420-5A6-AB Application Development II
Fall 2021

August 30th
Worth 8%

Assignment I

(Check on Moodle)

Design and implement an Android `Activity` and `Fragment` for a user to create a *task*. EThey will be able to modify a task's description, priority (low, medium or high), due date and status (pending, complete, etc…). A simple undo feature is included in the editor.

The aim of the fragment is to obtain a `Task` object that will be used by the rest of the app, completed in the following assignments.

The user interface is will look like this. Each part of the UI is outlined in the following sections.

1. Authoring a task description.
2. Add a priority and/or due date to a task.
3. The task's priority, either low, medium or high.
4. The task's due date.
5. Undo: remove each change since the start of editing.
6. Mark a completed task.

1 UI

The UI is demonstrated in the video alongside these instructions. Please watch it before continuing on. Below are the requirements and additional resources.

1.1 Description

Requirements. The description can occupy more than one line. When left empty, the description widget should contain an explanation of it's purpose in the form of a "hint".

Notes. Hints can be setup in the properties of an `EditText`: `https://developer.android.com/reference/android/widget/EditText`

1.2 Add Priority or Due Date.

Requirements. The two icons at the bottom right, `priority_high` and `add_alert`, add either a priority or a due date to the task. Each open their respective "panes" described below. For priority, the default is set to `HIGH`. For the due date, the process for choosing a date and time is initiated with a default value of the next day at 8:00am. After adding a priority or due date they will be disabled.

Notes.

- Android has icons available to use in your app. Right-click the `res` folder and add a new Vector Asset.
- Views have a visibility property that you can use to hide and show them programatically. See `https://developer.android.com/reference/android/view/View.html#setVisibility(int)` on how to use them.
- You can let Android animate the layout changes done at runtime. Just set the property `animateLayoutChanges` to `true` in the root layout of your fragment.

1.3 Priorities

Requirements. Each task can be given an (optional) priority: low, medium or high, represented by the labelled priority radio button. Clicking the `cancel` icon will clear the task's priority, close the pane (make it invisible) and re-enable the `priority_high` icon below.

Notes.

- Radio buttons and radio groups: `https://developer.android.com/guide/topics/ui/controls/radiobutton`

1.4 Reminders

Requirements. Each task may contain an due date and time set by the user. The date and time are chosen using predefined Android dialog windows. The chosen date is displayed in a user-friendly format, ex: Monday, August 30th at 8:00am. Clicking on the due date will allow the user to choose another due date using the dialogs. Clicking the `cancel` icon will clear the task's due date, close the pane (make it invisible) and re-enable the `add_alert` icon below.

The first time a due date is chosen, the UI dialogs should default to "tomorrow at 8:00am". In subsequent modifications, the previously chosen date and time will be used as the default value of the dialogs. Cancelling either dialog leaves the reminder date unchanged. Choosing a date in the past will display an Alert dialog stating "Invalid due date, Please select a date in the future."

Notes. Use the provided `DatePickerDialogFragment` and `TimePickerDialogFragment`. See the tutorial video in the course notes on how to use them.

1.5 Undo

Requirements. Each edit can be undone by pressing the undo icon. This includes text changes to the description, priority and due date. When undoing the addition/removal of priority or due date to the task, the visibility of the appropriate UI components should change.

Notes.

- Hint: store a sequence of `Task` objects (see the section below on the Model classes). Use a data structure that will supports operations that behave like a "history" of edits.
- Be careful: the `EditText`'s method `setText( .. )` will cause events. You might have to remove and re-introduce an event handler if you want to call this method while listening for edits.

2 Model

Requirements. Use the provided model classes to store task authored in the fragment. Specifically, when needed, the fragment must produce a `Task` object with appropriate fields set.

Not all fields of the `Task` are used. Some might be used in subsequent assignments.

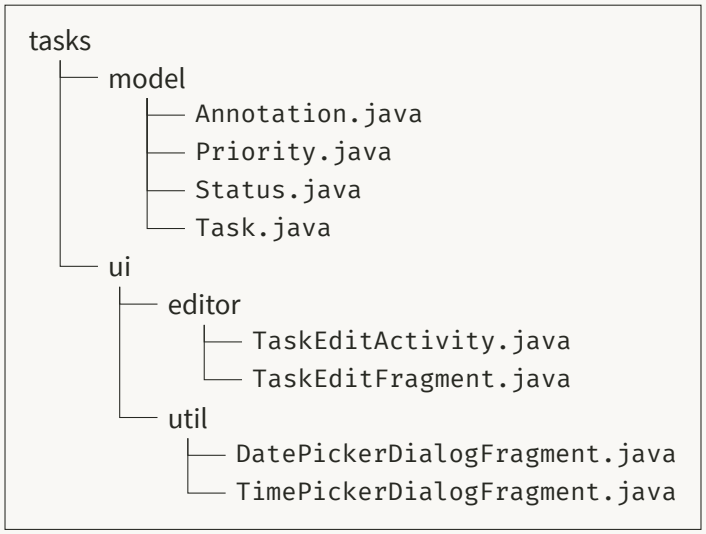
Be sure that you set the entry and modified timestamps correctly. The entry timestamp should only be set when a note is first edited. The modified timestamp should be set whenever any field is modified.

Notes.

- The `.copy()` method creates an identical copy of the `Task`. This should be useful when implementing the undo feature.

3 Project Structure

Use the following directory structure for your project's source code, separating the app into model, ui, activities and utilities:



Use refactoring to rename the original class names made when creating the project.

4 Other Helpful Resources

There are many tutorials online, but `vogella.com` has many including this nice introduction: `http://www.vogella.com/tutorials/Android/article.html`.

The Android API reference is a really important resource. Use it to look up specific classes and methods to understand how they work: `http://developer.android.com/reference/packages.html`

5 Requirements

- Your program should be clear and well commented. It must follow the "420-5A6 Style Guidelines" (on GitHub).
- Create an Android project with minimum SDK 26 - Android 8.0 (Oreo) or later.
- Your main activity uses a fragment.
- Use at least one `ConstraintLayout` and at least one `LinearLayout` in your fragment layout.
- The UI meets all the requirements above.
- The fragment produces a `Task` object with all fields set correctly.
- Submit using Git by following the instructions (on GitHub).