420-616-AB Oracle Technologies
Fall 2019

September 3rd
Worth 10%

Assignment I

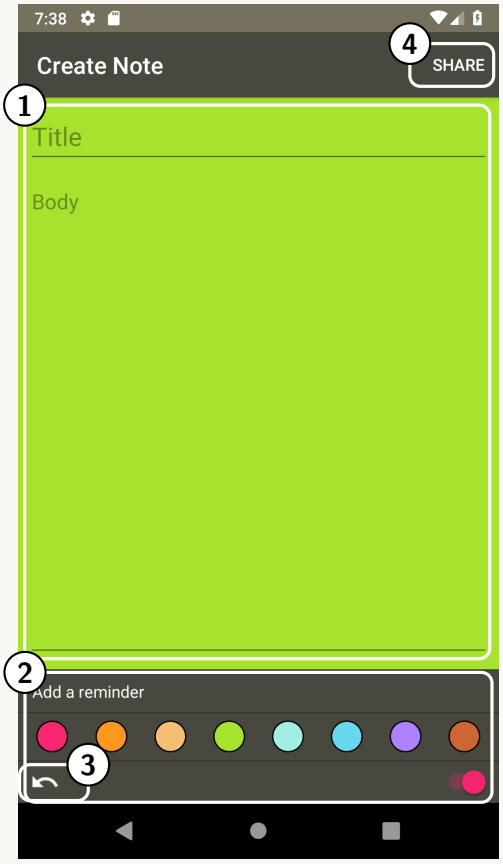
(Due on Check on Léa)

Design and implement an Android `Activity` and `Fragment` for a user to create a note. Each note will contain a title, body and some properties. A simple undo feature is included in the editor, as well as a simple export (share).

The aim of the fragment is to obtain a `Note` 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 note title and body.
2. The note's properties: a reminder date/time and the note category defined by a color.
3. Undo: remove each note change since the start of editing.
4. Share the note with other apps.



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 Title and Body

Requirements. The title should be on a single line, but the body will occupy more than one line. When left empty, the title and body should contain an explanation of their 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.html`

1.2 Properties

Requirements. A `Switch` at the bottom right corner of the fragment will show and hide the note's *properties* using the default Android animation.

Notes.

- 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 Categories

Requirements. Each note can be categorized by the user. The available categories are provided in bar of clickable circles at the bottom of the fragment. Selecting a category changes the background color behind the title and body of the note.

Notes.

- Use the provided `CircleView` class. See the tutorial video in the course notes on how to use it.

- Use named color resources in your implementation. See the tutorial video in the course notes on how to use it.
- The color scheme I used in my app is "base16 monokai", which has the following values:

Name	Code
base00	#272822
base01	#383830
base02	#49483E
base03	#75715E
base04	#A59F85
base05	#F8F8F2
base06	#F5F4F1
base07	#F9F8F5
base08	#F92672
base09	#FD971F
base0A	#F4BF75
base0B	#A6E22E
base0C	#A1EFE4
base0D	#66D9EF
base0E	#AE81FF
base0F	#CC6633

1.4 Reminders

Requirements. Each note may contain an reminder set by the user. The date and time are chosen using predefined Android dialog windows. The first time a date is chosen, the initial date in the dialogs is the next day at 8:00am. Subsequently, the previous date is displayed in the dialogs. Cancelling either dialog leaves the reminder date unchanged.

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 title and body, a change in the reminder and a change in the category.

Notes.

- Hint: store a sequence of `Note` 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.
- I used the Material Design "undo" icon in the sample app. You can find it here: `https://material.io/tools/icons/`.

2 Share

Requirements. Add a menu item called "Share" that will allow to user to send the text representation to other apps. Currently, only a text representation is shared.

Notes.

- See the tutorial video in the course notes on how to add a menu item.
- See the tutorial video in the course notes on how to access a fragment from an activity.
- Use the following code to share text data with other apps:

```
String data = ...;
Intent sendIntent = new Intent();
sendIntent.setAction(Intent.ACTION_SEND);
sendIntent.putExtra(Intent.EXTRA_TEXT, data);
sendIntent.setType("text/plain");
startActivity(sendIntent);
```

we will see "intents" in more detail later in the course.

3 Model

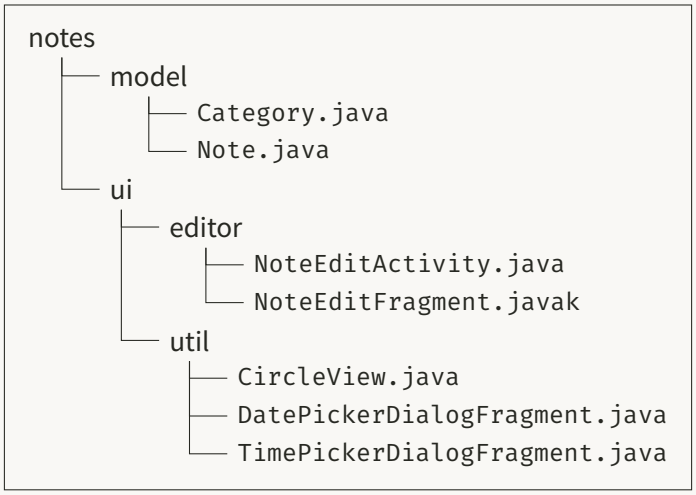
Requirements. Use the provided class `Note` to store the notes. Specifically, when needed, the fragment must produce a `Note` object with appropriate fields set.

Notes.

- You should ignore the ID field for this assignment.
- Make sure that you set the created and modified timestamps correctly. The created timestamp should only be set when a note is first edited. The modified timestamp should be set whenever any field is modified.
- The `.clone()` method creates an identical copy of the `Note`. This should be useful when implementing the undo feature.

4 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.

5 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`

6 Requirements

- Your program should be clear and well commented. It must follow the “420-616 Style Guidelines” (on Léa).
- Create an Android project with minimum SDK 23 - Android 6.0 (Marshmallow) 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 `Note` object with all fields set correctly.
- Submit using Git by following the instructions (on Léa).