# Android – Eine Einführung

## Toasts & Notifications

Andreas Wilhelm

CSC Computer-Schulung & Consulting GmbH

# Contents

# Toasts

# Contents

# Überblick

- Nachrichten bezüglich gestarteten, laufenden oder beendeten Operationen
- Kleine Anzeige, die nur benötigten Platz einnimmt
- Darunterliegende Oberfläche weiterhin nutzbar
- Keine Interaktion mit Toasts

```
Toast toast = Toast.makeText(getApplicationContext(), "Hello World!", Toast.LENGTH_SHORT);
toast.show();
```

Listing : Ein erster Toast

# Positionierung

- Normalerweise Anzeige zentriert am unteren Bildschirmrand
- Änderung der Position mit *setGravity()*
- Erstes Argument Positionierung auf Bildschirm
- Zweites und drittes Verschiebung

```
Toast toast = Toast.makeText(this, "Hello World!", Toast.LENGTH_LONG);
toast.setGravity(Gravity.CENTER_HORIZONTAL|Gravity.CENTER_VERTICAL, 0, 0);
toast.show();
```

Listing : Ein zentierter Toast

# Eigene Layouts

- Eigene Layouts mit *setView()* zuweisen
- Dabei sollte Hintergund mit Shapes deklariert werden

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:background="@drawable/toast_shape"
    android:orientation="horizontal"
    android:padding="8dp" >

    <ImageView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:src="@drawable/ubuntu"
        android:padding="3dip" />

    <TextView
        android:id="@+id/toastUbuntuName"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:textColor="@color/white"
        android:gravity="left|center_vertical"
        android:padding="7dip" />
</LinearLayout>
```

Listing : Das Toast layout

# Shapes

```
<?xml version="1.0" encoding="UTF-8"?>
<shape
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:shape="rectangle" >
    <solid
        android:color="#aa000000" />
    <corners
        android:bottomLeftRadius="7dp"
        android:bottomRightRadius="7dp"
        android:topLeftRadius="7dp"
        android:topRightRadius="7dp" />
</shape>
```

Listing : Der Toast Hintergrund

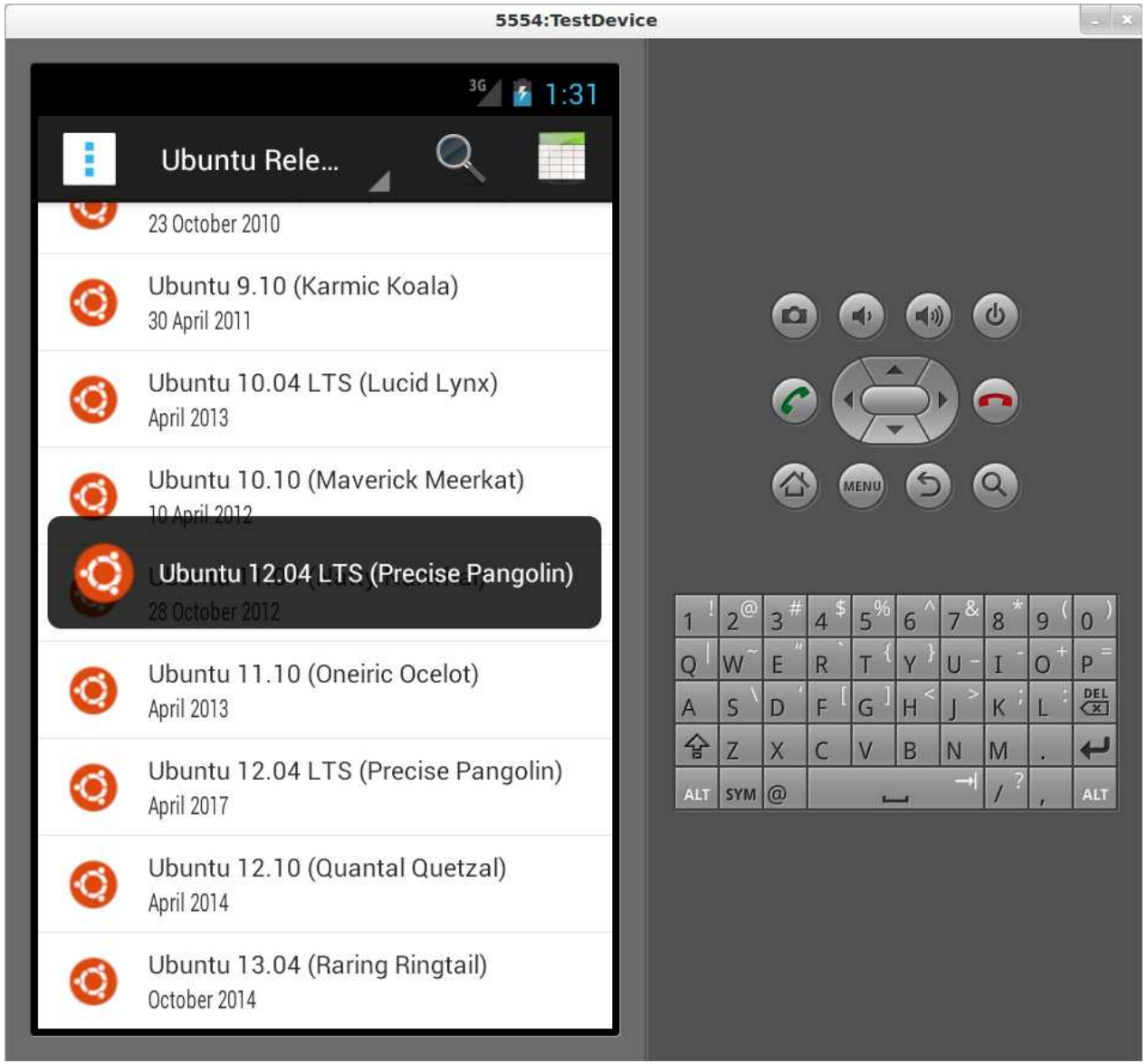
# Nutzung eigenes Layouts

- Laden des Layouts mit LayoutInflater
- Instanzierung eines Toasts mit Konstruktor (nicht *makeText*)
- Zuweisung des Layouts mit *setView()*

