Assignment I

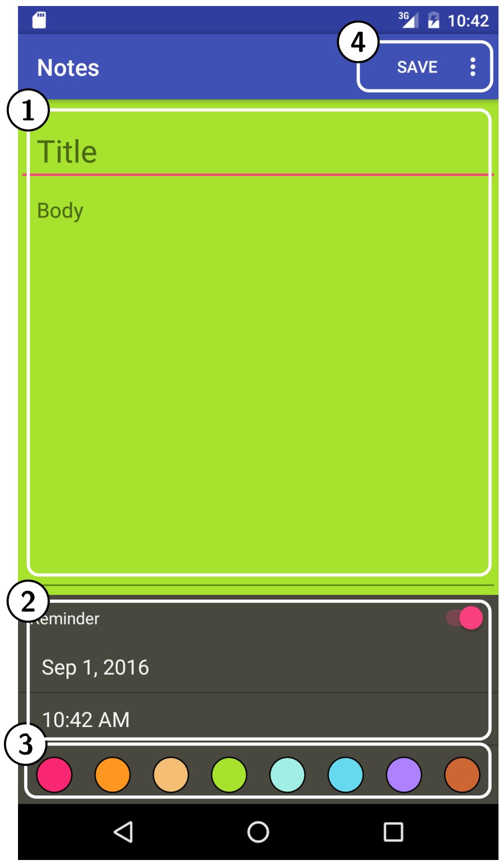
(Due date is on Léa)

Design and implement an Android `Fragment` for a user to create a note. Each note will contain a title and body, a category (indicated by a color) and an optional reminder set to a specific date and time. The goal of the fragment is to produce a `Note` object that will be used in the next Assignments.

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

The user will interact with a single fragment inside a single activity. It will include UI components for:

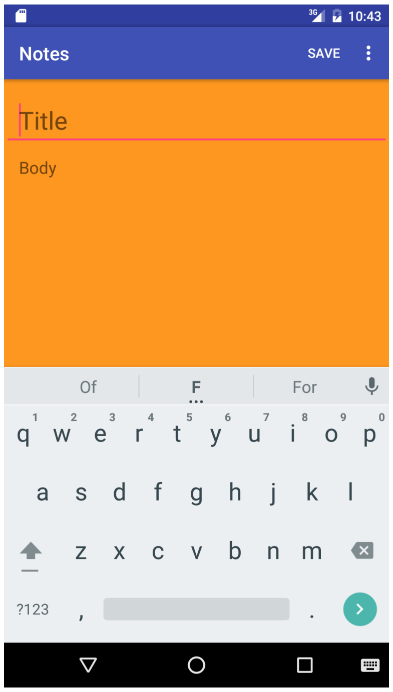
1. Authoring a note title and body.
2. Turning on a reminder, then setting the date and time.
3. Setting the category of the note as one of 8 colors.
4. Saving the note.



1 Title and Body

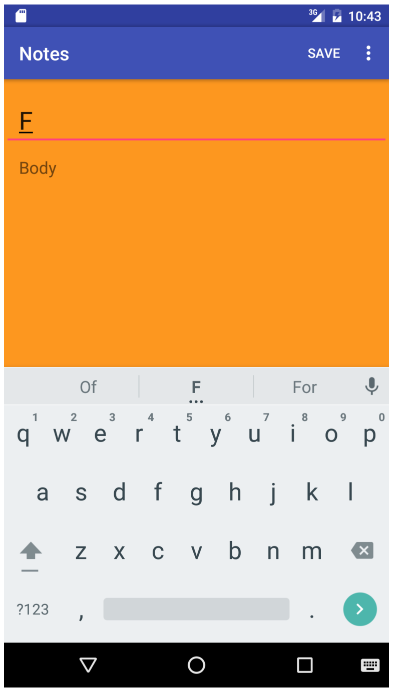
The title and body of the note are authored in the top part of the fragment. The title should be on a single line, but the body can occupy more than one line.

Hints. When left empty, the title and body should contain an explanation of their purpose in the form of a “hint”. Hints can be setup in the properties of an `EditText`.



Title and body UI.

$\Longleftrightarrow$



Adding text removes hint.

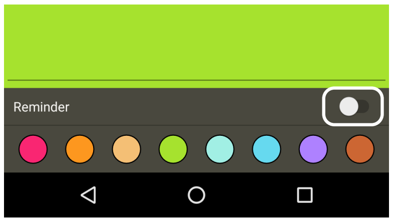
`https://developer.android.com/reference/android/widget/EditText.html`

2 Reminders

A note contains an optional reminder that be turned on/off with a `Switch`. When turned on, the date and time UI components are visible and vice versa. When they are visible, the date and time can be modified by clicking. This will bring up date-picker or time-picker dialog, respectively.

