VICTORIA UNIVERSITY

NIT3213 Mobile Application Development

College of Engineering and Science

Assignment 1 – Mobile Timetabler Android Application

Due Date: Week 8

This assignment will test your Android development skills and is worth 10% of your overall unit mark.

You should work with a partner, two students per group is the maximum.

Android Development

This assignment requires you to develop a simple Android application that uses many Activities, which are related using Intents. To write the application you will need to use the Android Studio (IDE) 2.3.3..

Your application must perform all tasks listed in this document, and can run on any Android device with a minimum API version of 19 (KitKat 4.4) up to the latest version of Android.

Your application should be created to display correctly on an emulated **Nexus 5X** (1080 X 1920 pixels in portrait orientation) - however, by using sizes specified in "density independent pixels" (dp) the application should also be able to display correctly on any Android device with any sized screen and resolution.

Plagiarism Policy

I highly recommend that you familiarise yourself with what is and what is not considered plagiarism, but I'll summarise the key aspects below:

- **Do NOT share code with fellow students' other than your group**. I mean it. I have to mark all these projects, and I look at them closely. If I see two projects with identical code, all students involved get zero for the assignment.

- **You may, and are in fact encouraged to, *discuss* the project with your fellow students.** The goal being that you understand how to create good, high-quality Android applications – and that understanding then flows into the exams where you'll be asked questions on Android application development. Just don't share projects/code – code it yourself, learn from your mistakes, and become a great developer.

NIT3213 Mobile Application Development

College of Engineering and Science

Application Specification

Project Settings

Your Android application should have the following settings:

Minimum SDK: `API 19 Android 4.4`

Application Name: `Mobile Timetabler`

Project Name: `MobileTimetabler<your-student-ID-number>`

For example, `MobileTimetabler30078589`

Package name: `au.edu.vu.timetable<your-student-ID-number>`

For example, `au.edu.vu.timetable30078589`

Application Description

Your **Mobile Timetabler** application will do exactly what it sounds like. When the application first launches, it will display the days of the week Monday through to Friday, and underneath each day, it will display some details (which you will enter, and which will be saved into a simple SQLite database) about what classes / labs you have on that day.

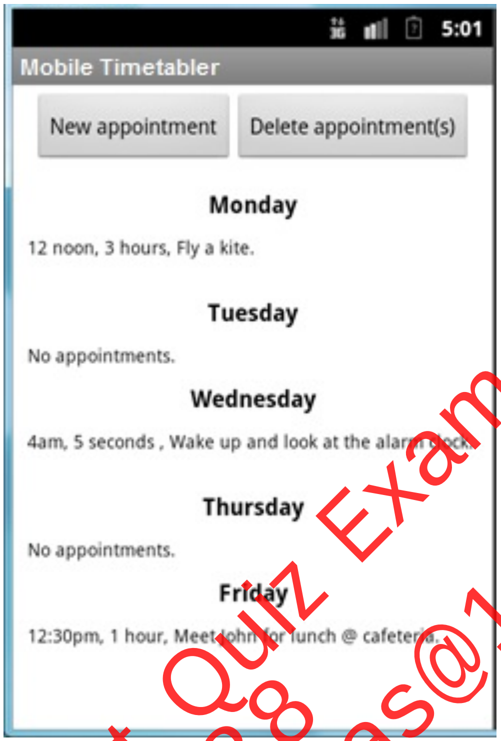
When the application starts it should have two buttons which are displayed at the top of the screen:

- A button to create a **New appointment**, and
- A button to **Delete appointment(s)**.

Under this, in a LinearLayout inside a ScrollView, there should be 5 centred TextViews with the days of the week Monday through Friday.

Under each day should be the text for one or more appointments. If there are no appointments saved for that day then the text "No appointments." is displayed.

An example of the MainActivity layout is shown below in figure 1 (Figures provided in this assignment are as sample only):



Monday

12 noon, 3 hours, Fly a kite.

Tuesday

No appointments.