```
@Override
protected void onListItemClick(ListView l, View v, int position, long id) {
   LayoutInflater inflater = getLayoutInflater();
   View layout = inflater.inflate(R.layout.toast_layout, null);

   TextView ubuntuName = (TextView) layout.findViewById(R.id.toastUbuntuName);
   ubuntuName.setText(this.adapter.getItem(position));

   Toast toast = new Toast(this);
   toast.setGravity(Gravity.CENTER_HORIZONTAL|Gravity.CENTER_VERTICAL, 0, 0);
   toast.setDuration(Toast.LENGTH_LONG);
   toast.setView(layout);
   toast.show();
}
```

Listing : Ein angepasster Toast

# Screenshot


5554:TestDevice
3G
1:31
Ubuntu Rele...
23 October 2010
Ubuntu 9.10 (Karmic Koala)
30 April 2011
Ubuntu 10.04 LTS (Lucid Lynx)
April 2013
Ubuntu 10.10 (Maverick Meerkat)
10 April 2012
Ubuntu 12.04 LTS (Precise Pangolin)
28 October 2012
Ubuntu 11.10 (Oneiric Ocelot)
April 2013
Ubuntu 12.04 LTS (Precise Pangolin)
April 2017
Ubuntu 12.10 (Quantal Quetzal)
April 2014
Ubuntu 13.04 (Raring Ringtail)
October 2014
MENU

# Anmerkung

## Toast Konstruktor

Es ist unbedingt darauf zu achten, dass der öffentliche Konstruktor der Klasse Toast nur dann verwendet wird, wenn ein eigenes Layout dem Toast zugewiesen werden soll. In allen anderen Fällen sollte die statische Methode *makeText()* verwendet werden.

# Notifications

# Contents

# Allgemeines

- Nachrichten in einer von der Applikation unabhängigen Umgebung
- Anzeige in Bar am oberen Bildschirmrand (NotificationBar)
- Zuerst nur Anzeige eines Icons
- Anzeige weiterer Informationen in Notification-DropDown
- Erstellen von Notifications mit *Notification.Builder*
- Bestandteil der API seit Android Version 3.0
- Verfügbar über Android Support Library
- Einführung weiterer Ansichten in Android 4.1

## NotificationCompat.Builder

*Notification.Builder* wurde erst in Android 3.0 (API-Version 11) eingeführt. Verwendung nur mit minimal unterstützter API-Version der Applikation 11 oder höher. Unterstützung älterer API-Versionen mit NotificationCompat.Builder, der Teil der Android Support Library ist.

# Android Support Library

## Android Support Library

Die Android Support Library enthält Bibliotheken, die in älteren API-Versionen noch nicht zur Verfügung standen oder Werzeuge bereitstellen, die nicht Teil der Standard-API sind.

Man muss allerdings bei der Entwicklung darauf achten, dass die Support Library mehrere Bibliotheken mit unterschiedlichen Anforderungen an die minimal unterstützte API enthält. Welche Bibliothek welche Anforderungen stellt, kann man ganz einfach der Verzeichnisstruktur der Support Library entnehmen. Alle Bibliotheken mit einer minimalen API-Version 4 findet man unter *v4*.

# Normale Notifications

- Höhe normaler Notifications bis zu 64 dp
- Titel am oberen Rand
- Großes Icon am linken Rand
- Zusatztext am unteren Rand
- Kleine Zusatzinformation und ein kleines Icon am rechten unteren Rand
- Zeitpunkt am rechten oberen Rand

# Erweiterte Notifications

- Eingeführt in Android 4.1
- Gleiches Aussehen, wie normale Notifications
- Können allerdings ausgeklappt werden
- Zustätzlicher Bereich zwischen Titelzeile und der Textzeile am unteren Rand
- Verschiedene Anzeigen für
  - Großes Bild
  - Längeren Text
  - Zeilenweiser Text (Liste)

# Allgemeines

- Deklarierung mit *Notification.Builder*
- Erzeugen mit Methode *build()*
- Weitergabe an System mit *NotificationManager*
- Verschiedene Einstellung im *Notification.Builder* möglich

| Methode | Beschreibung |
|---|---|
| *setSmallIcon()* | Ein kleines Icon, dass in der NotificationBar angezeigt wird |
| *setContentTitle()* | Der Titel der Notification |
| *setContentText()* | Ein kurzer Text mit Zusatzinformationen |

# Näheres zu Attributen

- Zuweisung von Icons oder Sounds
- Zuweisung einer Aktion immer sinnvoll
- Aktion wird als PendingIntent an *setContentIntent()* übergeben
- PendingIntent erlaubt anderer Applikation eine Aktivität zu starten

