warwick C<>ding

by Wesley Haigh

Lecture 5

Databases

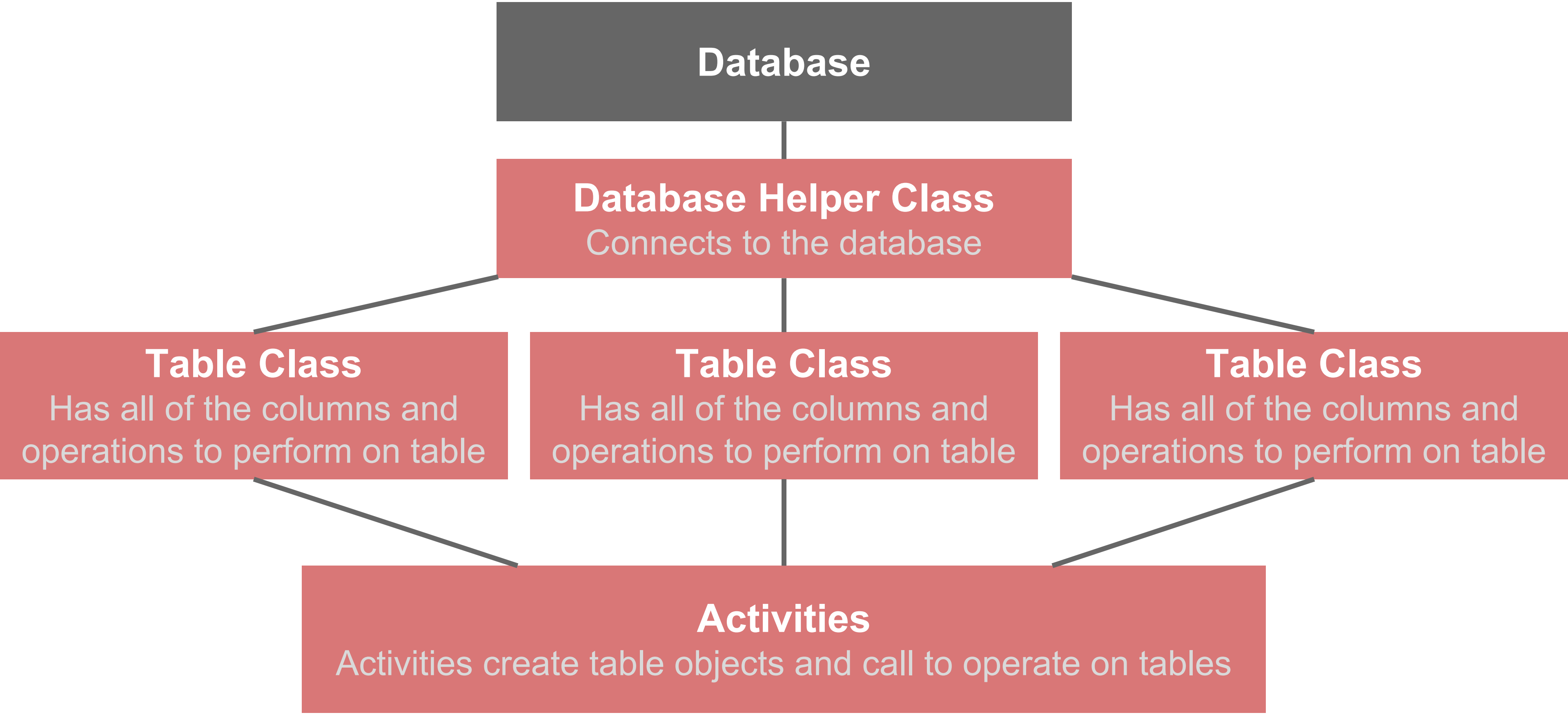
Recap

- 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

Android Databases

Adding a User table

SQLite - Basic Structure

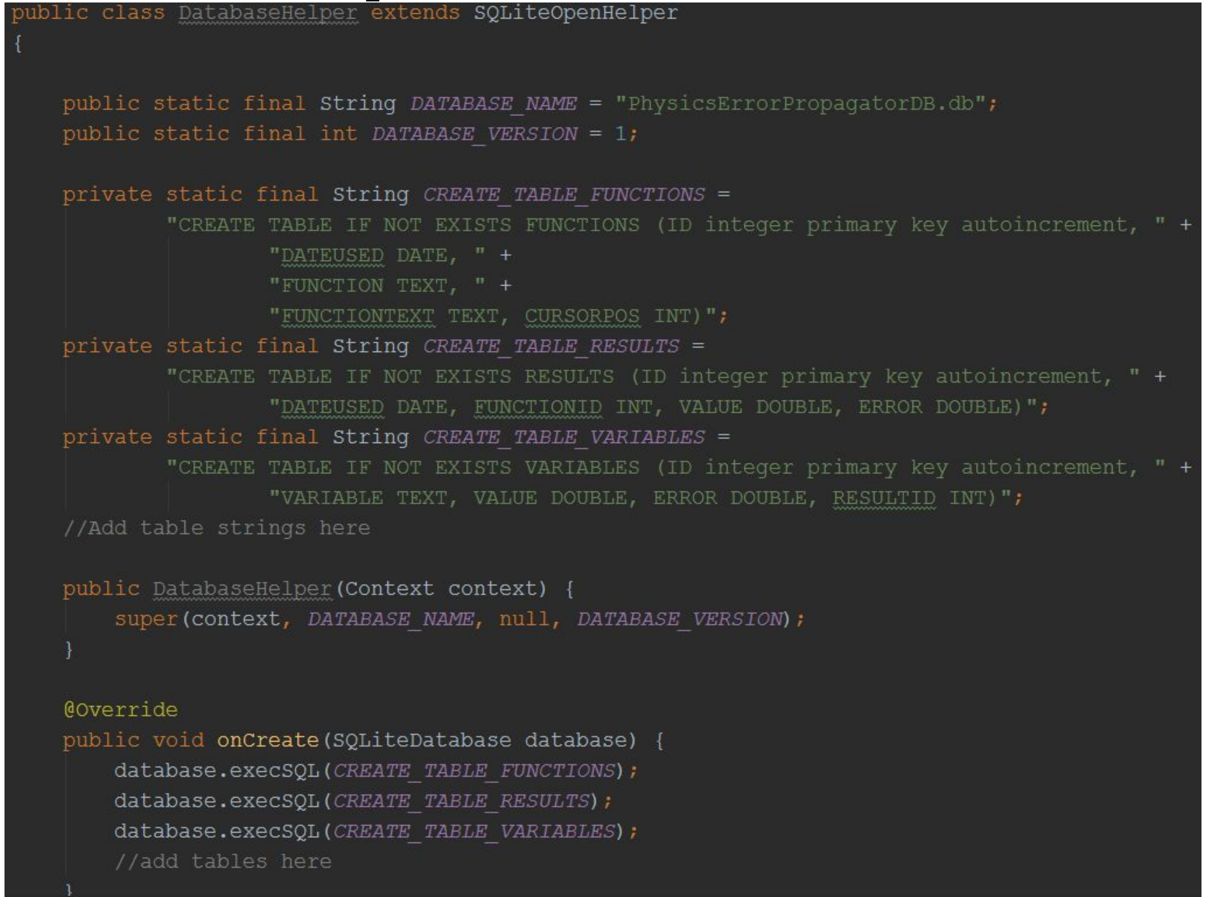


Database
Database Helper Class
Connects to the database
Table Class
Has all of the columns and operations to perform on table
Table Class
Has all of the columns and operations to perform on table
Table Class
Has all of the columns and operations to perform on table
Activities
Activities create table objects and call to operate on tables

Database Helper Class

- Sometimes called a **Database Adapter** as well
- It is the line of connection to the database
- Create a new **package** called **DatabaseClasses**
- Create a new java class in that package called **DatabaseAdapter**
- Go to the Google Drive folder and download and copy the contents of **DBAdapter.java** into your class

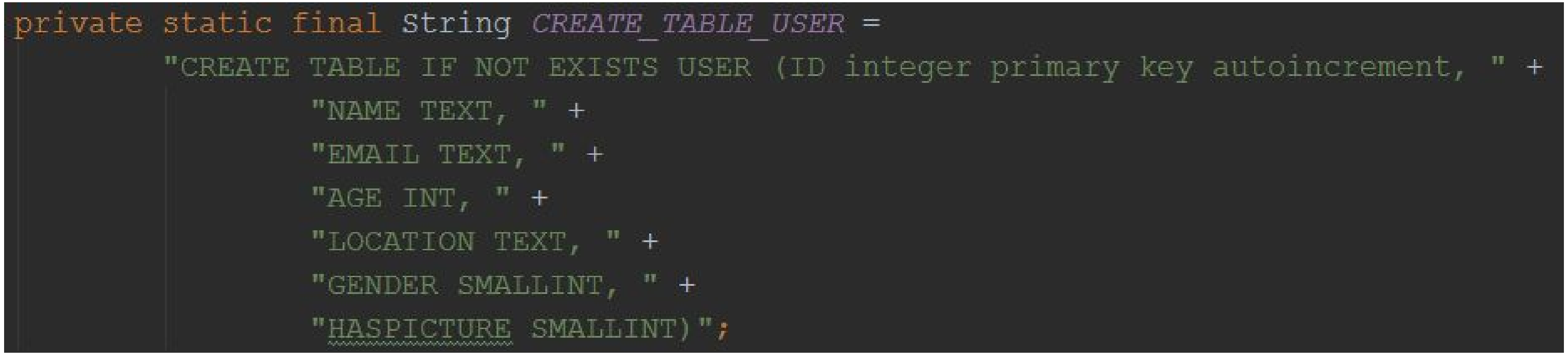
Database Helper Class



```
public class DatabaseHelper extends SQLiteOpenHelper
{

    public static final String DATABASE_NAME = "PhysicsErrorPropagatorDB.db";
    public static final int DATABASE_VERSION = 1;

    private static final String CREATE_TABLE_FUNCTIONS =
            "CREATE TABLE IF NOT EXISTS FUNCTIONS (ID integer primary key autoincrement, " +
                    "DATEUSED DATE, " +
                    "FUNCTION TEXT, " +
                    "FUNCTIONTEXT TEXT, CURSORPOS INT)";
