`

The Roster App

Assignment 1 – Prog 8215

Brahmpreet Singh (7845159)

Mobile Application Development (Semester 2)

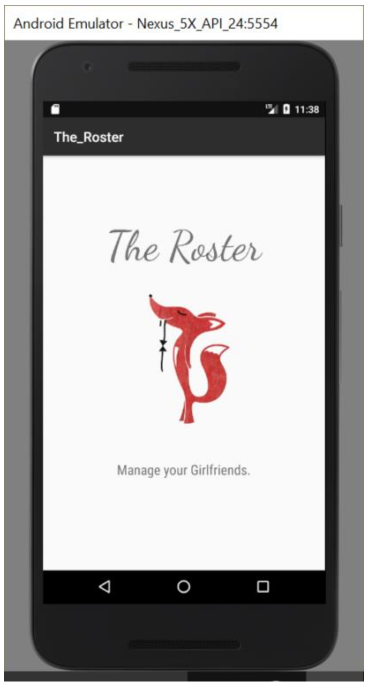
`

Components used:

Layout file	Component	Xml counterpart	Java counterpart
1. activity_main.xml	LinearLayout		
	Label	xLabel	jLabel
	EditText	xEditText	jEditText
	CheckBox	xCheckBox	jCheckBox
	6 TextViews	TextView	N/A
	Spinner	xSpinner	jSpinner
	Button	xBirthdayButton	jBirthdayButton
	TextView	TextView	N/A
	RadioGroup	xRadioGroup	jRadioGroup
	RadioButton	x_XS	j_XS
	RadioButton	x_S	j_S
	RadioButton	x_M	j_M
	RadioButton	x_L	j_L
	RadioButton	x_XL	j_XL
	RadioButton	x_XXL	j_XXL
	SeekBar	xSeekBar1	jSeekBar1
	TextView	xTextViewSeek1	jTextViewSeek1
	SeekBar	xSeekBar2	jSeekBar2
	TextView	xTextViewSeek2	jTextViewSeek2
	SeekBar	xSeekBar3	jSeekBar3
	TextView	xTextViewSeek3	jTextViewSeek3
	Button	xButtonSave	jButtonSave
	Button	xButtonShow	jButtonShow
2. activity_splash_screen.xml	LinearLayout		
	ImageView	xImageViewSplash1	jTextViewSplash1
	ImageView	xImageViewSplash2	jImageViewSplash2
	TextView	xTextViewSplash2	jTextViewSplash2
3. row_layout.xml	LinearLayout		
	TextView	xTextView1_Row	jTextView1_Row
	TextView	xTextView2_Row	jTextView2_Row
	TextView	xTextView3_Row	jTextView3_Row
	TextView	xTextView4_Row	jTextView4_Row
	TextView	xTextView5_Row	jTextView5_Row
4. activity_data_list.xml	LinearLayout		
	ListView	xListView	jListView

`

Activity Sequence



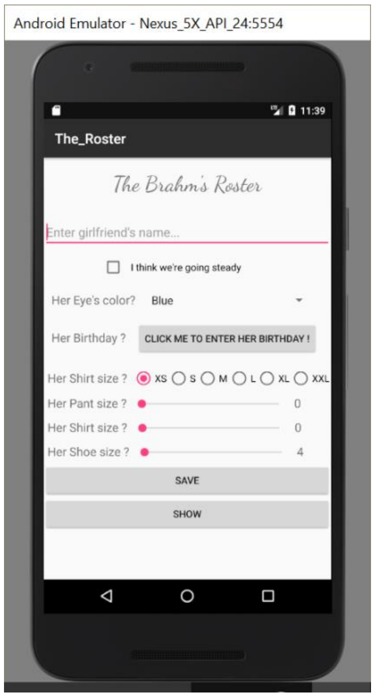
Android Emulator - Nexus_5X_API_24:5554

11:38

The_Roster

The Roster

Manage your Girlfriends.



Android Emulator - Nexus_5X_API_24:5554

11:39

The_Roster

The Brahm's Roster

Enter girlfriend's name...

☐ I think we're going steady

Her Eye's color? Blue

Her Birthday ? CLICK ME TO ENTER HER BIRTHDAY !

