warwick C<>ding

by Wesley Haigh

Lecture 7

Multiple Tables and Comments

Recap

- When making an activity plan how it will look and what it will do
- Once the planning is complete
    - **First:** break the xml into the different containers and then construct them
    - **Second:** fill in the contents of the containers
    - **Third:** add the basics to the java file (getXMLControls() etc.)
    - **Forth:** go through each bit of functionality and add it, testing each bit as you go

CommentTable

Steps to Adding a Table To an App

1. Plan what columns you want in your table
2. Create the Object class (e.g. User.java)
3. Modify your DatabaseAdapter class
4. Create the table class (e.g. UserTable.java)
5. Delete the app on the testing device and reinstall to ensure the table is created in the database

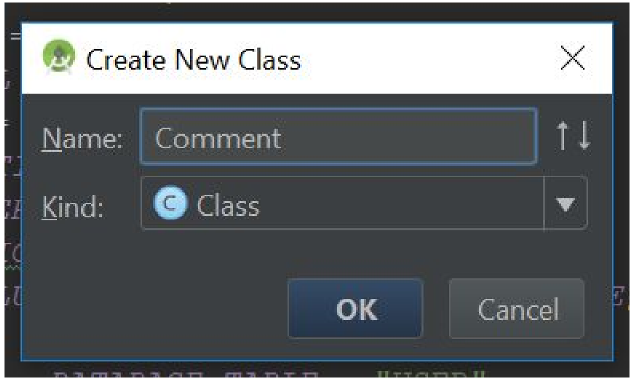
Now you can now use your new table in your code!

1. Columns for Comment Table

- **id** - integer, primary key
- **user Id** - integer, connects the comment to the user with this id
- **comment** - text, the actual comment

2. Creating a Comment Object

- Like as we did with the **User.java** class we need a **Comment.java** class
- Right click on ModelClasses and go New → Java Class
- Call the new class **Comment.java**



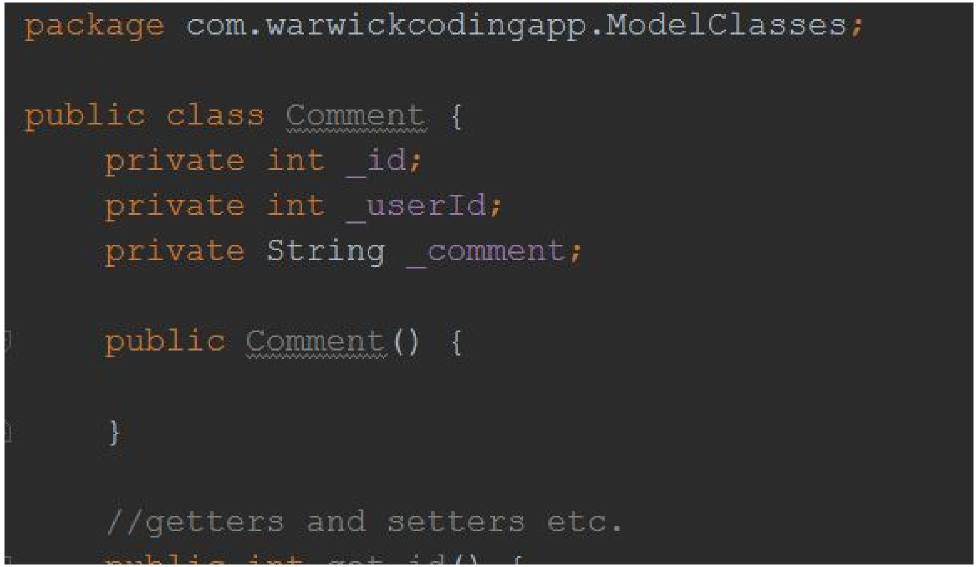
Create New Class

Name: Comment

Kind: Class

2. Comment Class

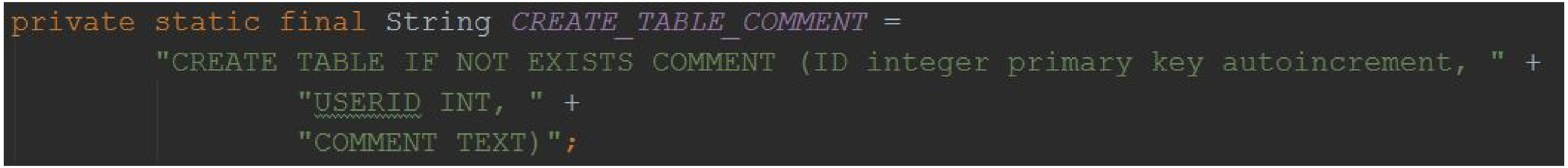
- Add an empty constructor to the Comment class
- As with the User class, add the following properties (and their getters and setters) to the Comment class:
    - id (int)
    - userId (int)
    - comment (String)



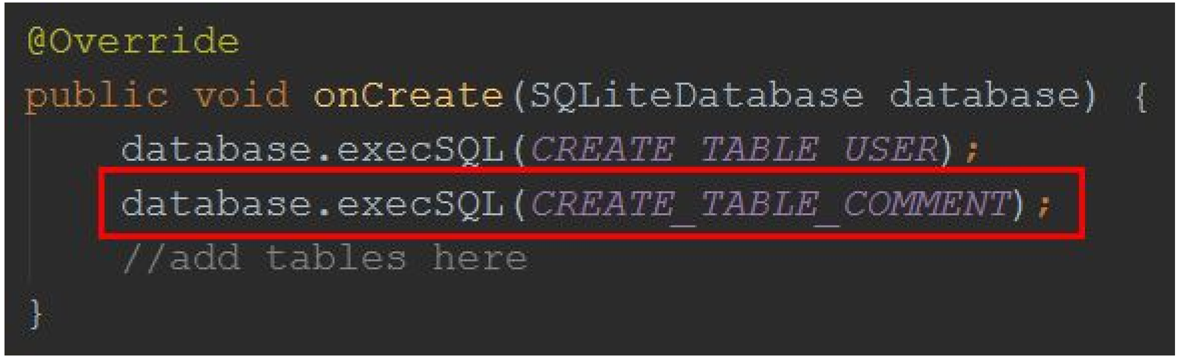
```
package com.warwickcodingapp.ModelClasses;

public class Comment {
    private int _id;
    private int _userId;
    private String _comment;

    public Comment() {

    }

    //getters and setters etc.
```