```
// Initialize the notification builder and assign the
// important notification data.
Notification.Builder releaseBuilder = new Notification.Builder(this)
    .setSmallIcon(R.drawable.ubuntu)
    .setContentTitle("New Ubuntu Release")
    .setContentText("Ubuntu 12.10 – Quantal Quetzal released!");

// Create an explicit intent to start the Ubuntu Release List activity.
Intent ubuntuIntent = new Intent(this, HelloList.class);

// Setup the pending intent to start the Ubuntu Release List activity.
PendingIntent pendingIntent = PendingIntent.getActivity(
      this, 0, ubuntuIntent, PendingIntent.FLAG_UPDATE_CURRENT);
releaseBuilder.setContentIntent(pendingIntent);

// Get the notification manager and show the notification.
NotificationManager notificationManager = (NotificationManager) getSystemService(Context.NOTIFICATION_SERVICE);
notificationManager.notify(RELEASE_NOTIFICATION, releaseBuilder.build());
```

Listing : Erstellen einer Notification

# Hinweise

## Methode *build()*

Seit API-Version 16 ist die Methode *getNotification()* der Klasse Notification.Builder als *deprecated* gekennzeichnet. Man sollte daher wenn möglich auf die Methode *build()* zurückgreifen, die allerdings in älteren Versionen nicht unterstützt wird.

