

The background is a light cream color, decorated with various abstract and organic shapes. In the top left, there's a dark teal square, an orange circle, and a small orange triangle. The top right features large, flowing organic shapes in orange and light peach. The bottom left has a brown leafy branch, a wavy light peach line, and three small circles (two orange, one dark teal). The bottom center shows a stylized rainbow with orange, yellow, and dark teal bands. The bottom right includes another brown leafy branch, a cluster of small orange diamonds, and a dark teal rectangle.

# Reminder App

Jillian Atkins

# New Concepts

## Alarms

- Set Ringtone
- Set Date and Time

## Notifications

- Notification Manager
- Build a Custom Notification

## Dialogs

- Custom Popup Dialogs with Buttons

## FAB

- F**loating **A**ction **B**utton
- Button that “Floats” above other content

## Existing Knowledge

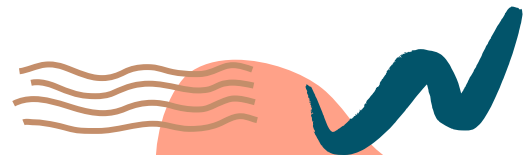
- Cursor Adapter -SQLite Database -ListView
- OnClickListener -Custom Classes

# Implement a Floating Action Button



- Add dependencies to the build.gradle file (implementation 'com.google.android.material:material:<version>')
- MainActivity extends AppCompatActivity
- Change Main Activity layout file to Coordinator Layout
- Add different styling
- Instantiate it in MainActivity.java (same way you would a regular button)

<https://material.io/develop/android/components/floating-action-button>



# Implement a Notification Manager

- This is a helper class to build and set the reminder notification that appears - it extends `BroadcastReceiver` (which is sort of like a “messaging” or “alert” system).
- This is also where the notification sound is set.



# Build a Custom Class for Reminder Objects

This concept is the same as the Chatter.java class, and the ListTitle/ListItem classes for Lab02.



## Build a Database Manager

DBManager is based on the SQLite lessons/examples we were taught.



# Implement a Pop-Up Dialog

- Create a new Layout Resource file (ReminderApp has 2 different dialogs, `floating_popup` and `floating_edit_popup`). These layout files are similar to the ones for an ordinary screen layout.
- Create a class-level Dialog variable in MainActivity.
- I put each of the dialogs in separate methods that run when a user clicks a button or a ListView item. These methods are called `addReminder()` and `editReminder()`.
- After instantiating the dialog variable, you need to set it to a view (one of the layout files that were created for the dialog).
- Important things to remember when implementing dialogs are `dismiss()` and `show()`. If I have a dialog called *dialog*, `dialog.show();` will display the dialog popup and `dialog.dismiss();` will close the dialog after an action has been completed.

# Implement an Alarm

- Much like the date & time demo, ReminderApp uses a date and time picker dialog to assign a date and time to variables and passes those to an Alarm Manager
- The Alarm Manager has many pre-defined constants and methods, ReminderApp uses `RTC_WAKEUP`, which will wake up the device when the alarm goes off.