CMSC436: Programming Handheld Systems

Fall 2017

Alarms

Today’s Topics

Alarms

AlarmManager APIs

Alarm Types

Example Application

Alarms

Mechanism for sending Intents at some point in the future

Allows one application to make code execute, even when that application is no longer running

Alarms

Once registered, Alarms remain active even if the device is asleep

Can configure Alarms to wake a sleeping device

Alarms are canceled on device shutdown/restart

Alarm Examples

MMS - Retry Scheduler

Settings - Bluetooth Discoverable timeout

Phone - User Info Cache

AlarmManager

You create and manage Alarms by interacting with the AlarmManager

Get a reference to the AlarmManager by calling the Context class’

getSystemService(Context.ALARM_SERVICE)

Creating Alarms

```
// One-shot Alarm with inexact timing. If there is already an alarm
// scheduled for the same IntentSender, that previous alarm will first
// be canceled
```

```
void set(int type, long triggerAtTime, PendingIntent operation)
```

Creating Alarms

```
// One-shot Alarm with exact timing
void setExact (int type, long triggerAtMillis, PendingIntent operation)
```

Creating Alarms

```
// Repeating alarm with inexact trigger criteria

void setRepeating (int type, long triggerAtMillis,
                                    long intervalMillis, PendingIntent operation)
```

Alarm Types

- Two degrees of configurability
  - How to interpret time
  - What to do if the device is sleeping when the Alarm fires

Interpreting Time

Realtime - relative to system clock

Elapsed - relative to time since last boot

Sleeping Devices

Wake up device now & deliver Intent

Wait to deliver Intent until device wakes up

Alarm Type Constants

RTC_WAKEUP

RTC

ELAPSED_REALTIME

ELAPSED_REALTIME_WAKEUP

PendingIntent

A description of an Intent and target action to perform with it

Can be handed to other applications so that they can perform actions on your behalf at a later time