3. DatabaseAdapter

- We need to modify the **DatabaseAdapter.java** class so that it creates the **Comment** table



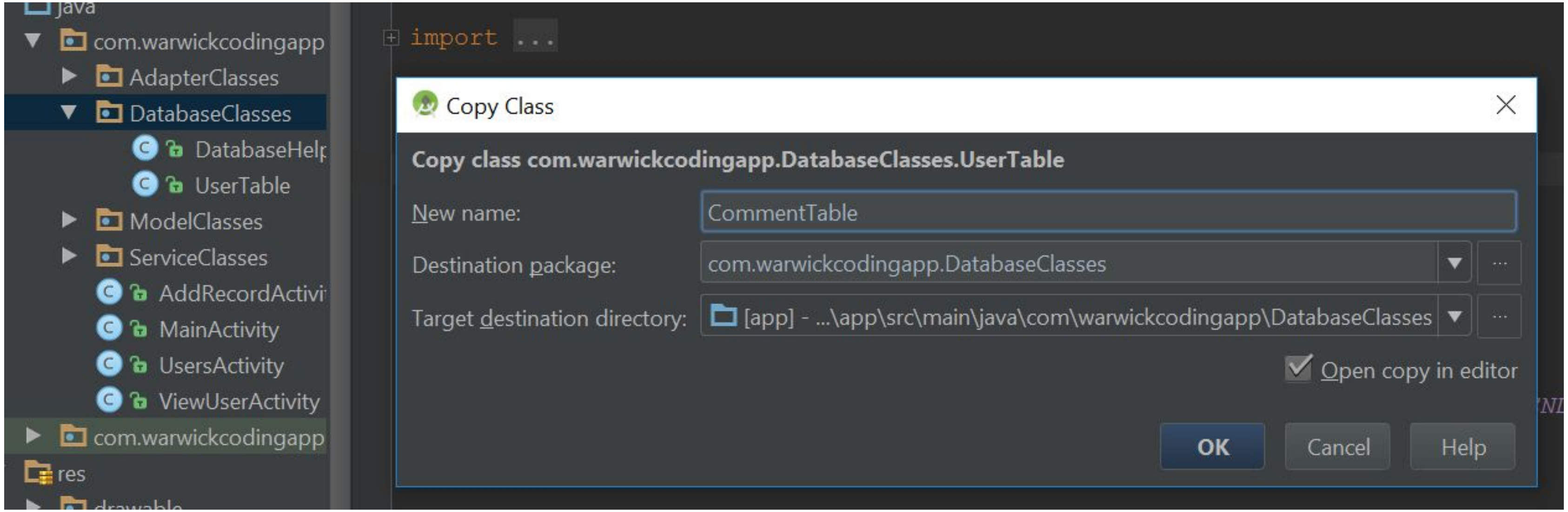
```java
private static final String CREATE_TABLE_COMMENT =
        "CREATE TABLE IF NOT EXISTS COMMENT (ID integer primary key autoincrement, " +
                "USERID INT, " +
                "COMMENT TEXT)";
```



```java
@Override
public void onCreate(SQLiteDatabase database) {
    database.execSQL(CREATE_TABLE_USER);
    database.execSQL(CREATE_TABLE_COMMENT);
    //add tables here
}
```

4. Creating CommentTable.java

- Right click and copy **UserTable.java** and paste it in the same folder
- Rename it to **CommentTable.java**



java
com.warwickcodingapp
AdapterClasses
DatabaseClasses
DatabaseHelp
UserTable
ModelClasses
ServiceClasses
AddRecordActivi
MainActivity
UsersActivity
ViewUserActivity
com.warwickcodingapp
res
drawable

```
import ...
```

Copy Class

Copy class com.warwickcodingapp.DatabaseClasses.UserTable

New name: CommentTable

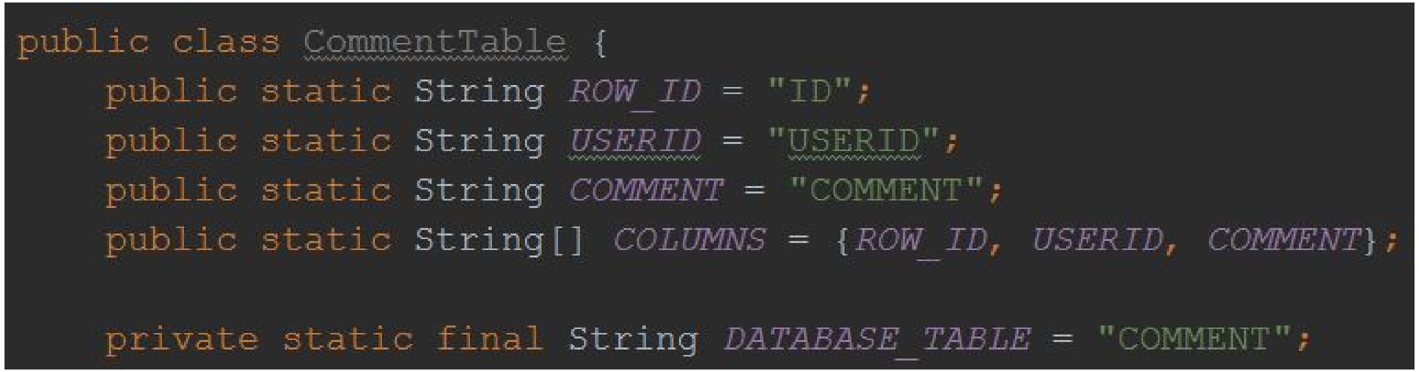
Destination package: com.warwickcodingapp.DatabaseClasses

