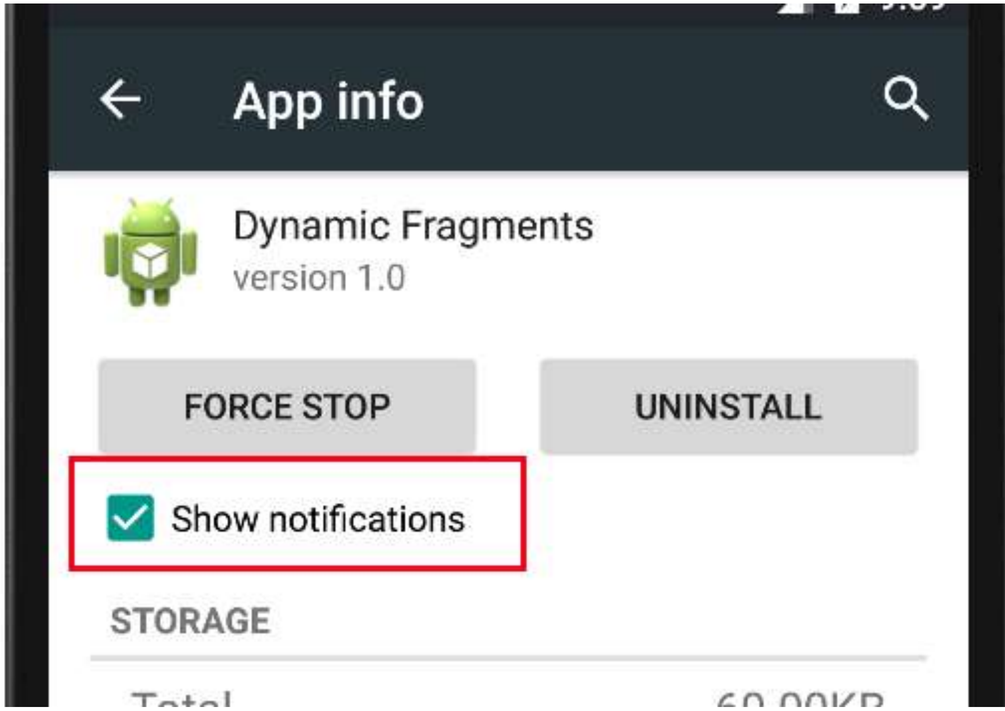
Notification

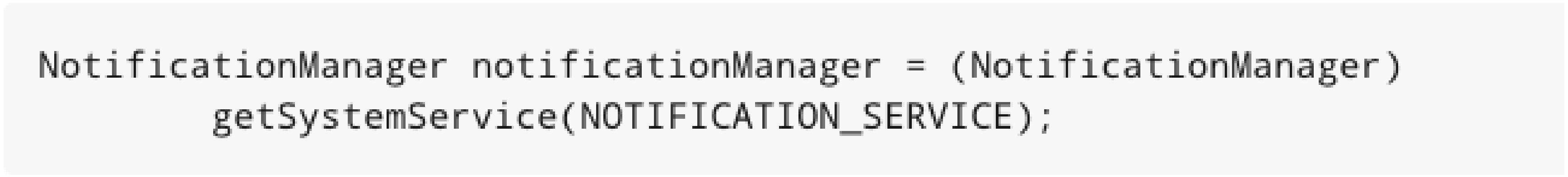
Preventing Notification

- Select Apps in the Settings, select the app you are interested in and unset the Show notifications checkbox to prevent the app to show notifications.



