Concordia University
COEN/ELEC 390
Winter 2021
Technical Assignment 1

Deadline:	February 2, 2021 @ 11:55 pm

Objective

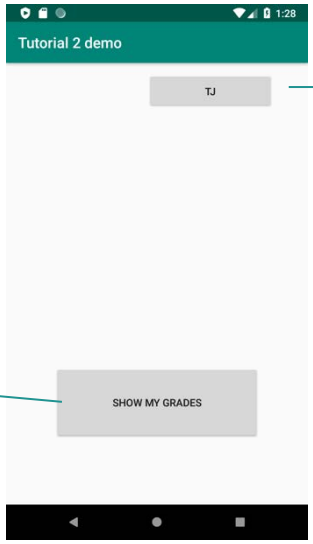
Design and implement an android mobile application to view a list of courses with their assignments and grades with user profile information saved using SharedPreferences in an MVC structure. By the end of the assignments we will end up with a simple grades management application where students can store assignment grades for the courses they are taking and be able to view the courses with their grades.

Application Description

Three Activities: mainActivity, gradeActivity and profileActivity.

- mainActivity is the launcher activity of the application
- profileActivity is a child Activity to mainActivity.
- gradeActivity is a child Activity to mainActivity

mainActivity:



A button with a label of the name of the profile saved that takes you to the profileActivity.

If there is no name saved already, the user will be redirected to the profileAcitivity.

A button labeled “Show My Grades” that takes you to the gradeActivity.

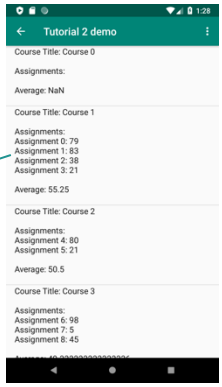
gradeActivity:

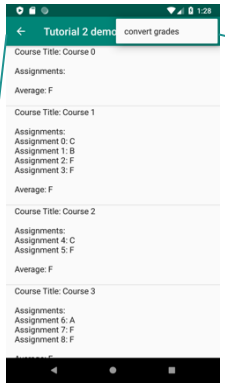
When gradeActivity is created:

Generate random number (1 to 5) of Course instances (Course class explained below).

Display the Courses with their Assignments in a ListView.

Calculate the Average of the grades per Course and display it, assuming all grades are out of 100.



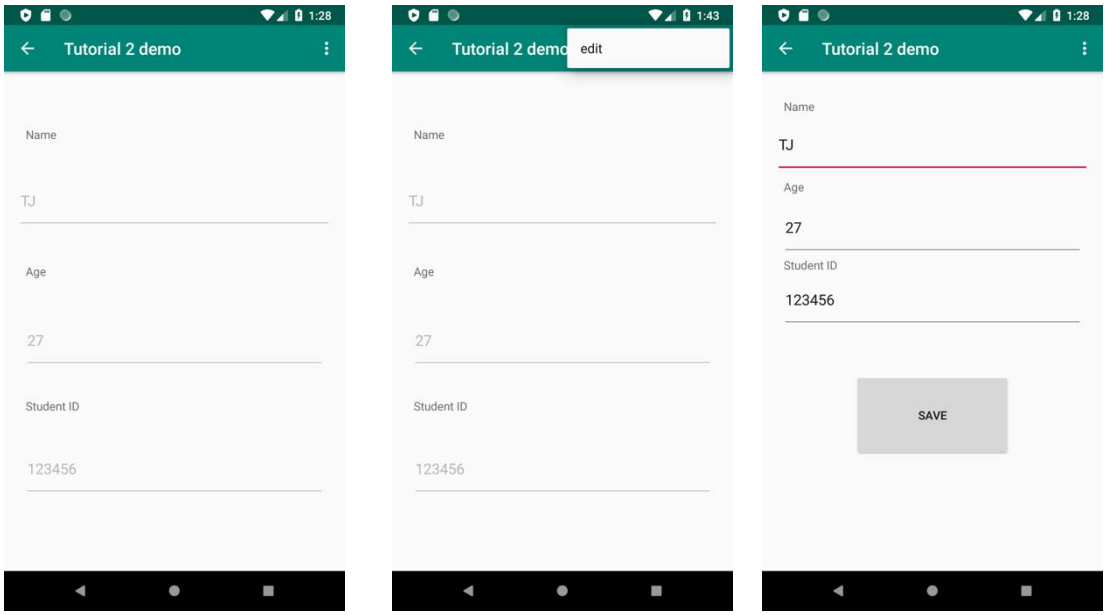


An Action in the action bar, when pressed the grades displayed will switch from number grades to letter grades (and vice versa).

