warwick
C<>ding

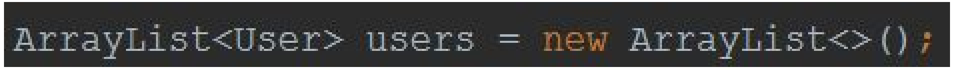
by Wesley Haigh

Lecture 4

A 3rd Activity & Databases

Recap

- Android makes use of **Object Oriented Programming**
- In OOP we have **Objects**
- These **Objects** correlate to actual entities eg. an animal object
- If we want to have a list of these **Objects** then we need to use an **ArrayList**
- An ArrayList is defined thus:

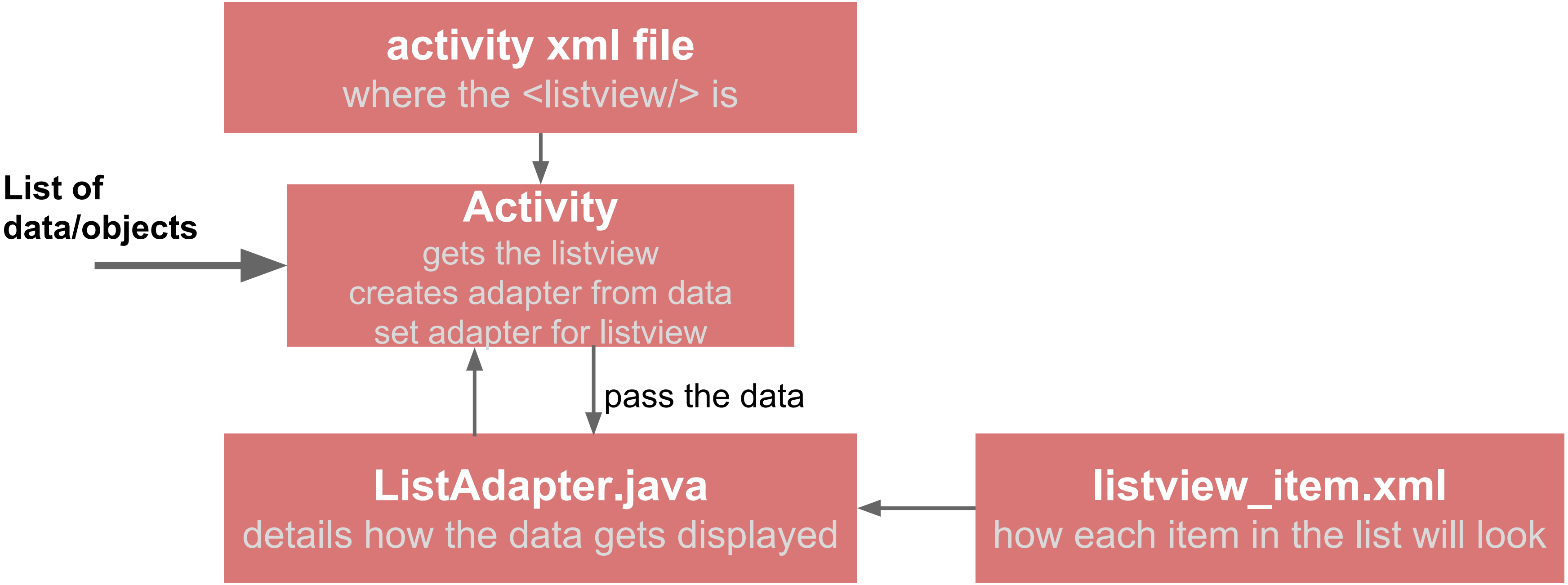


```
ArrayList<User> users = new ArrayList<>();
```

- To display a list in an app we need to use a **ListView**
- A **ListView** needs an adapter to go along with it.

Recap

- The diagram of how all the components needed to show a list interlink is:



List of data/objects

activity xml file
where the <listview/> is

Activity
gets the listview
creates adapter from data
set adapter for listview

pass the data

ListAdapter.java
details how the data gets displayed

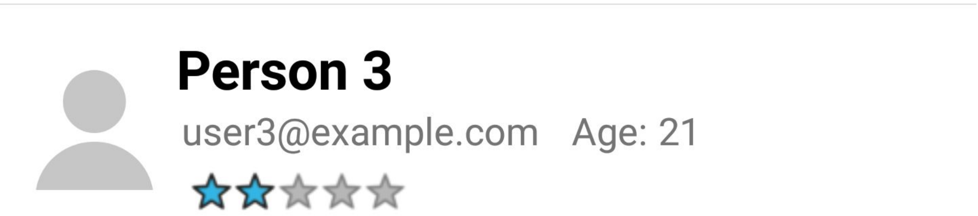
listview_item.xml
how each item in the list will look

Recap Exercise

- Change the items in the listview so that they display the age, and rating of the user as well.

