Concordia University
COEN/ELEC 390
Fall 2023
Technical Assignment 1

Deadline:	October 6, 2023 @ 11:55 pm
Late Submission:	-20% for every 1 day late after due date

WARNING

You should do this assignment entirely by yourself, individually, using only materials provided in this document and by the Teaching assistants.

Objective

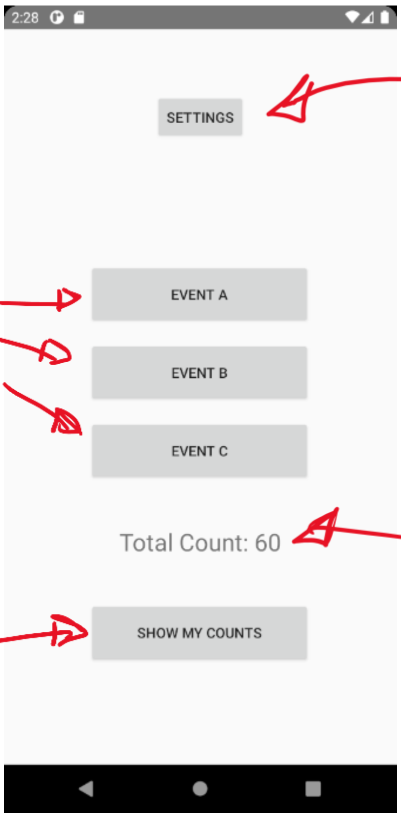
Design and implement an android mobile application with three activities. Key information saved using SharedPreferences in an MVC structure. By the end of the assignment, you will end up with a simple manual event counting application where you can store sequences of three distinct events, view the history of counted events and total amount counted each event type.

Application Description

Three Activities: mainActivity, dataActivity and SettingsActivity .

- mainActivity is the launcher activity of the application.
- SettingsActivity is a child Activity to mainActivity.
- dataActivity is a child Activity to mainActivity.

MainActivity



A button that takes you to the SettingsActivity

Three buttons representing individual events to count. Each showing the name of the event.

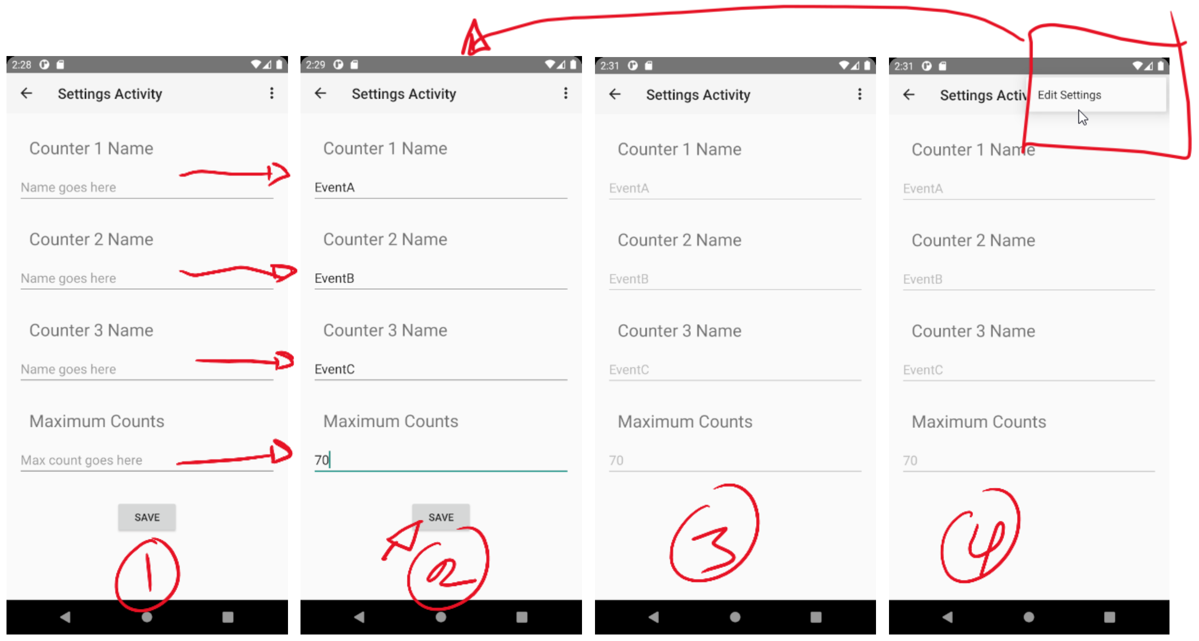
If there are no names saved already, the user will be redirected to the SettingsActivity.