Wednesday

4am, 5 seconds , Wake up and look at the alarm clock.

Thursday

No appointments.

Friday

12:30pm, 1 hour, Meet John for lunch @ cafeteria.

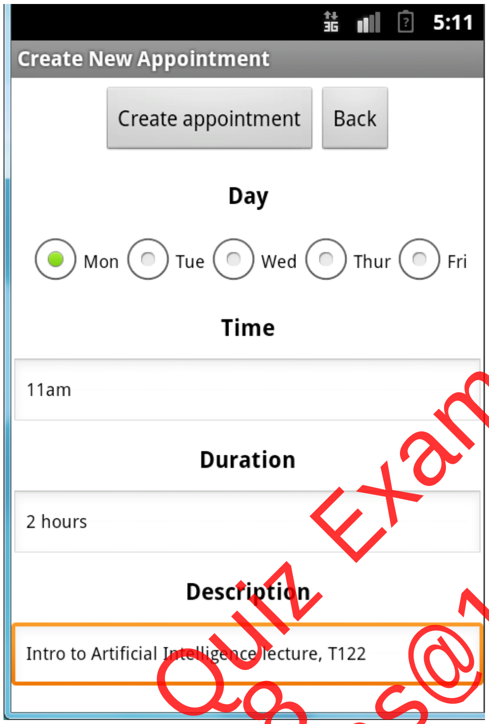
Figure 1 - The MainActivity layout. There are 3 appointments saved, one on Monday, another on Wednesday and the final one on Friday.

When the user clicks the **[New appointment]** button, an "onClick" method is executed which creates a new Intent, and starts a new **NewAppointmentActivity** using that intent.

The **NewAppointmentActivity** layout has two buttons, one to **[Create appointment]** which triggers the creation of a new **Appointment** object which must be saved to the SQLite database, and a second **[Back]** button which simply finishes the NewAppointmentActivity and returns to the MainActivity.

Under these two buttons are 5 radio buttons with the text "Mon", "Tue", "Wed", "Thur", and "Fri" on them inside a horizontal RadioGroup. It also has **TextViews** which display the words "Time", "Duration" and "Description". Under each of these three TextViews are **EditText** views which allow the user to enter text for the time, duration and description of the appointment.

The layout of the **NewAppointmentActivity** is shown in Figure 2, below:



Day

● Mon ○ Tue ○ Wed ○ Thur ○ Fri

Time

11am

Duration

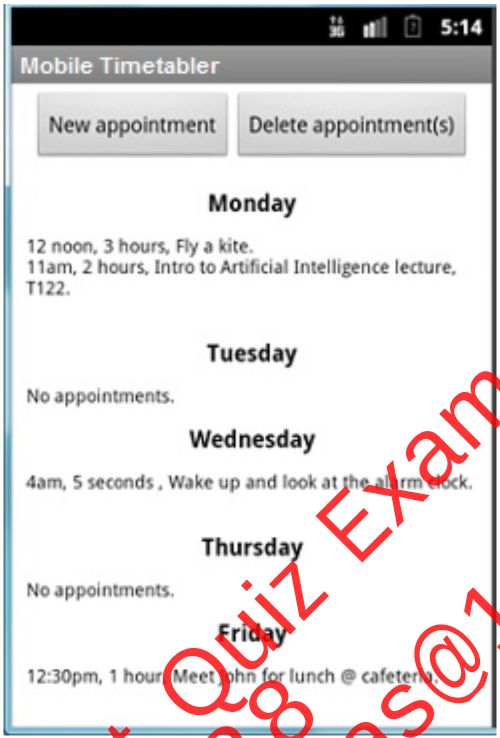
2 hours

Description

Intro to Artificial Intelligence lecture, T122

Figure 2 - The CreateNewAppointment Activity

Once the user clicks the **New appointment** button, the appointment must be saved in the database, and the NewAppointmentActivity finishes so we return to the MainActivity, which in both the **onCreate** and **onResume** methods calls another method called **populateSchedule** - which reads all the appointments from the database and puts them under the correct day for which they are scheduled. For example, after saving the above appointment, our new schedule is shown below in Figure 3.