## NotificationCompat.Builder

Seit API-Level 11 (Android Version 3.0) bietet Android die Klasse *TaskStackBuilder*, die sich um die Verwaltung des Back-Stacks bei der Navaigation kümmert. Die Navigation mit Hilfe des *Zurück*-Buttons ist nur innerhalb einer Applikation festgelegt, aber nicht der Wechsel zwischen Applikationen.

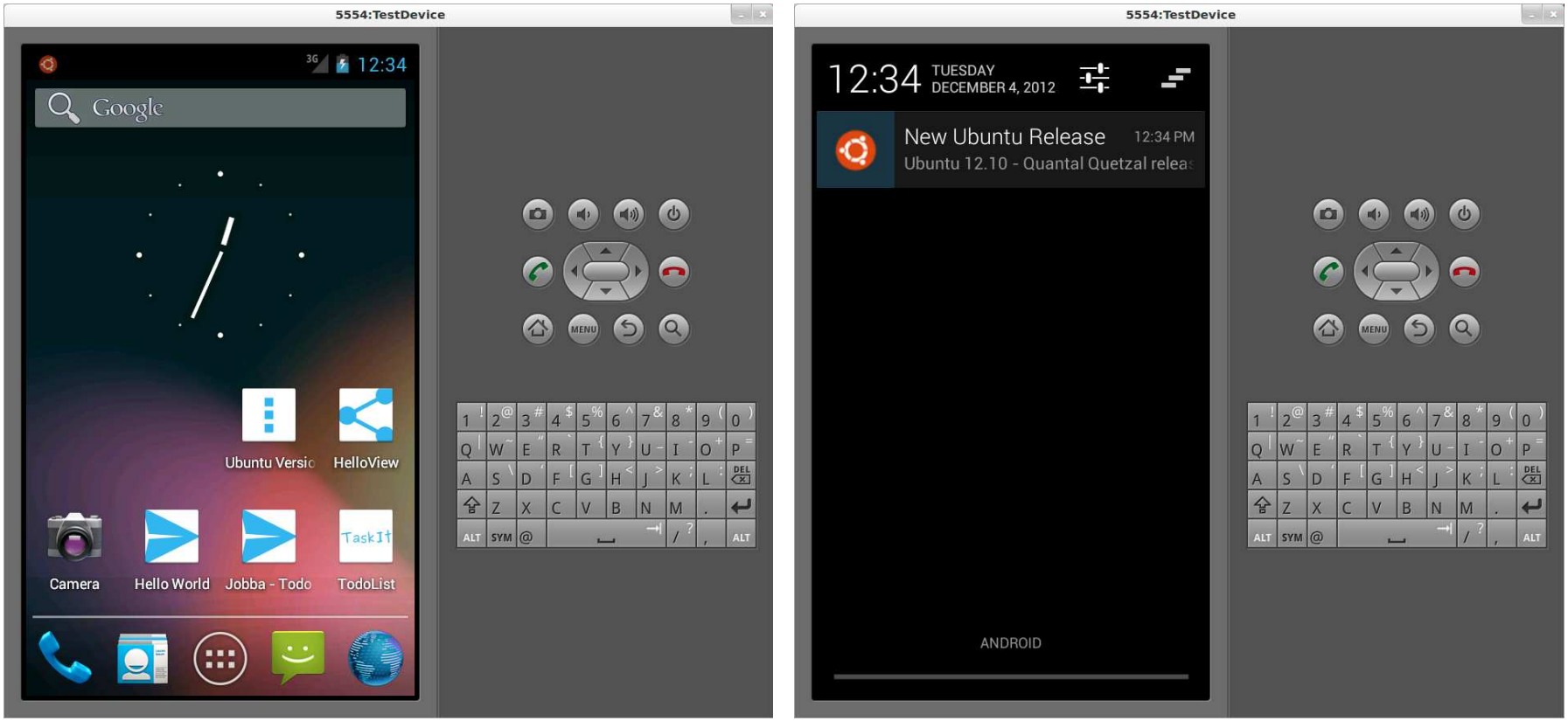
Der *TaskStackBuilder* ist auch kompatibel zu älteren Versionen von Android. So wird seit Android 3.0 beim Aufruf von *startActivities()* oder *getPendingIntent(int, int)* ein seperater Back-Stack angelegt, während in älteren Versionen nur die oberste Aktivität des übergebenen Stacks ausgeführt und so die Navigation zur aufrufenden Applikation ermöglicht wird.

# Einbinden des *TaskStackBuilders*

```
Notification.Builder releaseBuilder = new Notification.Builder(this)
    .setSmallIcon(R.drawable.ubuntu)
    .setContentTitle("New Ubuntu Release")
    .setContentText("Ubuntu 12.10 – Quantal Quetzal released!");

// Create an explicit intent to start the Ubuntu Release List activity.
Intent ubuntuIntent = new Intent(this, HelloList.class);

// Setup the back stack for the activity used by back navigation.
TaskStackBuilder stackBuilder = TaskStackBuilder.create(this);
stackBuilder.addParentStack(HelloList.class);
stackBuilder.addNextIntent(ubuntuIntent);

// Setup the pending intent to start the Ubuntu Release List activity.
PendingIntent pendingIntent = stackBuilder.getPendingIntent(
    0, PendingIntent.FLAG_UPDATE_CURRENT);
releaseBuilder.setContentIntent(pendingIntent);