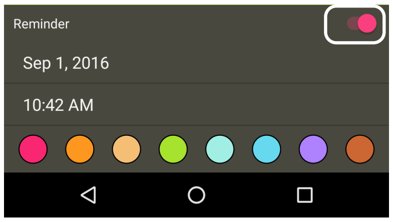
Visibility Each `View` has a visibility setting that can be one of three values: `VISIBLE`, `INVISIBLE`, or `GONE`.

`https://developer.android.com/reference/android/view/View.html#setVisibility(int)`



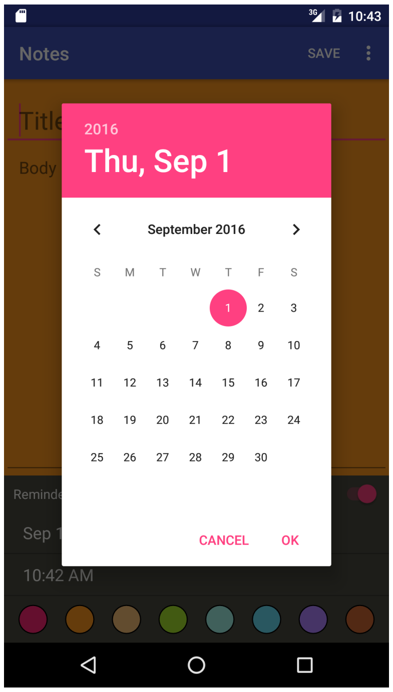
No reminder.

⟺

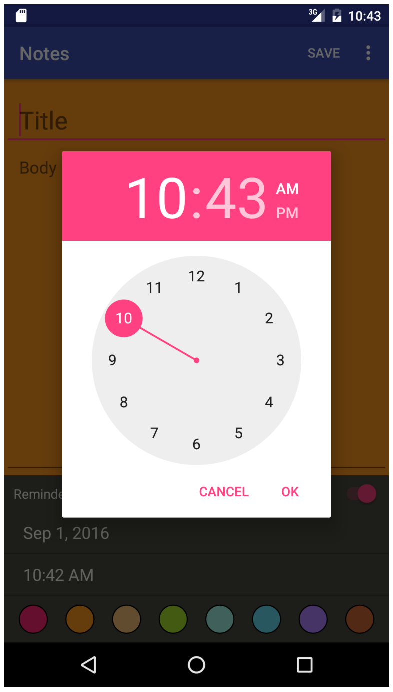


Reminder.

Dialogs: Time and Date Pickers The user can change the reminder date and time values by clicking on them. When they do a *dialog* is displayed with the previous date/time allowing them to change it. Initially the date/time should be set to a reasonable default, like the next day at 8:00am.

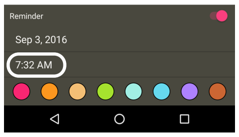


Date change dialog.



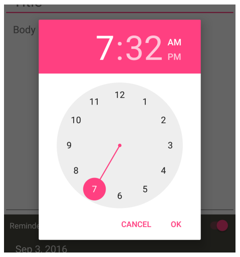
Time change dialog.

The follow steps change a time value (similar for date):



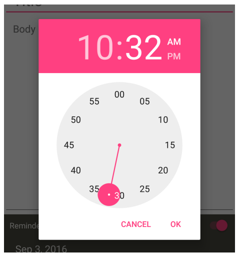
Select time.

⇒



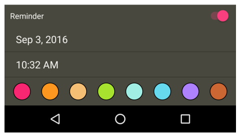
Start picker.

⇒



Choose new time.

⇒



New time.

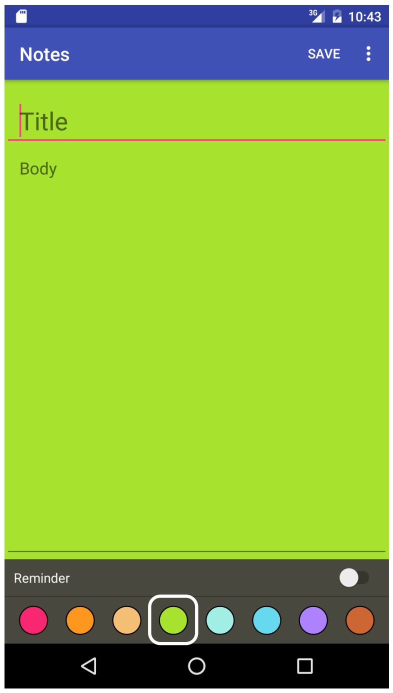
Dialogs are implemented in a class. For this Assignment, use the provided `DatePickerDialogFragment` and `TimePickerDialogFragment`. Check their documentation for usage, but here is code snippet that creates a time-picker dialog with a do-nothing event handler.

```
Date initial = new Date();   // TODO: use previous value or "tomorrow at 8:00"

// create and show the TimePicker with starting time
DialogFragment dialogFragment = TimePickerDialogFragment.create(
        initial,
        new TimePickerDialog.OnTimeSetListener() {
            @Override
            public void onTimeSet(TimePicker view, int hourOfDay, int minute) {
                // TODO
            }
        });
dialogFragment.show(getFragmentManager(), "timePicker");
```