- Use a **<RatingBar/>** to create the ratings try looking up the properties of a rating bar

- Will need to modify:
  - listview_user_item.xml
  - UserListAdapter.java
    - getView()
    - ViewHolder
  - UserActivity.java



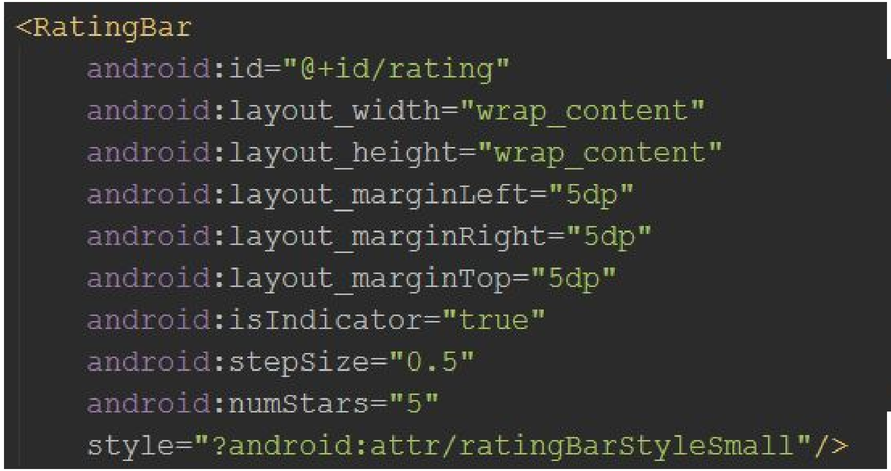
Person 3

user3@example.com Age: 21

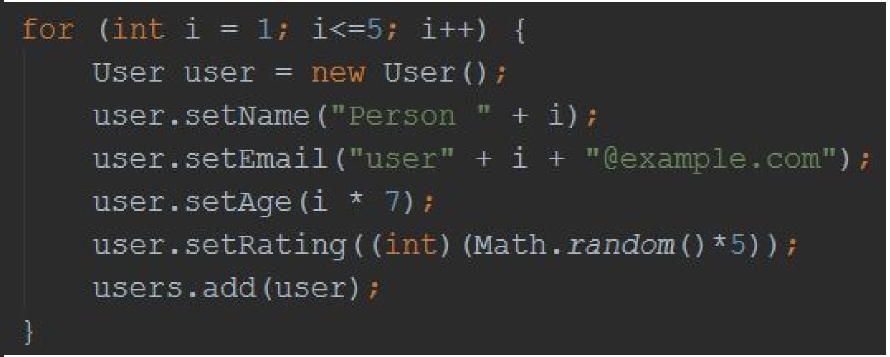
★★☆☆☆

Recap Exercise

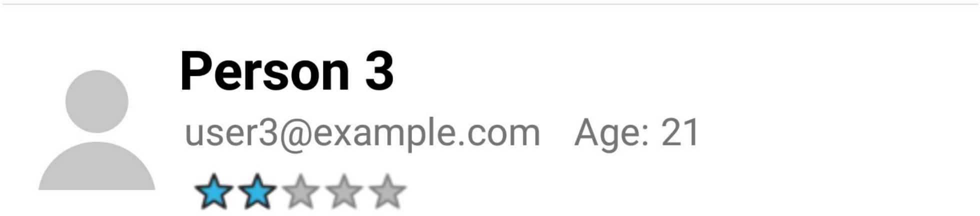
- XML for the rating bar and generating ratings:



```
<RatingBar
    android:id="@+id/rating"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginLeft="5dp"
    android:layout_marginRight="5dp"
    android:layout_marginTop="5dp"
    android:isIndicator="true"
    android:stepSize="0.5"
    android:numStars="5"
    style="?android:attr/ratingBarStyleSmall"/>
```



```
for (int i = 1; i<=5; i++) {
    User user = new User();
    user.setName("Person " + i);
    user.setEmail("user" + i + "@example.com");
    user.setAge(i * 7);
    user.setRating((int)(Math.random()*5));
    users.add(user);
}
```



Person 3

user3@example.com Age: 21

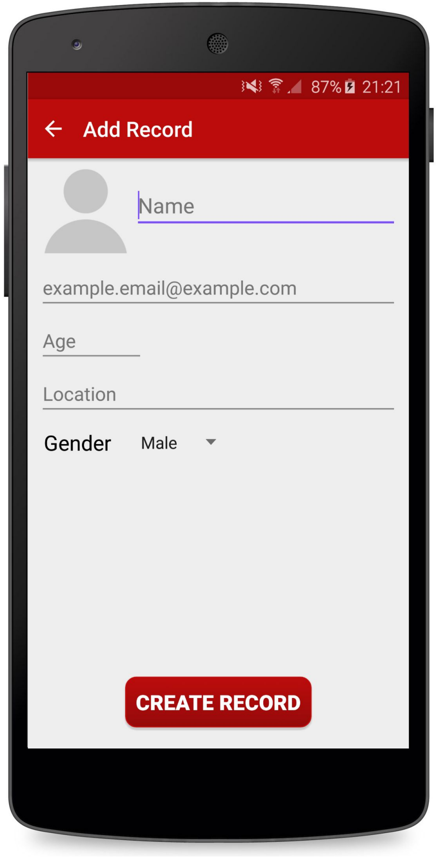
★★☆☆☆

A 3rd Activity

AddUserActivity

Add a New Activity

- Create an activity called **AddRecordActivity**
- Remove the Floating Action Button (FAB)
- We are going to change the layout so that it looks like on the right