A TextView that show the current total event count. **Updates every time a counter button is pressed**.

A button labeled "Show My Counts" that takes you to the dataActivity.

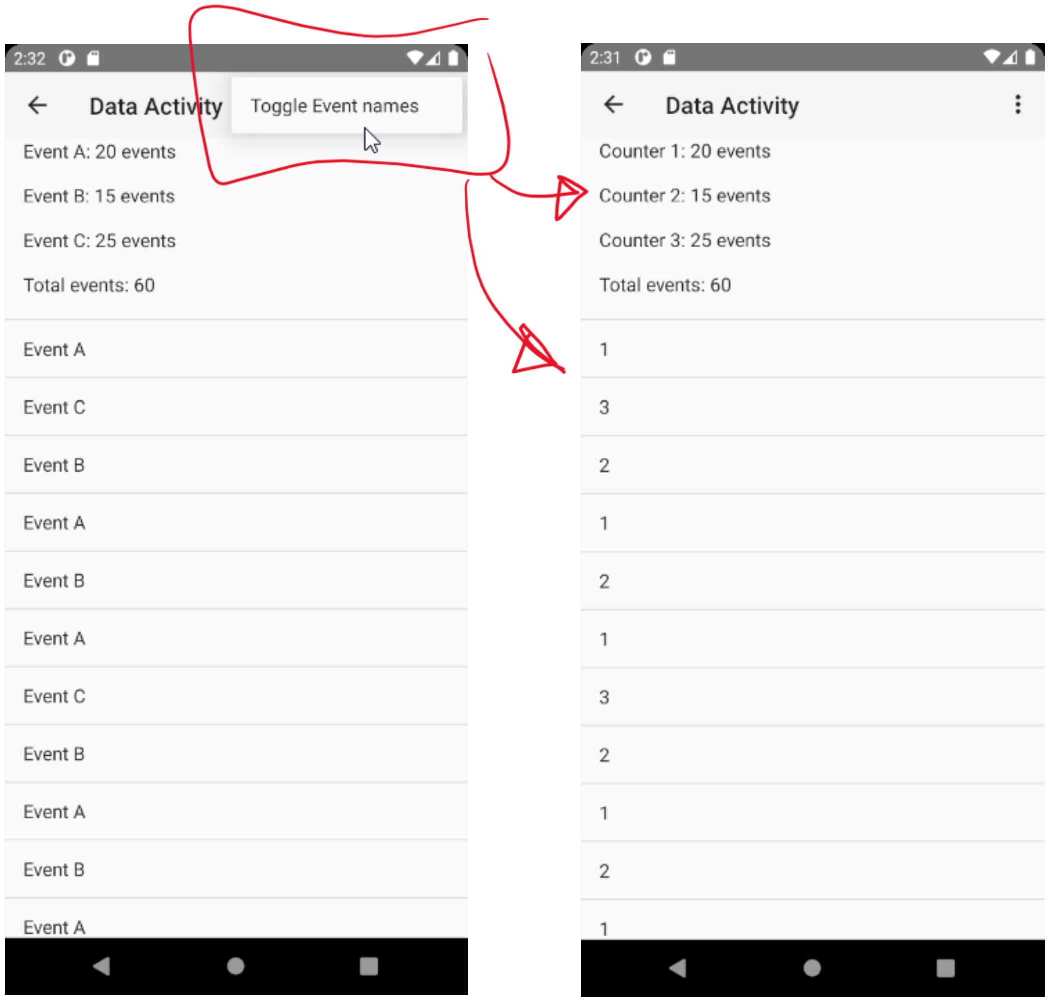
SettingsActivity

- When SettingsActivity is created:
    - The user will be able to see the counter button information displayed.
    - The user can switch to “edit mode” by pressing an action from the action bar.
    - The user can edit the fields of the activity when in “edit *mode*”
    - The user can save the information that were input by the user.
- The SettingsActivity will have two modes:
    - **Display mode:** The SettingsActivity is by default in display mode; the EditTexts displaying the information will not be editable. Meaning the user cannot write any text but can only view the text. The information is displayed on the activity layout as they were saved in the SharedPreference file. If the information does not exist, the text of every field of information <u>will be empty</u> (hints are ok, but not text in the field) and the SettingsActivity will automatically switch from the default display mode to the edit mode.
    - **Edit mode:** in the action bar, when the “edit” action is pressed, the activity switches to edit mode. The EditTexts fields become editable, and the user can enter information. A Button will show up at the bottom of the activity labeled “Save”. When this button is pressed (if no wrong values were entered) the data will be saved to SharedPreferences, the Save button will disappear, and the SettingsActivity will switch to display mode. If any value entered is wrong the user will get **a Toast message**, **nothing will be saved**, and the activity **will stay in edit mode**.
- The Settings Activity has the following 4 fields with the corresponding valid information:
    - **Button 1 name**: alphabetical characters and spaces. 20 char max length
    - **Button 2 name**: alphabetical characters and spaces. 20 char max length
    - **Button 3 name**: alphabetical characters and spaces. 20 char max length
    - **Maximum number of events**: number from 5 to 200
- The SettingsActivity has an up navigation that goes to the MainActivity when pressed.



