TU Clausthal

Informatikwerkstatt

Android – Konzepte:
Intents, Threads, NFC

11.12.2018

Bildquelle: https://developer.android.com/images/brand/Android_Robot_200.png?hl=de

Rico Jimenez
TU-Clausthal

Literaturhinweis

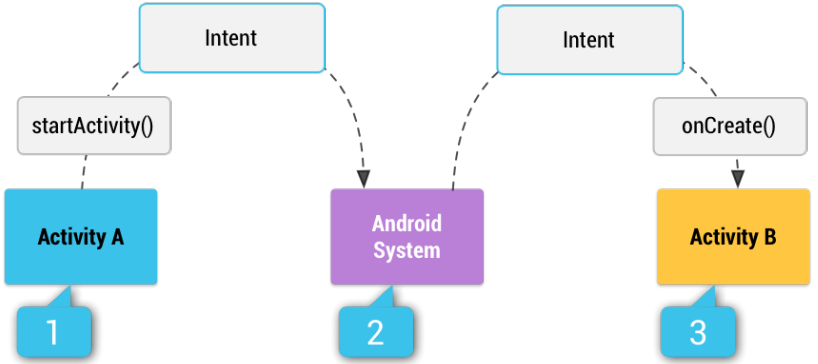
- Folien und Inhalte beruhen teilweise auf
  - Folien von Dr.-Ing. A. Reinhardt, TU-Clausthal, 2016
  - Folien von Dr.-Ing. A. Reinhardt, TU-Clausthal, 2017

Inhalt der heutigen Vorlesung

- **Intents und ihr Nutzen**
  - Was ist ein Intent?
  - Welche Arten gibt es?
  - Wofür und wie werden Intents genutzt?
  - Wie werden Intents zugeordnet?
- **Prozesse und Threads**
  - Was sind Prozesse?
  - Wofür werden Threads benötigt?
  - Was ist ein AsyncTask?
- **NFC – Near Field Communication**
  - Was ist NFC/RFID?
  - Wie kann NFC unter Android benutzt werden?

Intent

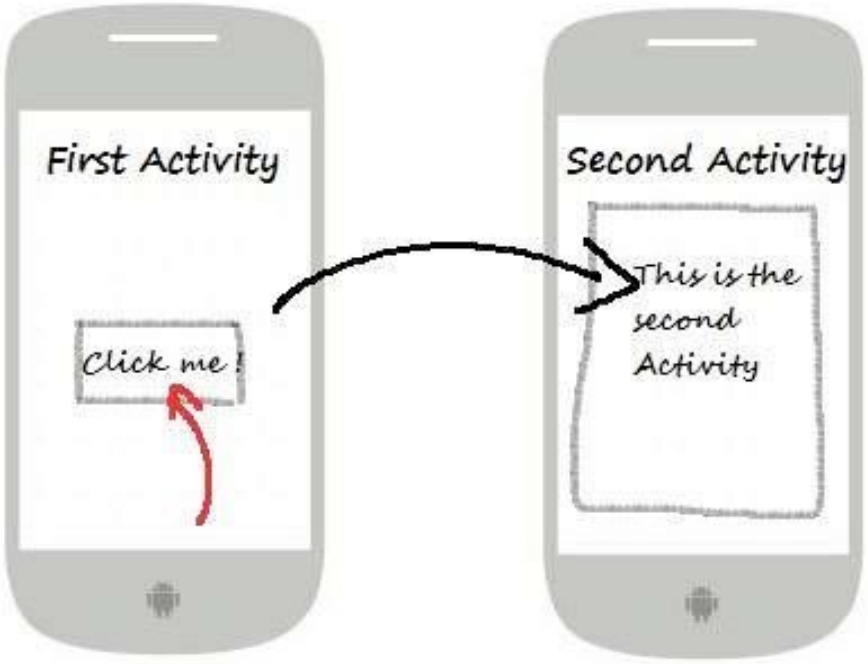
- Asynchrone Nachricht, die man verwenden kann, um Aktionen von anderen App-Komponenten zu beziehen, oder andere App-Komponenten über Ereignisse zu informieren
- Über Intents kann man bspw. innerhalb einer App von einer Activity zu einer anderen Activity springen“