    private static final String CREATE_TABLE_RESULTS =
            "CREATE TABLE IF NOT EXISTS RESULTS (ID integer primary key autoincrement, " +
                    "DATEUSED DATE, FUNCTIONID INT, VALUE DOUBLE, ERROR DOUBLE)";
    private static final String CREATE_TABLE_VARIABLES =
            "CREATE TABLE IF NOT EXISTS VARIABLES (ID integer primary key autoincrement, " +
                    "VARIABLE TEXT, VALUE DOUBLE, ERROR DOUBLE, RESULTID INT)";
    //Add table strings here

    public DatabaseHelper(Context context) {
        super(context, DATABASE_NAME, null, DATABASE_VERSION);
    }

    @Override
    public void onCreate(SQLiteDatabase database) {
        database.execSQL(CREATE_TABLE_FUNCTIONS);
        database.execSQL(CREATE_TABLE_RESULTS);
        database.execSQL(CREATE_TABLE_VARIABLES);
        //add tables here
    }
```

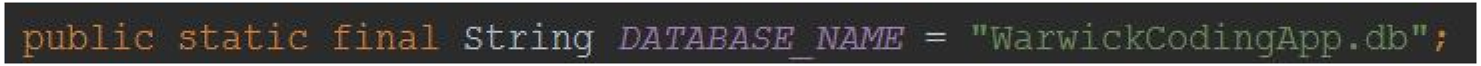
Database Helper Class

- Change it so that we make a **User** table



```
private static final String CREATE_TABLE_USER =
        "CREATE TABLE IF NOT EXISTS USER (ID integer primary key autoincrement, " +
                "NAME TEXT, " +
                "EMAIL TEXT, " +
                "AGE INT, " +
                "LOCATION TEXT, " +
                "GENDER SMALLINT, " +
                "HASPICTURE SMALLINT)";
```

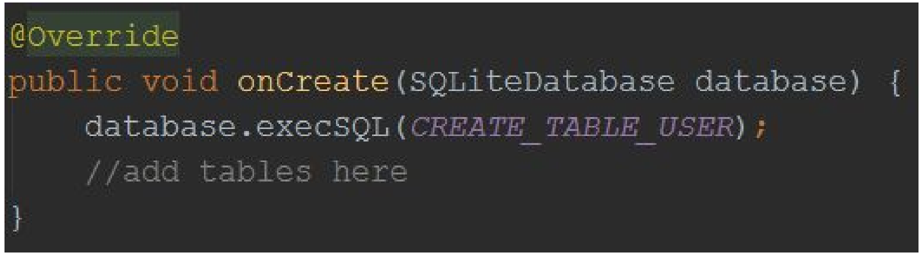
- Change the database name



```
public static final String DATABASE_NAME = "WarwickCodingApp.db";
```