Target destination directory: [app] - ...\app\src\main\java\com\warwickcodingapp\DatabaseClasses

☑ Open copy in editor

4. CommentTable.java

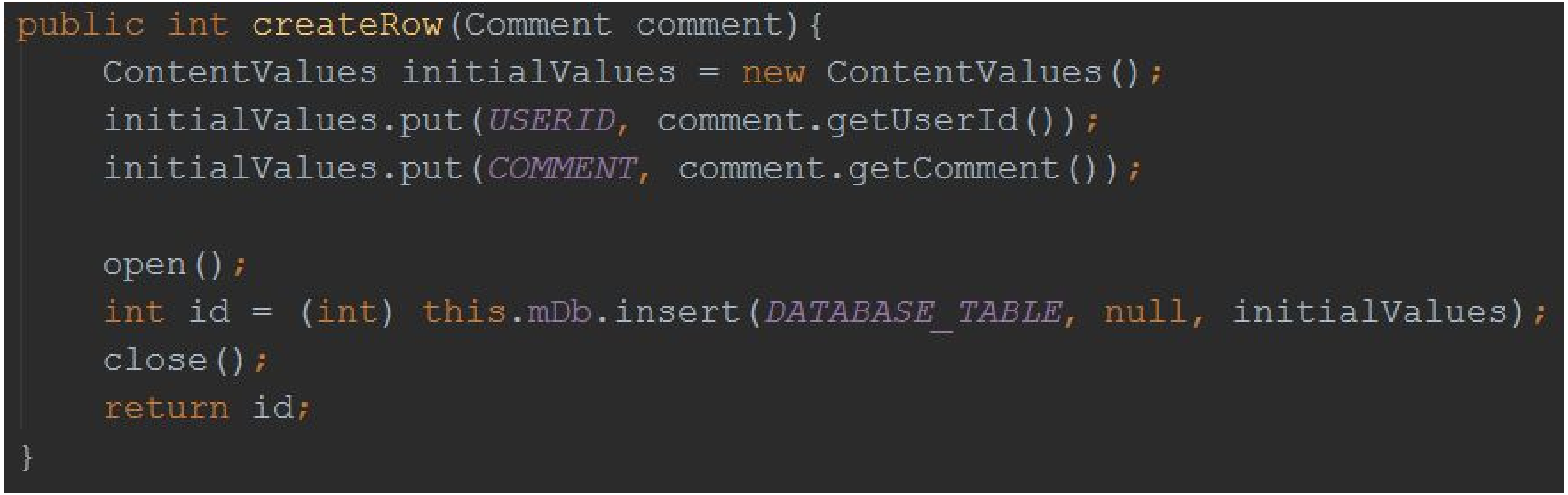
- Now we need to modify this table to fit its purpose
- First modify the **columns** and **table name**



```
public class CommentTable {
    public static String ROW_ID = "ID";
    public static String USERID = "USERID";
    public static String COMMENT = "COMMENT";
    public static String[] COLUMNS = {ROW_ID, USERID, COMMENT};

    private static final String DATABASE_TABLE = "COMMENT";
```

