CSCI 322/522 **16. Android SQLite Databases (beginning)**

Introduction

- In order to manage persistent data, we can store the data in a file or in a database.
- In this chapter, we learn how to use SQLite, a light relational database management system (RDBMS) available on all Android devices.
- An RDBMS is a software program that manages relational databases.
- Although SQLite uses flat files to store data and is therefore not optimized for speed as compared to a regular RDBMS and to use SQL syntax.
- The typical SQL operations are insert, delete, update, and select.
- In this chapter, we also explore how to use menus: each menu item corresponds to an SQL (insert, delete, update, select) operation.
- To keep things simple, we only allow the device to run in vertical orientation.

Menus and Menu Items: Candy Store App

- When we start an app using the Basic Activity template, Android Studio generates two layout XML files and one menu XML file: `activity_main.xml`, `content_main.xml`, and `menu_main.xml`.
- This is different from the Empty Activity template, which only generates the `activity_main.xml`.
- At this point, create a new app named CandyStore but this time choose the Basic Activity template instead of Empty Activity.
- The following activity_main.xml file is automatically generated by the new Empty Activity:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.coordinatorlayout.widget.CoordinatorLayout
  xmlns:android="http://schemas.android.com/apk/res/android"
  xmlns:app="http://schemas.android.com/apk/res-auto"
  xmlns:tools="http://schemas.android.com/tools"
  android:layout_width="match_parent"
  android:layout_height="match_parent"
  tools:context=".MainActivity">

  <com.google.android.material.appbar.AppBarLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:theme="@style/AppTheme.AppBarOverlay">

    <androidx.appcompat.widget.Toolbar
      android:id="@+id/toolbar"
      android:layout_width="match_parent"
```

```
        android:layout_height="?attr/actionBarSize"
        android:background="?attr/colorPrimary"
        app:popupTheme="@style/AppTheme.PopupOverlay" />

    </com.google.android.material.appbar.AppBarLayout>

    <include layout="@layout/content_main" />

    <com.google.android.material.floatingactionbutton.FloatingActionButton
      android:id="@+id/fab"
      android:layout_width="wrap_content"
      android:layout_gravity="bottom|end"
      android:layout_margin="@dimen/fab_margin"
      app:srcCompat="@android:drawable/ic_dialog_email" />

  </androidx.coordinatorlayout.widget.CoordinatorLayout>
```

- The above code uses a `CoordinatorLayout` to arrange the elements.

- This layout is typically used as a top-level container for an app and as a container for specific interactions with one or more child Views.

- It includes three elements:

  1. An `AppBarLayout` itself includes a `ToolBar`. This is the action bar and this is where the menu items go.

  2. The View defined by `context_main.xml`, a `RelativeLayout` with a `TextView` inside it.

  3. A `FloatingActionButton` positioned at the bottom right and whose icon is a standard email icon from the Android icon library. In this app, we do not want that functionality, so we will delete those lines in bold above.

- In the `onCreate()` method of `MainActivity.java`, remove all of the following lines of code meant to handle the `FloatingActionButton` we just deleted in the code above:

```
FloatingActionButton fab =
    (FloatingActionButton) findViewById(R.id.fab);
fab.setOnClickListener(new View.OnClickListener()
{
  @Override
  public void onClick(View view)
  {
    Snackbar.make(view, "Replace with your own action",
         Snackbar.LENGTH_LONG).setAction("Action", null).show();
  }
});
```

(continued)

- The following shows the generated code in `menu_main.xml` of our project:

```
<menu
  xmlns:android="http://schemas.android.com/apk/res/android"
  xmlns:app="http://schemas.android.com/apk/res-auto"
  xmlns:tools="http://schemas.android.com/tools"
  tools:context="edu.niu.students.decker.CandyStore.MainActivity">
  <item
    android:id="@+id/action_settings"
    android:orderInCategory="100"
    android:title="@string/action_settings"
    app:showAsAction="never" />
</menu>
```

- This code defines a menu with one item.

- Menu items are shown in the action bar, starting from the right.

- However, if we run the skeleton app, no menu items show because the only menu item has the value `never` for the attribute `app:showAsAction` in the second to last line of the code above.

- The `Menu` interface of `android.view` encapsulates a menu.

- The `MenuItem` interface encapsulates a menu item within a menu.

- Attribute `android:title` has methods `setTitle(int)` and `setTitle(CharSequence)`.

- Constants of `MenuItem` that are possible arguments of the `setShowAsAction` and corresponding values for the `showAsAction` XML attribute:

Constant	**Attribute Value**	**Description**
`SHOW_AS_ACTION_NEVER`	never	Never show the item in the action bar.
`SHOW_AS_ACTION_ALWAYS`	always	Always show the item in the action bar.
`SHOW_AS_ACTION_IF_ROOM`	ifRoom	Show the item in the action bar if there is room.

- Open menu_main.xml and change/add the code in bold below:

```
<menu
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    tools:context="edu.niu.your z-id.CandyStore.MainActivity">

    <item
        android:id="@+id/action_add"
        android:title="@string/add"
        app:showAsAction="ifRoom" />
    <item
        android:id="@+id/action_delete"
        android:title="@string/delete"
        app:showAsAction="ifRoom" />

    <item
        android:id="@+id/action_update"
        android:title="@string/update"
        app:showAsAction="ifRoom" />

</menu>
```

- Open the `strings.xml` file and add the code in bold below:

```
<resources>
  <string name="app_name">CandyStore</string>
  <string name="action_settings">Settings</string>
  <string name="add">ADD</string>
  <string name="delete">DELETE</string>
  <string name="update">UPDATE</string>

  <string name="label_name">Name</string>
  <string name="label_price">Price</string>
  <string name="button_add">ADD</string>
  <string name="button_back">BACK</string>
</resources>
```

The remainder of this chapter is found in a document named *322 16. Android SQLite Databases – Notes (continued).pdf* in the Slides & Notes folder named 16. Android SQLite Databases.