DataActivity

- When dataActivity is created:
  - The user will be able to see the count for each individual event
  - The user will be able to see the total count number
  - The user will be able to see a scrollable list of events in the order in which they happened, oldest at the top to newest at the bottom.
- The dataActivity will have two modes:
  - **Event name mode:** default mode; shows the event names as defined in the settings activity for the counters and the scrollable event list.
  - **Event Button # mode:** in the action bar, when the "toggle event name" action is pressed, the activity switches to Event Button # mode. Instead of the event names, number from 1 to 3 are used.
- The dataActivity has an up navigation that goes to the MainActivity when pressed.



Event A: 20 events
Event B: 15 events
Event C: 25 events
Total events: 60

Event A
Event C
Event B
Event A
Event B
Event A
Event C
Event B
Event A
Event B
Event A

Counter 1: 20 events
Counter 2: 15 events
Counter 3: 25 events
Total events: 60

1
3
2
1
2
1
3
2
1
2
1

Things to help you with the assignment.

It might take some time depending on your programming skills but if you **read the tutorials** provided below and **go through the examples** provided then the assignment will basically be applying everything together in one application.

Providing up navigation (activities hierarchy)

https://developer.android.com/training/appbar/up-action
https://stackoverflow.com/questions/15686555/display-back-button-on-action-bar/37185334#37185334

Android List View

https://developer.android.com/guide/topics/ui/layout/listview.html
https://www.tutorialspoint.com/android/android_list_view.htm

Adding an action to the Action Bar

https://developer.android.com/training/appbar/actions.html

Detecting invalid inputs:

To detect an invalid input, you can read the input from the edit text when the save button is pressed and implement a method to check if the input is valid or not.

However, some of the inputs can be managed in a way to prevent the user to input any invalid information. For example, for text names you can add the following line to the EditText element in the xml layout file to make sure only alphabetical characters are allowed (this also disables spaces):

```
android:digits="abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ"
```

source: http://stackoverflow.com/questions/2361497/how-to-create-edittext-accepts-alphabets-only-in-android

For the Maximum Count, you can use a numerical edit text and specify **android:maxLength** in the xml file to specify the max length of 6. AND/OR you can do it in the activity java class.

Example of input validation for the edit text of the name in the demo project:

```
nameEditText = (EditText) findViewById(R.id.editText);
int maxLength = 6;
InputFilter[] FilterArray = new InputFilter[1];
FilterArray[0] = new InputFilter.LengthFilter(maxLength);
nameEditText.setFilters(FilterArray);
```

Preventing the user from writing invalid inputs will reduce the work when reading the input and saving it. For example, for the button names, you do not need to check every character if it is an alphabet or not anymore, you just need to check if the field is empty or not. Etc...

Hint, saving Lists in SharedPreferences

SharedPreferences are not typically used for lists. However, in a pinch, a list can be defined as a series of characters (a String), with each character being an element of the list. Comma separated values (CSV) Strings are also an option (more flexible, but heavier). This is not optimal, but in a pinch, it can be a quicker solution to program than using an SQLite database.