Monday

12 noon, 3 hours, Fly a kite.
11am, 2 hours, Intro to Artificial Intelligence lecture, T122.

Tuesday

No appointments.

Wednesday

4am, 5 seconds , Wake up and look at the alarm clock.

Thursday

No appointments.

Friday

12:30pm, 1 hour, Meet John for lunch @ cafeteria.

Figure 3 - The result of adding a new appointment

If the user then clicks the **[Delete appointment(s)]** button the **DeleteAppointmentActivity** is executed. There are two ways that you can write the **DeleteAppointmentActivity** - a standard way.

DeleteAppointmentActivity - Standard Version

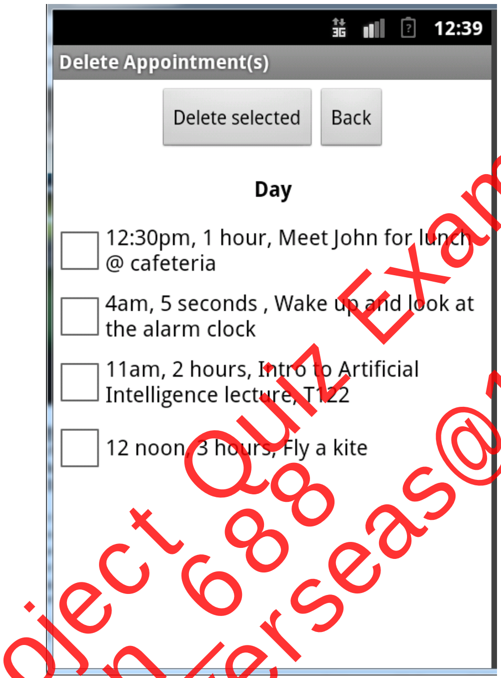
When a **DeleteAppointmentActivity** is launched, the database is queried for all saved appointments, and each appointment is displayed in a list of dynamically created checkboxes, one appointment per checkbox.

The **id** value from the database is retrieved (along with all other fields for each appointment), and the program sets the id value of the checkbox to be the id of the appointment. This way, the id value of each checked checkbox is the id value of the appointment we want to delete from the database.

A user can check some, all or none of the checkboxes and then click the **[Delete selected]** button, which will trigger deletion of the appointments from the database. This is achieved by creating a List of the checkbox id values where the checkbox is checked, and then passing that list to a **deleteScheduleItems** method which goes through each value in the list and deletes the record with that id from the database.

If the user clicks the **[Back]** button the activity finishes and we are returned to the MainActivity without making any changes to the database.

An example layout for the standard version of the DeleteAppointmentActivity is shown below in Figure 4:



□ 12:30pm, 1 hour, Meet John for lunch @ cafeteria

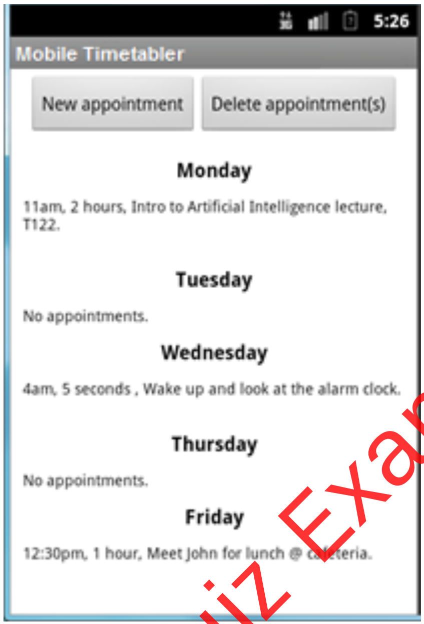
□ 4am, 5 seconds , Wake up and look at the alarm clock

□ 11am, 2 hours, Intro to Artificial Intelligence lecture, T122

□ 12 noon, 3 hours, Fly a kite

Figure 4 - The standard version of the DeleteAppointmentActivity

After deleting the "fly a kite" appointment above, our appointment list in the MainActivity will now look like Figure 5, below:



Mobile Timetabler

Monday

11am, 2 hours, Intro to Artificial Intelligence lecture, T122.