Database Helper Class

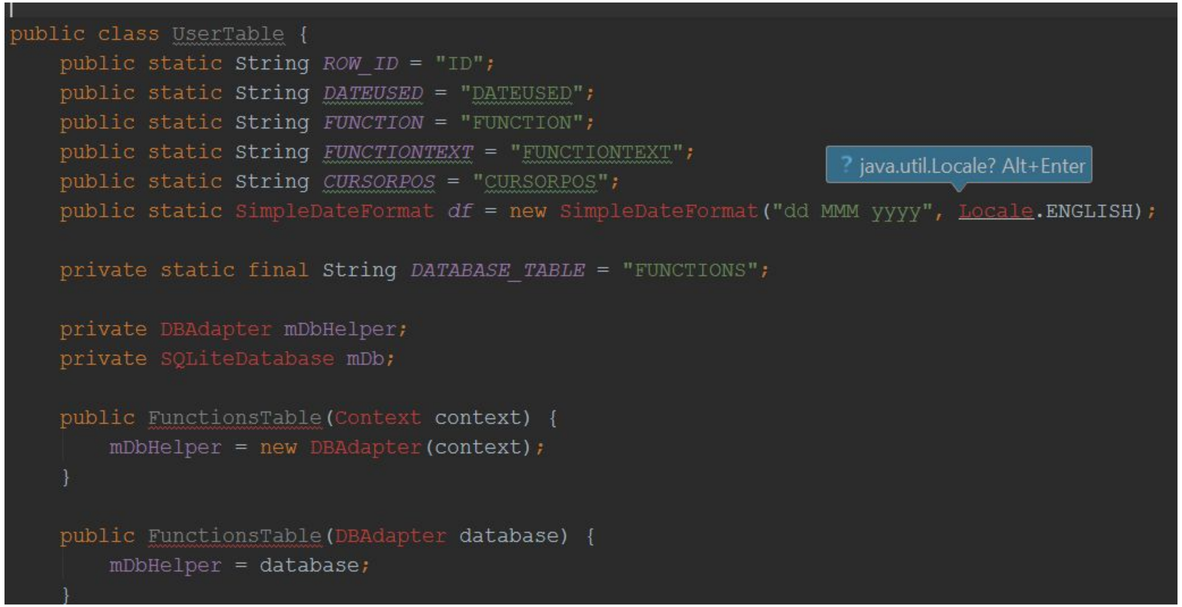
- Edit the **onCreate()** function:



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

UserTable Class

- Create another class in the **DatabaseClasses** package called **UserTable**
- Go to the Google Drive and copy the contents of **FunctionsTable**

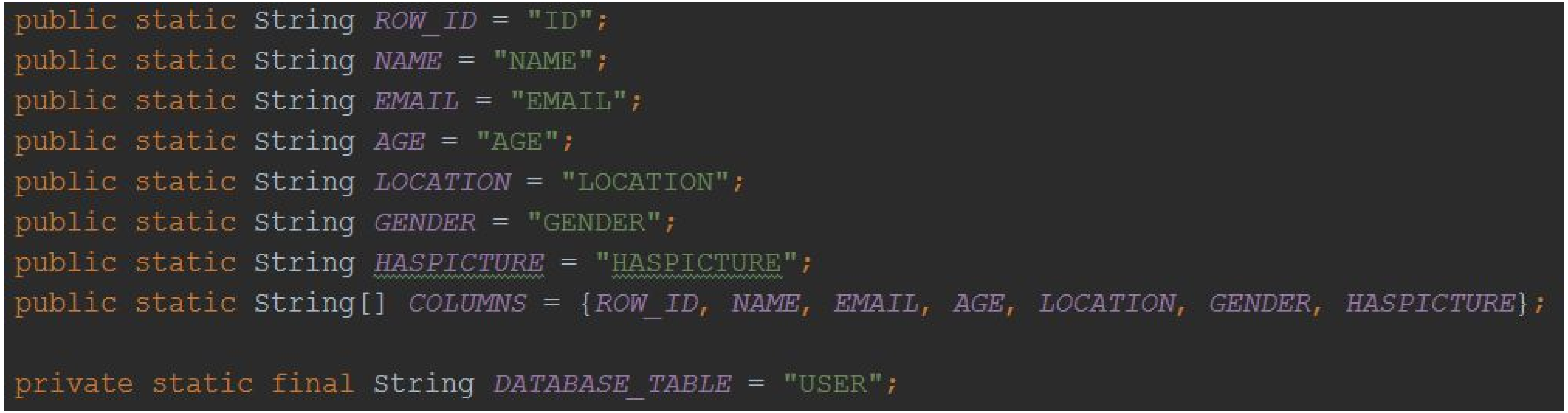


```
public class UserTable {
    public static String ROW_ID = "ID";
    public static String DATEUSED = "DATEUSED";
    public static String FUNCTION = "FUNCTION";
    public static String FUNCTIONTEXT = "FUNCTIONTEXT";
    public static String CURSORPOS = "CURSORPOS";
    public static SimpleDateFormat df = new SimpleDateFormat("dd MMM yyyy", Locale.ENGLISH);

    private static final String DATABASE_TABLE = "FUNCTIONS";

    private DBAdapter mDbHelper;
    private SQLiteDatabase mDb;

    public FunctionsTable(Context context) {
        mDbHelper = new DBAdapter(context);
    }

    public FunctionsTable(DBAdapter database) {
        mDbHelper = database;
    }
```

