Alarms

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 set 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

Create & manage alarms indirectly, 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
void set(int type, long triggerAtTime, "
                        PendingIntent operation)
```

Creating Alarms

```
// repeating alarm
void setRepeating(int type, "
                          long triggerAtTime, "
                          long interval, "
                          PendingIntent operation)
```

Creating Alarms

```
// repeating alarm with inexact trigger criteria
void setInexactRepeating(int type, "
                                    long triggerAtTime, "
                                    long interval, "
                                    PendingIntent operation)
```

Interval options

- INTERVAL_FIFTEEN_MINUTES
- INTERVAL_HALF_HOUR
- INTERVAL_HOUR
- INTERVAL_HALF_DAY
- INTERVAL_DAY

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

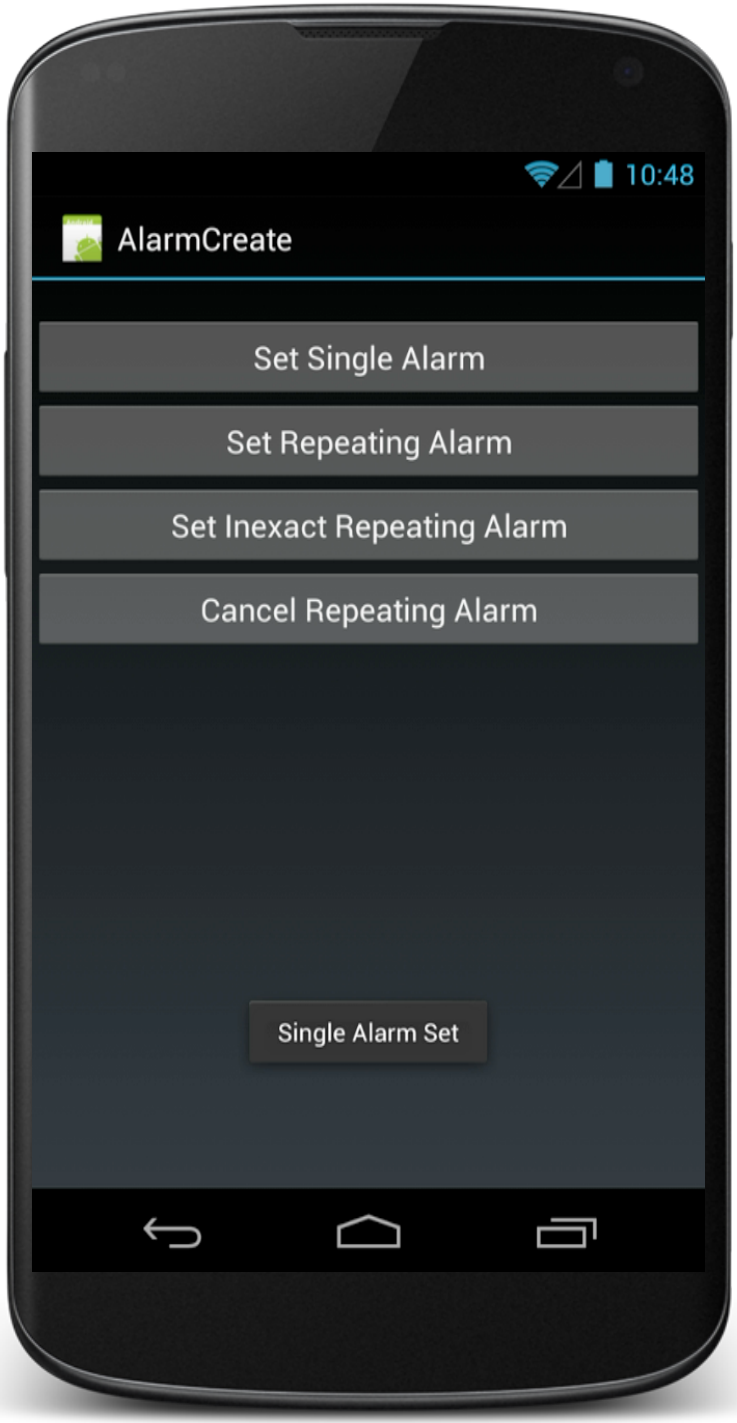
ELAPSED_REALTIME_WAKEUP

PendingIntent

```
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)
```


10:48
AlarmCreate
Set Single Alarm
Set Repeating Alarm
Set Inexact Repeating Alarm
Cancel Repeating Alarm
Single Alarm Set

AlarmCreate

```
@Override
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
    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
    mLoggerReceiverIntent = new Intent(AlarmCreateActivity.this,
            AlarmLoggerReceiver.class);

