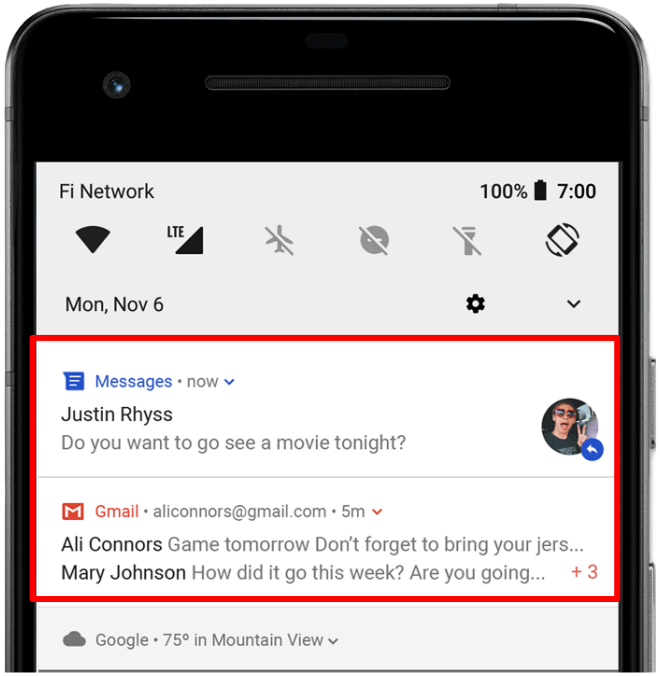
SSE3052: Embedded Systems Practice

Jinkyu jeong
jinkyu@skku.edu
Computer Systems Laboratory
Sungkyunkwan University
http://csl.skku.edu

Notification

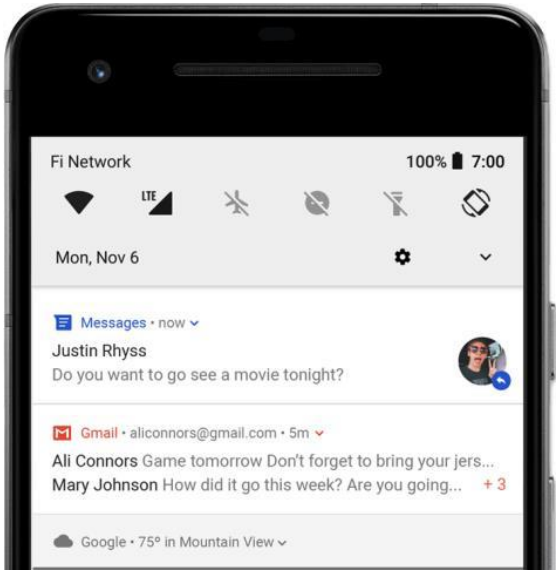
- A message that android displays outside your app's UI to provide the user with reminders, communication from other people, or other timely information from your app



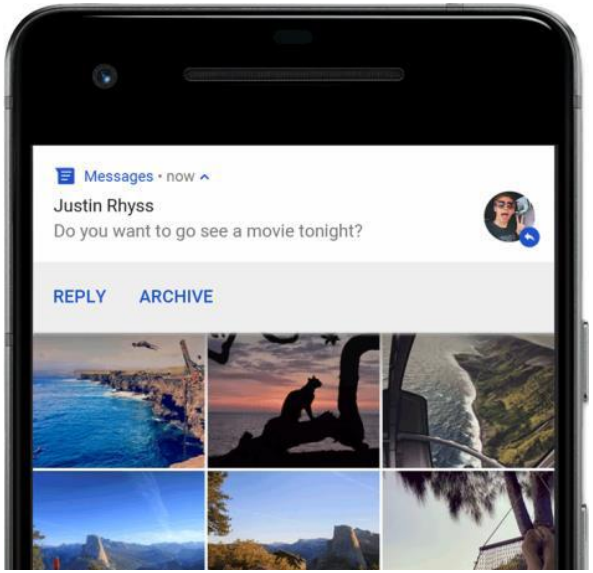
Appearances on a Device

- Notifications appear to users in different locations and formats
  - Status bar, notification drawer, heads-up notification, lock screen, app icon badge

