# CS 475/575

Alarm Manager

Notifications

# Alarm Manager

- Fire Intents at predetermined intervals or times
- Do not need app running to work
- Bit more overhead than Timers
- Trigger whatever (broadcast intents, services, notifications, even activities with some work)
- Allow effective resource management

# Alarm Manager-Create, Set, Cancel

- Get Reference to AlarmManager service

```
AlarmManager alarmManager = (AlarmManager) getSystemService(Context.ALARM_SERVICE);
```

- Decide type
  - RTC_WAKEUP – wakes and fires pending intent at time specified
  - RTC – same as RTC_WAKEUP **no** wakeup
  - ELAPSED_REALTIME_WAKEUP - wakes and fires pending intent after specific amount of time has elapsed since system boot
  - ELAPSED_REALTIME - same as ELAPSED_REALTIME_WAKEUP **no** wakeup

# Alarm Manager-Create, Set, Cancel

- ELAPSED_REALTIME_WAKEUP
- ELAPSED_REALTIME
  - Starts specific time after device boots

```
alarmMgr = (AlarmManager)context.getSystemService(Context.ALARM_SERVICE);
alarmMgr.set(AlarmManager.ELAPSED_REALTIME_WAKEUP,
        SystemClock.elapsedRealtime() +
        60 * 1000, alarmIntent);
```

- RTC_WAKEUP
- RTC
  - Starts specific time from now

```
final int TEN_MINUTES = 6000000;  //60sec/min*10min*1000msec/sec
alarmManager.set(AlarmManager.RTC_WAKEUP, TEN_MINUTES, alarmIntent);
```

# Alarm Manager-Create, Set, Cancel

Wake up the device to fire a one-time (non-repeating) alarm in one minute

```
private AlarmManager alarmMgr;
private PendingIntent alarmIntent;
...
alarmMgr = (AlarmManager)context.getSystemService(Context.ALARM_SERVICE);
Intent intent = new Intent(context, AlarmReceiver.class);
alarmIntent = PendingIntent.getBroadcast(context, 0, intent, 0);

alarmMgr.set(AlarmManager.ELAPSED_REALTIME_WAKEUP,
        SystemClock.elapsedRealtime() +
        60 * 1000, alarmIntent);
```

Intents fired for your app by another app at a later time
Execute with same identity and permissions as your app

Cancel alarm

```
// If the alarm has been set, cancel it.
if (alarmMgr!= null) {
    alarmMgr.cancel(alarmIntent);
}
```

# Alarm Manager-Repeat Alarm

- setRepeating – for precise control over exact alarm interval
- setInExactRepeating – OS schedules multiple inExactRepeating alarms to execute at same time
  - INTERVAL_FIFTEEN_MINUTES, INTERVAL_HALF_HOUR, INTERVAL_HOUR, INTERVAL_HALF_DAY, INTERVAL_DAY
  - If do not require precision, then wake all alarms that are ‘close together’ at same time (verses multiple wakeups around same time)
  - Reduces battery drain

# Alarm Manager-Repeat Alarm

- setRepeating

```
final int TEN_MINUTES = 6000000;  //60sec/min*10min*1000msec/sec
alarmManager.setRepeating(AlarmManager.RTC_WAKEUP,
        TEN_MINUTES,
        AlarmManager.INTERVAL_DAY,
        alarmIntent);
```

- setInExactRepeating

```
final int TEN_MINUTES = 6000000;  //60sec/min*10min*1000msec/sec
alarmManager.setInexactRepeating(AlarmManager.RTC_WAKEUP,
        TEN_MINUTES,
        AlarmManager.INTERVAL_DAY,
        alarmIntent);
```

# Pending Intent

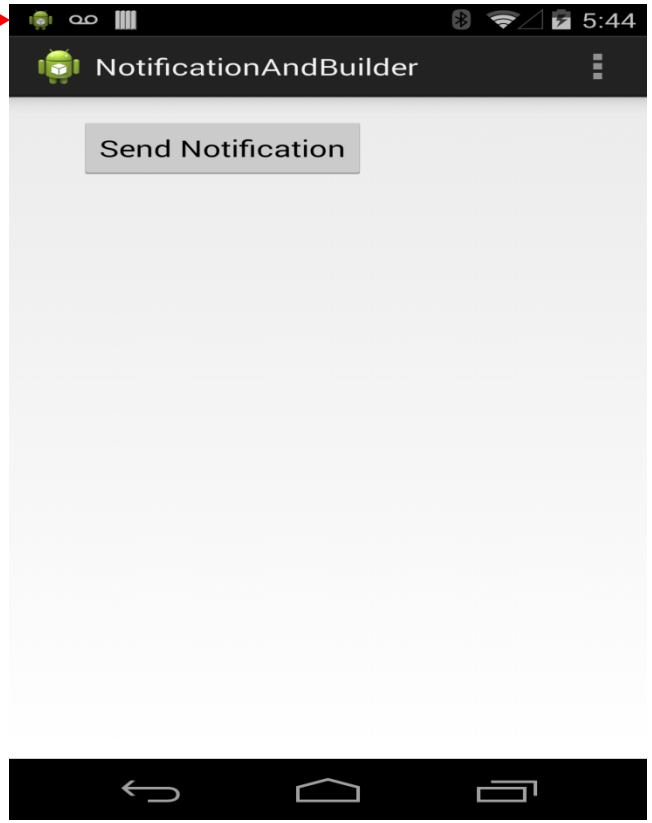
- Intents fired for your app by another app at a later time
- Execute with same identity and permissions as your app