Setting up Notifications

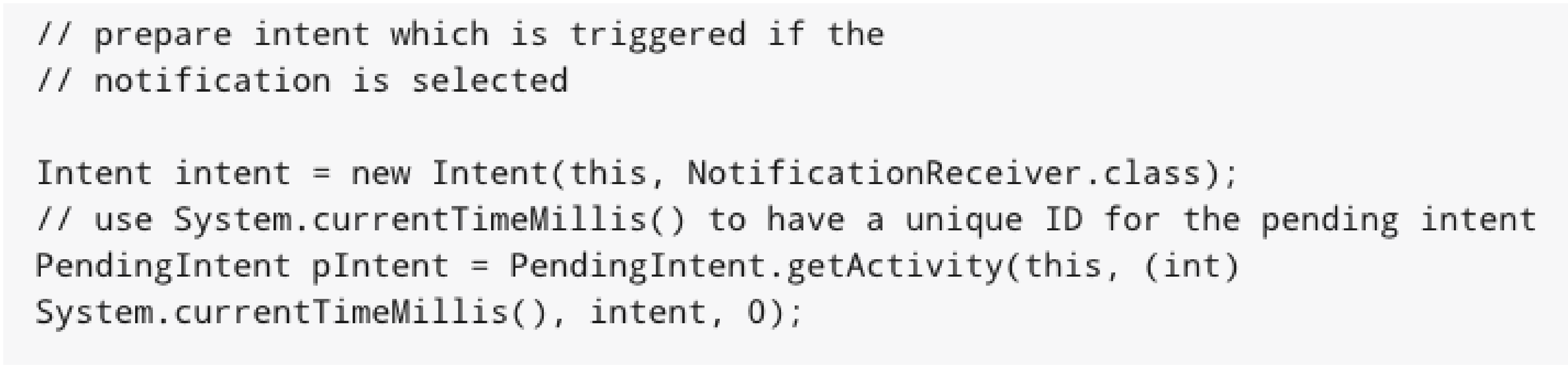
- Notification class
- NotificationManager class which can be received from the Context, e.g. an *activity* or a *service*, via the () getSystemService



```
NotificationManager notificationManager = (NotificationManager)
        getSystemService(NOTIFICATION_SERVICE);
```

- Notification.Builder provides an builder interface to create an Notification object.
- use a PendingIntent to specify the action which should be performed once the user select the notification.

- The Notification.Builder allows you to add up to three buttons with definable actions to the notification.



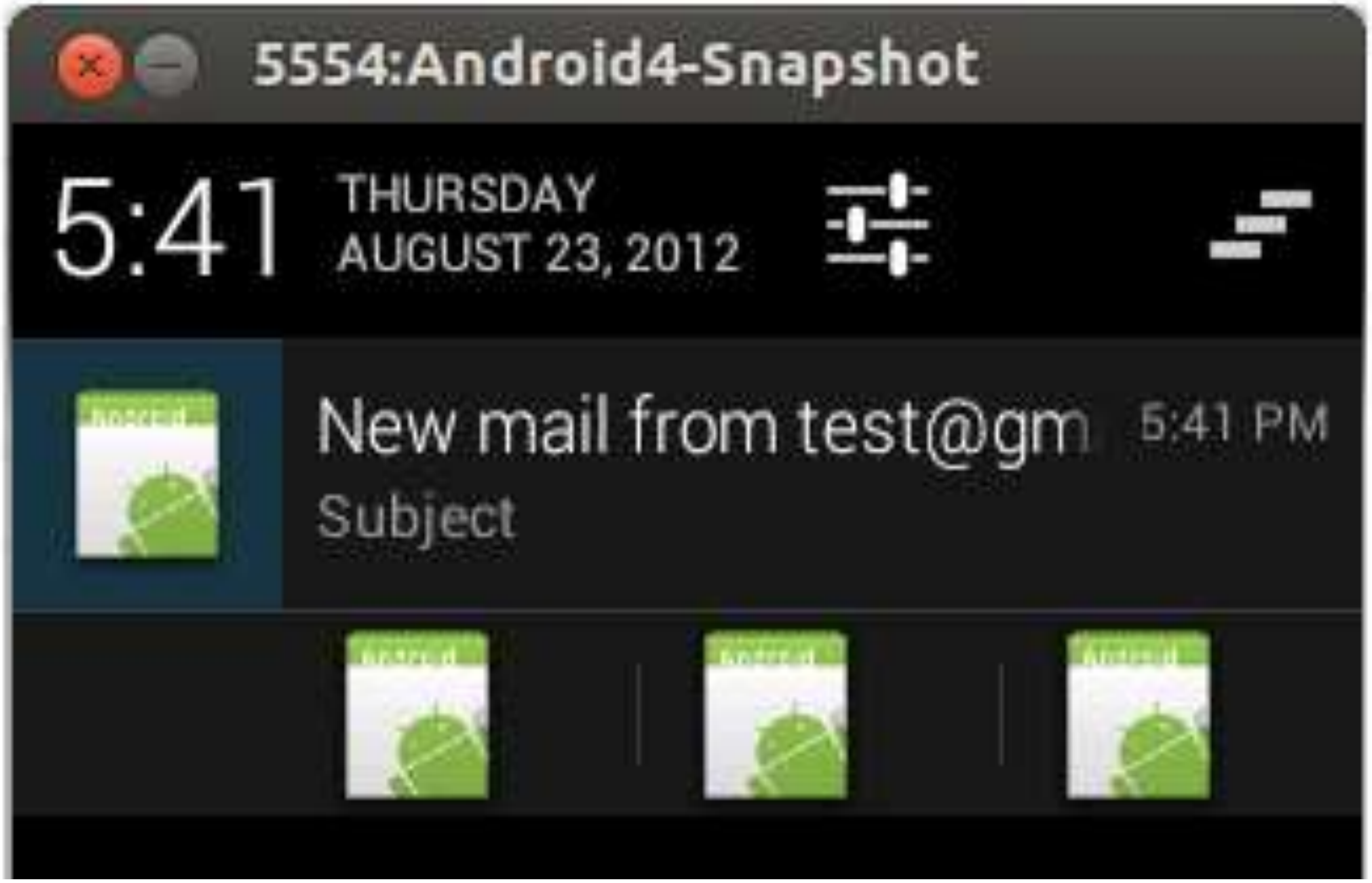
```
// prepare intent which is triggered if the
// notification is selected

Intent intent = new Intent(this, NotificationReceiver.class);
// use System.currentTimeMillis() to have a unique ID for the pending intent
PendingIntent pIntent = PendingIntent.getActivity(this, (int)
System.currentTimeMillis(), intent, 0);
```