4. CommentTable.java

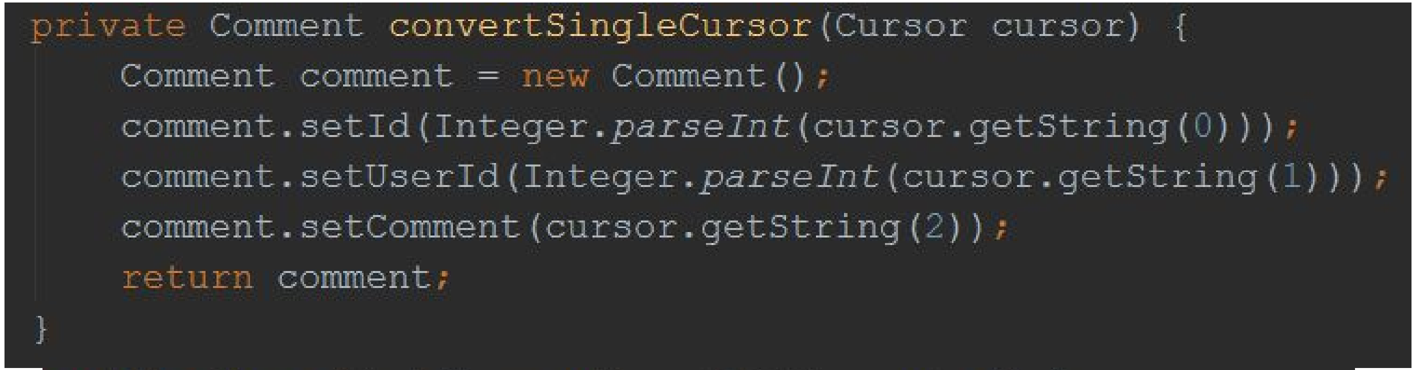
- Next the **create row** function



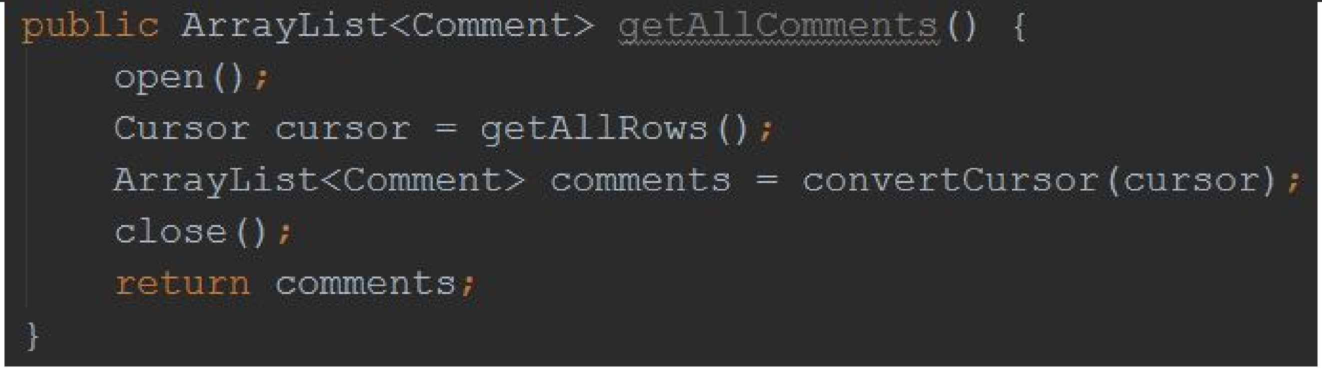
```
public int createRow(Comment comment){
    ContentValues initialValues = new ContentValues();
    initialValues.put(USERID, comment.getUserId());
    initialValues.put(COMMENT, comment.getComment());

    open();
    int id = (int) this.mDb.insert(DATABASE_TABLE, null, initialValues);
    close();
    return id;
}
```

4. CommentTable.java

- Then finally it is the **"getRow"** methods and **convertCursor**



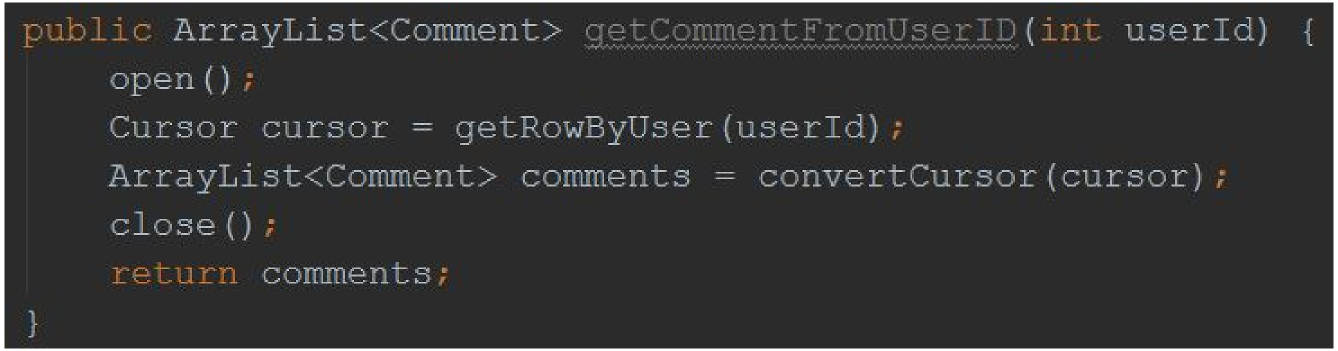
```java
private Comment convertSingleCursor(Cursor cursor) {
    Comment comment = new Comment();
    comment.setId(Integer.parseInt(cursor.getString(0)));
    comment.setUserId(Integer.parseInt(cursor.getString(1)));
    comment.setComment(cursor.getString(2));
    return comment;
}
```



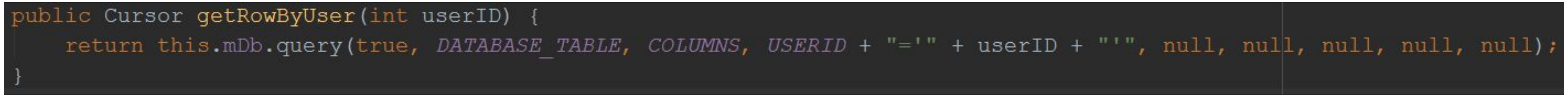
```java
public ArrayList<Comment> getAllComments() {
    open();
    Cursor cursor = getAllRows();
    ArrayList<Comment> comments = convertCursor(cursor);
    close();
    return comments;
}
```

4. CommentTable.java

- Basic rule of thumb is anywhere you see **"user"** you want to change it to **"comment"**
- Finally add a **getCommentsFromUserId(int userId)** function



```
public ArrayList<Comment> getCommentFromUserID(int userId) {
    open();
    Cursor cursor = getRowByUser(userId);
    ArrayList<Comment> comments = convertCursor(cursor);
    close();
    return comments;
}
```



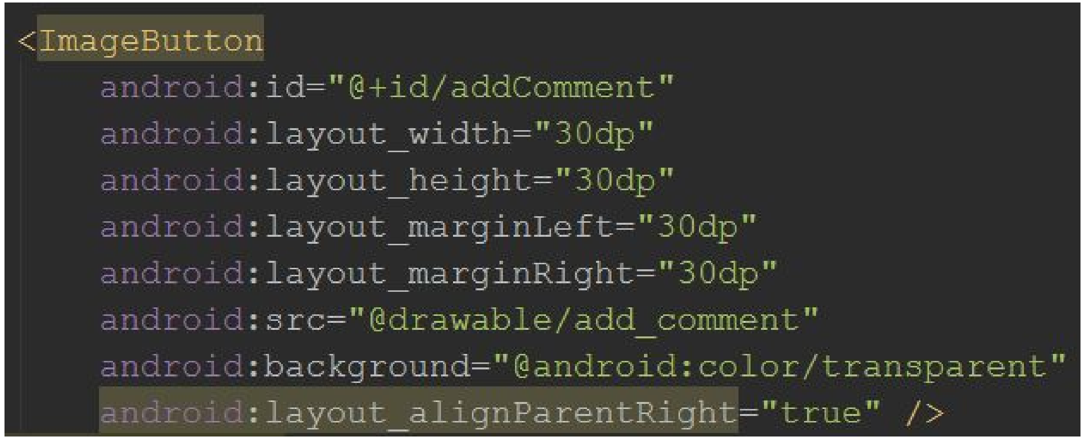
```
public Cursor getRowByUser(int userID) {
    return this.mDb.query(true, DATABASE_TABLE, COLUMNS, USERID + "='" + userID + "'", null, null, null, null, null);
}
```