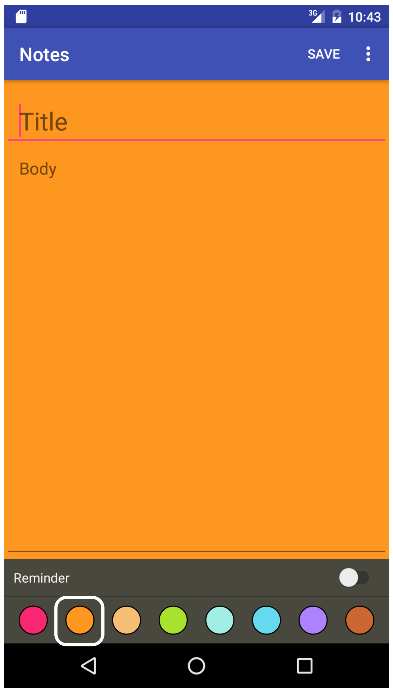
3 Categories

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.



Note in Green category.

$\Longleftrightarrow$



Note in Orange category.

Custom View: CircleView Each category circle can be constructed using the provided `CircleView` class. This class is an example of a customized `View`. You use a custom view in a similar way as normal views: they can be placed in your layout, they have IDs, you create references in your code to interact with them, etc…

After adding the file `CircleView.java` to your project, you can place a `CircleView` into your layout by first choosing "CustomView" from the widget palette (bottom of the list) and then choosing the `CircleView`.

`https://developer.android.com/guide/topics/ui/custom-components.html`

Color Resources In Android colors are represented as `ints`. You can generate Android colors from RGB components using the `Color.argb()` method.

```
int color = Color.argb(255, 0, 128, 128);
```

Instead of hardcoding the color values in this assignment, specify color values in the file `colors.xml`, located in the `res/values/` directory. For example, the colors scheme used in these instructions are given names:

```
<color name="base00">#272822</color>
<color name="base01">#383830</color>
<color name="base02">#49483E</color>
<color name="base03">#75715E</color>
<color name="base04">#A59F85</color>
<color name="base05">#F8F8F2</color>
<color name="base06">#F5F4F1</color>
<color name="base07">#F9F8F5</color>
<color name="base08">#F92672</color>
<color name="base09">#FD971F</color>
<color name="base0A">#F4BF75</color>
<color name="base0B">#A6E22E</color>
<color name="base0C">#A1EFE4</color>
<color name="base0D">#66D9EF</color>
<color name="base0E">#AE81FF</color>
<color name="base0F">#CC6633</color>
```

and are taken from the "base16 monokai" color theme (`https://github.com/chriskempson/base16`).

To use a color resource in your program, retrieve the color from the app's resources object:

```
int color = getResources().getColor(R.color.base02)
```

`https://developer.android.com/reference/android/graphics/Color.html` `https://developer.android.com/guide/topics/resources/more-resources.html#Color`

4 Save Menu Item

In this assignment, the “Save” menu item is a placeholder for future development. When the user saves the note, create an instance of the class `Note`, populate it with the data from the UI and print its `String` representation to the Android log using the method `Log.d()`. You should be able to verify the log using `logcat` in Android Studio.

```
public class Note {

    private String title;
    private String body;
    private int category;
    private boolean hasReminder;
    private Date reminder;

    // + constructor, setters/getters and toString() (use Code - Generate)
}
```

`https://developer.android.com/guide/topics/ui/menus.html`

5 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 21 or later.
- Your main activity uses a fragment.
- Design your UI using the simplest layout possible (ex: `LinearLayout` or `GridLayout` instead of `RelativeLayout` whenever possible).
- The “Save” menu creates a `Note` object and outputs its `String` representation to the Android log-file.
- Submit your Android Studio project folder on Léa.