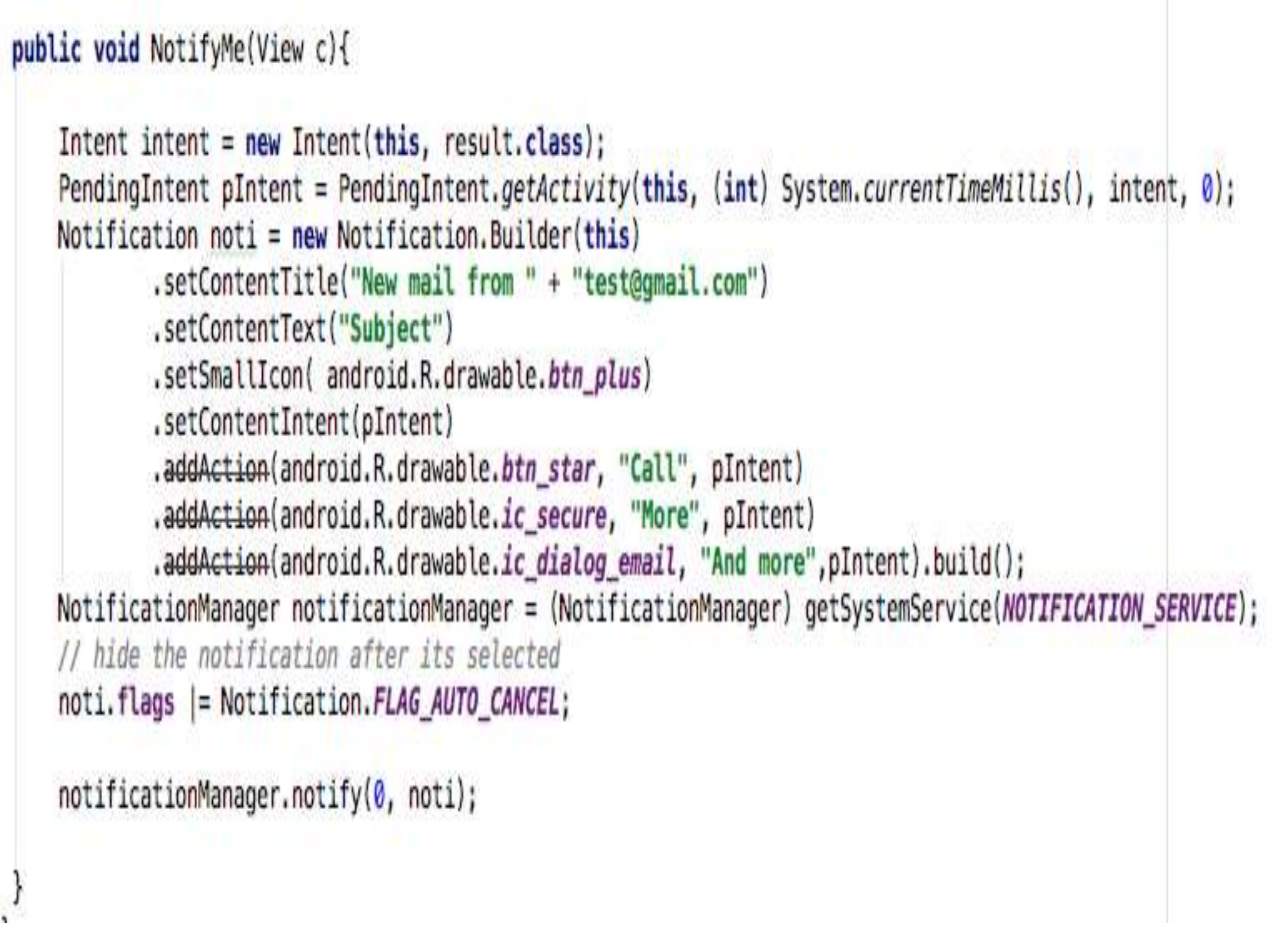
```
// build notification
// the addAction re-use the same intent to keep the example short
Notification n  = new Notification.Builder(this)
                                .setContentTitle("New mail from " +
"test@gmail.com")
                                .setContentText("Subject")
                                .setSmallIcon(R.drawable.icon)
                                .setContentIntent(pIntent)
                                .setAutoCancel(true)
                                .addAction(R.drawable.icon, "Call", pIntent)
                                .addAction(R.drawable.icon, "More", pIntent)
                                .addAction(R.drawable.icon, "And more",
pIntent).build();
```

```
NotificationManager notificationManager =
        (NotificationManager) getSystemService(NOTIFICATION_SERVICE);

notificationManager.notify(0, n);
```