87% 21:21

Add Record

Name

example.email@example.com

Age

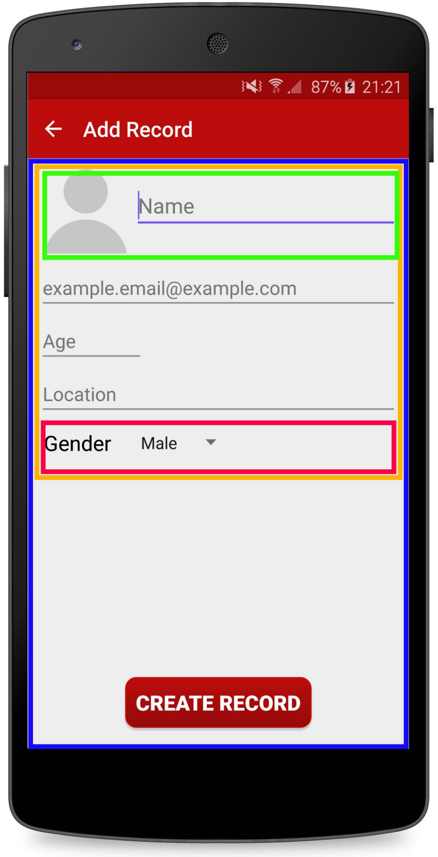
Location

Gender Male

CREATE RECORD

First Step - XML

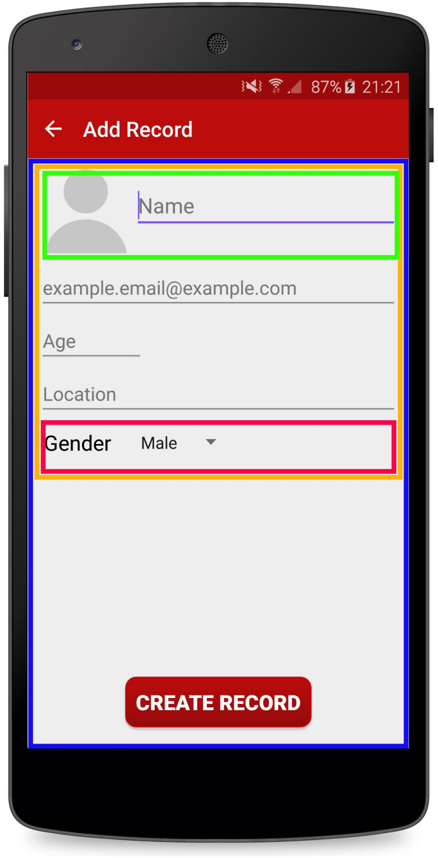
- Break the layout down. What Containers do we need?
- Look at the contents of each container
  - Blue RelativeLayout
    - Button at the bottom
    - LinearLayout in the rest
  - Red LinearLayout
    - TextView
    - Spinner (male/female selector)



First Step - XML

Remember:

- Add relevant String resources to **strings.xml**
- Give the EditTexts, Imageview, and Button ids
- Set the input types for some of the EditTexts
  - email field has **email** input type
  - age field has **number** input type