Her Shirt size ? [✓ XS] S M L XL XXL

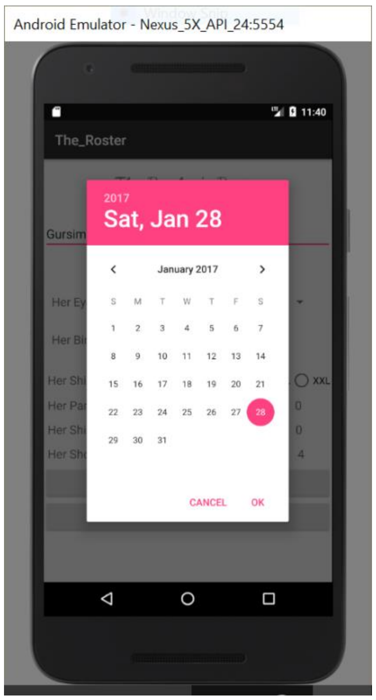
Her Pant size ? 0

Her Shirt size ? 0

Her Shoe size ? 4

SAVE

SHOW



Android Emulator - Nexus_5X_API_24:5554

11:40

The_Roster

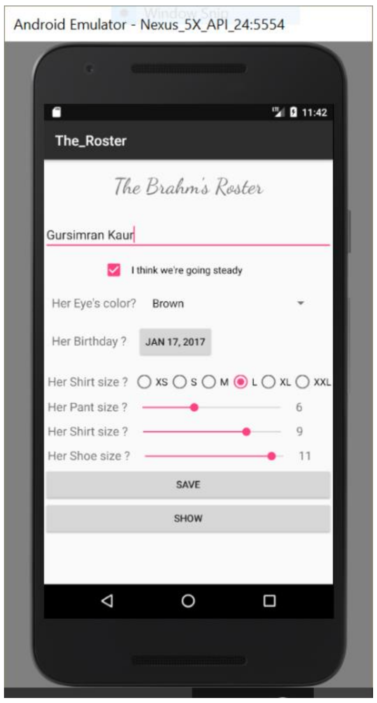
2017

Sat, Jan 28

January 2017

S	M	T	W	T	F	S
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	[✓ 28]
29	30	31				