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

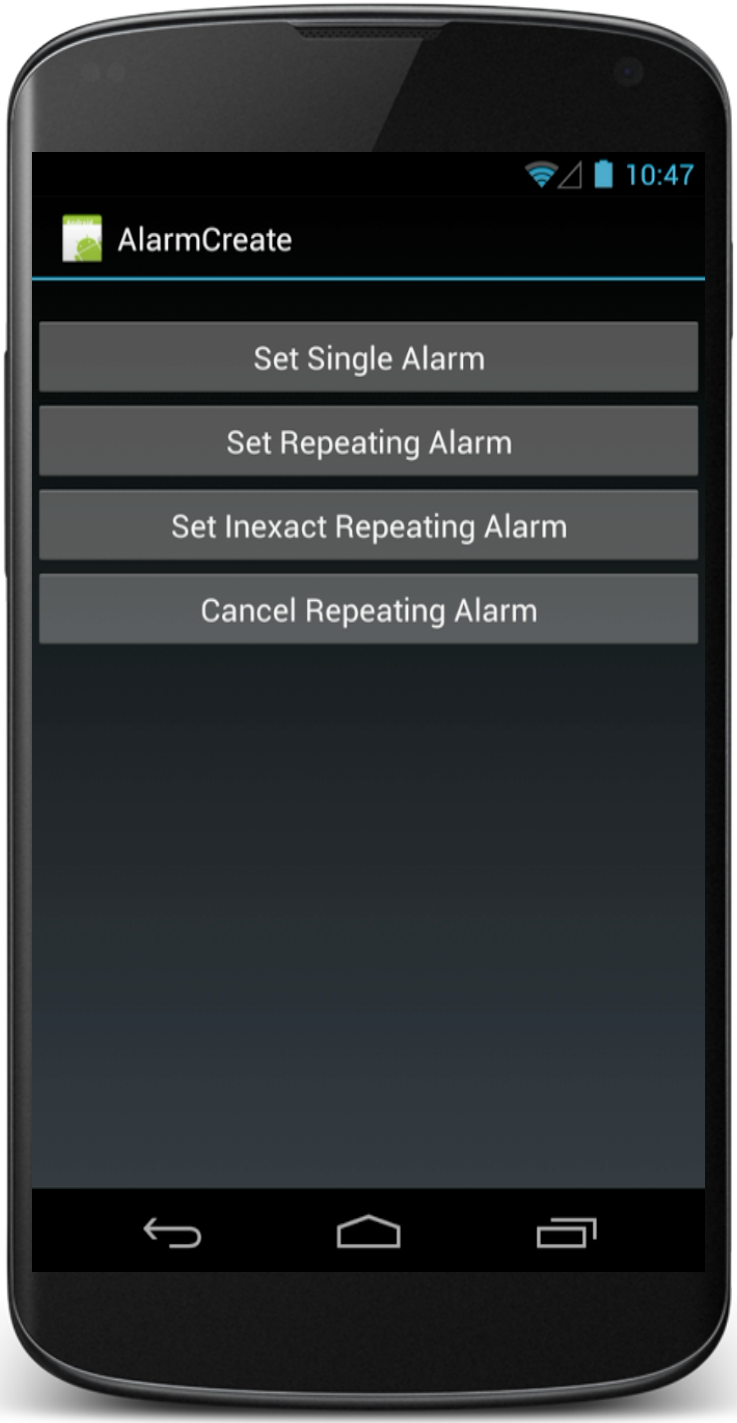
AlarmCreate

```
// Set up single alarm Button
final Button singleAlarmButton = (Button) findViewById(R.id.single_alarm_button);
singleAlarmButton.setOnClickListener(new OnClickListener() {
    @Override
    public void onClick(View v) {

        // Set single alarm
        mAlarmManager.set(AlarmManager.RTC_WAKEUP,
                System.currentTimeMillis() + INITIAL_ALARM_DELAY,
                mNotificationReceiverPendingIntent);

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

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


10:47
AlarmCreate
Set Single Alarm
Set Repeating Alarm
Set Inexact Repeating Alarm
Cancel Repeating Alarm

AlarmCreate

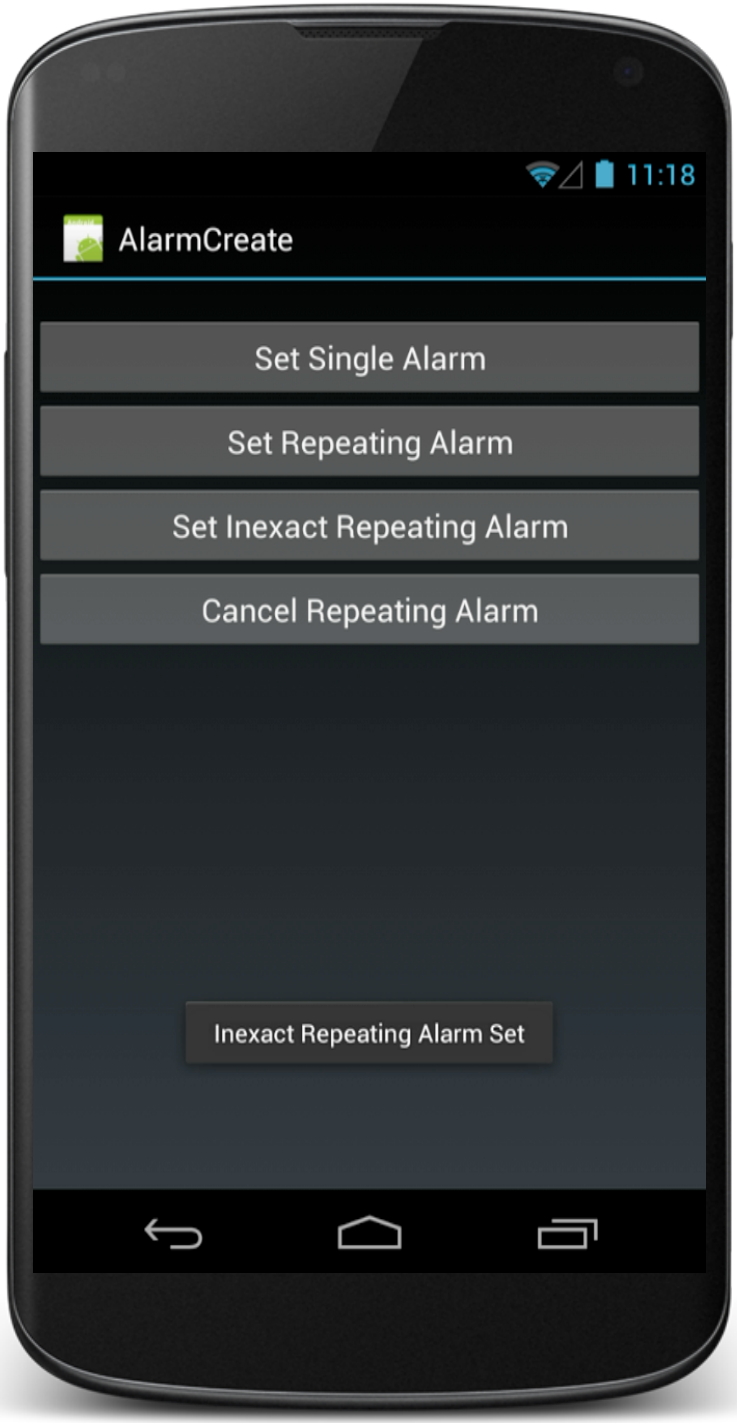
```
// Set up repeating Alarm Button
final Button repeatingAlarmButton = (Button) findViewById(R.id.repeating_alarm_button);
repeatingAlarmButton.setOnClickListener(new OnClickListener() {