? java.util.Locale? Alt+Enter

UserTable Class

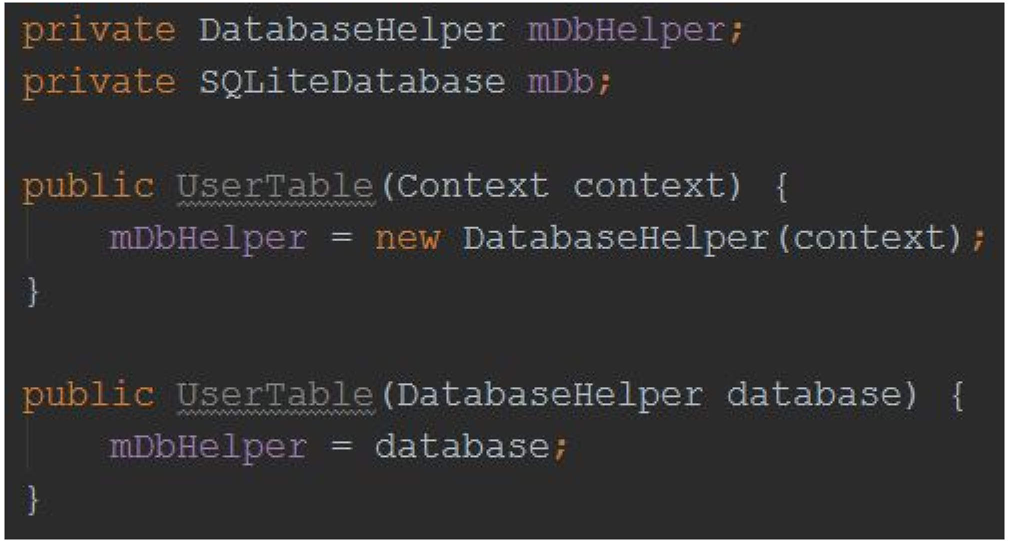
1. Change the table columns:



```
public static String ROW_ID = "ID";
public static String NAME = "NAME";
public static String EMAIL = "EMAIL";
public static String AGE = "AGE";
public static String LOCATION = "LOCATION";
public static String GENDER = "GENDER";
public static String HASPICTURE = "HASPICTURE";
public static String[] COLUMNS = {ROW_ID, NAME, EMAIL, AGE, LOCATION, GENDER, HASPICTURE};

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

UserTable Class

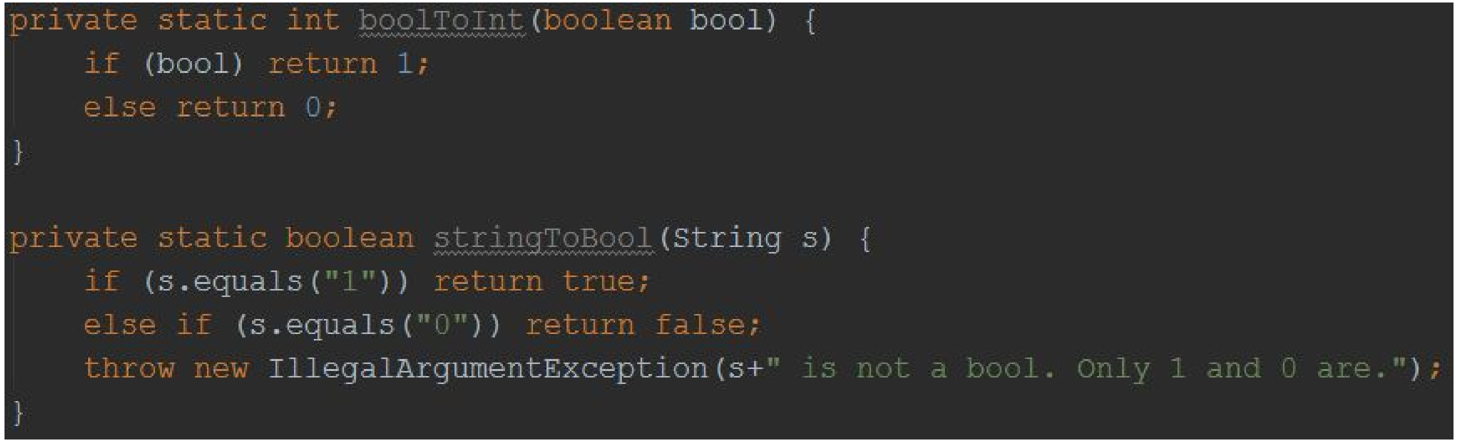
2. Import and modify constructors:



```
private DatabaseHelper mDbHelper;
private SQLiteDatabase mDb;

public UserTable(Context context) {
    mDbHelper = new DatabaseHelper(context);
}

public UserTable(DatabaseHelper database) {
    mDbHelper = database;
}
```

UserTable Class

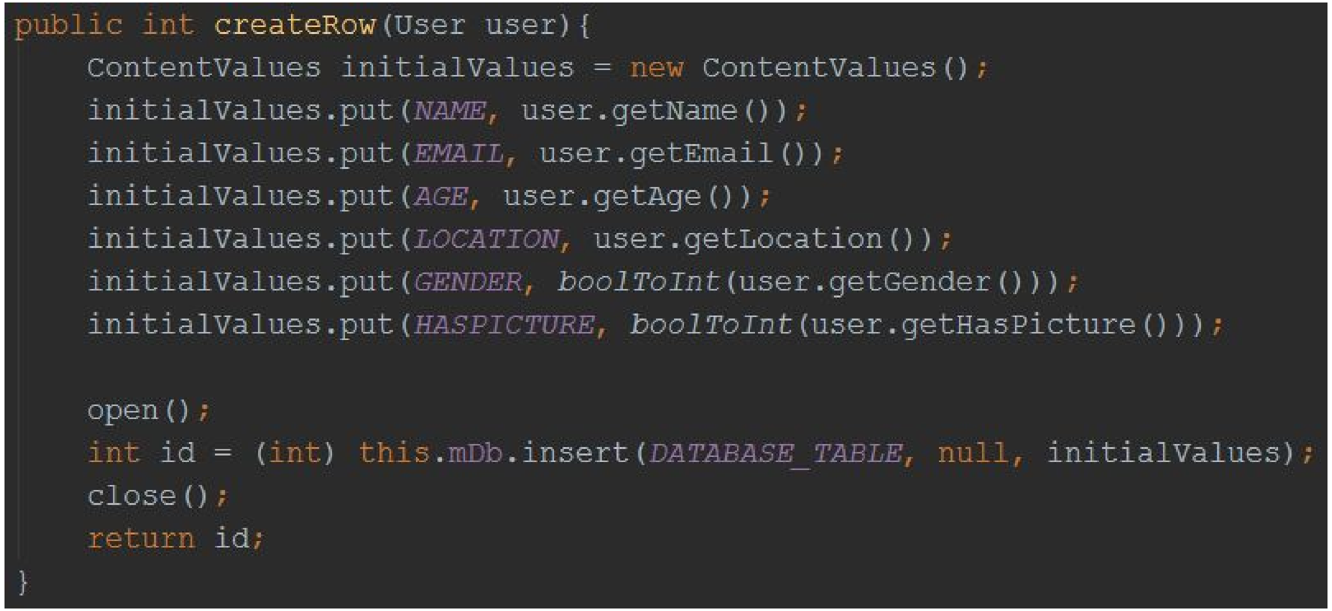
3. Add boolean → Smallint converter