Using Comments

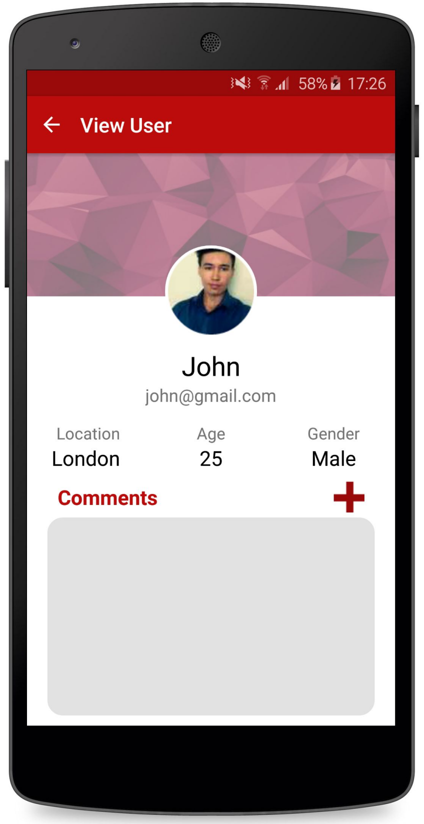
Remember to delete the app before continuing

Adding Comments to a User

- Add an image button to content_view_user.xml which will be used to add a comment
- Image seen can be found in the drive folder: **add_comment.png**

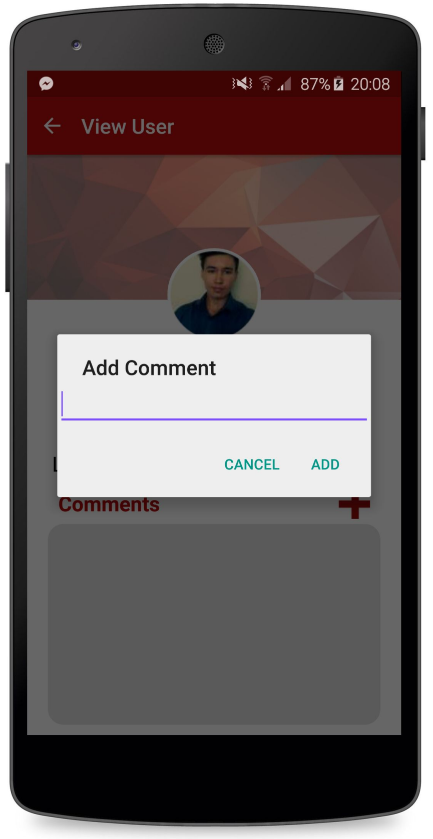


```
<ImageButton
    android:id="@+id/addComment"
    android:layout_width="30dp"
    android:layout_height="30dp"
    android:layout_marginLeft="30dp"
    android:layout_marginRight="30dp"
    android:src="@drawable/add_comment"
    android:background="@android:color/transparent"
    android:layout_alignParentRight="true" />
```



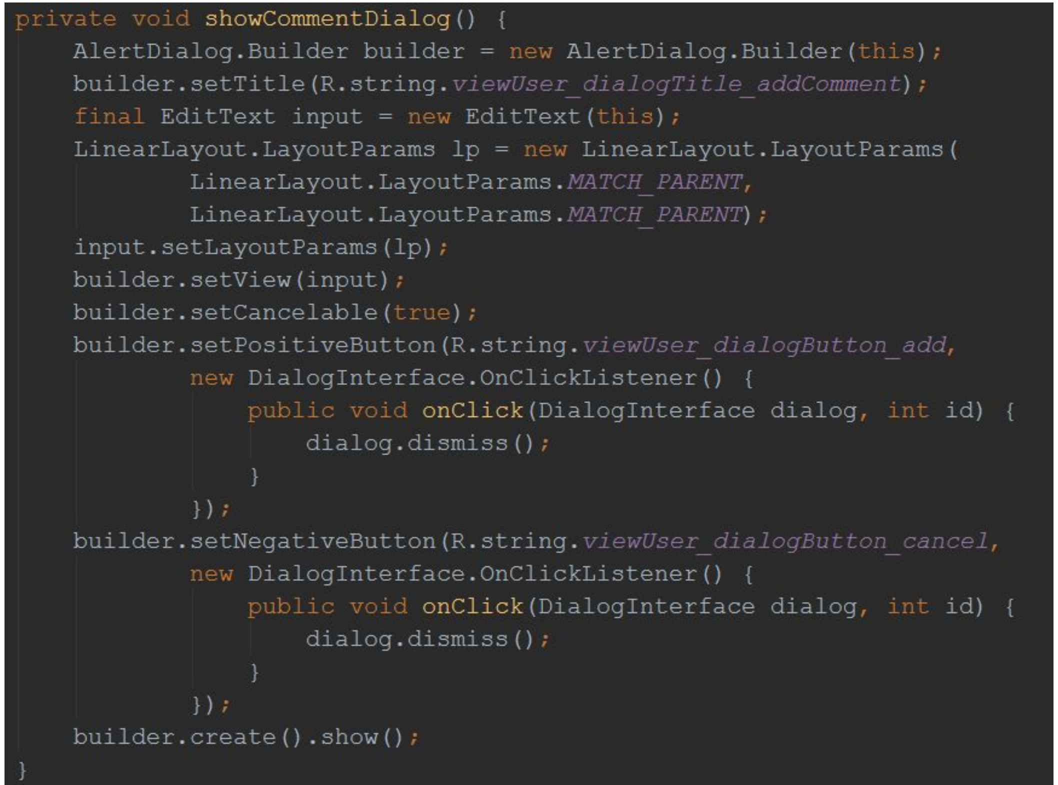
Adding Comments to a User

- **Functionality**: when the user clicks on the button a dialog will be displayed with an EditText to enter the comment and 1 button to add, and 1 button to cancel.
- As shown on the right