    @Override
    public void onClick(View v) {

        // Set repeating alarm
        mAlarmManager.setRepeating(AlarmManager.ELAPSED_REALTIME,
                SystemClock.elapsedRealtime() + INITIAL_ALARM_DELAY,
                AlarmManager.INTERVAL_FIFTEEN_MINUTES,
                mNotificationReceiverPendingIntent);

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

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


11:18
AlarmCreate
Set Single Alarm
Set Repeating Alarm
Set Inexact Repeating Alarm
Cancel Repeating Alarm
Inexact Repeating Alarm Set

AlarmCreate

```
// Set up inexact repeating alarm Button
final Button inexactRepeatingAlarmButton = (Button) findViewById(R.id.inexact_repeating_alarm_button);
inexactRepeatingAlarmButton.setOnClickListener(new OnClickListener() {

    @Override
    public void onClick(View v) {

        // Set inexact repeating alarm
        mAlarmManager.setInexactRepeating(
                AlarmManager.ELAPSED_REALTIME,
                SystemClock.elapsedRealtime() + INITIAL_ALARM_DELAY,
                AlarmManager.INTERVAL_FIFTEEN_MINUTES,
                mNotificationReceiverPendingIntent);
        // Set inexact repeating alarm to fire shortly after previous alarm
        mAlarmManager.setInexactRepeating(
                AlarmManager.ELAPSED_REALTIME,
                SystemClock.elapsedRealtime() + INITIAL_ALARM_DELAY
                        + JITTER,
                AlarmManager.INTERVAL_FIFTEEN_MINUTES,
                mLoggerReceiverPendingIntent);

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

AlarmCreate

```
    // Set up cancel repeating alarm Button
    final Button cancelRepeatingAlarmButton = (Button) findViewById(R.id.cancel_repeating_alarm_button);
    cancelRepeatingAlarmButton.setOnClickListener(new OnClickListener() {

        @Override
        public void onClick(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();
        }
    });
}
```