Tuesday

No appointments.

Wednesday

4am, 5 seconds , Wake up and look at the alarm clock.

Thursday

No appointments.

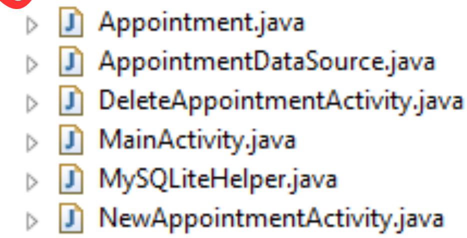
Friday

12:30pm, 1 hour, Meet John for lunch @ cafeteria.

Figure 5 - The result of deleting an appointment from the list of appointments stored for Monday.

Project Files

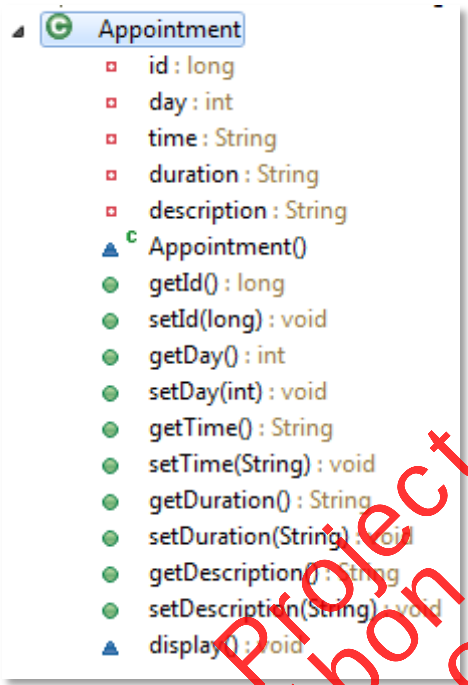
The classes used to in this program are as below (your project name and package should contain your student ID number, as specified in the project settings section on page 2 of this document):



Appointment.java
AppointmentDataSource.java
DeleteAppointmentActivity.java
MainActivity.java
MySQLiteHelper.java
NewAppointmentActivity.java

Class Specifications

The Appointment class



```
Appointment
    id : long
    day : int
    time : String
    duration : String
    description : String
    Appointment()
    getId() : long
    setId(long) : void
    getDay() : int
    setDay(int) : void
    getTime() : String
    setTime(String) : void
    getDuration() : String
    setDuration(String) : void
    getDescription() : String
    setDescription(String) : void
    display() : void
```

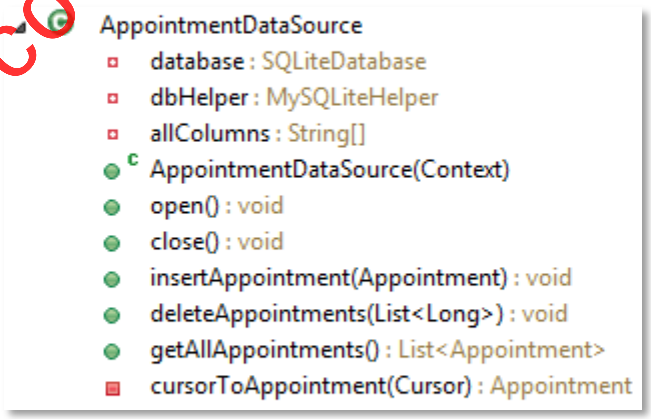
The appointment class stores an **id** value which uniquely identifies an appointment.

The day field is a value between 1 and 5 where 1 means "Monday" and 5 means "Friday".

The time, duration and description properties are all plain Strings.

Getters and Setters work as normal, and the **display()** method is just used to print out the fields of an appointment to the console, which comes in useful when debugging your application.