Adding Comments to a User

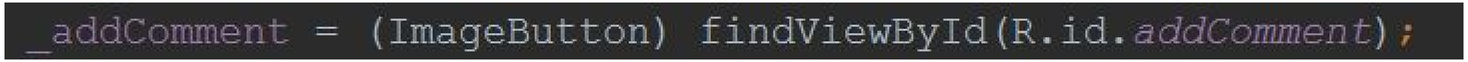
- Add **showCommentDialog()** function in ViewUserActivity.java



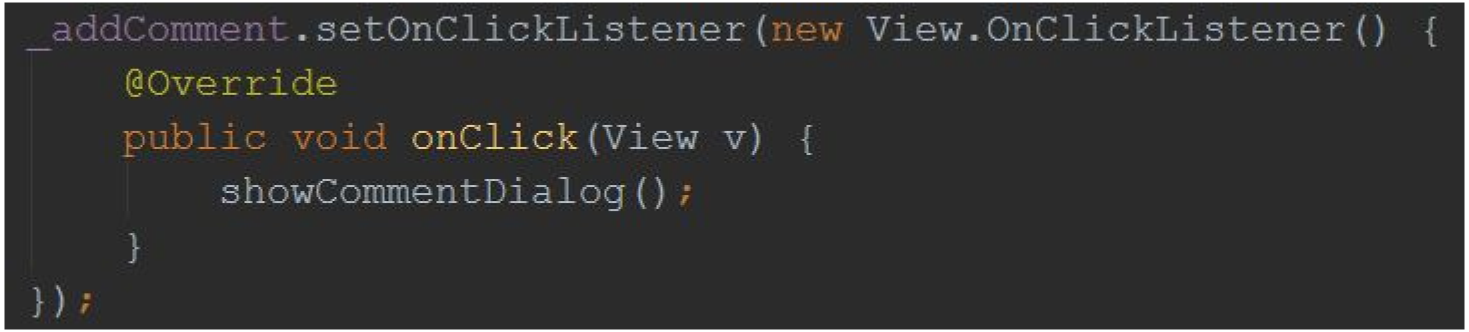
```java
private void showCommentDialog() {
    AlertDialog.Builder builder = new AlertDialog.Builder(this);
    builder.setTitle(R.string.viewUser_dialogTitle_addComment);
    final EditText input = new EditText(this);
    LinearLayout.LayoutParams lp = new LinearLayout.LayoutParams(
            LinearLayout.LayoutParams.MATCH_PARENT,
            LinearLayout.LayoutParams.MATCH_PARENT);
    input.setLayoutParams(lp);
    builder.setView(input);
    builder.setCancelable(true);
    builder.setPositiveButton(R.string.viewUser_dialogButton_add,
            new DialogInterface.OnClickListener() {
                public void onClick(DialogInterface dialog, int id) {
                    dialog.dismiss();
                }
            });
    builder.setNegativeButton(R.string.viewUser_dialogButton_cancel,
            new DialogInterface.OnClickListener() {
                public void onClick(DialogInterface dialog, int id) {
                    dialog.dismiss();
                }
            });
    builder.create().show();
}
```

Adding Comments to a User

- Get hold of button in getXMLControls and set its onClickListener()



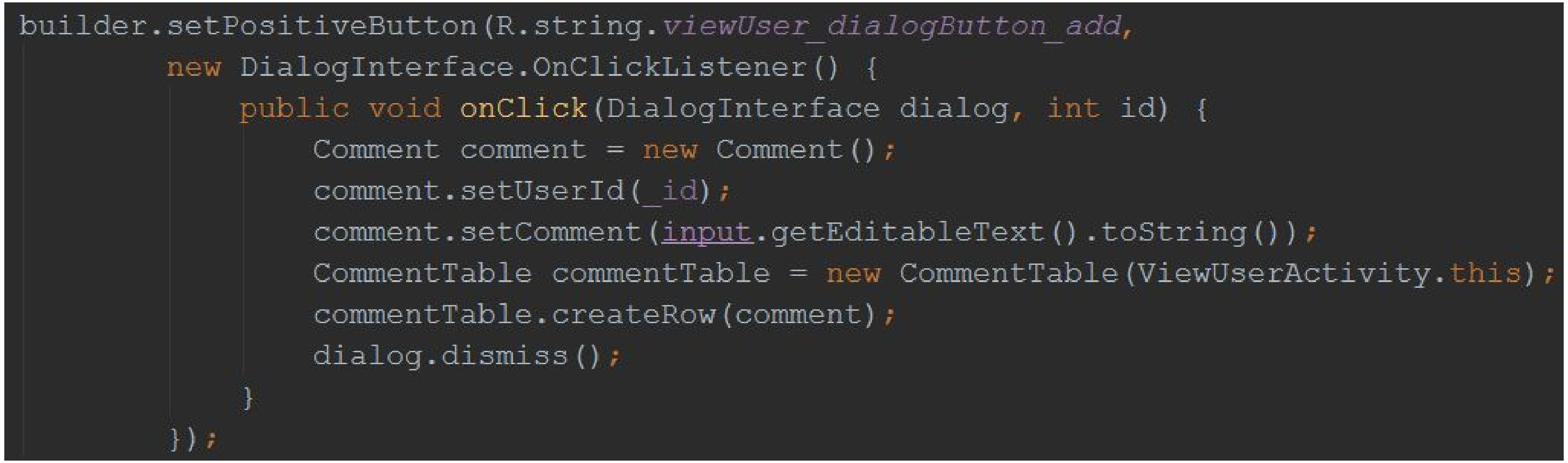
```
_addComment = (ImageButton) findViewById(R.id.addComment);
```



```
_addComment.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        showCommentDialog();
    }
});
```