// Get the notification manager and show the notification.
NotificationManager notificationManager = (NotificationManager) getSystemService(Context.NOTIFICATION_SERVICE);
notificationManager.notify(RELEASE_NOTIFICATION, releaseBuilder.build());
```

Listing : Einbinden des TaskStackBuilders

# Screenshot




(a) Icon in NotificationBar

(b) Eintrag in Notification-DropDown

Abbildung: Notifications im Einsatz

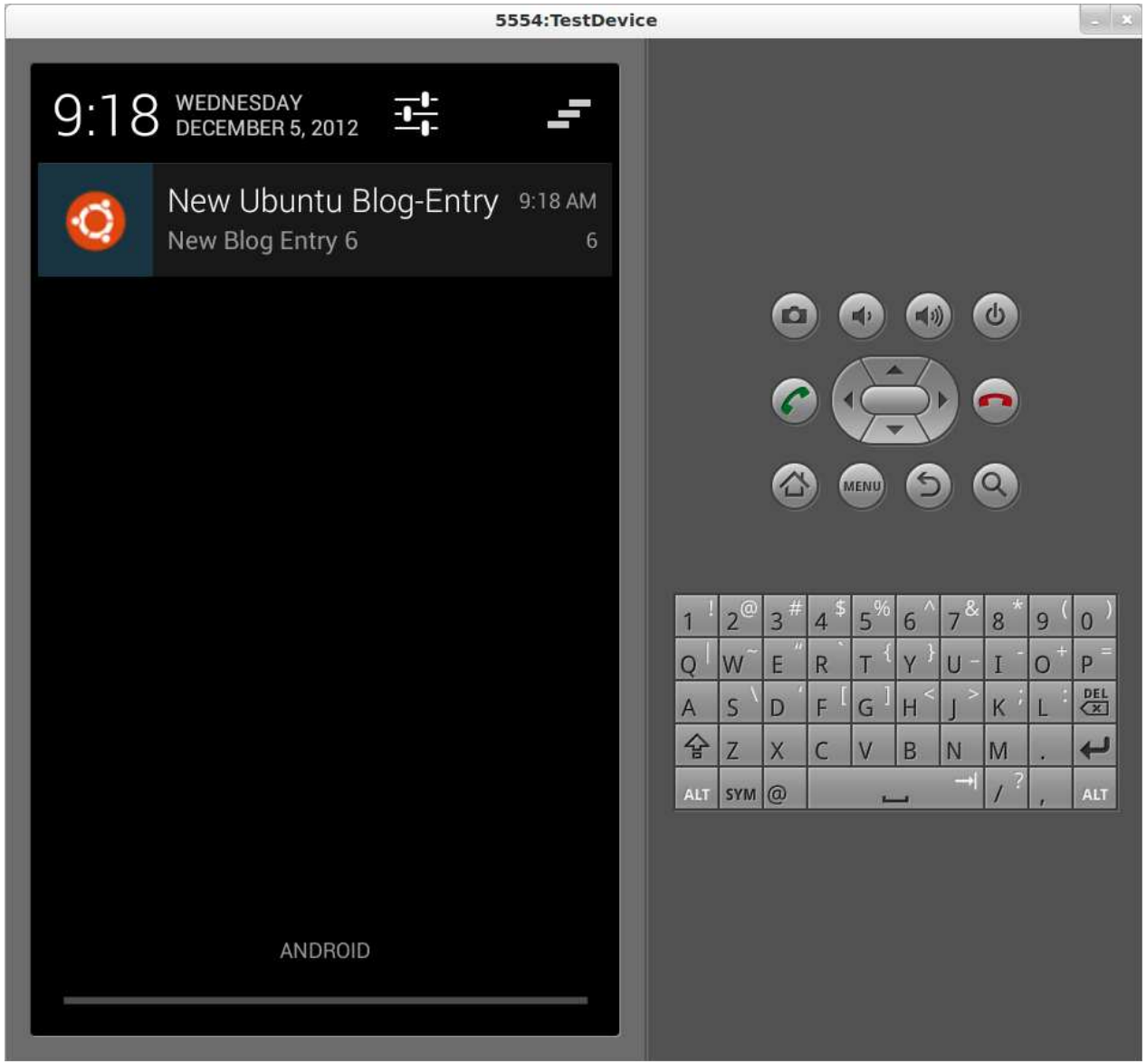
# Allgemeines

- Benachrichtigung des Anwenders in kurzen Zeitabständen
- Aktualisierung von Notifications entlastet System und Anwender
- Bei Übergabe an das System mit *NotificationManager* ID angeben
- System aktualisiert automatisch Notifications mit selber ID

```
// Get the notification manager.
NotificationManager notificationManager = (NotificationManager) getSystemService(Context.NOTIFICATION_SERVICE);

// Initialize the notification builder and assign the
// important notification data.
Notification.Builder entryBuilder = new Notification.Builder(this)
    .setSmallIcon(R.drawable.ubuntu)
    .setContentTitle("New Ubuntu Blog-Entry")
    .setContentText("Ubuntu 12.10 - Quantal Quetzal released!")
    .setNumber(1);

// Create an explicit intent and pending intent to start the activity.
...