The AppointmentDataSource class

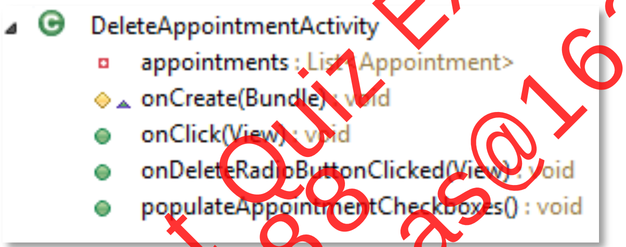


```
AppointmentDataSource
    database : SQLiteDatabase
    dbHelper : MySQLiteHelper
    allColumns : String[]
    AppointmentDataSource(Context)
    open() : void
    close() : void
    insertAppointment(Appointment) : void
    deleteAppointments(List<Long>) : void
    getAllAppointments() : List<Appointment>
    cursorToAppointment(Cursor) : Appointment
```

The **AppointmentDataSource** class allows us to create an AppointmentDataSource object, which provides methods to:

- Open the database,
- Insert an appointment into the database,
- Delete one or more appointments by providing a **List** of **id** values of the record(s) to delete,
- Return a **List** of all the appointments in the database, and
- Close the database.

The DeleteAppointmentActivity class - Standard Version



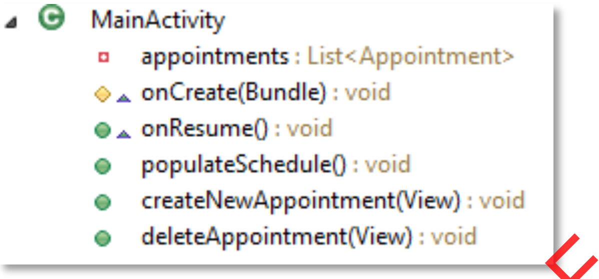
```
DeleteAppointmentActivity
    appointments : List<Appointment>
    onCreate(Bundle) : void
    onClick(View) : void
    onDeleteRadioButtonClicked(View) : void
    populateAppointmentCheckboxes() : void
```

When the **DeleteAppointmentActivity** starts, it populates a List of Appointments from the database. The text for each appointment is created through appending the time, duration and description data in a **StringBuilder** object, and then creating a new **CheckBox** and setting that appointment text on it. Additionally, each checkbox is assigned the **id** value of the appointment it represents.

When the **Delete appointment(s)** button is clicked, we go and find which checkboxes are currently checked, and add the **id** value(s) of the CheckBox to the list of appointments to delete.

We then call the **deleteAppointments** method (on a AppointmentDataSource object), passing it the list of appointment id's to delete.

The MainActivity class



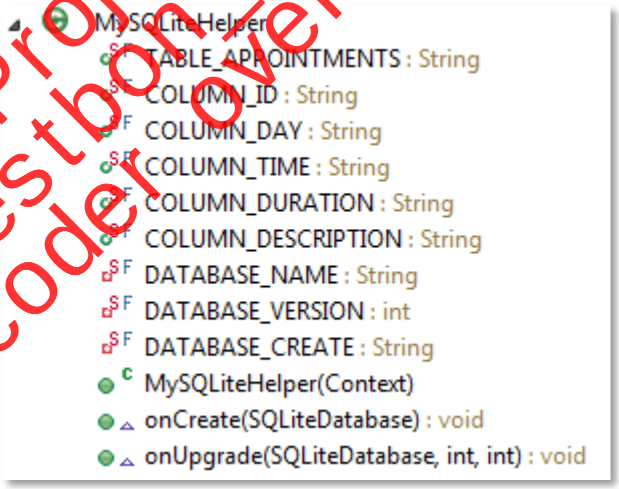
MainActivity
appointments : List<Appointment>
onCreate(Bundle) : void
onResume() : void
populateSchedule() : void
createNewAppointment(View) : void
deleteAppointment(View) : void

The MainActivity class starts by running the **populateSchedule()** method, which opens the database, retrieves all the saved appointments, and adds the text of those appointments to the **timetableLayout** displayed.