CANCEL OK



Android Emulator - Nexus_5X_API_24:5554

11:42

The_Roster

The Brahm's Roster

Gursimran Kaur

☑ I think we're going steady

Her Eye's color? Brown

Her Birthday ? JAN 17, 2017

Her Shirt size ? XS S M [✓ L] XL XXL

Her Pant size ? 6

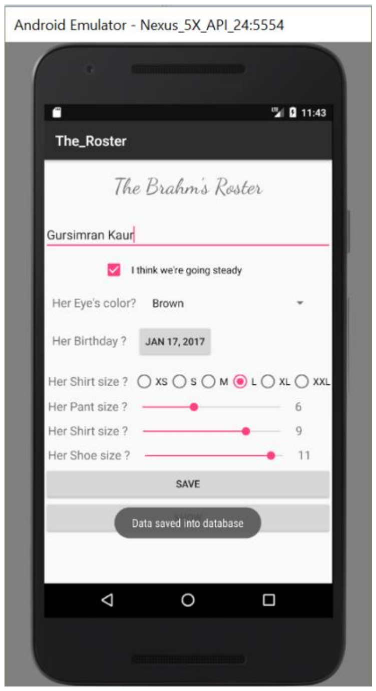
Her Shirt size ? 9

Her Shoe size ? 11

SAVE

SHOW

`


Android Emulator - Nexus_5X_API_24:5554

11:43

The_Roster

The Brahm's Roster

Gursimran Kaur

[✓] I think we're going steady

Her Eye's color? Brown

Her Birthday ? JAN 17, 2017

Her Shirt size ? ( ) XS ( ) S ( ) M [✓ L] ( ) XL ( ) XXL

Her Pant size ? 6

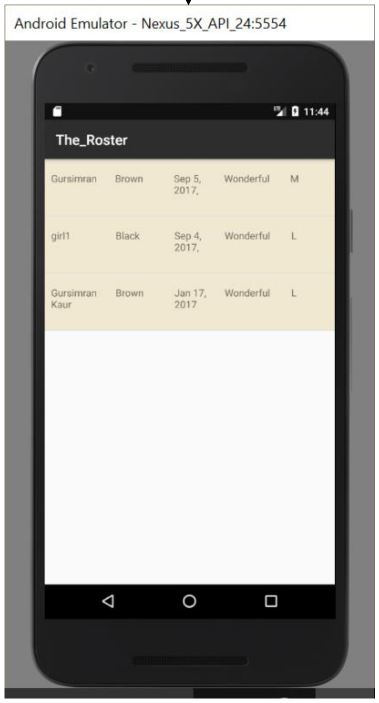
Her Shirt size ? 9

Her Shoe size ? 11

SAVE

Data saved into database

`


Android Emulator - Nexus_5X_API_24:5554

11:44

The_Roster

Gursimran	Brown	Sep 5, 2017,	Wonderful	M
girl1	Black	Sep 4, 2017,	Wonderful	L
Gursimran Kaur	Brown	Jan 17, 2017	Wonderful	L

`

Java files created:

1. SplashScreen.java (Launcher)
2. MainActivity.java
3. UserContract.java
4. UserDbHelper.java
5. ListDataAdapter.java
6. DataProvider.java
7. DataListActivity.java

Functionality:

1. **SplashScreen.java –**
   a) Java Initialization of all the xml-counterparts of 'activity_main.xml' in this file.
   b) Involves usage of Predefined classes like '**Animation**' & '**AnimationUtils**' & inbuild methods in these classes like '**loadAnimation()**' & '**startAnimation()**' applied on various components to make the animations work as per the user-defined '**anim**' resource file. Also involves 'Intent' functionality to switch to 'activity_main.xml'.

2. **MainActivity.**java –

   a) Java Initialization of all the xml-counterparts of 'activity_main.xml' in this file.
   b) **j_RG1.setOnClickListener()** method to fetch value into '**radioValue' variable**, so that that could be later use to save info into database. (JUST TO KNOW WHICH RADIO BUTTON WAS CLICKED)
   c) **jBirthdayButton.setOnClickListener()** method to define onClick() method and thereby having an instantiated object of inbuilt '**Calendar**' class and thereby using this reference, to perform various Calendar operations later in program. Next, we are calling '**dialogAppear()**' within **onClick()** in which we have instantiated the inbuilt '**DatePickerDialog**' class by passing the required

`

parameters to its constructor(involves using object ‘**calendar**’ here to fetch year,month,day and passing them as parameter)

Parameter 1 →**context**
Parameter 2 → **d** – which is an object of onDateSetListener, thereby letting computer know what to do when user will select a date from DatePicker. We are making processor save picked date to be saved in ‘calendar’ object.
Parameter 3 → **Year** (Fetched using calendar object)
Parameter 4 →**Month** (Fetched using calendar object)
Parameter 5 → **Day_Of_Month** (Fetched using calendar object)

Next we’re calling ‘**updateTextViewDate()**’ whose functionality merely includes setting text to the button using methods of inbuilt class ‘**DateFormat**’. That is we have instantiated the DateFormat class earlier and we are using object to format date value in the required format.

d) **jSeekBar1.setOnSeekBarChangeListener()** – By setting this functionality to the seekbar component we are defining what should be the output when we perform any of these 3 operations.
    - **onProgressChanged()** – We will be setting progress value to textView adjacent to it.
    - **onStartTrackingTouch()** - We will be setting progress value to textView adjacent to it.
    - **onStopTrackingTouch()** - We will be setting progress value to textView adjacent to it.

NOTE – We have 3 seekbars. All of the three seekbars have been defined pretty much the same way.

e) **showMethod()** – This method shall be called when the user will press the button with ‘Show’ text on it. In this method we have setup an **intent** to make sure that than when user click show method, his screen must be switched to ‘**DataListActivity.class’**. In other words, the screen that is assosciated with this class. This class contains the set of rows into which information is getting saved.
f) **saveMethod()** –
    i. This method shall be called when user shall click on the Button with ‘SAVE’ text on it. In this method we are basically using 3 to 4 string

`

variables to fetch values from the java components that user has setup (filled in information into them).