Bildquelle: https://developer.android.com/images/components/intent-filters@2x.png

Intent

- Vergleichbar mit einem Kellner
  - Kellner regelt Zugriff auf Essen / Getränke
  - System regelt mittels Intents Zugriff auf
    - **Activities**
    - BroadcastReceiver
    - ContentProvider
    - Services

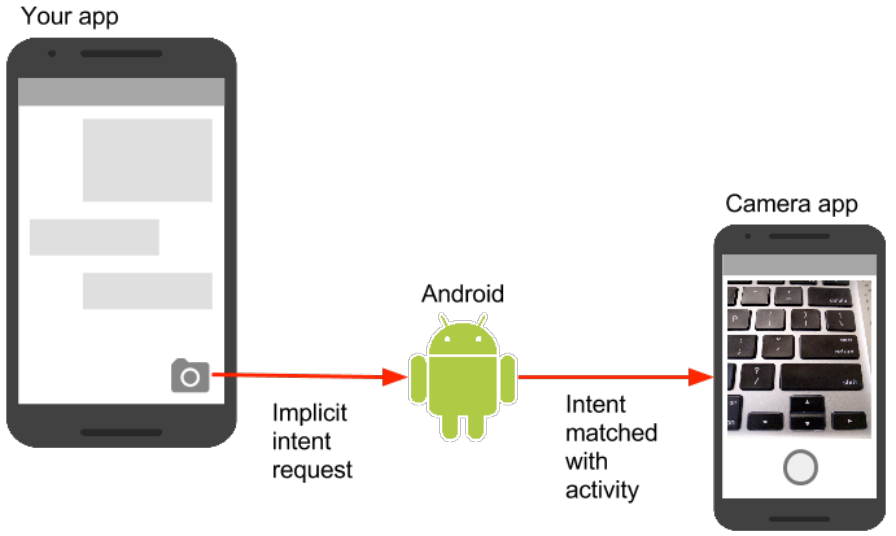


Bildquelle: https://o7planning.org/de/10425/cache/images/i/8274465.jpeg

Intent: Zwei Arten der Nutzung

- Explizite Intents benennen die Klasse, die sie aufrufen
  - Meist in der eigenen App
  - Beispielsweise eine konkrete Activity
- Implizite Intents benennen, was zu tun ist
  - Keine konkrete Klasse
  - Aufruf anderer Apps

Intent – Beispiel (1)



Bildquelle: https://google-developer-training.gitbooks.io/android-developer-fundamentals-course-concepts/content/en/images/2_3_C_images/implicit-intent.png

- Es gibt zwei Apps
  - Eine App möchte etwas senden
    - *ACTION_SEND*
  - Die andere App kann als Messenger Text und Bilder verarbeiten
- Es wird ein Impliziter Intent ausgelöst, der zuerst aufgelöst und dann bearbeitet wird

Impliziter Intent – Beispiel (1)

- Impliziten Intent starten

```
Intent sendIntent = new Intent();
sendIntent.setAction(Intent.ACTION_SEND);
sendIntent.putExtra(Intent.EXTRA_TEXT, "Message");
sendIntent.setType("text/plain");

// Prüfung, ob es einen passenden Empfänger gibt

if (sendIntent.resolveActivity(
       getPackageManager()) != null ) {
          startActivity(sendIntent);
}
```

Impliziter Intent – Beispiel (1)

- Manifest der empfangenden App für die Activity

```
<activity android:name="ShareActivity">
 <intent-filter>
  <action android:name="android.intent.action.SEND"/>
  <category
    android:name="android.intent.category.DEFAULT"/>
  <data android:mimeType="text/plain"/>
  <data android:mimeType="image/*"/>
 </intent-filter>
</activity>
```

Impliziter Intent – Beispiel (1)

- Bearbeitung des Impliziten Intent in der Activity:

```
@Override
protected void onCreate(Bundle savedInstanceState) {
  super.onCreate(savedInstanceState);
  setContentView(R.layout.main);
  // Intent abfragen
  Intent intent = getIntent();
  Uri data = intent.getData();
  // Nach Datentyp das Handling anpassen
  if (intent.getType().indexOf("image/") != -1) {
       // Bilder weiterleiten
  } else if (intent.getType().equals("text/plain")) {
       // text verarbeiten und senden...
  }
}
```