Status bar

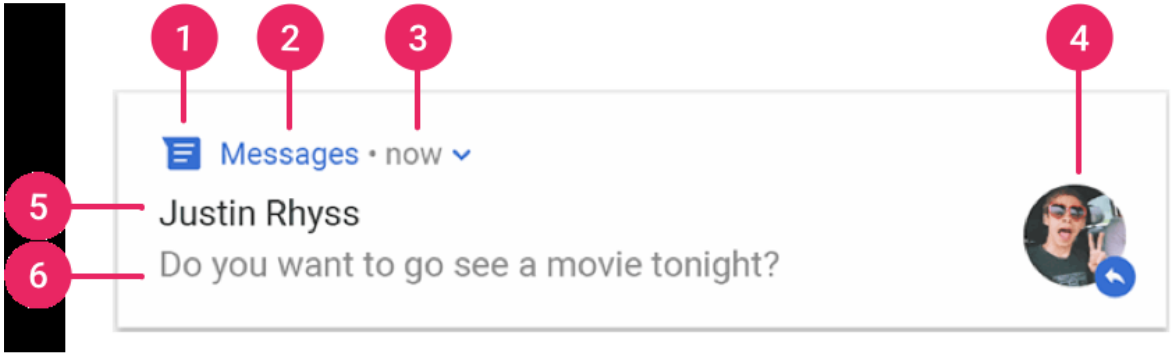


Notification drawer



Heads-up notification

Notification Anatomy



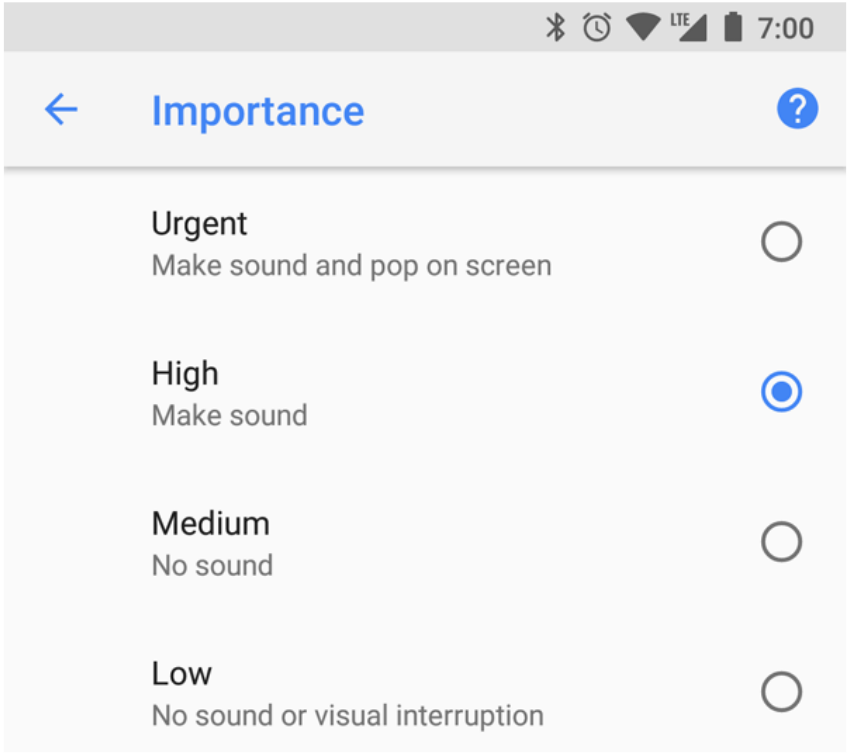
1. Small icon: This is required and set with setSmallIcon()
2. App name: This is provided by the system
3. Time stamp: This is provided by the system but you can override with setWhen() or hide it with setShowWhen(false)
4. Large icon: This is optional (usually used only for contact photos; do not use it for your app icon) and set with setLargeIcon()
5. Title: This is optional and set with setContentTitle()
6. Text: This is optional and set with setContentText()

Notification Channels

- Starting in Android 8.0 (API level 26), all notifications must be assigned to channel or it will not appear
- By categorizing notifications into channels for your app, and users can control the visual and auditory options for each channel

