

Lab 7: Custom Notifications

Due Nov 15 by 3:45pm **Points** 10 **Submitting** a text entry box or a file upload
Available Nov 5 at 12am - Nov 30 at 11:59pm 26 days

This assignment was locked Nov 30 at 11:59pm.

LAB 7: Custom Notifications

Introduction:

In this lab we will learn to create custom notifications in Android Studio. A notification is a message that Android displays outside your app's UI to provide the user with reminders, communication from other people, or other timely information from your app. Users can tap the notification to open your app or take an action directly from the notification. To see the final product of the lab, look in the 'Deliverables' section.

Note: This lab is designed for devices with Android Build Version 26 or higher. The application may not work as intended on devices with lower Android versions.

Background:

Appearances on Devices:

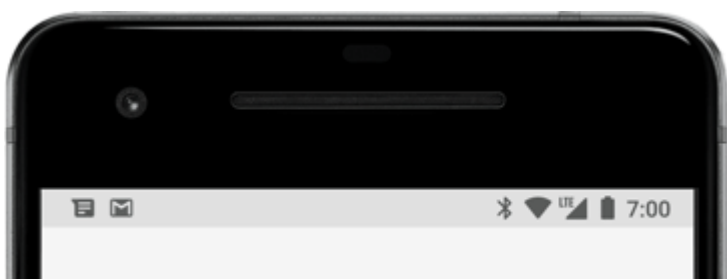
Notifications appear to users in different locations and formats, such as:

- An icon in the status bar
- A more detailed entry in the notification drawer
- A badge on the app's icon
- On paired wearable devices

The type of notifications we will be looking at in this lab include the status bar and notification drawer.

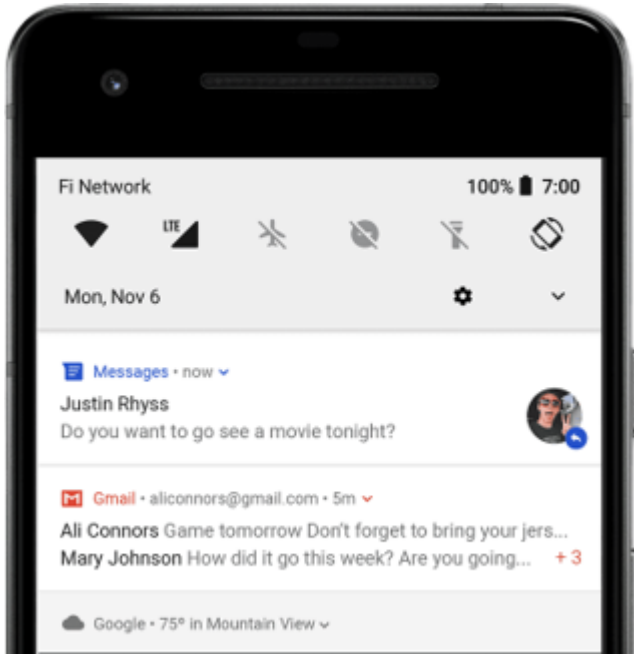
Status Bar and Notification Drawer:

- When you issue a notification, it first appears as an icon in the status bar as shown below.
- Notification icons appear on the left side of the status bar.
- Users can swipe down on the status bar to open the notification drawer, where they can view more details and take actions with the notification



Notifications in the notification drawer:

- Here, users can drag down on a notification in the drawer to reveal the expanded view, which shows additional content and action buttons, if provided.
- A notification remains visible in the notification drawer until dismissed by the app or user.