// Finally show the notification and pass a notification ID.
notificationManager.notify(RELEASE_NOTIFICATION, entryBuilder.build());

// Increase the number of new blog entries in a loop.
for(int num = 2; num < 7; num++) {
    // Update the notification ...
    entryBuilder.setContentText("New Blog Entry " + num)
        .setNumber(num);

    // ... and show it.
    notificationManager.notify(RELEASE_NOTIFICATION, entryBuilder.build());
}
```

Listing : Notifications ändern

# Screenshot


5554:TestDevice
9:18
WEDNESDAY
DECEMBER 5, 2012
New Ubuntu Blog-Entry
9:18 AM
New Blog Entry 6
6
ANDROID
MENU

# Entfernen von Notifications

- Notifications werden nur durch Benutzeraktion gelöscht
- Alternativ kann Notification beim anklicken gelöscht werden (*setAutoCancel()*)
- Löschen durch NotificationManager mit *cancelAll()* oder *cancel(ID)*

# Allgemeines

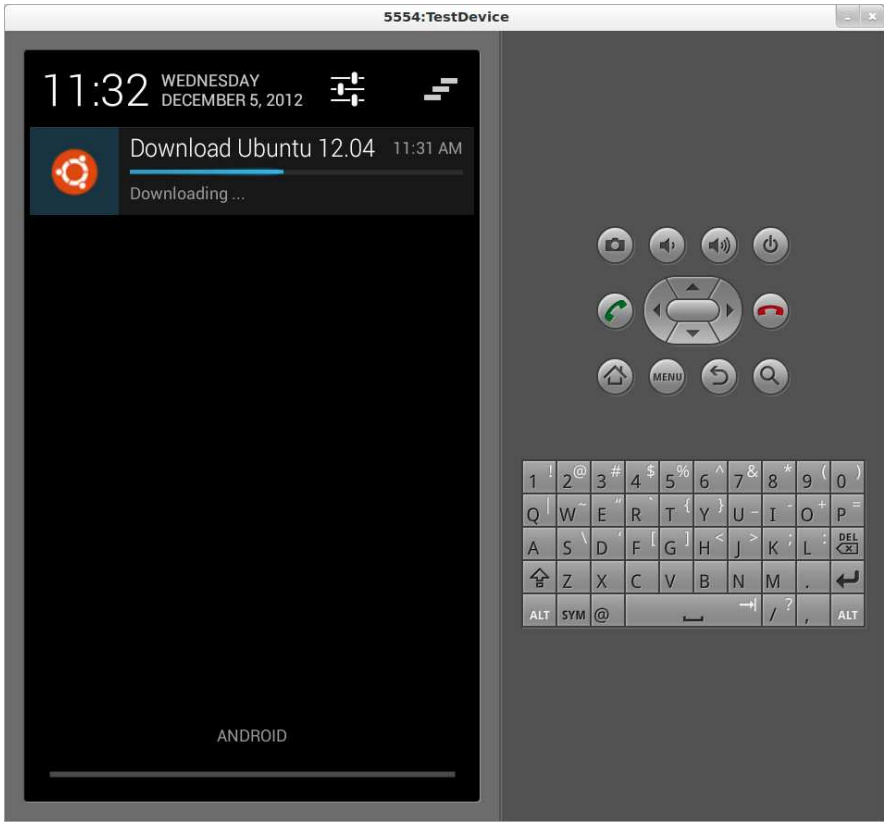
- Anzeige einer ProgressBar in Notification
- Verwendung der View-Klasse ProgressBar
- Unterscheidung in deterministische und nicht deterministische ProgressBar
- Bis Android 4.0 Einbindung über eigenes Layout
- Seit Android 4.0 Einbindung mit *setProgress()*
- Methode *setProgress()* aktualisiert auch den Zustand

# Implementierung

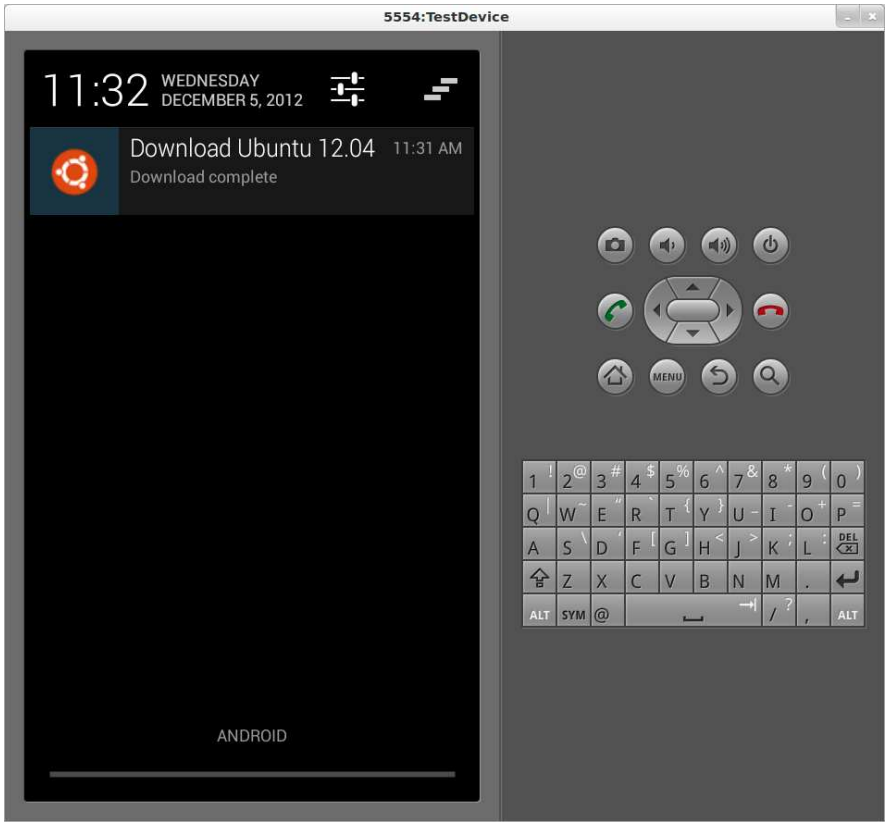
```
// Initialize the notification builder and assign the
// important notification data.
final Notification.Builder downloadBuilder = new Notification.Builder(this)
   .setSmallIcon(R.drawable.ubuntu)
   .setContentTitle("Download Ubuntu 12.04")
   .setContentText("Downloading ...");