Importance of Notification

- Android uses the importance of a notification to determine how much the notification should interrupt the user

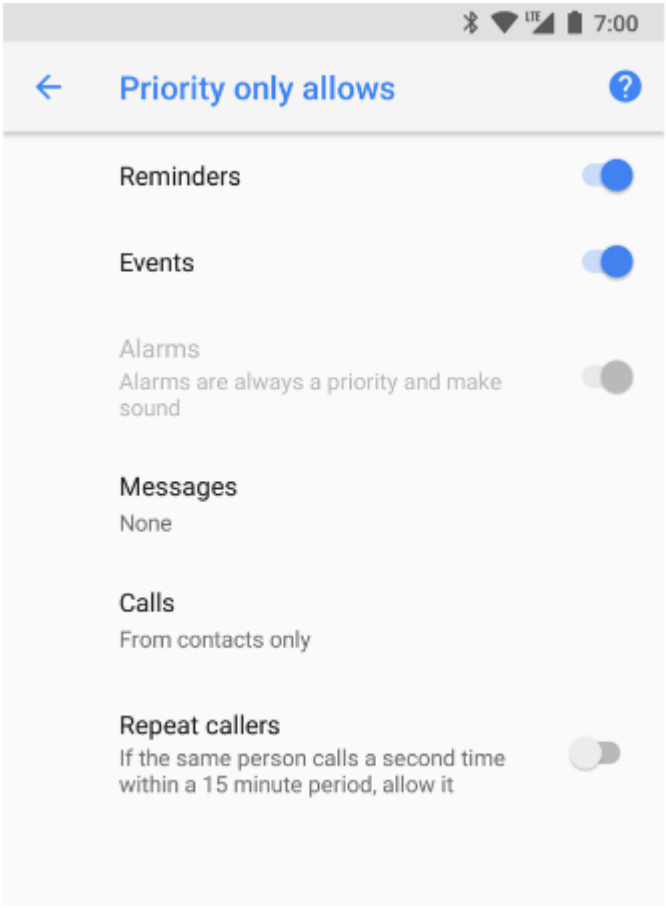


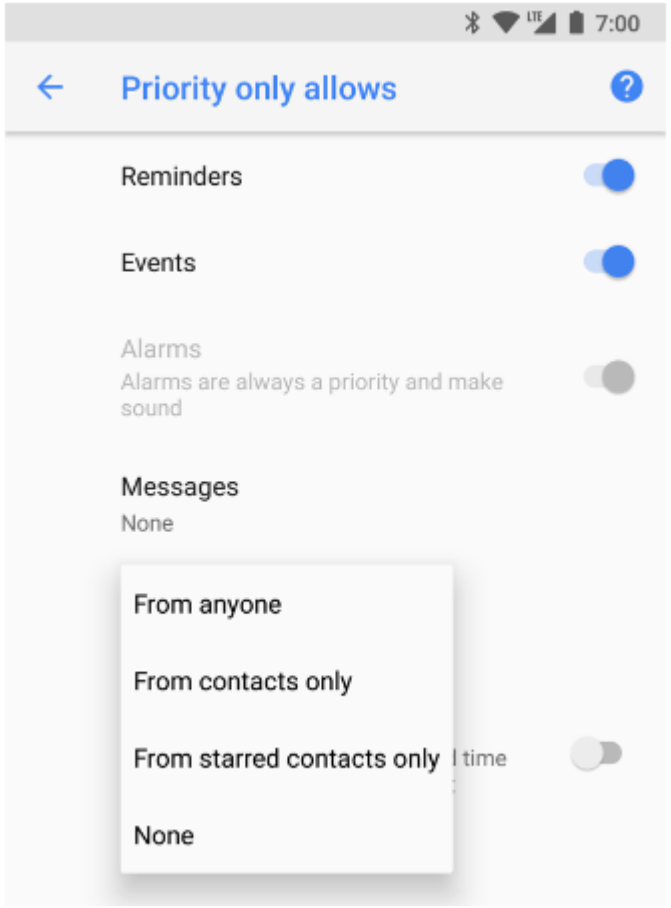
Do Not Disturb Mode

- Starting in Android 5.0 (API level 21), users can enable Do Not Disturb mode, which silences sounds and vibration for all notification
- Three levels in Do Not Disturb mode:
  1. Total silence: blocks all sounds and vibrations
  2. Alarm only: blocks all sounds and vibrations, except alarms
  3. Priority only: users can configure which system-wide categories can interrupt them (such as only alarms, reminders, events, calls, or messages)