Adding Comments to a User

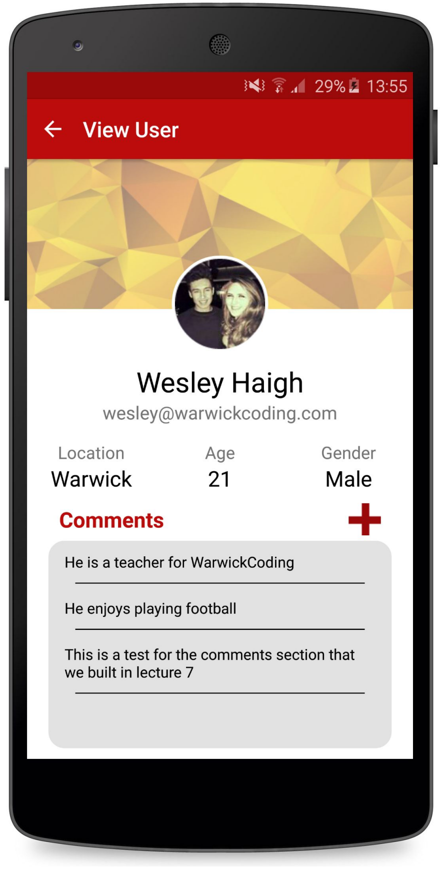
- Currently the buttons of the dialog (defined in **showCommentDialog**) just close the dialog.
- Modify **Add** button so that it adds a comment to the table (in showCommentDialog)



```
builder.setPositiveButton(R.string.viewUser_dialogButton_add,
        new DialogInterface.OnClickListener() {
            public void onClick(DialogInterface dialog, int id) {
                Comment comment = new Comment();
                comment.setUserId(_id);
                comment.setComment(input.getEditableText().toString());
                CommentTable commentTable = new CommentTable(ViewUserActivity.this);
                commentTable.createRow(comment);
                dialog.dismiss();
            }
        });
```

End of Lecture Exercise

- Display the comments for a user as shown on the right
  - We will need a listview in **content_view_user.xml**
  - And then to get hold of it in **ViewUserActivity.java**
  - We will need a **listview_comment_item.xml** file to define how an entry in the list will look
  - Finally we will need an adapter (**CommentListAdapter**) for the listview