// Get the notification manager and show the notification.
final NotificationManager notificationManager = (NotificationManager) getSystemService(Context.
      NOTIFICATION_SERVICE);

new Thread(new Runnable() {
   public void run() {
      for(int stat = 0; stat < 100; stat++) {
         try {
            downloadBuilder.setProgress(100, stat, false);
            notificationManager.notify(DOWNLOAD_NOTIFICATION, downloadBuilder.build());
            Thread.sleep(5000);
         } catch (InterruptedException e) {
            e.printStackTrace();
         }
      }

      // When the download finished, update the notification.
      downloadBuilder.setContentText("Download complete")
         .setProgress(0, 0, false);
      notificationManager.notify(DOWNLOAD_NOTIFICATION, downloadBuilder.build());
   }
}).start();
```

Listing : Deterministische ProgressBar in Notifications

# Screenshot




(a) ProgressBar während des Downloads




(b) ProgressBar nach dem Download

Abbildung: Eine determistische ProgressBar

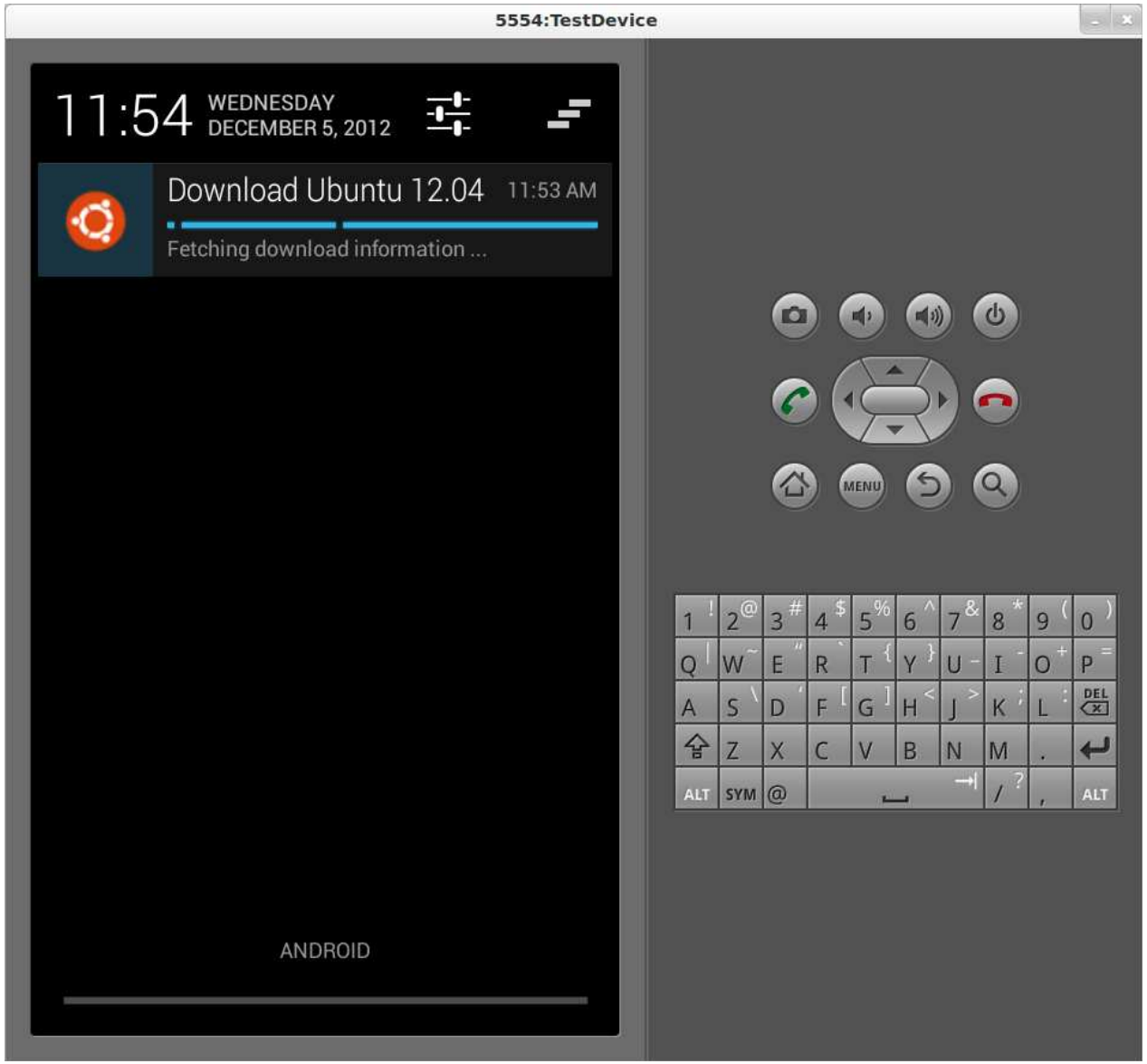
# Nicht deterministische ProgressBar

- Dritter Parameter von *setProgress* ist *true*
- Erster und zweiter Parameter werden ignoriert

```
// Initialize the notification builder and assign the
// important notification data.