Do Not Disturb Mode (2)

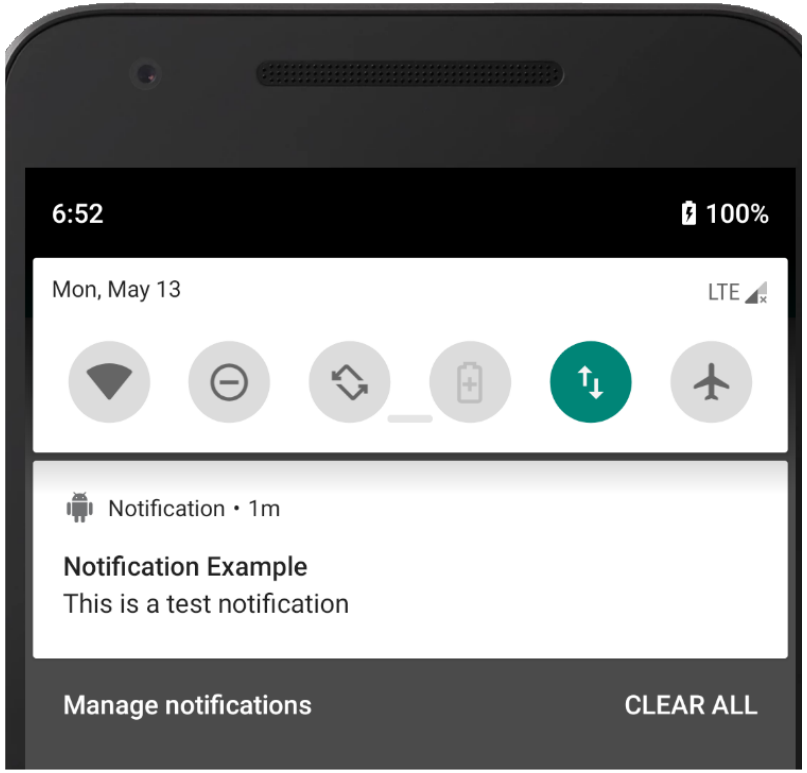
- Priority only





Create a Basic Notification

- A Notification in its most basic and compact form displays an icon, a title, and a small amount of content text



Create a Notification Channel

- Construct a NotificationChannel object with a unique channel ID, a user-visible name, and an importance level
- Optionally, specify the description that the user sees in the system settings with setDescription()
- Register the notification channel by passing it to createNotificationChannel()

```
private void createNotificationChannel() {
    // Create the NotificationChannel, but only on API 26+ because
    // the NotificationChannel class is new and not in the support library
    if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O) {
        CharSequence name = getString(R.string.channel_name);
        String description = getString(R.string.channel_description);
        int importance = NotificationManager.IMPORTANCE_DEFAULT;
        NotificationChannel channel = new NotificationChannel(CHANNEL_ID, name, importance);
        channel.setDescription(description);
        // Register the channel with the system; you can't change the importance
        // or other notification behaviors after this
        NotificationManager notificationManager = getSystemService(NotificationManager.class);
        notificationManager.createNotificationChannel(channel);
    } }
```

Set the Importance Level

```
int importance = NotificationManager.IMPORTANCE_DEFAULT;
NotificationChannel channel = new NotificationChannel(CHANNEL_ID, name, importance);
```