Expliziter Intent – Beispiel (2)

- Expliziten Intent starten

```
String intentText = "New Activity";
Intent meinIntent =
  new Intent(MainActivity.this, Main2Activity.class);
meinIntent.putExtra("NEXTACTIVITY",intentText);
// zur Activity die im Intent benannt wurde wechseln
startActivity(meinIntent);
```

Expliziter Intent – Beispiel (2)

- Bearbeitung des Expliziten Intent in der Activity:

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main2);

    Intent aufruf = getIntent();
    String intentText = "";
    if (aufruf.getExtras() != null) {
     intentText =
       aufruf.getExtras().get("NEXTACTIVITY").toString();
    }
}
```

Intent

- Attribute eines Intent
  - Name (optional)
  - Action
    - `ACTION_VIEW` – Datei zeigen / wiedergeben
    - `ACTION_PICK` – Zum Beispiel einen Kontakt auswählen
    - `ACTION_DIAL` – Nummer wählen
  - Daten, die verarbeitet werden sollen
    - `ACTION_VIEW content://contacts/people/1`
    - `ACTION_DIAL content://contacts/people/1`

Intent

- Zusätzlich ist es nützlich, eine Kategorie zu setzen
- `Category` schränkt die kompatiblen Apps ein
  - `Action` ist die erste Einschränkung
  - `CATEGORY_LAUNCHER` als Activity im System Launcher aufgeführt
- Flags
  - Kontrollieren, wie der Intent behandelt wird
- Extras
  - Schlüssel-Wert-Paare werden als `Bundle` gespeichert
  - Es gibt vordefinierte Extras
    - `intent.putExtra(Intent.EXTRA_SUBJECT, "Betreff");`
    - `EXTRA_TITLE` setzt Titel der Actionauswahl

Intent

- Wie kann ein impliziter Intent zugeordnet werden?
  - Das System sucht eine passende App automatisch
  - Gibt es mehrere Apps, kann der Nutzer eine auswählen
  - Es ist möglich, eine Standard App bei der Auswahl auszuwählen
- Wonach beurteilt das System, ob eine App die passende ist?
  - Apps legen selber *Intent Filter* fest

Intent Filter

- Intent Filter beschreiben mögliche Aufgaben
  - Eine Activity kann mehrere Intent Filter beinhalten
- Sie werden im Manifest unter dem Bereich der Activity aufgelistet
- Filter-Möglichkeiten
  - Action
  - Data
  - Category
- Mehrere Elemente des gleichen Typs sind möglich

Intent Filter – Beispiel (ACTION_SEND)

```
<activity android:name=".MainActivity">
    <intent-filter>
        <action android:name="android.intent.action.MAIN" />
        <category android:name="android.intent.category.LAUNCHER" />
    </intent-filter>
</activity>
```