87% 21:21

Add Record

Name

example.email@example.com

Age

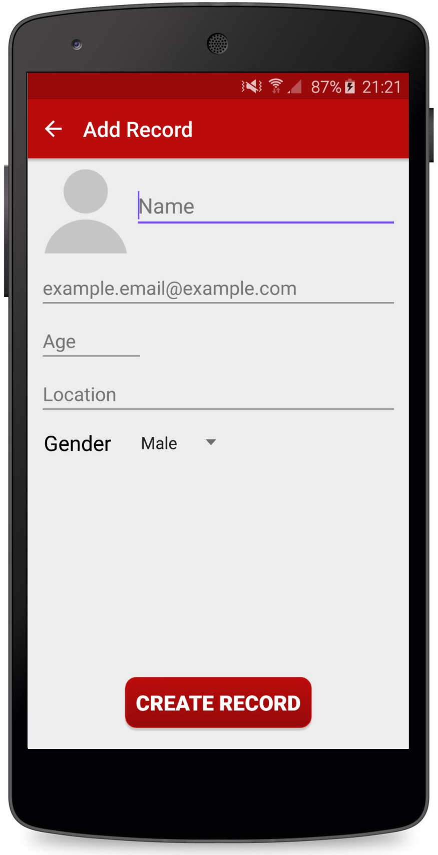
Location

Gender Male

CREATE RECORD

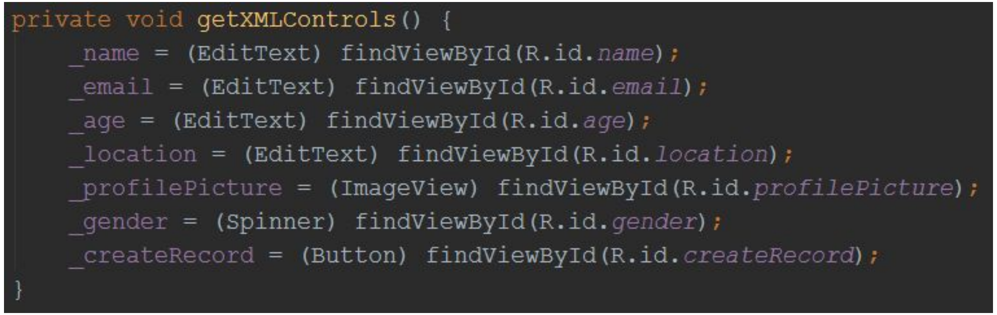
Second Step - Java

- Functionality of the activity:
    - Create button: creates a User object and fills it with the input on screen
    - Can click on the picture to select a picture from the gallery on the phone
    - Have some validity checks for things like age and email



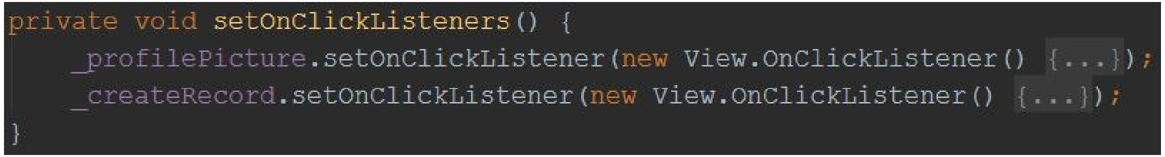
Java

- Set up the basics in AddRecordActivity.java
  - **getXMLControls()**



```
private void getXMLControls() {
    _name = (EditText) findViewById(R.id.name);
    _email = (EditText) findViewById(R.id.email);
    _age = (EditText) findViewById(R.id.age);
    _location = (EditText) findViewById(R.id.location);
    _profilePicture = (ImageView) findViewById(R.id.profilePicture);
    _gender = (Spinner) findViewById(R.id.gender);
    _createRecord = (Button) findViewById(R.id.createRecord);
}
```

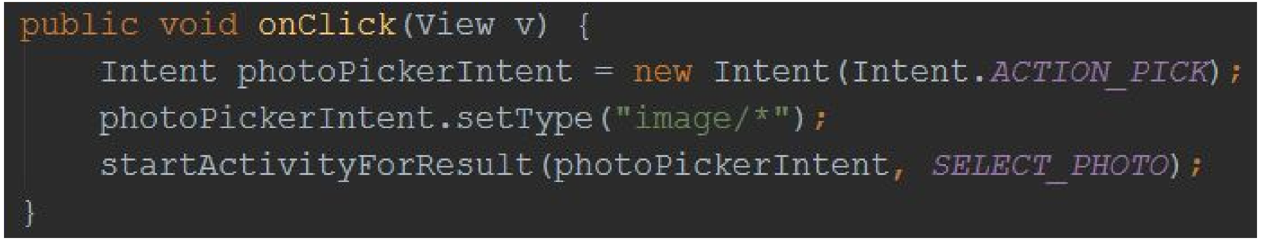
  - **setOnClickListeners()**



```
private void setOnClickListeners() {
    _profilePicture.setOnClickListener(new View.OnClickListener() {...});
    _createRecord.setOnClickListener(new View.OnClickListener() {...});
}
```

Java - Select Profile Picture

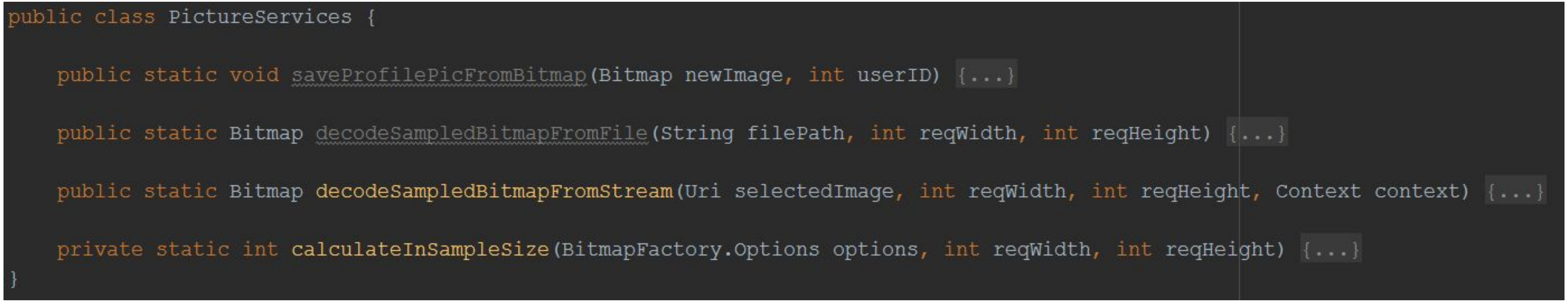
- Add property to **User** class called **_hasPicture** (boolean)
- Add class variable to AddRecordActivity called **_profileBitmap** (Bitmap)
- Add another class variable called **SELECT_PHOTO** (final int) = 101
- Add this to your onClickListener (or your onClick function):



```
public void onClick(View v) {
    Intent photoPickerIntent = new Intent(Intent.ACTION_PICK);
    photoPickerIntent.setType("image/*");
    startActivityForResult(photoPickerIntent, SELECT_PHOTO);
}
```