```
private static int boolToInt(boolean bool) {
    if (bool) return 1;
    else return 0;
}

private static boolean stringToBool(String s) {
    if (s.equals("1")) return true;
    else if (s.equals("0")) return false;
    throw new IllegalArgumentException(s+" is not a bool. Only 1 and 0 are.");
}
```

UserTable Class

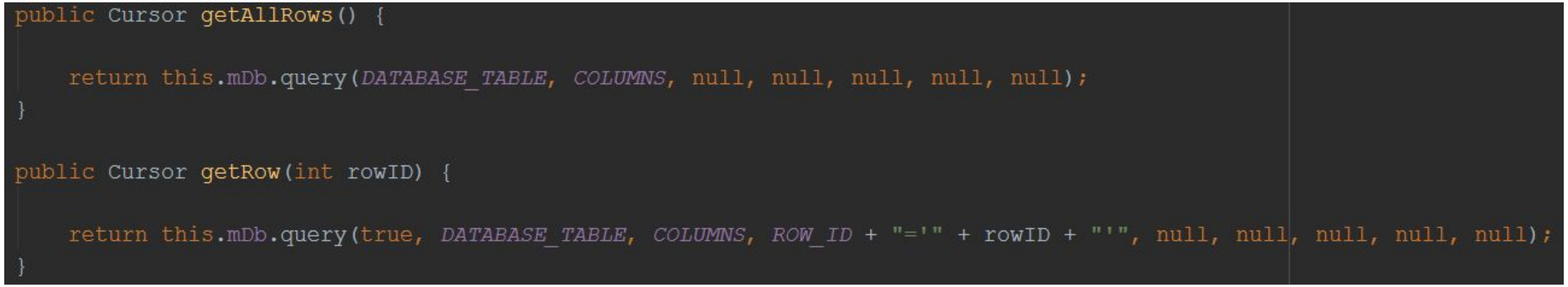
4. Correct other functions:



```
public int createRow(User user){
    ContentValues initialValues = new ContentValues();
    initialValues.put(NAME, user.getName());
    initialValues.put(EMAIL, user.getEmail());
    initialValues.put(AGE, user.getAge());
    initialValues.put(LOCATION, user.getLocation());
    initialValues.put(GENDER, boolToInt(user.getGender()));
    initialValues.put(HASPICTURE, boolToInt(user.getHasPicture()));

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

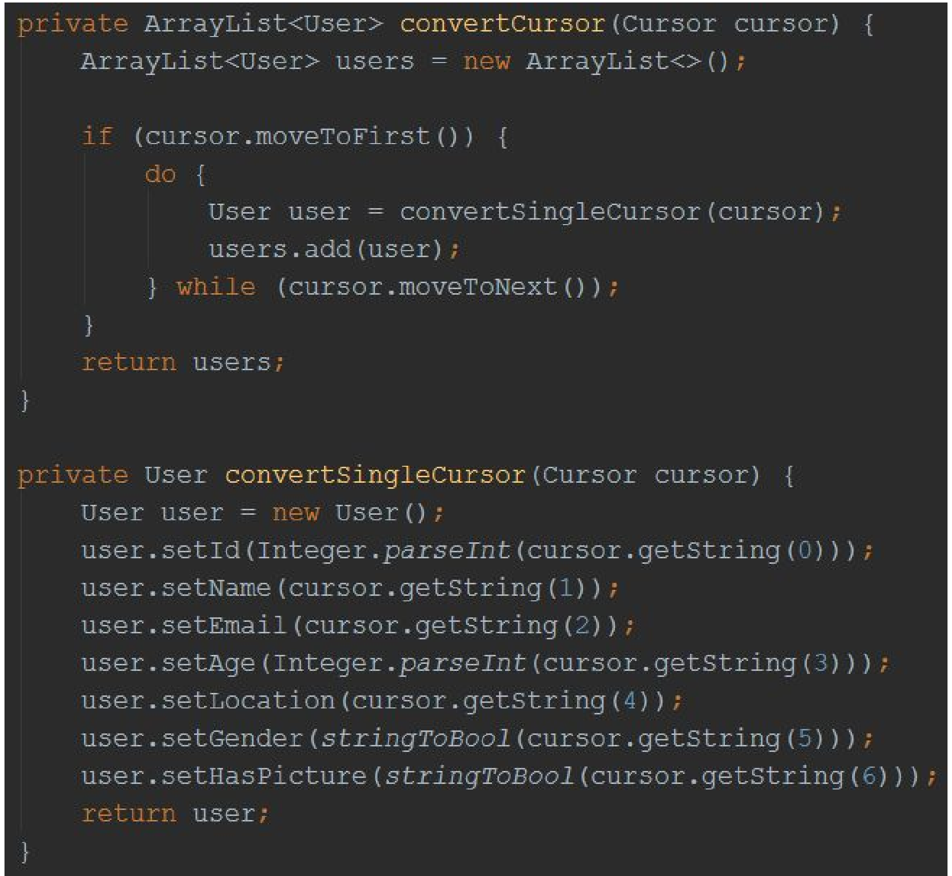
UserTable Class

4. Correct other functions **continued**:



```
public Cursor getAllRows() {

    return this.mDb.query(DATABASE_TABLE, COLUMNS, null, null, null, null, null);
}

public Cursor getRow(int rowID) {

    return this.mDb.query(true, DATABASE_TABLE, COLUMNS, ROW_ID + "='" + rowID + "'", null, null, null, null, null);
}
```

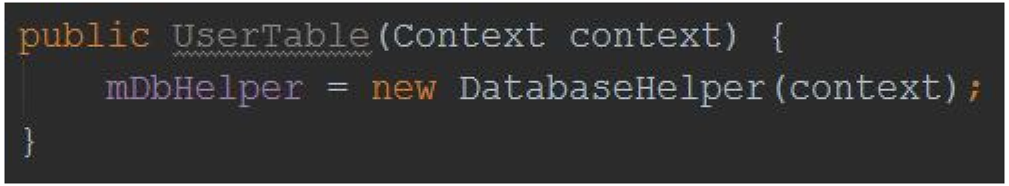
UserTable Class



```
private ArrayList<User> convertCursor(Cursor cursor) {
    ArrayList<User> users = new ArrayList<>();

    if (cursor.moveToFirst()) {
        do {
            User user = convertSingleCursor(cursor);
            users.add(user);
        } while (cursor.moveToNext());
    }
    return users;
}

private User convertSingleCursor(Cursor cursor) {
    User user = new User();
    user.setId(Integer.parseInt(cursor.getString(0)));
    user.setName(cursor.getString(1));
    user.setEmail(cursor.getString(2));
    user.setAge(Integer.parseInt(cursor.getString(3)));
    user.setLocation(cursor.getString(4));
    user.setGender(stringToBool(cursor.getString(5)));
    user.setHasPicture(stringToBool(cursor.getString(6)));
    return user;
}
```