Milestone 1

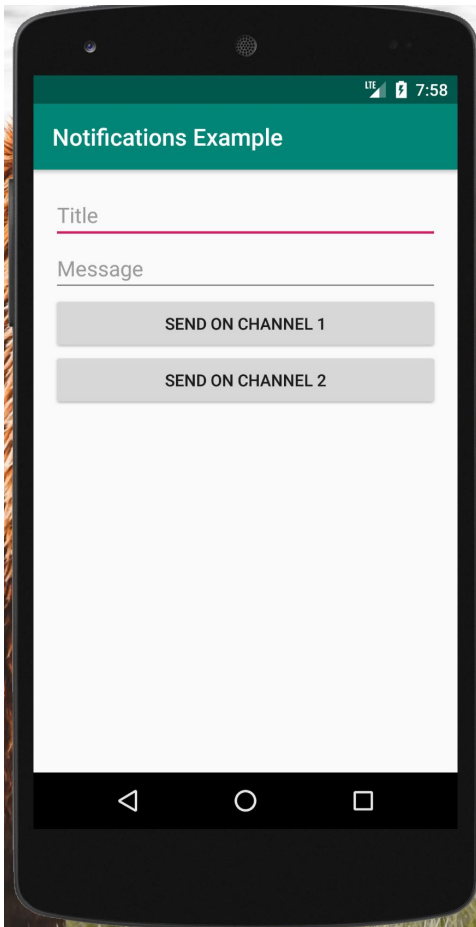
Background

In this milestone, you will learn how to create a basic notification. To do this you will start off by creating a notification channel. Starting in Android 8.0 (API level 26), all notifications must be assigned to a channel. For each channel, you can set the visual and auditory behavior that is applied to all notifications in the channel. Then, users can change these settings and decide which notification channels from your app should be intrusive or visible at all. After you create a notification channel, you cannot change the notification behaviors - the user has complete control at that point. Though you can still change a channel's name and description. Best practice is to create a channel for each distinct type of notification you need to send. You can also create notification channels to reflect choices made by users of your app. For example, you can set up separate notification channels for each conversation group created by a user in a messaging app.

In this milestone we will create two different notification channels and send a notification in each of these channels with different priority levels.

Setup

To start, create a new project and create a UI resembling the picture bellow:



Channels:

Now, it is time to create our channels! Although we could create channels in an adhoc fashion whenever we needed them, it is far better practice to make them one time, when the application starts. To do this we can create an application class under *app > java > [your package name]* called *App*.

In this class, we can add an `OnCreate` function. This `OnCreate` function calls another function, defined by us, called `CreateNotificationChannels()`. The `App.java` class and the code to create the first notification channel are bellow:

```

public class App extends Application {
    public static final String CHANNEL_1_ID = "channel1";

    @Override
    public void onCreate() {
        super.onCreate();

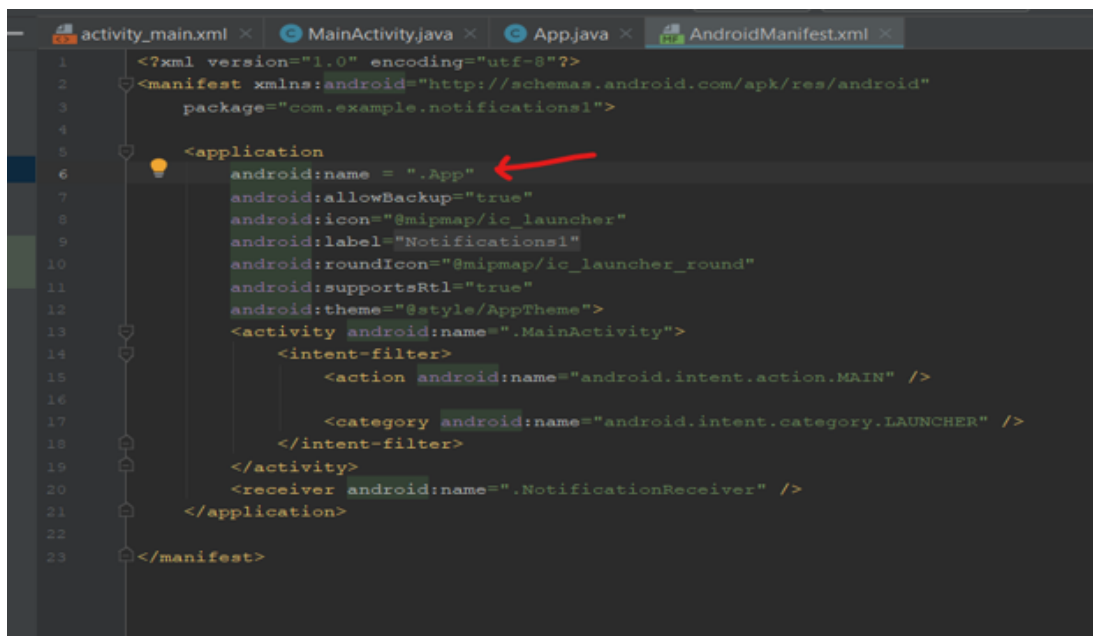
        createNotificationChannels();
    }

    private void createNotificationChannels() {
        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O) {
            NotificationChannel channel1 = new NotificationChannel(
                CHANNEL_1_ID,
                name: "Channel 1",
                NotificationManager.IMPORTANCE_HIGH
            );
            channel1.setDescription("This is Channel 1");

            NotificationManager manager = getSystemService(NotificationManager.class);
            manager.createNotificationChannel(channel1);
        }
    }
}