Java - Select Profile Picture

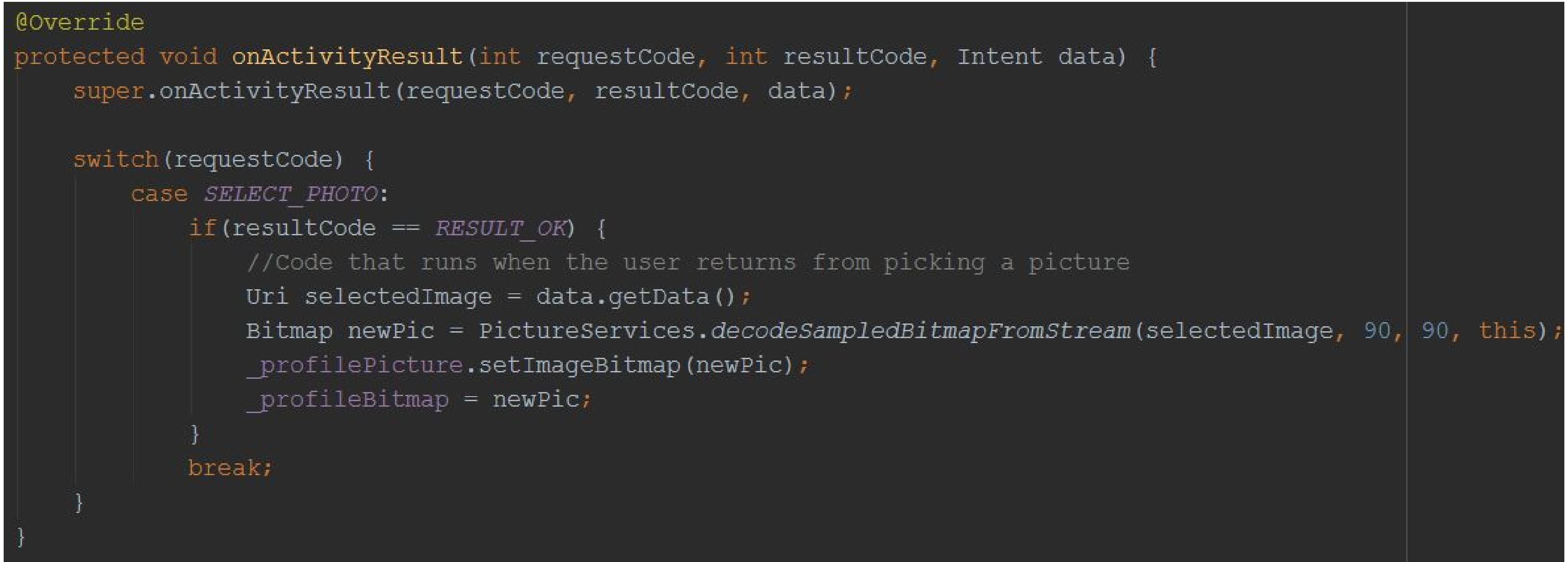
- Create a **ServiceClasses** package. Right click com.warwickcodingapp → New → package
- Create a java class in this package called **PictureServices**
- Go to the Google drive folder and download the PictureServices java file and copy the contents into your file.



```
public class PictureServices {

    public static void saveProfilePicFromBitmap(Bitmap newImage, int userID) {...}

    public static Bitmap decodeSampledBitmapFromFile(String filePath, int reqWidth, int reqHeight) {...}

    public static Bitmap decodeSampledBitmapFromStream(Uri selectedImage, int reqWidth, int reqHeight, Context context) {...}

    private static int calculateInSampleSize(BitmapFactory.Options options, int reqWidth, int reqHeight) {...}
}
```

Java - Select Profile Picture

- Need code for when user returns from selecting picture
- That is what **SELECT_PHOTO** helps
- Add this to **UsersActivity**