```
<activity android:name="ShareActivity">
    <intent-filter>
        <action android:name="android.intent.action.SEND"/>
        <category android:name="android.intent.category.DEFAULT"/>
        <data android:mimeType="text/plain"/>
    </intent-filter>
</activity>
```

Intent resolution

- Das System übermittelt der App Komponente nur dann ein Intent, wenn der Intent mit dem Filter
  - bei mindestens einem action-Element übereinstimmt, oder
  - alle category-Elemente dort vorkommen, oder
  - die data-Elemente dort vorkommen
    - Komplexere Prüfung

@LET‘S TRY

- Erstellen Sie eine weitere Activity in Ihrem „Hello World“- Projekt
- Fügen Sie einen Button hinzu, der mit Hilfe eines **expliziten Intents** auf die neue Activity leitet
- Übergeben Sie in dem Intent Ihren Namen, der in einer TextView eingegeben wird
- Geben Sie den Namen in der neuen Activity in einer TextView aus. Machen Sie kenntlich das der Name aus der MainActivity stammt.
  - Beispiel: „<Ihr_Name> hat mich übermittelt“

Prozesse und Threads

- Prozesse sind die Ablaufumgebung eines Programms
  - Eine Verwaltungseinheit im Betriebssystem
  - Jede Anwendung in Android hat ihren eigenen Prozess
  - Prozesse haben Ressourcen (Adressbereich, Programmcode)
- Im Allgemeinen hat eine Applikation nur einen Prozess
- Threads bearbeiten eine sequentielle Teilaufgabe innerhalb eines Prozesses
  - Mehrere Teilaufgaben können in mehreren Threads nebenläufig (= parallel) innerhalb eines Prozess abgearbeitet werden

Threads

- Wann werden weitere Threads nötig?
  - Hauptthread (UI Thread) verwaltet Oberflächenereignisse
    - Reaktionen auf Benutzereingaben
  - Lange laufende Berechnungen stoppen UI-Aktualisierung
  - Programm „wirkt aufgehängt“
  - Nach derzeit 5 Sekunden `ActivityNotResponding`-Fehler
- Potentiell lang laufende Aktivitäten sollten in *Arbeiterthreads* ausgelagert werden, z.B.
  - Netzwerkaktivitäten wie Abruf von Informationen einer Webseite
  - Lange Rechenoperationen

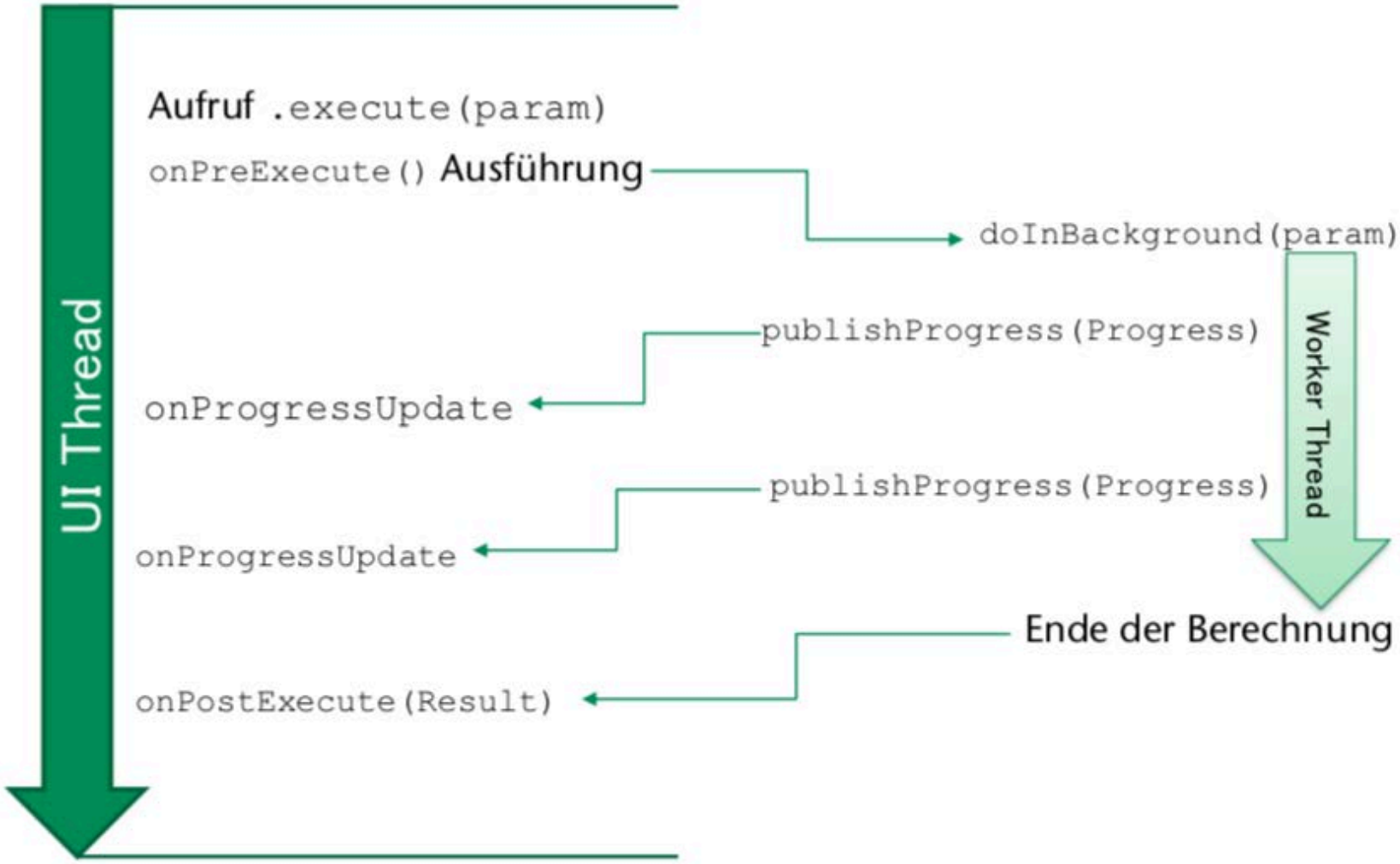
Threads

- Zugriff auf UI aber **nur** aus UI Thread
  - Sonst gibt es eine Fehlermeldung
- Android bietet mehrere Möglichkeiten dafür, Threads zu nutzen
  - Thread anpassen und starten
    - Üblicherweise mit Runnable und Handler
  - Klasse mit Oberklasse `AsyncTask` nutzen

AsyncTask

- Androids Klasse für Hintergrundthreads
- Schritte eines AsyncTask (entsprechen den zugehörigen Methoden)
  - `onPreExecute()`
    - im UI Thread, Task vorbereiten, z.B. Dialog einblenden
  - `doInBackground(Params...)`
    - im Hintergrund Thread ausführen direkt nach `onPreExecute()`
  - `onProgressUpdate(Progress...)`
    - Im UI Thread, wenn z.B. in `doInBackground()` `publishProgress(Progress...)` aufgerufen wurde
  - `onPostExecute(Result)`
    - Im UI Thread ausführen nach Ende der Hintergrundberechnung

AsyncTask



Bildquelle: Dr.-Ing. A. Reinhardt, TU-Clausthal, 2016

AsyncTask

- Um von AsyncTask zu erben, müssen stets Datentypen für drei Parameter angegeben werden
  - `Params` – Datentyp der Eingabeparameter
  - `Progress` – Datentyp der Fortschrittseinheit, für die Ausgabe von Statusangaben
  - `Result` – Datentyp des Ergebnisses
  - Wird ein Typ nicht gebraucht, wird `void` angegeben
- `doInBackground(Params...)` muss überschrieben werden

AsyncTask - Beispiel