```

Note, as with Activity Classes, we will have to make sure to update our Manifest to include this new class:



```

<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.notifications1">

    <application
        android:name=".App"
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="Notifications1"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportRtl="true"
        android:theme="@style/AppTheme">
        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <receiver android:name=".NotificationReceiver" />
    </application>

</manifest>

```

Notification #1

Now that we have seen how to create a notification channel, we can now explore how to create the notifications themselves in the MainActivity Class.

We want our notifications to display the text from our two EditText fields on our UI. These fields will be used as inputs where you can enter the title of the notification and the message that the notification will display. As such, we will need to create references to our EditText fields to access their contents.

We also need to create a reference for a NotificationManagerCompat object, which will help us display our notifications, see the below code:

```

public class MainActivity extends AppCompatActivity {

    private NotificationManagerCompat notificationManager;
    private EditText editTextTitle;
    private EditText editTextMessage;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        notificationManager = NotificationManagerCompat.from(this);
    }
}

```

To display the notification, we need a `sendOnChannel` function, which will be executed when the corresponding button is pressed. To make the notifications we use the following template:

```

NotificationCompat.Builder builder = new NotificationCompat.Builder(this, CHANNEL_ID)
    .setSmallIcon(R.drawable.notification_icon)
    .setContentTitle(textTitle)
    .setContentText(textContent)
    .setPriority(NotificationCompat.PRIORITY_DEFAULT);

```

We can add more options and tweak them as shown in the code below for Channel 1:

```

public void sendOnChannel1(View v) {
    String title = editTextTitle.getText().toString();
    String message = editTextMessage.getText().toString();

    Notification notification = new NotificationCompat.Builder(context, CHANNEL_1_ID)
        .setSmallIcon(R.drawable.ic_chat_black_24dp)
        .setContentTitle(title)
        .setContentText(message)
        .setPriority(NotificationCompat.PRIORITY_HIGH)
        .setCategory(NotificationCompat.CATEGORY_MESSAGE)
        .build();

    notificationManager.notify(1, notification);
}

```

To choose custom icons for each notification you can: right click on drawable > new > vector asset > choose an icon > click finish. In this code we choose a messaging icon, but you can choose whatever icon you prefer.

As you can see in the code, once you have created the notification, all you have to do is use the notification manager to display the notification. When you do this, you will also have to pass the corresponding channel id along with the notification i.e. (`notificationManager.notify(CHANNEL_ID, NOTIFICATION_OBJECT)`);

Notification #2

Awesome job, you just learned how to create a new notification - now it is your turn! We want you to create a second notification based on the code we gave you for the first notification. To do this you will need to create a new notification channel and a new `sendOnChannel` method specific to this notification. To see what the second notification should look like, reference the video under the deliverables section at the end of the lab. The second notification will be very similar to the first, with a only a few differences:

1. We want our second notification channel to be created with the `IMPORTANCE_LOW` instead of `IMPORTANCE_HIGH` designation.
2. We want our second notification to have a `PRIORITY_LOW` instead of `PRIORITY_HIGH` designation.
3. We want our second notification to have a different icon than the first notification.