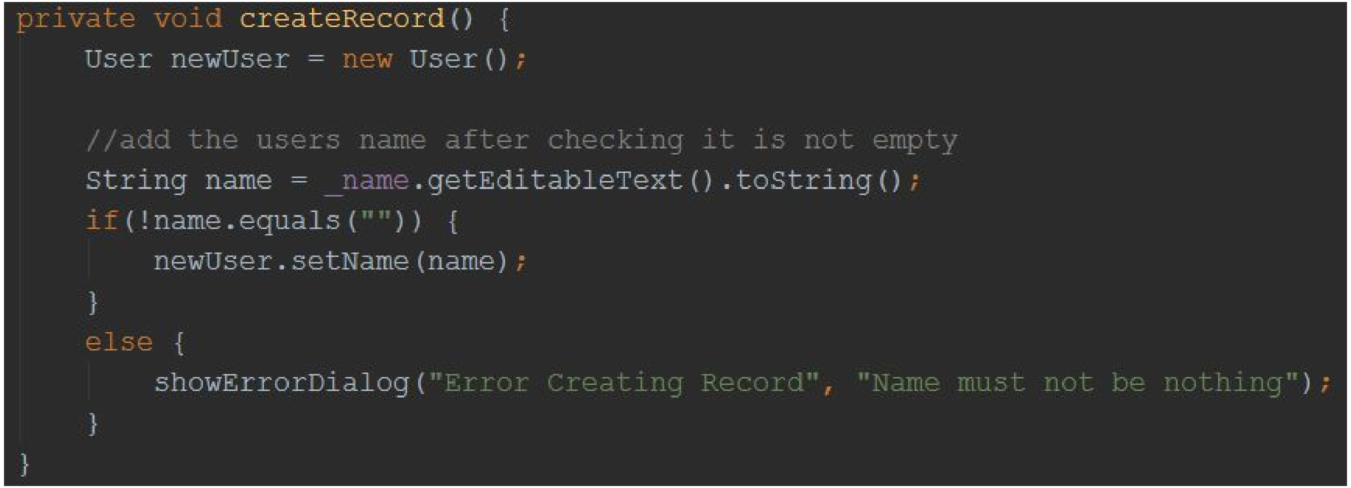
```java
@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
    super.onActivityResult(requestCode, resultCode, data);

    switch(requestCode) {
        case SELECT_PHOTO:
            if(resultCode == RESULT_OK) {
                //Code that runs when the user returns from picking a picture
                Uri selectedImage = data.getData();
                Bitmap newPic = PictureServices.decodeSampledBitmapFromStream(selectedImage, 90, 90, this);
                _profilePicture.setImageBitmap(newPic);
                _profileBitmap = newPic;
            }
            break;
    }
}
```

Java - Create Record Button

- Need to add some functionality when the user clicks create record
    - Create empty user object
    - Fill it with the information entered. Performing any relevant validity checks for each bit of info
    - If any check fails don’t “create” the new record
    - Otherwise “create” the user (for now we will just check that all info is correct

Java - Create Record Button

- New function in UsersActivity called **createRecord()**

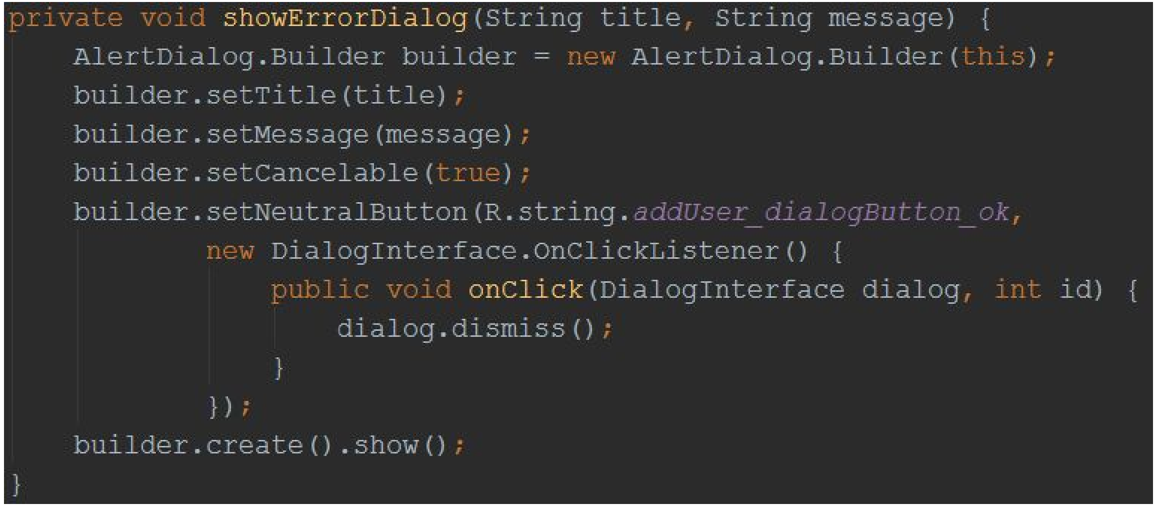


```
private void createRecord() {
    User newUser = new User();

    //add the users name after checking it is not empty
    String name = _name.getEditableText().toString();
    if(!name.equals("")) {
        newUser.setName(name);
    }
    else {
        showErrorDialog("Error Creating Record", "Name must not be nothing");
    }
}
```

