"
    tools:context=".MainActivity" >

    <ListView
        android:id="@+id/listViewContacts"
        android:layout_width="match_parent"
        android:layout_height="665dp"
        android:layout_above="@+id/buttonAddContact"
        android:layout_marginBottom="5dp" />

    <Button
        android:id="@+id/buttonAddContact"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentEnd="true"
        android:layout_alignParentBottom="true"
        android:layout_marginEnd="145dp"
        android:layout_marginBottom="15dp"
        android:layout_weight="1"
        android:gravity="center"
        android:text="Add Contact" />
</RelativeLayout>
```

In the Main activity Class we wire a n ArrayAdapter with the s*imple_list_item_1* and a list of the contactNames. This will allow us to view the contact names on the Main activity. The new adapter is attached to the list view and An onItemClickListener is attached to the listview as well. The onItemClickListener will start the *ShowContactsActivity for the specific contact.* An arrayList is used to store the contactNames and a hash Map has the contact name as a key for the Contact object. The values of the contact object are extracted and added to the intent to initialize the show contact intent.

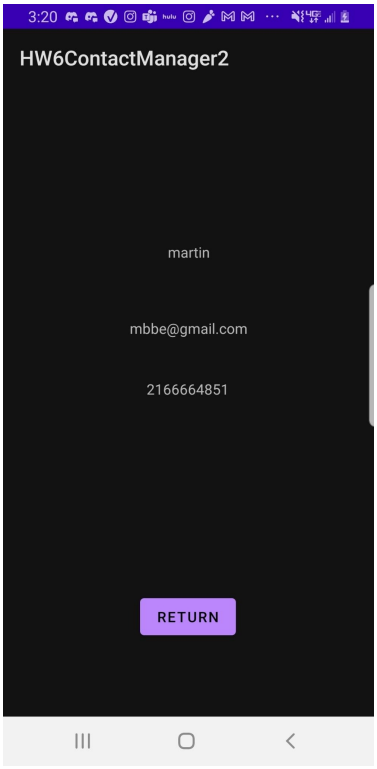
```
lvContacts      = findViewById(R.id.listViewContacts);
lvContacts.setAdapter(contactsAdapter);
lvContacts.setOnItemClickListener(new AdapterView.OnItemClickListener() {
    @Override
    public void onItemClick(AdapterView<?> adapterView, View view, int i, long l) {
        Contact c = contacts.get(contactNames.get(i));
        //TODO get View Contact

        Intent showContactIntent =  new Intent(getApplicationContext(), ShowContactActivity.class);
        showContactIntent.putExtra(ContactConstants.CONTACT_NAME, c.getName());
        showContactIntent.putExtra(ContactConstants.CONTACT_EMAIL, c.getEmail());
        showContactIntent.putExtra(ContactConstants.CONTACT_PHONE, c.getPhone());

        startActivity(showContactIntent);

    }
});
```

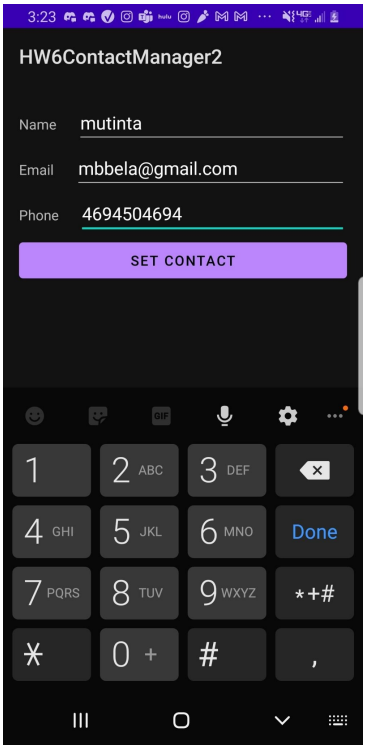
Using the new Android Pattern for activity results. I created an activityResultsLauuncher with an activity result Contract. This will get the information after a contact has been created in the create contact activity and add it to the contacts HashMap, ContactNames list and will finally tell the adapter to update the list. We will then a listener to the btnAddContact to launch the CreateContactActivity

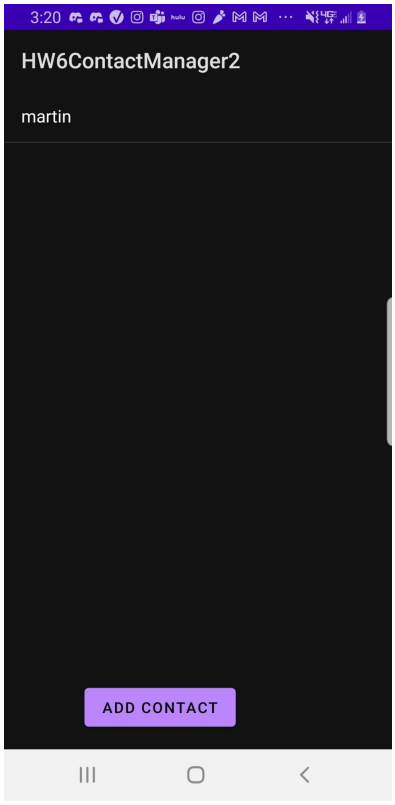
```
mGetContactInformation = registerForActivityResult(new ActivityResultContracts.StartActivityForResult(), new ActivityResul
    @Override
    public void onActivityResult(ActivityResult result) {
        Contact c =  new Contact();
        c.setName(result.getData().getStringExtra(ContactConstants.CONTACT_NAME));
        c.setPhone(result.getData().getStringExtra(ContactConstants.CONTACT_PHONE));
        c.setEmail(result.getData().getStringExtra(ContactConstants.CONTACT_EMAIL));

        contacts.put(c.getName(), c);
        contactNames.add(c.getName());
        contactsAdapter.notifyDataSetChanged();

    }
});

btnAddContact.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        Intent startCreateContentIntent =  new Intent(getApplicationContext(), CreateContactActivity.class);
        mGetContactInformation.launch(startCreateContentIntent);          }
});
```

RESULTS

User is able to create contacts, populating the list view in the main activity and can further view these contacts in another activity on clicking on them.


3:20
HW6ContactManager2
martin
mbbe@gmail.com
2166664851
RETURN


3:23
HW6ContactManager2
Name
mutinta
Email
mbbela@gmail.com
Phone
4694504694
SET CONTACT
1
2 ABC
3 DEF
4 GHI
5 JKL
6 MNO
Done
7 PQRS
8 TUV
9 WXYZ
*+#
0 +
#
,


3:20
HW6ContactManager2
martin
ADD CONTACT