Creating the database

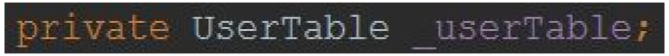
- To create the database all you need to do is **create our DatabaseHelper** class
- Given the way we have it set up we will create the database when we create a **UserTable** object to use to get or add records



```
public UserTable(Context context) {
    mDbHelper = new DatabaseHelper(context);
}
```

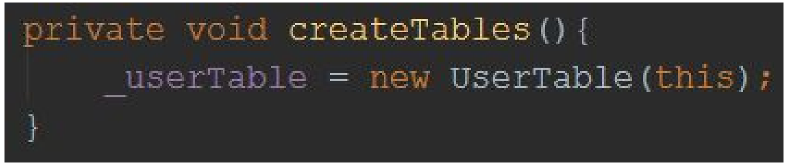
Adding a Record to UserTable

- Let’s modify our **Create Record** button so that it adds a record to the database
- Go to **AddRecordActivity.java**
- Add a class variable **_userTable** (UserTable)



```java
private UserTable _userTable;
```

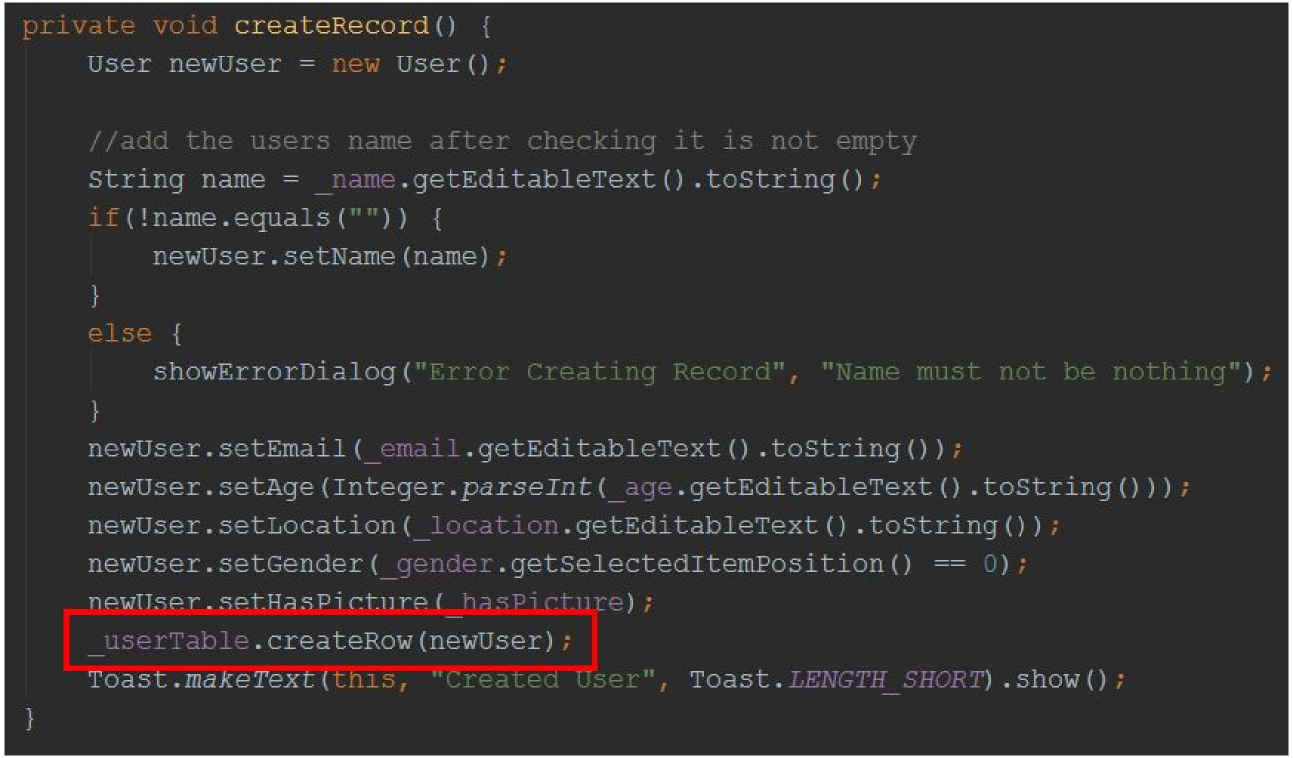
- Add a **createTables()** function (Remember to call it in OnCreate)



```java
private void createTables(){
    _userTable = new UserTable(this);
}
```

Adding a Record to UserTable

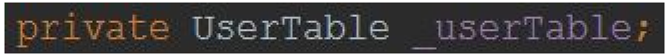
- Modify **createRecord()** function:



```
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(_hasPicture);
    _userTable.createRow(newUser);
    Toast.makeText(this, "Created User", Toast.LENGTH_SHORT).show();
}
```

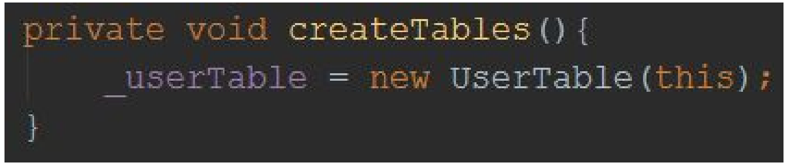
Getting Records From UserTable

- Go to **UsersActivity.java**
- Same as before, create a class variable called **_userTable** (UserTable)



```
private UserTable _userTable;
```

- Add **createTables()** function

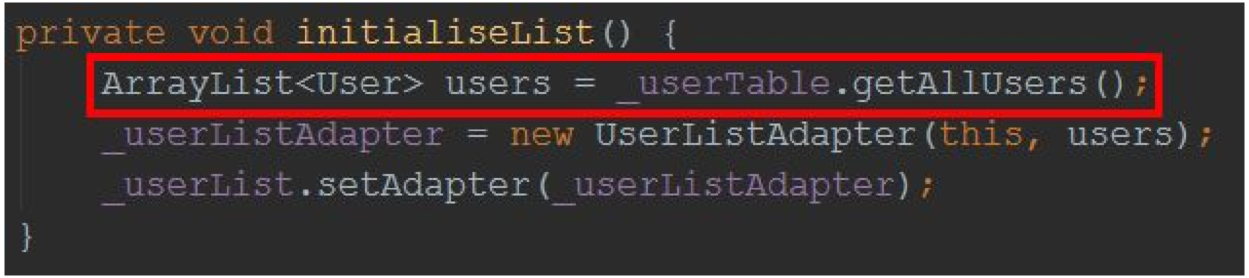


```
private void createTables(){
    _userTable = new UserTable(this);
}
```

- Now we can modify the **initialiseList()** function so that it actually gets data from the database and doesn't just make it up