Note the differences between the notifications caused by the changes we made here!

Milestone 2

From Milestone 1, you may have noticed that tapping on our notifications currently does not do anything. In milestone 2, we will learn how to customize your notification and add a button on the notification to make a toast.

Once the user taps the notification, we want to go to our app, which is the `MainActivity`. Recall that pending Intents can help us accomplish this task. So let's create a `PendingIntent` that takes us back to the `MainActivity` when we tap on the notification:

```

35 public void sendOnChannel1(View v) {
36     String title = editTextTitle.getText().toString();
37     String message = editTextMessage.getText().toString();
38
39     Intent activityIntent = new Intent( packageContext, MainActivity.class);
40     PendingIntent contentIntent = PendingIntent.getActivity( context, this,
41         requestCode, activityIntent, flags);
42
43
44     Notification notification = new NotificationCompat.Builder( context, CHANNEL_1_ID)
45         .setSmallIcon(R.drawable.ic_chat_black_24dp)
46         .setContentTitle(title)
47         .setContentText(message)
48         .setPriority(NotificationCompat.PRIORITY_HIGH)
49         .setCategory(NotificationCompat.CATEGORY_MESSAGE)
50         .setColor(Color.BLUE)

```

One way to customize the notification is by setting up a button on the notification and assigning an action to it. To do this, let us set up a broadcast receiver. We will use this broadcast receiver to make a Toast. So, we pass this broadcast receiver to a `PendingIntent` to display the Toast message.

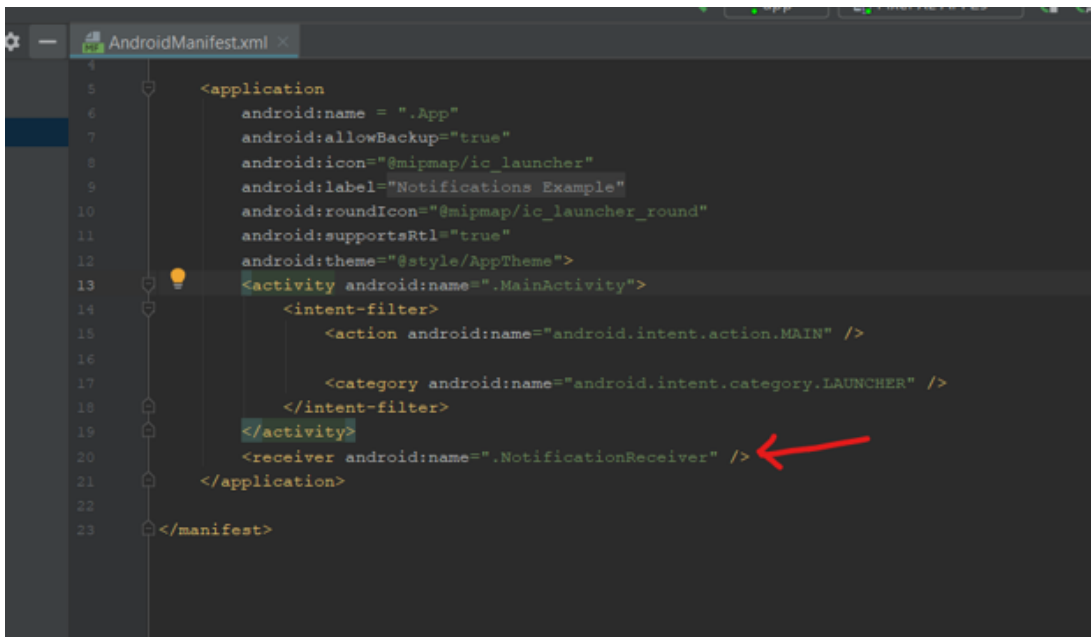
To set up a broadcast receiver let us create another class in our package. Let us call this class *NotificationReceiver* which extends *BroadcastReceiver*. The content of this class are shown below:

```

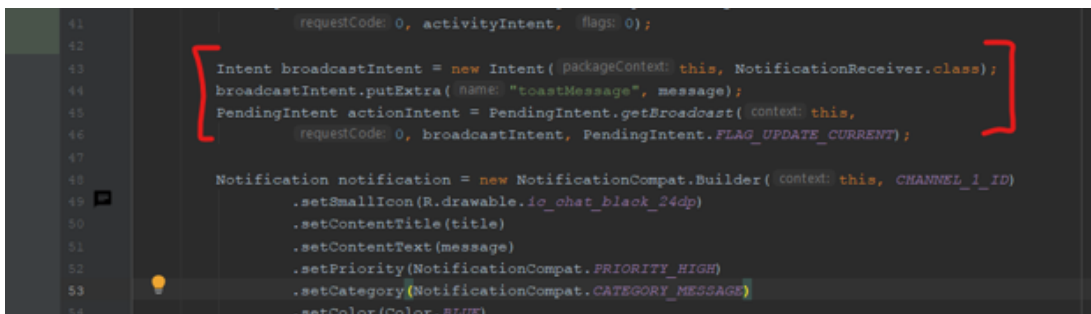
4 import android.content.BroadcastReceiver;
5 import android.content.Context;
6 import android.content.Intent;
7 import android.widget.Toast;
8
9
10 public class NotificationReceiver extends BroadcastReceiver {
11
12     @Override
13     public void onReceive(Context context, Intent intent) {
14         String message = intent.getStringExtra( name: "toastMessage");
15         Toast.makeText(context, message, Toast.LENGTH_SHORT).show();
16     }
17 }