Wesley Haigh
wesley@warwickcoding.com

Location	Age	Gender
Warwick	21	Male

Comments

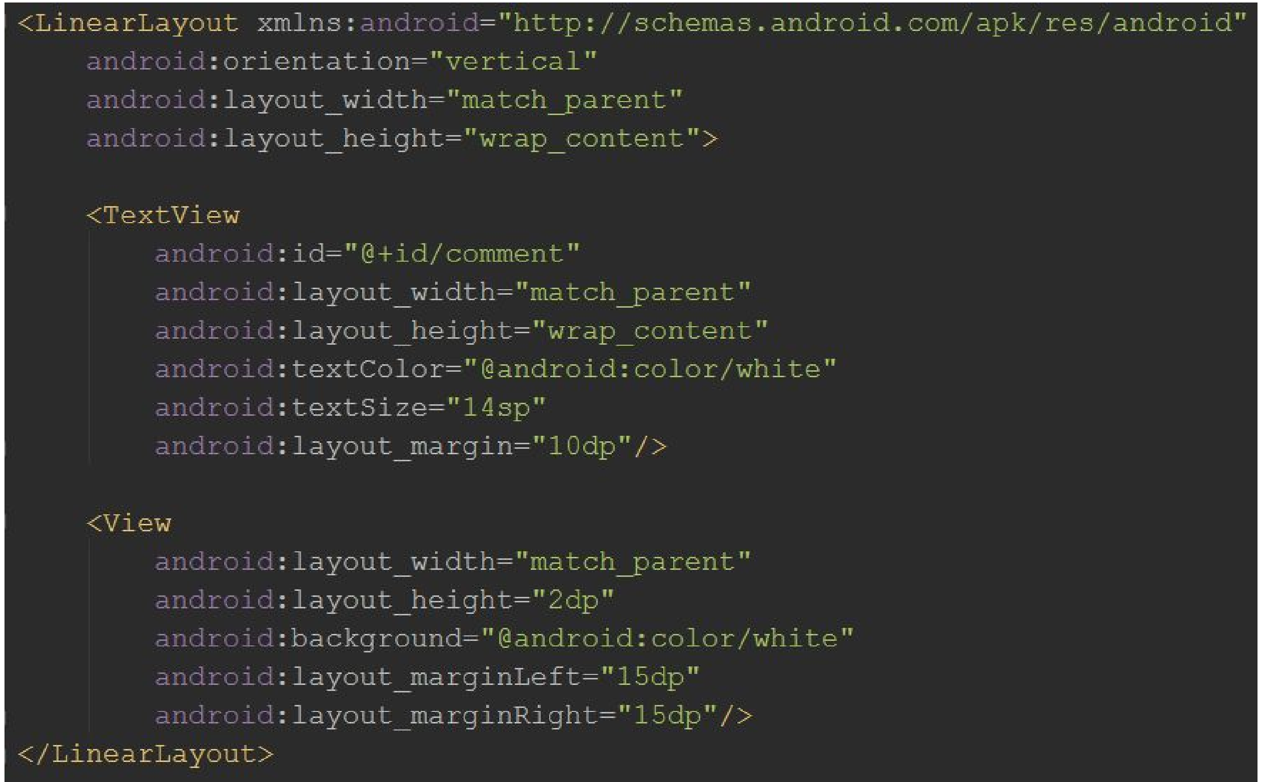
He is a teacher for WarwickCoding

He enjoys playing football

This is a test for the comments section that we built in lecture 7

End of Lecture Exercise

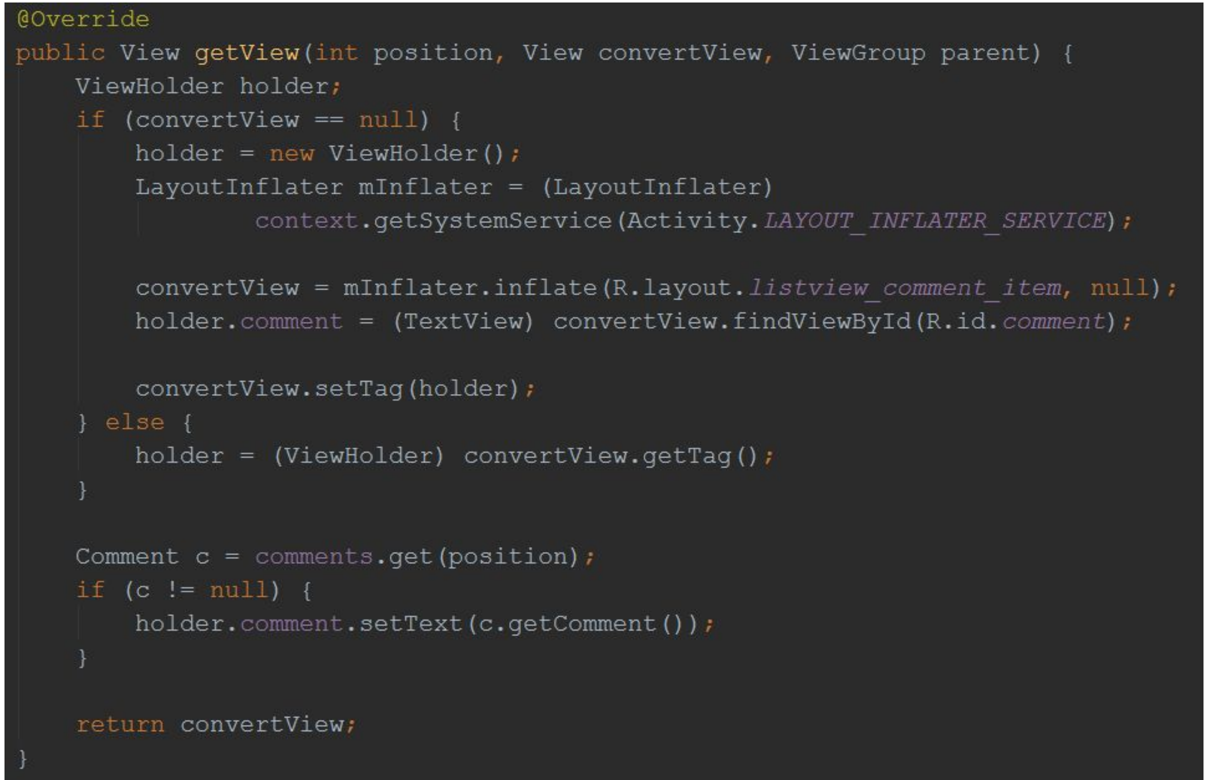
- listview_comment_item.xml



```xml
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="wrap_content">

    <TextView
        android:id="@+id/comment"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:textColor="@android:color/white"
        android:textSize="14sp"
        android:layout_margin="10dp"/>

    <View
        android:layout_width="match_parent"
        android:layout_height="2dp"
        android:background="@android:color/white"
        android:layout_marginLeft="15dp"
        android:layout_marginRight="15dp"/>
</LinearLayout>
```

End of Lecture Exercise

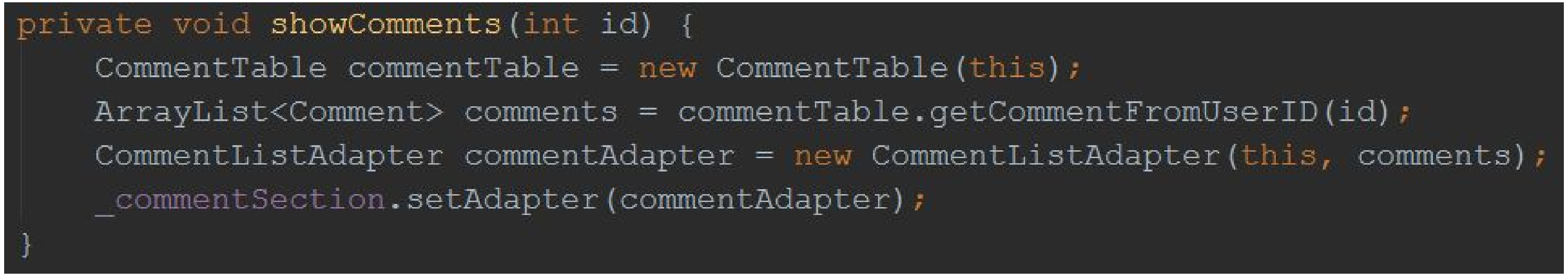
- CommentListAdapter.java → getView() function



```java
@Override
public View getView(int position, View convertView, ViewGroup parent) {
    ViewHolder holder;
    if (convertView == null) {
        holder = new ViewHolder();
        LayoutInflater mInflater = (LayoutInflater)
                context.getSystemService(Activity.LAYOUT_INFLATER_SERVICE);

        convertView = mInflater.inflate(R.layout.listview_comment_item, null);
        holder.comment = (TextView) convertView.findViewById(R.id.comment);

        convertView.setTag(holder);
    } else {
        holder = (ViewHolder) convertView.getTag();
    }

    Comment c = comments.get(position);
    if (c != null) {
        holder.comment.setText(c.getComment());
    }

    return convertView;
}
```

End of Lecture Exercise

- ViewUserActivity.java → **showComments(int id)** function. Call this at the end of setUserDetails(User user)



```java
private void showComments(int id) {
    CommentTable commentTable = new CommentTable(this);
    ArrayList<Comment> comments = commentTable.getCommentFromUserID(id);
    CommentListAdapter commentAdapter = new CommentListAdapter(this, comments);
    _commentSection.setAdapter(commentAdapter);
}
```

warwick
C<>ding