Getting Records From UserTable

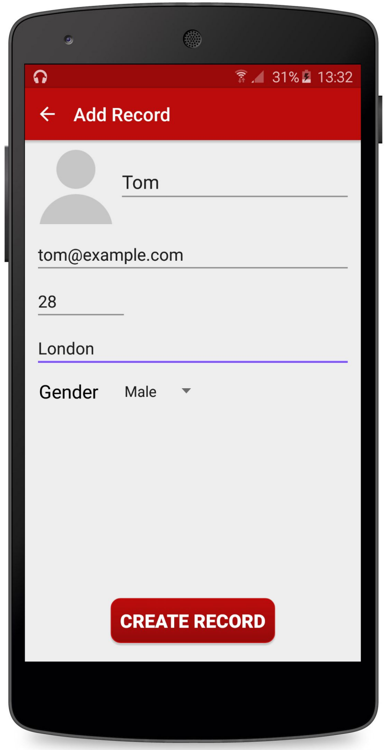
- Modify **initialiseList()**



```
private void initialiseList() {
    ArrayList<User> users = _userTable.getAllUsers();
    _userListAdapter = new UserListAdapter(this, users);
    _userList.setAdapter(_userListAdapter);
}
```

UserTable Implementation Complete

- Should have something that works as below:



31% 13:32

Add Record

Tom

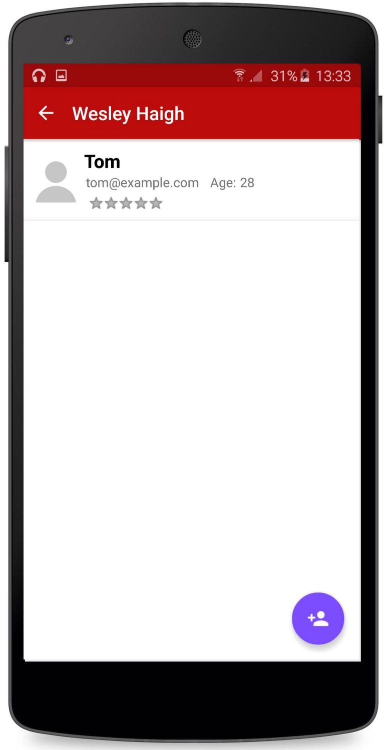
tom@example.com

28

London

Gender Male

CREATE RECORD



31% 13:33

Wesley Haigh

Tom

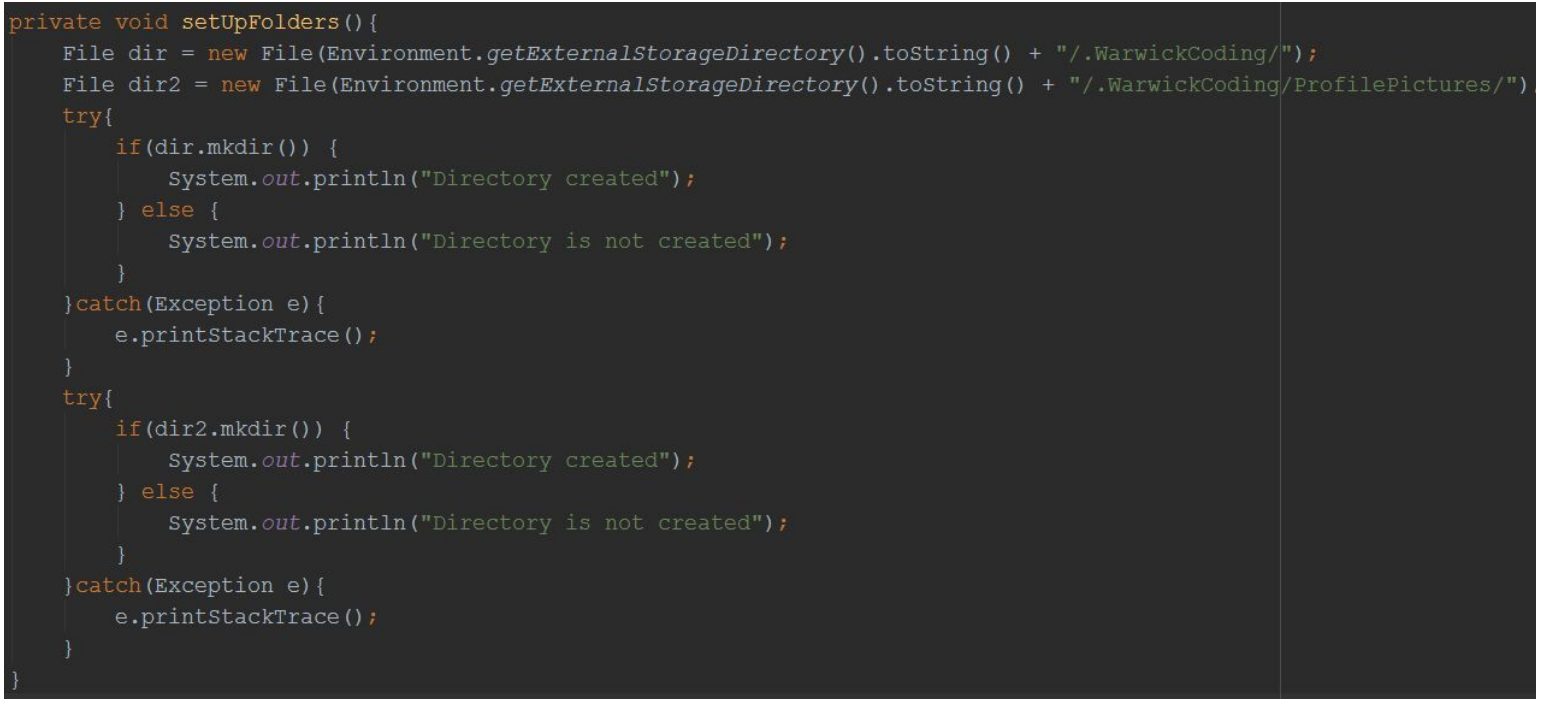
tom@example.com Age: 28

End of Lecture Exercise

- To get the profile pictures working:
  - Need to modify **MainActivity** so that it creates a folder to store the pictures
  - Need to save pictures to this folder when record is created
  - Need to get these pictures when user **hasPicture**