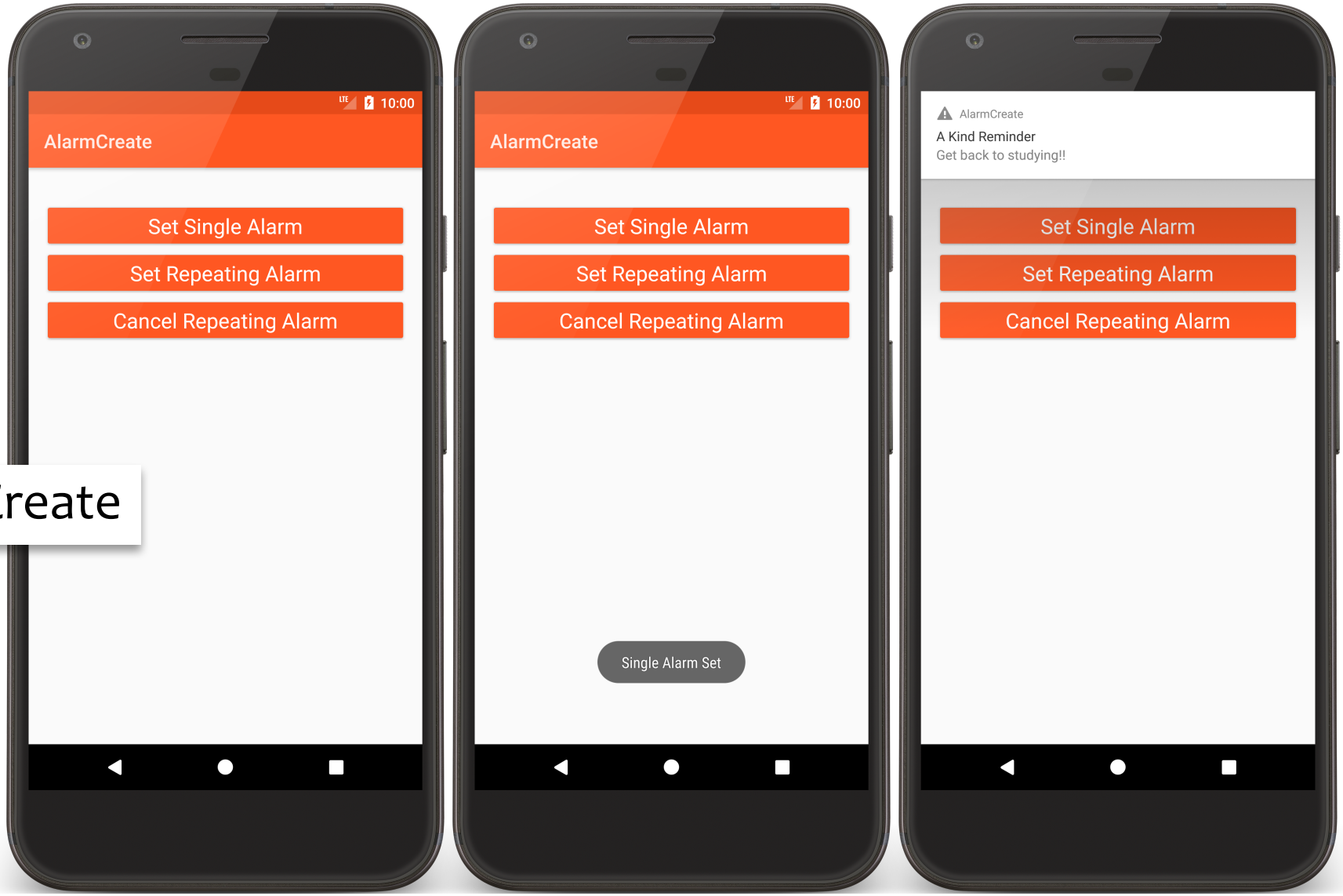
PendingIntent

PendingIntent getActivity(Context context, int requestCode,
Intent intent, int flags, Bundle options)

PendingIntent getBroadcast(Context context, int requestCode,
Intent intent, int flags)

PendingIntent getService(Context context, int requestCode, Intent intent, int flags)

AlarmCreate



```
public class AlarmCreateActivity extends Activity {
…
   public void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.main);

      // Get the AlarmManager Service
      mAlarmManager = (AlarmManager) getSystemService(ALARM_SERVICE);

      // Create an Intent to broadcast to the AlarmNotificationReceiver
      Intent mNotificationReceiverIntent = new Intent(AlarmCreateActivity.this,
            AlarmNotificationReceiver.class);

      // Create an PendingIntent that holds the NotificationReceiverIntent
      mNotificationReceiverPendingIntent = PendingIntent.getBroadcast(
            AlarmCreateActivity.this, 0, mNotificationReceiverIntent, 0);
```

```
...
    // Create an Intent to broadcast to the AlarmLoggerReceiver
    Intent mLoggerReceiverIntent = new Intent(AlarmCreateActivity.this,
            AlarmLoggerReceiver.class);

    // Create PendingIntent that holds the mLoggerReceiverPendingIntent
    mLoggerReceiverPendingIntent = PendingIntent.getBroadcast(
            AlarmCreateActivity.this, 0, mLoggerReceiverIntent, 0);

}
```

```
// Set single Alarms
 public void onClickButton(View v) {

    // Set first alarm to fire immediately
     mAlarmManager.set(AlarmManager.RTC_WAKEUP,
           System.currentTimeMillis(),
           mNotificationReceiverPendingIntent);

    // Set single alarm to fire shortly after previous alarm
    mAlarmManager.set(AlarmManager.RTC_WAKEUP,
           System.currentTimeMillis() + JITTER,
           mLoggerReceiverPendingIntent);

    // Show Toast message
    Toast.makeText(getApplicationContext(), "Single Alarm Set",
           Toast.LENGTH_LONG).show();
 }
```

```
public class AlarmLoggerReceiver extends BroadcastReceiver {

  public void onReceive(Context context, Intent intent) {

    // Log receipt of the Intent with timestamp
    Log.i(TAG, context.getString(R.sting.logging_at_string) +
                                    DateFormat.getDateTimeInstance().format(new Date()));
  }
}
```

```
public class AlarmNotificationReceiver extends BroadcastReceiver {
    …
  public void onReceive(Context context, Intent intent) {
     mContext = context;
     // Create and Send Notification

     // Log occurrence of notify() call
     Log.i(TAG, mContext.getString(R.string.sending_not_string)
           + DateFormat.getDateTimeInstance().format(new Date()));
  }
```

```
// Set repeating Alarms
public void onClickRepButton(View v) {
    // Set first repeating Alarm
    mAlarmManager.setRepeating(AlarmManager.ELAPSED_REALTIME,
            SystemClock.elapsedRealtime(), REPEAT_INTERVAL,
            mNotificationReceiverPendingIntent);

    // Set repeating alarm to fire shortly after previous alarm
    mAlarmManager.setRepeating(AlarmManager.ELAPSED_REALTIME,
            SystemClock.elapsedRealtime() + JITTER, REPEAT_INTERVAL,
            mLoggerReceiverPendingIntent);

    // Show Toast message
    Toast.makeText(getApplicationContext(), "Repeating Alarm Set",
            Toast.LENGTH_LONG).show();
}
```

```
// Cancel existing Alarms
public void onClickCancelRepButton(View v) {

   // Cancel all alarms using mNotificationReceiverPendingIntent
   mAlarmManager.cancel(mNotificationReceiverPendingIntent);

   // Cancel all alarms using mLoggerReceiverPendingIntent
   mAlarmManager.cancel(mLoggerReceiverPendingIntent);

   // Show Toast message
   Toast.makeText(getApplicationContext(),
         "Repeating Alarms Cancelled", Toast.LENGTH_LONG).show();
}
```

Next Time

Networking