One Last Hint

To make an edit text editable:

```
mNameEditText.setEnabled(true);
```

To make an edit text un-editable

```
mNameEditText.setEnabled(false);
```

Building an MVC structure

Part of the assignment was built during the tutorial using the SharedPreferences. However, the project does not follow MVC structure. To do so, you will need a "Controller" for the SharedPreference. Which we create as a java class called SharedPreferenceHelper. See below for a Controller/Helper for profile name string values.

```
public class SharedPreferenceHelper {

    private SharedPreferences sharedPreferences;
    public SharedPreferenceHelper(Context context)
    {
        sharedPreferences = context.getSharedPreferences("ProfilePreference",
Context.MODE_PRIVATE );
    }

    public void saveProfileName(String name)
    {
        SharedPreferences.Editor editor = sharedPreferences.edit();
        editor.putString("profileName",name );
        editor.commit();
    }

    public String getProfileName()
    {
        return sharedPreferences.getString("profileName", null);
    }
}
```

and in the MainActivity for example, instead of implementing the usage of SharedPreference we use SharedPreferenceHelper as follow:

```java
public class MainActivity extends AppCompatActivity {


    protected SharedPreferenceHelper sharedPreferenceHelper;

    protected Button button = null;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        sharedPreferenceHelper = new SharedPreferenceHelper(MainActivity.this);

        button = (Button) findViewById(R.id.profileButton);


        button.setOnClickListener(new View.OnClickListener() {
            public void onClick(View v) {


                 goToSettingsActivity();

            }
        });


    }

    protected void onStart()
    {
        super.onStart();

        String name = sharedPreferenceHelper.getProfileName();
        if(name == null)
            goToSettingsActivity ();
        else
            button.setText(name);

    }

    void goToSettingsActivity ()
    {
        Intent intent = new Intent(MainActivity.this, Profile.class);
        startActivity(intent);
    }


    }
```

This example only works for saving and getting a String representing the profile name. However, in this assignment you will need to get Settings and save Settings.

Tip: create a Settings java class that holds all the information of the profile with getters and setters. And your SharedPreferenceHelper methods will take a Settings object as parameter (or return a Settings object) instead of a String.

Assignment submission and procedure

Original work

This is an individual assignment. Your submission must be your own work.

You may use online sources to understand, but you must write your own code. To ensure this, you should:

1) **Never copy and paste code.**
2) **Never have a window with code from a website open at the same time you have a window with your code open.**

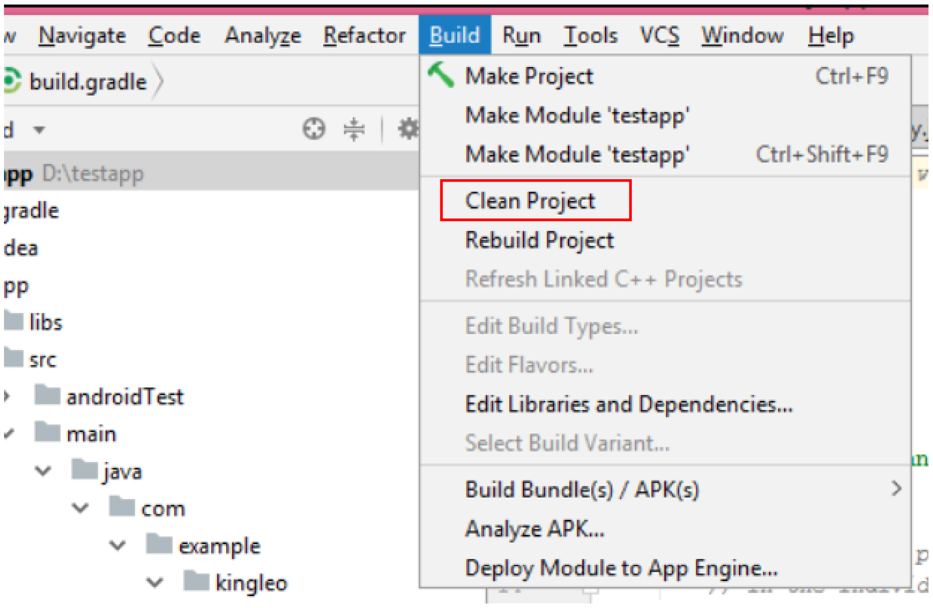
Also, you should never use code from another student in any way.

You must submit your assignment before midnight on the due date using moodle Assignment Submission **in the submission link tagged with the tutorial section you are registered in (very important, a wrong submission might be considered a late submission)**. The file submitted **must be a .zip (no .rar)** file named **StudentID_Ass1** containing **the entire android project folder**.

Before submitting your code make sure you clean the project.

Points may be deducted for uncleaned projects.

Use this: `Android Studio --> Build --> Clean Project`



Evaluation criteria and grading scheme

Meeting the requirements and use cases	80%
Using MVC design	15%
Clean code: well commented, proper naming, easy to read and understand.	5%
Possible bonus for quality UI design.	

If the project submitted does not compile and run the student will receive a grade of 0! So, make sure even if the assignment is not completely done that you submit an application that can be built and run. We will not grade none compiling code.