End of Lecture Exercise

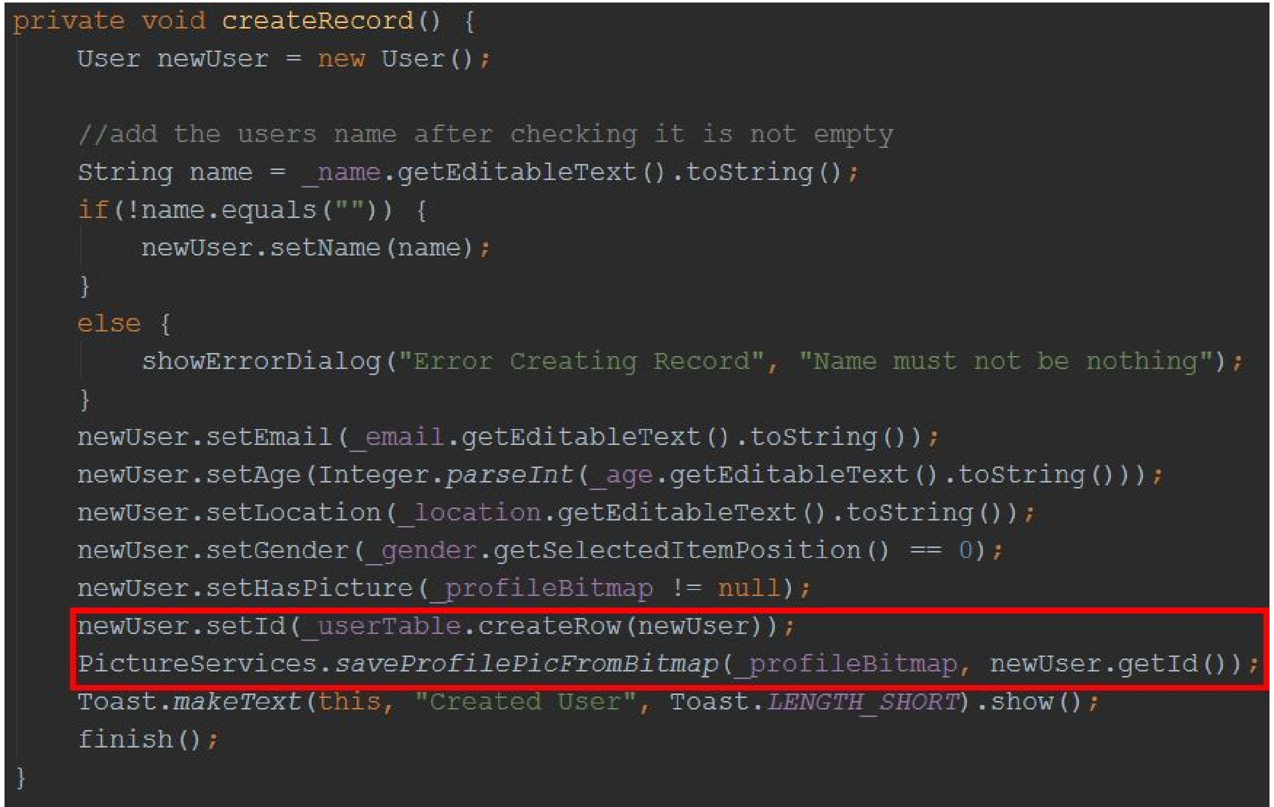
- Add **setUpFolders()** function in **MainActivity**



```
private void setUpFolders(){
    File dir = new File(Environment.getExternalStorageDirectory().toString() + "/.WarwickCoding/");
    File dir2 = new File(Environment.getExternalStorageDirectory().toString() + "/.WarwickCoding/ProfilePictures/")
    try{
        if(dir.mkdir()) {
            System.out.println("Directory created");
        } else {
            System.out.println("Directory is not created");
        }
    }catch(Exception e){
        e.printStackTrace();
    }
    try{
        if(dir2.mkdir()) {
            System.out.println("Directory created");
        } else {
            System.out.println("Directory is not created");
        }
    }catch(Exception e){
        e.printStackTrace();
    }
}
```

End of Lecture Exercise

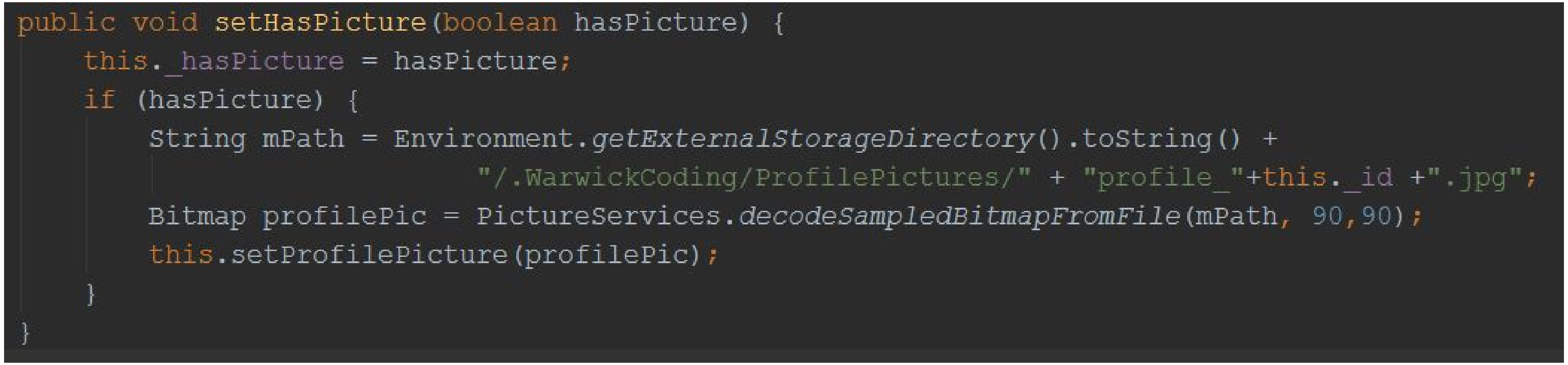
- Edit **createRecord()** function in **UsersActivty**



```
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);
    newUser.setId(_userTable.createRow(newUser));
    PictureServices.saveProfilePicFromBitmap(_profileBitmap, newUser.getId());
    Toast.makeText(this, "Created User", Toast.LENGTH_SHORT).show();
    finish();
}
```

End of Lecture Exercise

- Edit **setHasPicture()** function in **User**



```java
public void setHasPicture(boolean hasPicture) {
    this._hasPicture = hasPicture;
    if (hasPicture) {
        String mPath = Environment.getExternalStorageDirectory().toString() +
                                   "/.WarwickCoding/ProfilePictures/" + "profile_"+this._id +".jpg";
        Bitmap profilePic = PictureServices.decodeSampledBitmapFromFile(mPath, 90,90);
        this.setProfilePicture(profilePic);
    }
}
```

warwick
C<>ding