```
 // Innere Klasse HoleDatenTask führt den asynchronen Task auf eigenem Arbeitsthread aus

public class HoleDatenTask extends AsyncTask<String, Integer, String[]> {
        @Override
        protected String[] doInBackground(String... strings) {
            String[] ergebnisArray = new String[20];
            for (int i=0; i < 20; i++) {
                // Den StringArray füllen wir mit Beispieldaten
                ergebnisArray[i] = strings[0] + "_" + (i+1);
                if (i%5 == 4) { // Alle 5 Elemente geben wir den aktuellen Fortschritt bekannt
                    publishProgress(i+1, 20);
                }
            return ergebnisArray;
        }
        @Override
        protected void onProgressUpdate(Integer... values) {
            // Auf dem Bildschirm geben wir eine Statusmeldung aus, immer wenn
            // publishProgress(int...) in doInBackground(String...) aufgerufen wird
            Toast.makeText(getActivity(), values[0] + " von " + values[1] + " geladen",
                    Toast.LENGTH_SHORT).show();
        }
        @Override
        protected void onPostExecute(String[] strings) {
        // Hintergrundberechnungen sind jetzt beendet, darüber informieren wir den Benutzer
            Toast.makeText(getActivity(), "Aktiendaten vollständig geladen!",
                    Toast.LENGTH_SHORT).show();
        }
    }
}
```

Beispiel: Nutzung von Thread für Timer

- Soll nur ein Ereignis nach einer gewissen Zeit ausgelöst werden, ist dies einfach mit einem Thread umzusetzen
- Hierzu werden ein `Runnable` und ein `Handler` benötigt
  - `Runnable` befähigt einen Thread zu starten
  - `Handler` kümmert sich um Nachrichten in dem UI Thread
    - Ermöglicht Änderungen am User Interface zur Laufzeit

Beispiel: Timer