- Create similar bits for the other inputs (don’t always need a validation check)

Java - Create Record Button

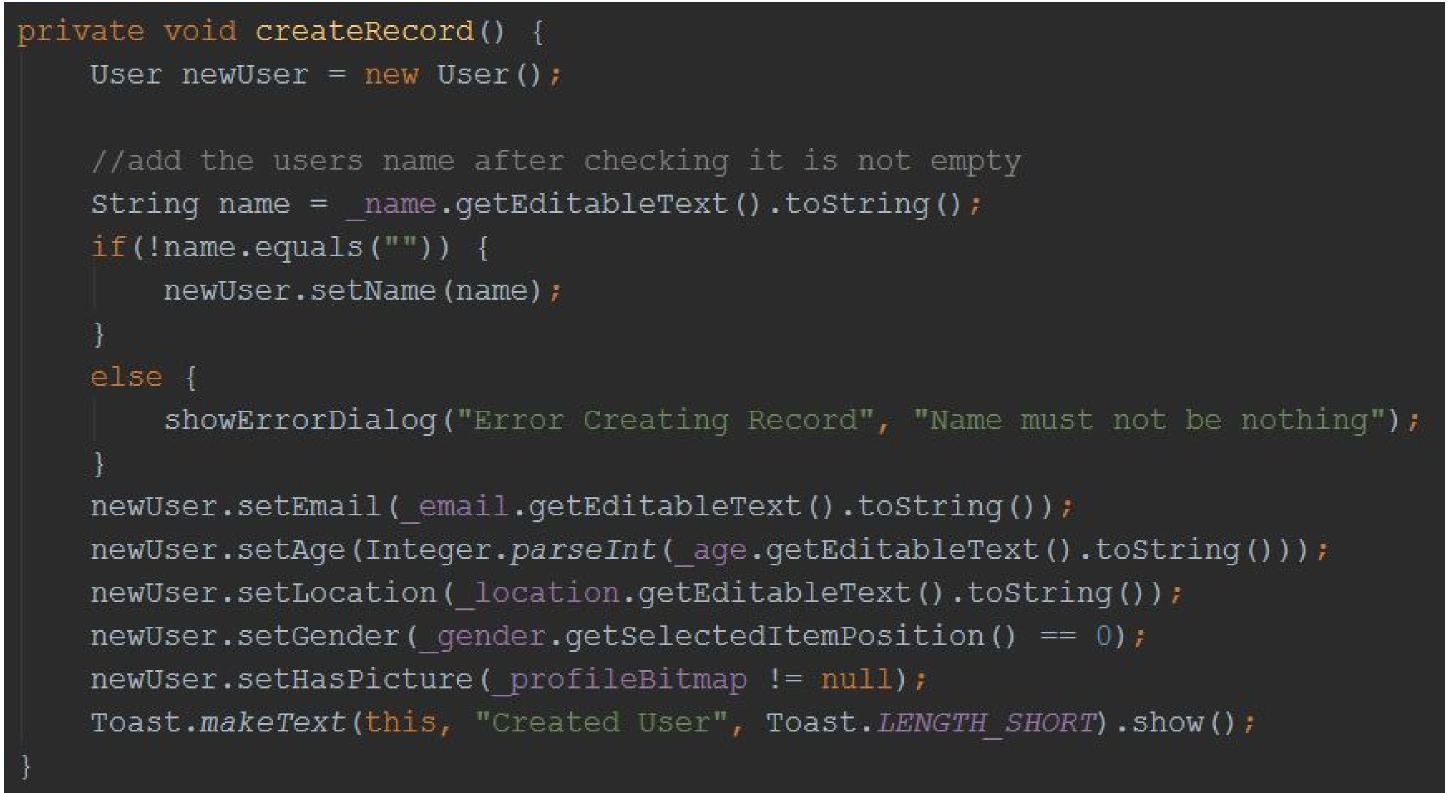
- showErrorDialog(String title, String message)



```java
private void showErrorDialog(String title, String message) {
    AlertDialog.Builder builder = new AlertDialog.Builder(this);
    builder.setTitle(title);
    builder.setMessage(message);
    builder.setCancelable(true);
    builder.setNeutralButton(R.string.addUser_dialogButton_ok,
            new DialogInterface.OnClickListener() {
                public void onClick(DialogInterface dialog, int id) {
                    dialog.dismiss();
                }
            });
    builder.create().show();
}
```