```

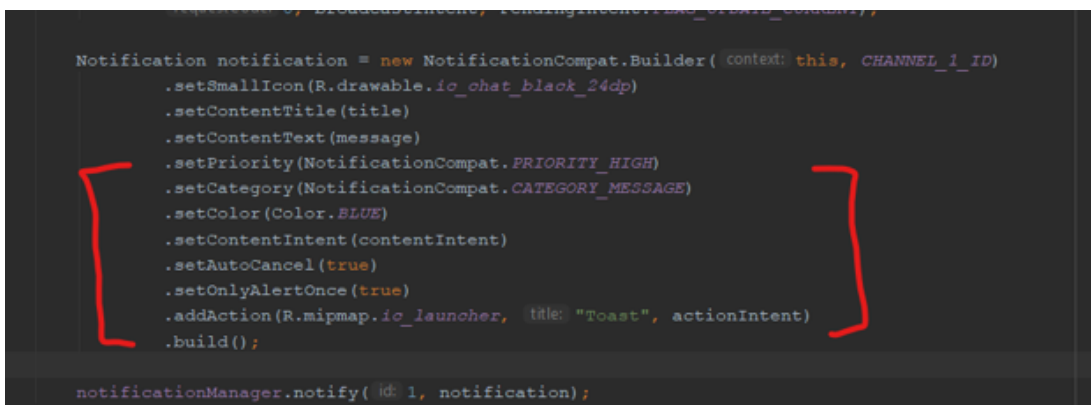
We also have to add this receiver in the android manifest file as shown below:



Now, let us create another pending intent to call this broadcast receiver in the MainActivity.



Once we have completed all these steps all we have to do is update the notification to reflect these changes by calling the appropriate methods. It will look something like this:



Once you complete these steps you will notice that on tapping the button a toast appears. If you tap the notification then it will open the app. You can also notice that we changed the color of the notification to blue. Here we used the following code to make the necessary changes:

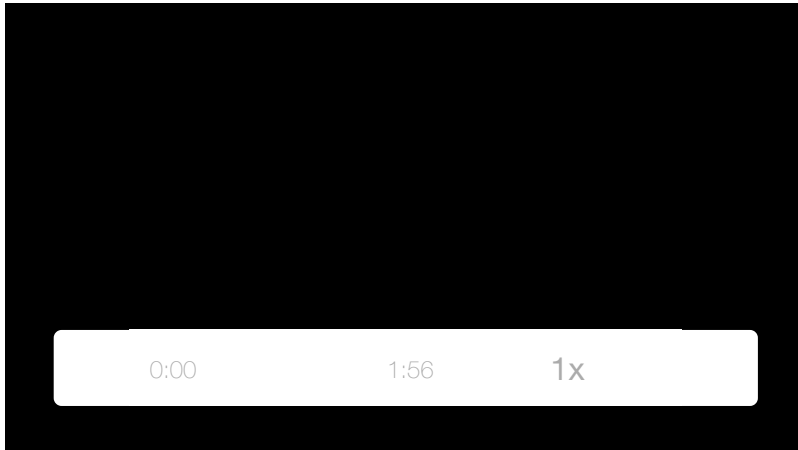

```
Notification notification = new NotificationCompat.Builder(this, CHANNEL_1_ID)
    .setSmallIcon(R.drawable.ic_chat_black_24dp)
    .setContentTitle(title)
    .setContentText(message)
    .setPriority(NotificationCompat.PRIORITY_HIGH)
    .setCategory(NotificationCompat.CATEGORY_MESSAGE)
    .setColor(Color.BLUE) This line sets the notification color to blue
    .setContentIntent(contentIntent) This will assign an action when the notification has been clicked
    .setAutoCancel(true) This dismisses the message after it has been clicked
    .setOnlyAlertOnce(true) This will make the notification alert the device only once
    .addAction(R.mipmap.ic_launcher, "Toast", actionIntent) This will assign an action to the button in the
notification
    .build();
```

Conclusion:

Great job getting through this lab! Now you can integrate your knowledge of notifications in your application to notify users of your app. You can play around with the different functions provided by the notification class to customize and personalize your notifications to match your applications needs.

Deliverables:

- Please push your code for both Milestones to your personal Github.
- Get your Lab checked off either via Video Submission (see bellow) or In person/virtually during office hours.
- Your App should function as shown in the below video (for better quality, simply select gear on bottom right of video and increase the streaming rate) :



Video Option for Lab Checkoff:

In addition to the in-person (during class) and virtual (TA office hours) check-off options, we have added a video option for lab checkoff. The video option is for you to do a screen recording of you showing your functioning app with a voice over explaining what you are doing. These videos will need to be submitted through Canvas.

For an example of such a video, please see the above Deliverables section.

Note:

1. Please do the screen recording as a single video and definitely with voice over. It should not be multiple videos that are edited and stitched together.

2. We recommend using Zoom's screen recording feature to capture the video, but you can do so by other means as well.

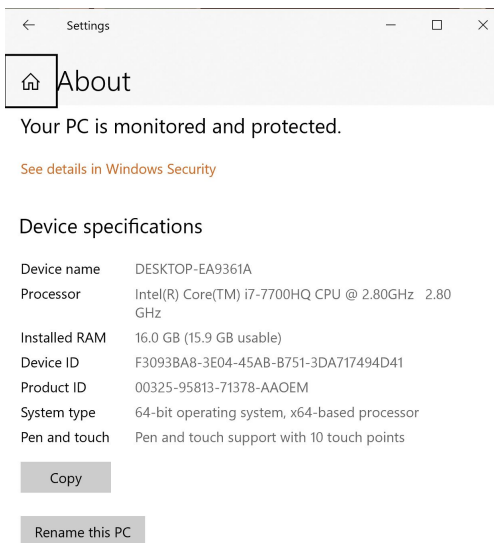
3. Please include a link your Github repos in the provided text box when you submit

4. Device identification:

For mac, go to the Apple icon in the top left, and click "About this Mac".



For windows, simply search 'About your PC' in the bottom left search bar:



Some Rubric (1)

Criteria	Ratings		Pts
Milestone 1 Student's Milestone 1 functions as per the assignment specifications (see video in deliverables for an example)	5 pts Full Marks	0 pts No Marks	5 pts
Milestone 2 Student's Milestone 2 functions as per the assignment specifications (see video in deliverables for an example)	5 pts Full Marks	0 pts No Marks	5 pts
Total Points: 10			