New mail from test@gm 5:41 PM
Subject



```java
public void NotifyMe(View c){

    Intent intent = new Intent(this, result.class);
    PendingIntent pIntent = PendingIntent.getActivity(this, (int) System.currentTimeMillis(), intent, 0);
    Notification noti = new Notification.Builder(this)
            .setContentTitle("New mail from " + "test@gmail.com")
            .setContentText("Subject")
            .setSmallIcon( android.R.drawable.btn_plus)
            .setContentIntent(pIntent)
            .addAction(android.R.drawable.btn_star, "Call", pIntent)
            .addAction(android.R.drawable.ic_secure, "More", pIntent)
            .addAction(android.R.drawable.ic_dialog_email, "And more",pIntent).build();
    NotificationManager notificationManager = (NotificationManager) getSystemService(NOTIFICATION_SERVICE);
    // hide the notification after its selected
    noti.flags |= Notification.FLAG_AUTO_CANCEL;

    notificationManager.notify(0, noti);

}
```

Scheduling tasks

- If you have a repetitive task in your Android app, you need to consider that activities and services can be terminated by the Android system to free up resources. Therefore you can not rely on standard Java schedule like the TimerTasks class.

Options for scheduling

- The (outdated) AlarmManager
- the JobScheduler API.

JobScheduler to schedule background jobs

- The Android 5.0 Lollipop (API 21) release introduces a job scheduler API via the JobScheduler class
- In general this API can be used to schedule everything that is not time critical for the user.

- Compared to a custom SyncAdapter or the alarm manager,
  - the JobScheduler supports batch scheduling of jobs.
  - The Android system can combine jobs so that battery consumption is reduced.
  - JobManager makes handling uploads easier as it handles automatically the unreliability of the network. It also survives application restarts.

- Tasks that should be done once the device is connect to a power supply
- Tasks that require network access or a Wi-Fi connection.
- Task that are not critical or user facing
- Tasks that should be running on a regular basis as batch where the timing is not critical

JobInfo

- A unit of work is encapsulated by a JobInfo object.