```
import android.os.Handler; //Auf richtigen Import achten!
//...
final Handler handler = new Handler();
final Runnable r = new Runnable(){
public void run() {
//Ihr Code was zu tun ist
//Hier können auch UI Elemente beeinflusst werden
 }
};
handler.postDelayed(r,1500);//führt den Code in run() nach 1500ms aus
```

- Muss in der UI Klasse (hier: MainActivity) sein
- Muss im Lebenszyklus aufgeräumt werden!
  - Spätestens in `onDestroy()`
  - Aufruf stoppen mit `handler.removeCallbacks(r)`

NFC – Near Field Communication

- Drahtlose Funk-Kommunikation für kurze Entfernungen
  - RFID: Lesen und Schreiben eines kleinen Speichers
    - z.B. Studierendenausweis
  - NFC: Gerät-zu-Gerät Kommunikation zum Datenaustausch
- Erster Schritt:
  - Im Manifest den Zugriff freigeben (vor <**application...**)

```
<uses-permission android:name="android.permission.NFC" />
```

NFC – Near Field Communication

- Zweiter Schritt:
  - Eine abstrakte Klasse `NfcActivity` zu Ihrem Package hinzufügen, die `Activity.java` erweitert (`NfcActivity` siehe folgende Folie):
  - Dann ihre Klasse als Unterklasse definieren:
    - `public class MainActivity extends NfcActivity`
- Dritter Schritt:
  - `nfcRead(Intent i){}` überschreiben
    - Hinweis: `@Override` verwenden

NfcActivity.java

```
public abstract class NfcActivity extends Activity {
    PendingIntent pendingIntent;
    NfcAdapter nfcAdapter;

    @Override
    public void onResume() {
        super.onResume();
        pendingIntent = PendingIntent.getActivity(this, 0, new Intent(this,
getClass()).addFlags(Intent.FLAG_ACTIVITY_SINGLE_TOP), 0);
        nfcAdapter = NfcAdapter.getDefaultAdapter(this);
        nfcAdapter.enableForegroundDispatch(this, pendingIntent, null, null);
    }

    @Override
    protected void onPause() {
        super.onPause();
        nfcAdapter.disableForegroundDispatch(this);
    }

    @Override
    public void onNewIntent(Intent intent) {
        String action = intent.getAction();
        if(NfcAdapter.ACTION_TAG_DISCOVERED.equals(action))
            nfcRead(intent);
    }

    public abstract void nfcRead(Intent intent);
}
```

Auswerten des NFC-/RFID-Ereignisses

- Beim Erkennen einer NFC-/RFID-Karte wird von der abstrakten Klasse nun stets die Methode `NfcRead(Intent i)` aufgerufen
  - Über den Inhalt des Intents können Sie nun auf die Daten des Tags zugreifen
  - Hierzu wird der folgende Code hinzugefügt
    - `Tag tag = i.getParcelableExtra(NfcAdapter.EXTRA_TAG);`
    - `byte[] tagID = tag.getId();`
  - Die eindeutige Identifikationsnummer des Tags ist nun in einem Array gespeichert

Auswerten des NFC-/RFID-Ereignisses

- Mit folgender Methode lässt sich der Byte Array in einem String umwandeln

```
public static String byteArrayToHex(byte[] a) {
  StringBuilder sb = new StringBuilder(a.length * 2);
  for(byte b: a)
      sb.append(String.format("%02x",b & 0xff));
  return sb.toString();
}
```

- Mit dieser Methode lässt sich nun die ID als String auslesen
  - `String tagIdString = byteArrayToHex(tagID);`

@LET‘S TRY

- Führen Sie auf den vorigen Folien („NFC/RFID“) beschriebenen Schritte durch
- Lassen Sie die ID Ihres Studierendenausweis durch einen Toast mit Hilfe der `byteArrayToHex()` *Methode ausgeben*
- Halten Sie zur Probe Ihren Studierendenausweis an die Rückseite Ihres Tablets

Quellen

- http://www.programmierenlernenhq.de/tutorial-android-prozesse-threads-und-asynctask/
- https://developer.android.com/guide/topics/connectivity/nfc/
- https://developer.android.com/reference/android/content/Intent
- https://developer.android.com/reference/android/os/AsyncTask
- https://developer.android.com/guide/components/processes-and-threads
- https://developer.android.com/guide/topics/manifest/intent-filter-element