Notification.Builder downloadBuilder = new Notification.Builder(this)
    .setSmallIcon(R.drawable.ubuntu)
    .setContentTitle("Download Ubuntu 12.04")
    .setContentText("Fetching download information ...")
    .setProgress(0, 0, true);

// Get the notification manager and show the notification.
NotificationManager notificationManager = (NotificationManager) getSystemService(Context.NOTIFICATION_SERVICE);
notificationManager.notify(DOWNLOAD_NOTIFICATION, downloadBuilder.build());
```

Listing : Nicht deterministische ProgressBar in Notifications

# Screenshot II



# Hinweis

## Eigene Layouts

In älteren Android Versionen musste eine ProgressBar über ein eigenes Layout eingefügt werden. Natürlich ist es auch in neueren Versionen von Android möglich eigene Layouts zu deklarieren. Dabei ist allerdings darauf zu achten, dass die normale Ansicht nur 64dp und die erweiterte Ansicht 256dp hoch ist.

Deklaration eines eigenen Layouts erfolgt in XML. Das Layout wird als RemoteView geladen und mit *setContent()* der Klasse *Notification.Builder* bzw. *NotificationCompat.Builder* zugewiesen. Dabei sollten Hintergundbilder vermieden werden, da sonst die Schrift unlesbar werden könnte. Außerdem sollte auf komplexe Layouts verzichtet werden. Der Einsatz von System-Resourcen ist sehr zu empfehlen.