User-visible importance level	Importance (Android 8.0 and higher)	Priority (Android 7.1 and lower)
Urgent Makes a sound and appears as a heads-up notification	IMPORTANCE_HIGH	PRIORITY_HIGH or PRIORITY_MAX
High Makes a sound	IMPORTANCE_DEFAULT	PRIORITY_DEFAULT
Medium No sound	IMPORTANCE_LOW	PRIORITY_LOW
Low No sound and does not appear in the status bar	IMPORTANCE_MIN	PRIORITY_MIN

https://developer.android.com/reference/android/app/NotificationManager.html

Set the Notification Content

- To get started, you need to set the Notification's content and channel using a NotificationCompat.Builder object
- A small icon, set by setSmallIcon()
- A title, set by setContentTitle()
- The body text, set by setContentText()
- The Notification priority, set by setPriority()

```
NotificationCompat.Builder builder = new NotificationCompat.Builder(this,CHANNEL_ID)
        .setSmallIcon(R.drawable.notification_icon)
        .setContentTitle("Notification Example")
        .setContentText("This is a test notification")
        .setPriority(NotificationCompat.PRIORITY_DEFAULT);
```

Show the Notification

- To make the Notification appear, call NotificationManagerCompat.notify(), passing it a unique ID for the notification and the result of NotificationCompat.Builder.build()

```
NotificationManager manager =
(NotificationManager)getSystemService(Context.NOTIFICATION_SERVICE);
manager.notify(0,builder.build());
```

Notification ID

MainActivity of Basic Notification (1)

```
public class MainActivity extends AppCompatActivity {

    private static final String CHANNEL_ID = "channel_id";

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

        createNotificationChannel();

        Button b1 = (Button) findViewById(R.id.notiButton);
        b1.setOnClickListener( new Button.OnClickListener(){
            @Override
            public void onClick(View v) {
                addNotification();
            }
        });
    }
```

MainActivity of Basic Notification (2)

```
    private void addNotification(){
            NotificationCompat.Builder builder = new NotificationCompat.Builder(this,CHANNEL_ID)
                    .setSmallIcon(R.drawable.notification_icon)
                    .setContentTitle("Notification Example")
                    .setContentText("This is a test notification")
                    .setPriority(NotificationCompat.PRIORITY_DEFAULT);

            NotificationManager manager =
(NotificationManager)getSystemService(Context.NOTIFICATION_SERVICE);
            manager.notify(0,builder.build());
    }

    private void createNotificationChannel() {
        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O) {
            CharSequence name = getString(R.string.channel_name);
            String description = getString(R.string.channel_description);
            int importance = NotificationManager.IMPORTANCE_DEFAULT;
            NotificationChannel channel = new NotificationChannel(CHANNEL_ID, name, importance);
            channel.setDescription(description);
            NotificationManager notificationManager =
getSystemService(NotificationManager.class);
            notificationManager.createNotificationChannel(channel);
        }
    }
}
```

Set the Notification's Tap Action

- Every Notification should respond to a tap, usually to open an activity in your app
- To do, you must specify a content intent defined with a PendingIntent object and pass it to setContentIntent()

```
Intent notificationIntent = new Intent(this,MainActivity.class);
PendingIntent contentIntent = PendingIntent.getActivity(this, 0,
notificationIntent,PendingIntent.FLAG_UPDATE_CURRENT);
builder.setContentIntent(contentIntent);
//To automatically removes the notification when the user tap it.
builder.setContentIntent(contentIntent).setAutoCancel(true);
```

Update a Notification

- To update the notification after you've issued it, call NotificationManagerCompat.notify() again
  - Passing it a notification with the same ID you used previously
- You can optionally call setOnlyAlertOnce() so your notification interrupts the user only the first time the notification appears and not for later updates

Remove a Notification

- NotificationManager
  - Calls cancel() for a specific notification ID. This method deletes ongoing notifications
  - Calls cancelAll(), which removes all of the notifications
- NotificationCompat.Builder
  - The user clicks the notification, and you called setAutoCancel() when you created the notification
  - If you set a timeout when creating a notification using setTimeoutAfter(), the system cancel the notification after the specified duration elapses

Exercise – Notification Sender

- Two EditText views
  - textTitle: the user sets title of a notification
  - textContent: the user sets content of a notification

  Assume that both of EditText views always have value
- Three Buttons
  - Notifiy: issues a new notification
  - Update: updates the recent notification
  - RemoveAll: removes all notifications

Appendix

- https://developer.android.com/guide/topics/ui/notifiers/notifications
- https://developer.android.com/training/notify-user/build-notification#Actions