Up Navigation action, when pressed takes you to the parent activity (the MainActivity)

ProfileActivity

- When profileActivity is created:
  - The user will be able to see the profile 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 profile information when in “edit mode”
  - The user can save the information that were input by the user.
- The Profile has the following information: **Name, Age, Student ID.**
- The profileActivity will have two modes:
  - **Display mode:** The profileActivity 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 profile 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 will be empty and the profileActivity will automatically switch from the default display mode to the edit mode.
  - **Edit mode:** when the action edit, from the action bar, is pressed the activity switches to edit mode. The Edit Texts of the fields will switch to being editable where the user can write the information of his/her profile. A Button will show up at the bottom of the activity labeled “Save”. When this button is pressed the data (if no wrong values are entered) will be saved to the file and the Save button will disappear and the profileActivity will switch to display mode. If any value entered is wrong the user will get a message Toast and nothing will be saved, and the activity will stay in its current mode.
- Valid data for profile information:
  - Name: only alphabetical characters.
  - Age: 18-99
  - Student ID: numerical values maximum 6 digits.
- The profile activity has an up navigation that goes to the main activity when pressed.



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:

In order to detect an invalid input, you can do that by reading 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, few of the inputs that you have can be managed in a way to prevent the user to input any invalid information.

For example, for the name you can add the following line to EditText in the xml file to make sure only alphabetical characters are allowed:

`android:digits="abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ"`

source:

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

For the Student ID, you can use a numerical edit text and specify android:maxLength in the xml file to specify the max length of 6. Or you can do it in the activity java class.

Example of 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 Name you don't need to check every character if it's an alphabet or not anymore, you just need to check if the field is empty or not. Etc...

One Last Hint:

To make an edit text editable:

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

To make an edit text un-editable

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

Generating random courses with random assignments and grades

The following two Java Classes will allow you to generate a Course with auto generated Title and auto generated random number of assignments (max 4) with a random grade out of 100.

A sample code below will show you how to use the Course class to generate the courses that you will be displaying. Every time the user navigates to the Grades Activity a different set of Courses will be randomly generated and displayed in a List View manner.

Assignment.Java

```
import java.util.Random;

/**
 * Created by Tawfiq on 1/13/2017.
 */
public class Assignment {

    private static int assID = 0;       //static ID increments with every new
assignment created
    private String assignmentTitle;            //title of assignment
    private int assignmentGrade;               //grade of assignment

    //private constructor. Increments ID.
    private Assignment(String title, int grade)
    {
        assignmentTitle = title;
        assignmentGrade = grade;
        assID++;
    }

    //returns an Assignment instance with random values
    static public Assignment generateRandomAssignment()
    {
        Random rnd = new Random();
        String tempTitle = "Assignment " + assID;
        int tempGrade = rnd.nextInt(100) + 1;

        return new Assignment(tempTitle, tempGrade);
    }

    //****get methods*****//
    public String getAssignmentTitle()   {return assignmentTitle; }
    public int getAssignmentGrade()   {return assignmentGrade;}
}
```

Course.Java

```
import java.util.ArrayList;
import java.util.Random;

/**
 * Created by Tawfiq on 1/13/2017.
 */
public class Course {
    private static int courseID = 0;            //static ID increments with every new
Course created
    private String courseTitle;                 //cou
    private ArrayList<Assignment> assignments;

    private Course(String title, ArrayList<Assignment> assns)
    {
        courseTitle = title;
        assignments = assns;
        courseID++;
    }

    //returns a Course instant with random assignment values
    static public Course generateRandomCourse()
    {
        Random rnd = new Random();
        int assignmentNo = rnd.nextInt(5);
        ArrayList<Assignment> tempAssns = new ArrayList<Assignment>();

        for(int i=0; i < assignmentNo; i++)
            tempAssns.add(Assignment.generateRandomAssignment());

        return new Course("Course " + courseID, tempAssns);
    }

    //****get methods*****//
    public String getCourseTitle()  {return courseTitle;}
    public ArrayList<Assignment> getAssignments()   {return assignments;}
}
```

Both classes are important to be created inside your Android project.

The following is a sample code main function that you can test the two classes in a Java console App (not Android App):

```
public class Driver {

    public static void main(String[] a)
    {
        for(int j=0; j<5; j++) {
            Course course = Course.generateRandomCourse();
            ArrayList<Assignment> assignments = course.getAssignments();
            System.out.println(course.getCourseTitle());
            for (int i = 0; i < assignments.size(); i++) {
                System.out.println(assignments.get(i).getAssignmentTitle()
                        + "    " + assignments.get(i).getAssignmentGrade());
            }
        }
```

```
    }
}
```

The output of this main is:

Course 0
Assignment 0 3
Course 1
Assignment 1 60
Assignment 2 36
Course 2
Assignment 3 37
Assignment 4 51
Assignment 5 46
Course 3
Course 4
Assignment 6 83
Assignment 7 24
Assignment 8 58

Your grades activity will have a member:

```
ArrayList<Course> courses;
```

When the activity is first created you will generate a random number of courses. The above example generates 5 courses at all time. Then you will display the courses in a ListView.

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.

```
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:

```
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) {


                goToProfileActivity();
```

```
            }
        });


    }

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

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

    }

    void goToProfileActivity()
    {
        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 a Profile and save a Profile.

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

Assignment submission and procedure

You have to 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** file named **StudentID_Ass1** containing your android project. **Before submitting your code make sure you clean the project.**

```
Android Studio --> Build --> Clean
```

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%

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.