ii. **baseUserDbHelper = new UserDBHelper(context);**
By writing this, we are retrieving object of UserDbHelper class into 'baseUserDBHelper' object.

iii. **baseSqLiteDatabase = baseUserDbHelper.getWritableDatabase();**
Now with the help of 'baseUserDbHelper' object, we have got a writable database. <u>We have saved this writeable database into a 'sqlLiteDatabase' object</u>. We got this writable database by applying '**getWritableDatabse()**' method on '**baseUserDbHelper**' object.

iv. **baseUserDbHelper.addRow(name, eyecolor, dob, status, size, baseSqLiteDatabase);**
As we have a objectified reference of 'UserDbHelper' in the form of 'baseUserDbHelper' we are able to call '**addRow()**' method that exists in UserDbHelper class.
As we can see that addRow() method takes **6 parameters** of which 5 of them are merely the values that we have fetched from the components which user filled with information<u>. The most important one is the **6th parameter 'baseSqlLiteDatabase'**. So by passing this parameter we will make UserDbHelper class help us to write into it.</u>

v. **c = baseUserDbHelper.selectRow(baseSqLiteDatabase);**
Here '**c**' is the object of cursor and here we are trying to call the method '**selectRow()**' of 'UserDbHelper' class and also we are passing writable database (baseSqLiteDatabase) to 'selectRow()' method. As we know, <u>selectRow() method return Cursor object</u>. So we will fetch that cursor object into 'c' after processing. In the selectRow() method, using the passed 'baseSqLiteDatabase' object, we will apply **query() method on the database**.

vi. Toast to display that data is saved into database

vii. **baseUserDbHelper.close();**
Freeing resources.

`

3. **UserContract.**java – This file contains an inner abstract class '**NewUserInfo**' with the help of which we are able to define the structure of SQL table i.e columns required defined.
4. **UserDbHelper.java – '**UserDbHelper' class is majorly assosciated with performing operations the Database.
   a) Variables containing Database name, Database version, Create_Query information created.
   b) **UserDbHelper(Context context)** – This is a **constructor** which we will define and by calling super() method, we will be calling the constructor of Super class 'SQLiteOpenHelper' and **to that 'super()' we are passing 4 parameters:**
      i. Context
      ii. Database_name
      iii. Any cursor factory i.e. null
      iv. Database version

   By passing these parameters to super() constructor, we got database to work on.

   So when we typed, 'baseUSerDbHelper = new UserDbHelper(context);', indirectly we got a database in the 'baseUserDbHelper' object on which we later applied 'getWritableDatabse()' method.

   c) **onCreate(SqLiteDatabase db)** - This method gets automatically called when we try to instantiate UserDbHelper class in MainActivity.java, thereby 'db.execSQL(Create_Query);' gets executed which means that the table gets created, the moment we try to instantiate 'USerDbHelper' class.

   d) **public void addRow(String name, String Eyecolor, String BirthDate, String Status,String Size, SQLiteDatabase db) { }**
   Here we are getting the input dadta from user in the form of parameters and we need to save this data into table in the form of row.

```
                    ContentValues contentValues = new ContentValues();
          contentValues.put(UserContract.NewUserInfo.Girl_name, name);
  contentValues.put(UserContract.NewUserInfo.Girl_Eyecolor, Eyecolor);
    contentValues.put(UserContract.NewUserInfo.Birth_date, BirthDate);
           contentValues.put(UserContract.NewUserInfo.Status, Status);
               contentValues.put(UserContract.NewUserInfo.Size, Size);
```

   Here we are putting the values to the object of 'ContentValues' class which is an inbuilt class.

`

db.insert(UserContract.NewUserInfo.Table_name, null, contentValues);

Here we are just putting 'contentValues' into the Structure (Table) that we defined in UserContract class.

e) **public Cursor selectRow(SQLiteDatabase db) {}**
This method will return a cursor with saved information (as per cursor definition) after running query() method on db.

5. **DataProvider.java – '**DataProvider' class is merely assosciated to provide us data from each row as an object. Just a class assosciated with setters and getters.
6. **DataListActivity.java –** This class is assosciated with putting data into table, everytime as a row.
7. **ListDataAdapter.java –** This file is completely assosciated with inflating the rowlayout into the activity and putting the information by fetching from the database.