Java - Create Record Button

- The whole createRecord()



```java
private void createRecord() {
    User newUser = new User();

    //add the users name after checking it is not empty
    String name = _name.getEditableText().toString();
    if(!name.equals("")) {
        newUser.setName(name);
    }
    else {
        showErrorDialog("Error Creating Record", "Name must not be nothing");
    }
    newUser.setEmail(_email.getEditableText().toString());
    newUser.setAge(Integer.parseInt(_age.getEditableText().toString()));
    newUser.setLocation(_location.getEditableText().toString());
    newUser.setGender(_gender.getSelectedItemPosition() == 0);
    newUser.setHasPicture(_profileBitmap != null);
    Toast.makeText(this, "Created User", Toast.LENGTH_SHORT).show();
}
```

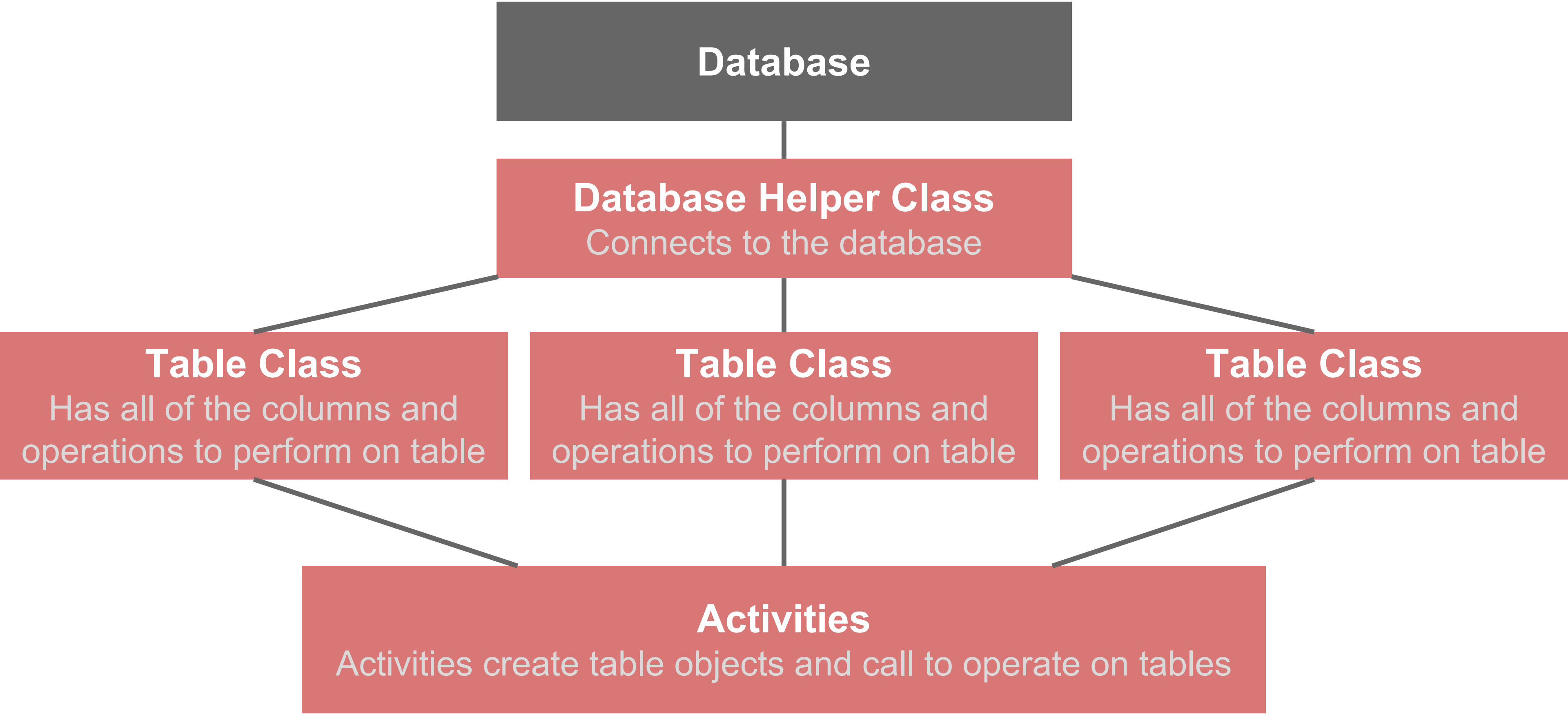
SQL Databases

Databases in Android

SQLite

- SQLite is an open source database. It supports basic SQL syntax and operations.
- SQLite is embedded into every Android device
- As a result it is the easiest way to use databases in Android

SQLite - Basic Structure



Database

Database Helper Class
Connects to the database

Table Class
Has all of the columns and operations to perform on table

Table Class
Has all of the columns and operations to perform on table

Table Class
Has all of the columns and operations to perform on table

Activities
Activities create table objects and call to operate on tables

Next Time

- We will create:
  - a database
  - a database helper class
  - a User table class
- And we will use them all together to add and get records from the database

warwick

C<>ding