# CS 475/575

Notifications

# Notifications

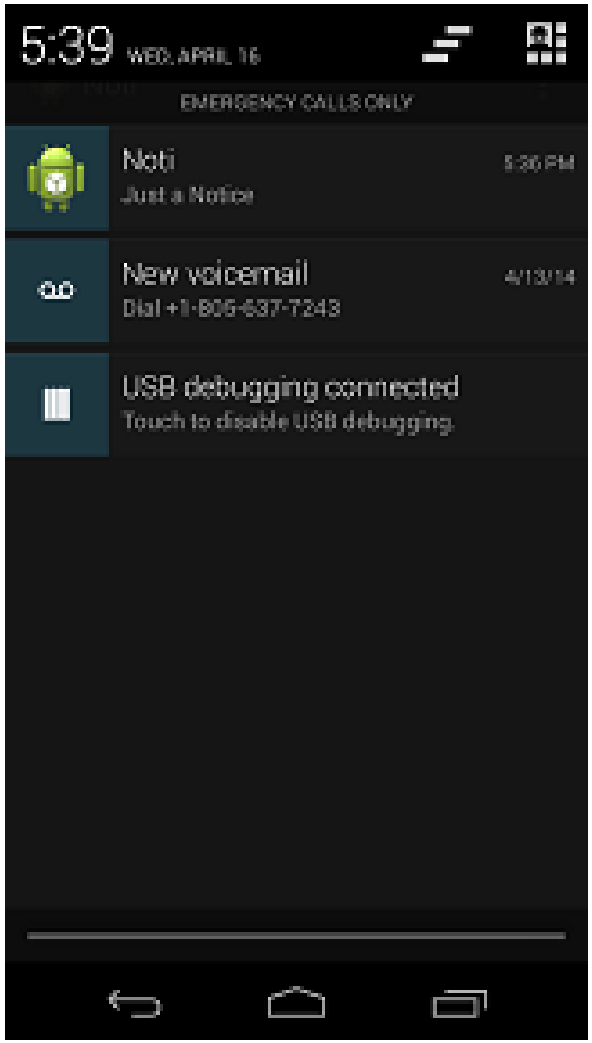
Messages and Information
That appear here



# Notifications

Status Bar
Settings, time, date

Notification Tray
Drag down from top



# Notifications

- Notices that do not require activity
- Handled by Notification Manager
  - Display a status bar icon
  - Flash Lights/LEDs
  - Vibrate Phone
  - Audible Alerts (Ringtones etc)
  - Display additional info in notification tray
  - Broadcast intents from notification tray

# Notifications

- Use Case
  1. Gradle Dependency
  2. Create a channel (if target API >=26)
  3. Build notification
  4. Send notification

# Notifications

- Gradle Dependency
- Be sure to include the proper androidx dependency in the build.gradle(app) file

```
dependencies {
    implementation 'androidx.appcompat:appcompat:1.1.0'
```

# Notifications

- Create a channel (if target API >=26)

```
private static final String CHANNEL_ID = "KP";
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 = "channel1";
        String description = "demo channel";
        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);
    }
}
```

# Notifications

- Build notification
  - CHANNEL_ID comes from channel created last slide

```
NotificationCompat.Builder builder = new NotificationCompat.Builder( context: this, CHANNEL_ID)
        .setSmallIcon(R.drawable.ic_launcher)
        .setContentTitle(getString(R.string.app_name))
        .setContentText("Just a Notice")
        .setPriority(NotificationCompat.PRIORITY_DEFAULT);
```

# Notifications

- Send notification

```
NotificationManagerCompat notificationManager = NotificationManagerCompat.from(this);
notificationManager.notify(MYNOTIFICATION, builder.build());
```

# Notifications

- Can show progress
- Retrigger
  - Pass in same ref ID with updated notification fields
- Cancel
  - notificationManager.cancel(NOTIFICATION_REF);

# Notifications Example

- See 12_Notification project

# References

- Create a Notification- Android Developer
- https://developer.android.com/training/notify-user/build-notification.html