When the **[Create appointment]** button is clicked we create a new Intent and start a new **NewAppointmentActivity**.

When the **[Delete appointment(s)]** button is clicked, we create a new Intent use it to start a new **DeleteAppointmentActivity**.

The MySQLiteHelper class



MySQLiteHelper
TABLE_APPOINTMENTS : String
COLUMN_ID : String
COLUMN_DAY : String
COLUMN_TIME : String
COLUMN_DURATION : String
COLUMN_DESCRIPTION : String
DATABASE_NAME : String
DATABASE_VERSION : int
DATABASE_CREATE : String
MySQLiteHelper(Context)
onCreate(SQLiteDatabase) : void
onUpgrade(SQLiteDatabase, int, int) : void

The **MySQLiteHelper** class creates our SQLite database.

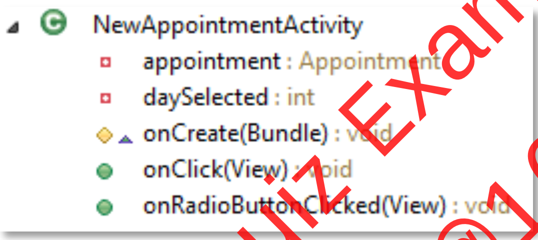
The **DATABASE_CREATE** string must contain commands to create the table with the following schema:

```
COLUMN_ID          - integer primary key autoincrement
COLUMN_DAY         - integer not null
```

```
COLUMN_TIME        - text not null
COLUMN_DURATION    - text not null
COLUMN_DESCRIPTION - text not null
```

The database name should be **appointments.db**

The NewAppointmentActivity class



```
NewAppointmentActivity
    appointment : Appointment
    daySelected : int
    onCreate(Bundle) : void
    onClick(View) : void
    onRadioButtonClicked(View) : void
```

The **NewAppointmentActivity** allows the user to select a day from a group of RadioButtons, and then enter text (as a String) for the time, duration and description.

If the **[Create appointment]** button is clicked the appointment is saved to the database. However, if the user opts to create a new appointment but has not entered any time, duration or description information (i.e. all three strings are empty) - then placeholder text is used for each field, which is specified as **"<No time>"**, **"<No duration>"** and **"<No description>"** respectively. After saving the appointment the activity finishes and we are returned to the MainActivity.

If the **[Back]** button is pressed then the activity finishes without saving the appointment and we are again returned to the MainActivity.

Submission and marking process

Create a single zip file containing the Android Studio project of your Mobile Timetabler and upload it to the upload location on VU Collaborate for Assignment 1 of this unit.

Assignments will be marked on the basis of fulfillment of the requirements.

In addition to the marking criteria, marks may be deducted for failure to comply with the assignment requirements, including (but not limited to):

- incomplete functionality,

- incomplete submissions (e.g. missing files)

Refer to the unit Description for details of the policy on marking of late assignments. Any applications for extensions or special consideration should be made as early as possible, and in all cases prior to the deadline for submission.

NIT3213 Mobile Application Development

College of Engineering and Science

Assignment 1 – Mobile Timetabler

Student name: **Student ID:**

Student name: **Student ID:**

Requirement	**Weight**	**Mark**
Correct usage of defining labels, the text on text views and buttons	2	
Correct usage of defining text styles for schedule items and TextViews	2	
Correct layout for MainActivity	1	
Correct layout for NewAppointmentActivity	1	
Correct layout for DeleteAppointmentActivity	1	
Ability to create database with correct schema	1	
Correct creation of the Appointment class	1	
Ability to save a new appointment to the SQLite database with details as entered by the user	3	
Ability to retrieve a previously saved appointment from the SQLite database	3	
Ability to display a retrieved appointment	3	
Ability to delete an appointment from the SQLite database	2	
Demonstration in Week 9 Lab	5	
Assignment mark total		/ 25
Contribution to unit mark (out of 10%)		%

Comments: