TU Clausthal

Informatikwerkstatt

Android – Diverse Konzepte: SQLite, Room, Sensoren

19.12.2018

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

Rico Jimenez
TU-Clausthal

Literaturhinweis

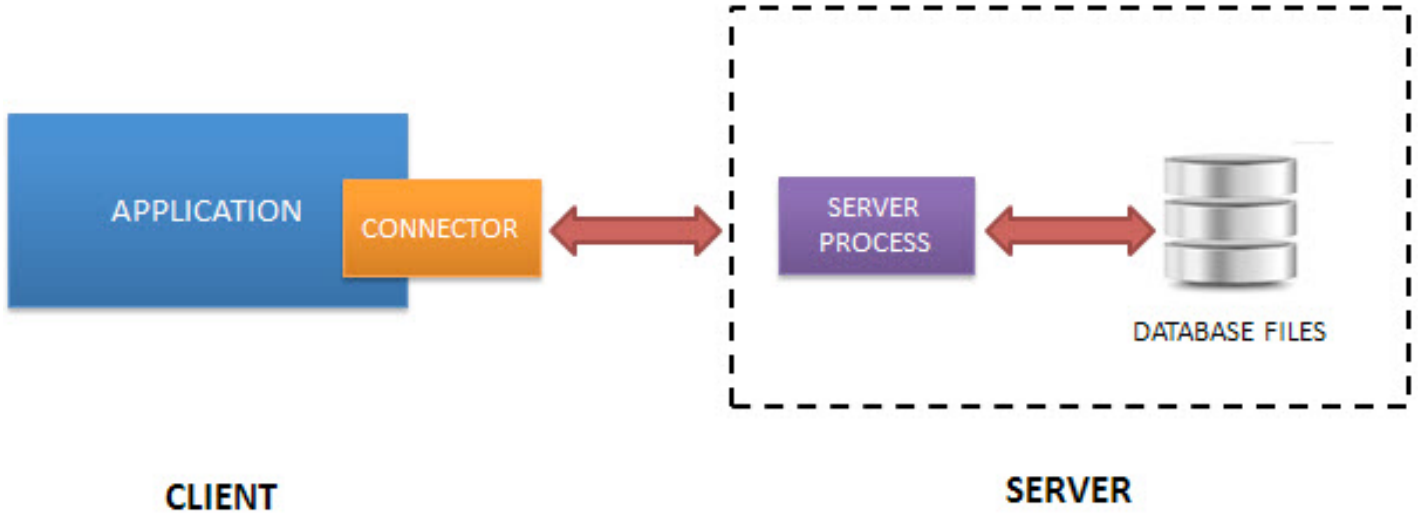
- Folien und Inhalte beruhen teilweise auf
  - Folien von Dr.-Ing. A. Reinhardt, TU-Clausthal, 2017

Inhalt der heutigen Vorlesung

- Datenbank
  - Wofür benötige ich eine Datenbank?
- SQLite mit Room
  - SQLite unter Android
  - Nutzung der Room-Annotationen
- Sensoren
  - Sensortypen in Android
  - Sensor, SensorManager, SensorEvent
  - Zugriff auf Sensoren

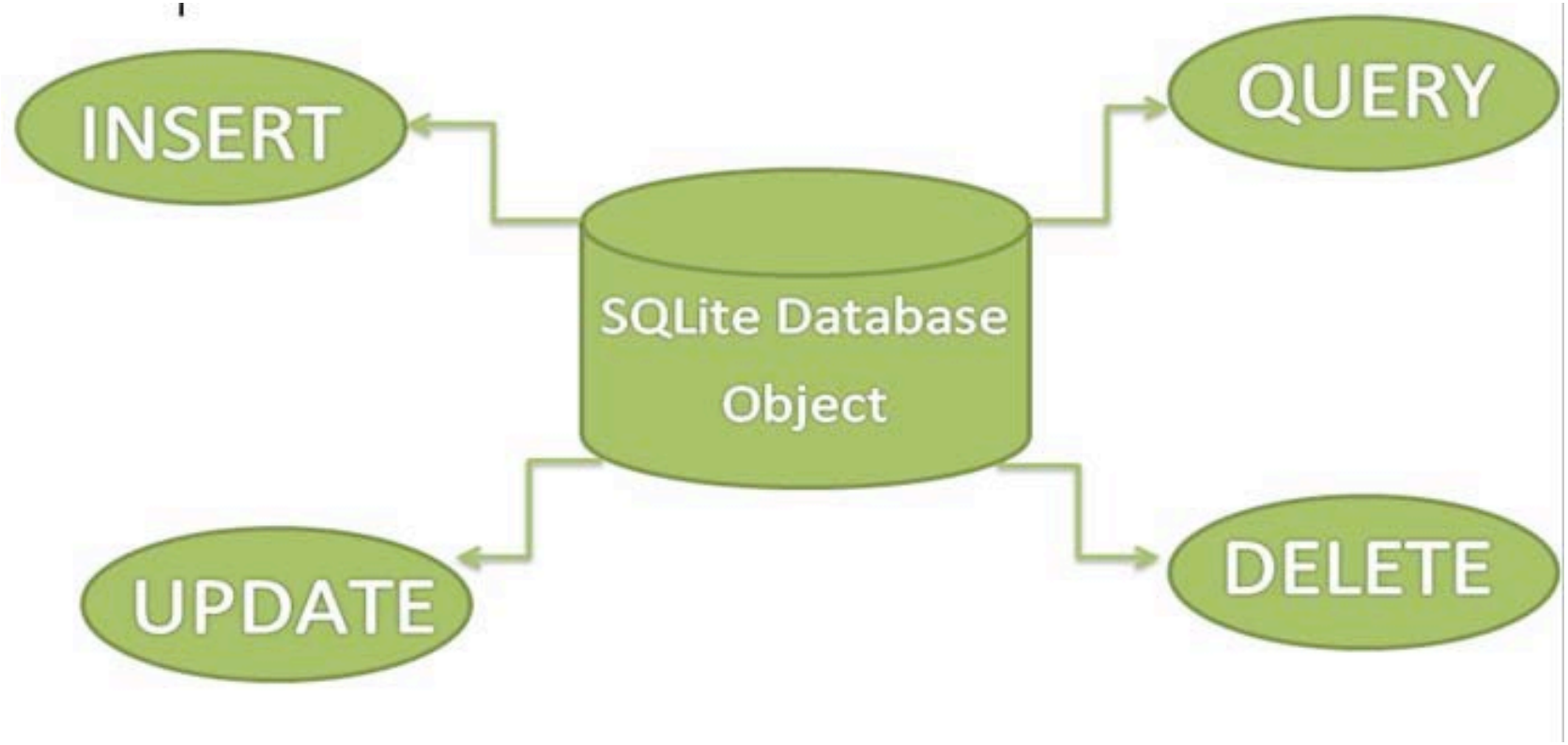
Datenbank

- Wofür benötige ich eine Datenbank?
  - Sammlung von Informationen
  - Aktualisierung von bestehenden Informationen
  - Große Datenmengen effizient und widerspruchsfrei aufarbeiten



Bildquelle: http://www.sqlitetutorial.net/wp-content/uploads/2015/12/RDBMS-Client-Server-Architecture.jpg

SQLite



Bildquelle: https://abhiandroid.com/database/wp-content/uploads/2017/03/SQLite-Database-Operations.jpg

SQLite mit Room

- Android nutzt zum das Datenbanksystem SQLite
- *Room* bietet eine Abstraktionsschicht über SQLite
  - stabiler Datenbankzugriff
  - volle Leistungsfähigkeit von SQLite nutzbar
- Mit *Room* kann sehr einfach über Annotationen ein Zugriff auf eine Datenbank (DB) erfolgen
- Um in einer DB Daten zu speichern oder Daten abzufragen, verwendet man *Data Access Objects* (DAOs)
  - Diese sind Interfaces mit `@Dao` annotiert, deren Methoden Abfragen darstellen und passende Annotationen erhalten

Vorbereitung von Room

- Es werden dazu benötigt:
  - Eine Klasse, die als Datenbank dient
  - DAO Klassen zum Abruf der Daten
  - Klassen, die die Entitäten (Tabellen) darstellen
- Der Zugriff auf *Room* muss eingerichtet werden
- Gradle-Abhängigkeiten müssen gesetzt werden
- Fügen Sie unter *Gradle Scripts* > *build.gradle* unter `dependencies` folgende zwei Zeilen ein

```
implementation
"android.arch.persistence.room:runtime:1.0.0"
annotationProcessor
"android.arch.persistence.room:compiler:1.0.0"
```

Entitäten anlegen

- In einer SQL-Datenbank sind Entitäten als Tabellen angelegt
- In der Objektorientierten Programmierung sind Entitäten üblicherweise Klassen mit auserwählten Eigenschaften
- Room erlaubt es Klassen so zu annotieren, dass sie einfach gespeichert werden können
- Hierzu müssen die zu speichernden Objekte öffentlich (public) sein, oder getter und setter im Java Beans Standard haben
- Für weitere Annotationen: https://developer.android.com/training/data-storage/room/defining-data.html

Entitäten anlegen

- Beispiel:

```
@Entity
public class User {
@PrimaryKey
public int id;
public String firstName;
public String lastName;
public int age;
}
```

- Ein Primärschlüssel wird zur eindeutigen Identifizierung eines Datensatzes verwendet
- `id` könnte ein eindeutig mitlaufender Integer als Klassenvariable sein
  - Weitere Möglichkeit: `UUID` von Java

Data Access Objects – DAO

- Durch die Klassenannotation sind die Tabellen definiert
- `@Insert` fügt alle übergebenen Einträge ein, gibt bei einem Eintrag ein `long` mit `rowId` zurück
  - bei mehreren entweder `long[]` oder `List<Long>`
- `@Update` überarbeitet bestehende Einträge
- `@Delete` löscht übergebene Einträge aus der Datenbank
- `@Query("<Query>")`definiert eine beliebige SQL-Abfrage
  - Fehler werden als Compilerfehler erkannt
  - Eine Liste der Queries ist unter folgendem Link zu finden

Data Access Objects – DAO

```
@Dao
public interface UserDao {
@Insert
void insertAll(User... users);
@Update
void updateOne(User user);
@Update
void updateMultiple(User... users);
@Delete
void delete(User user);
@Query("SELECT * FROM user")
List<User> getAll();
@Query("SELECT * FROM user WHERE id IN (:userIds)") List<User>
loadAllByIds(int[] userIds);

}
```

- https://developer.android.com/training/data-storage/room/accessing-data

Datenbank

- Sie muss eine öffentliche abstrakte Klasse sein
  - Die Klasse erbt von `RoomDatabase`
  - Sie enthält alle Entitäten als Annotation
- Für jede DAO gibt es eine abstrakte Klasse
  - Diese enthält eine parameterlose Methode, die das entspreche DAO zurückgibt

```
@Database(entities={User.class},version = 1)
public abstract class TestDatabase extends RoomDatabase {
public abstract UserDao userDao();

}
```

Datenbank – onCreate()

- Um Datenbank zu nutzen, muss im Programm eine Referenz erzeugt werden
- Hier wird der Name der Datenbankdatei (Testdatenbank) festgelegt
  - Der Name muss immer gleich bleiben

```
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);

     TestDatabase db =
    Room.databaseBuilder(getApplicationContext(),
       TestDatabase.class, "Testdatenbank").build();
    new DatabaseAsync().execute();
}
```

Datenbankzugriff

- Datenbankzugriff im MainThread erzeugt eine Exception
  - Der Zugriff könnte lange dauern
  - Zugriff muss daher asynchron geschehen
  - Hierzu können Sie beispielsweise wieder einen AsyncTask nutzen (Beispiel folgt auf nächster Folie)
- Alternativ kann der Zugriff über den MainThread erlaubt werden (nicht empfehlenswert)
  - `Room.databaseBuilder(this, TestDatabase.class, "Testdatenbank").allowMainThreadQueries().build();`

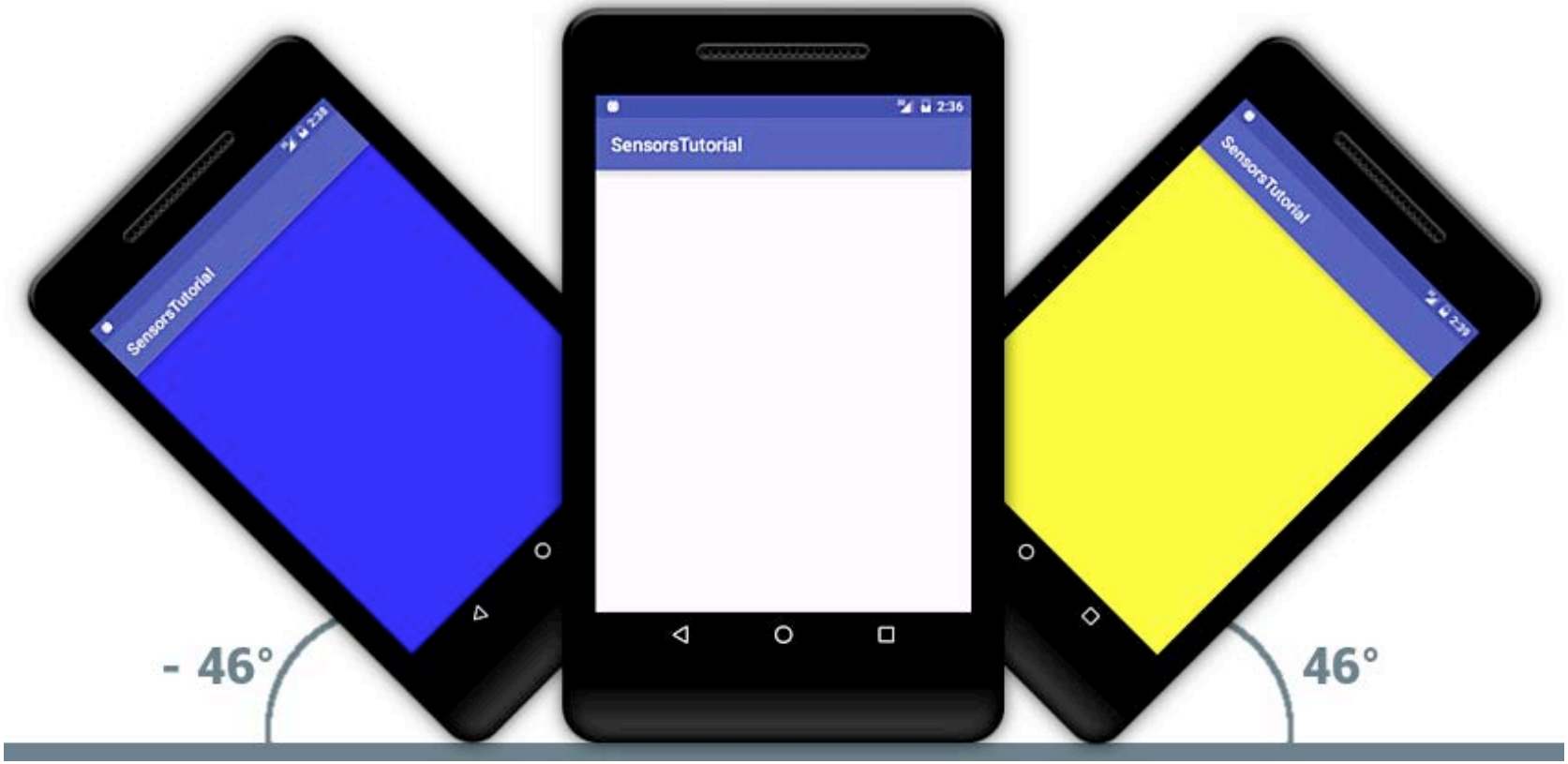
Beispiel - Datenbankzugriff

```
 private class DatabaseAsync extends AsyncTask<Void, Void, Void> {
            List<User> liste;
            @Override
            protected Void doInBackground(Void... params) {
                liste = db.userDao().getAll();
                return null;
            }
            @Override
            protected void onPostExecute(Void aVoid) {
                super.onPostExecute(aVoid);
                 TextView textfeld =
MainActivity.this.findViewById(R.id.textView);
                String text = "";
                for (User u: liste){
                    text += u.firstName;
                    text += " "+u.lastName+" \n";
                }
                    textfeld.setText(text);
            }
        }
}
```

@LET‘S TRY

- Implementieren Sie eine Datenbank in einem neuen Android Projekt
- Erstellen Sie die Klassen jeweils in einer eigenen Java Datei (TestDatabase.java, UserDao.java, User.java)
- Erstellen Sie in der Klasse *User getter* und *setter* Methoden

Sensoren



Bildquelle: https://cms-assets.tutsplus.com/uploads/users/362/posts/28084/image/q7.png

Sensoren

- Verschiedene Geräte bringen unterschiedliche Sensoren mit
- Android bietet für viele Arten von Sensoren die Möglichkeit diese auszulesen
- Die Sensoren sind in drei Typen unterteilt
  - Bewegung, Kräfte und Beschleunigungen(Gyroskop)
  - Positionen (Orientierung)
  - Umwelteigenschaften (Feuchtigkeit, Temperatur, Licht)
- Nicht alle Sensoren sind als Hardware vorhanden
  - Diese werden in der Software simuliert

Sensoren

- Zu den häufig genutzten und von Android unterstützen Sensoren gehören
  - Lichtsensor
  - Gyroscope
  - Rotation
  - Schrittzähler

- Vollständige Liste unter https://developer.android.com/guide/topics/sensors/sensors_overview

- Nicht jedes Gerät besitzt alle Sensoren, wir beschränken uns auf die oben genannten

Bildquelle: https://cms-assets.tutsplus.com/uploads/users/362/posts/28084/image/q4.png

Sensoren – Framework

- Ein *Framework* ist eine Software, die dem Programmierer den Entwicklungsrahmen für die Anwendungsprogrammierung zur Verfügung stellt
- Android stellt zum Zugriff das Sensor-*Framework* bereit, dieses enthält
  - `SensorManager` – Abrufen von Sensoren, Registrierung von Listenern für Sensoren
  - `Sensor` – Informationen zu den Sensoren
  - `SensorEventListener` – Änderungen der Sensorwerte und der Sensorgenauigkeit werden übermittelt

Sensoren – SensorManager

- Um einen Sensor auszulesen, wird eine Instanz des SensorManagers benötigt
  - `sensorManager = (SensorManager)getSystemService(Context.SENSOR_SERVICE);`
  - Der SensorManager wird in der onCreate() Methode initialisiert
  - Über den SensorManager ist es möglich Listener zu registrieren
- Daten von einem Sensor werden in einem SensorEvent übertragen
- Für die Registrierung der Listener muss SensorEventListener implementiert werden

Sensoren – `Sensor` (1/2)

- Um einen Sensor direkt anzusprechen wird der SensorManager verwendet
  - Der `Sensor` wird in der `onCreate()` Methode initialisiert

- Lichtsensor

```
lightSensor =
sensorManager.getDefaultSensor(Sensor.TYPE_LIGHT);
```

- Gyroskop
  -
```
gyroscopeSensor =
sensorManager.getDefaultSensor(
                        Sensor.TYPE_GYROSCOPE);
```

Sensoren – Sensor (2/2)

- Rotationssenor
  - `rotationSensor = sensorManager.getDefaultSensor(Sensor.TYPE_GAME_ROTATION_VECTOR);`
- Schrittzähler
  - `stepCounterSensor = sensorManager.getDefaultSensor(Sensor.TYPE_STEP_DETECTOR);`

Sensoren – `SensorEventListener`

- Das für den Listener zu implementierende Interface `SensorEventListener` enthält die Methoden:
  - `onSensorChanged(SensorEvent event)`
  - `onAccuracyChanged(Sensor sensor, int accuracy)`
    - Muss implementiert werden, um Compilerfehler zu vermeiden, wir müssen jedoch keine Änderungen innerhalb der Methode vornehmen

Sensoren

- Jetzt muss noch der Listener registriert werden
- Es ist wichtig, dass der Listener bei `onResume()` registriert und in `onStop()` unregistriert wird
  - Registriert und unregistriert wird ein Listener mithilfe des zuvor abgerufenen `SensorManager` und dem `Sensor`
  - `onResume()`
    - ```
      sensorManager.registerListener(this,lightSensor,
                                     SensorManager.SENSOR_DELAY_NORMAL);
      ```
  - `onStop()`
    - ```
      sensorManager.unregisterListener(this);
      ```

Sensoren – Beispiel – MainActivity (1/3)

```
@Override
    protected void onResume() {
        super.onResume();
        sensorManager.registerListener(this, lightSensor,
                SensorManager.SENSOR_DELAY_NORMAL);
        sensorManager.registerListener(this, gyroscopeSensor,
                SensorManager.SENSOR_DELAY_NORMAL);
        sensorManager.registerListener(this, rotationSensor,
                SensorManager.SENSOR_DELAY_NORMAL);
        sensorManager.registerListener(this, stepCounterSensor,
                SensorManager.SENSOR_DELAY_NORMAL);
    }
    @Override
    protected void onPause() {
        super.onPause();
        sensorManager.unregisterListener(this);
    }
}
```

Quellen: http://www.zoftino.com/android-sensors-examples
https://code.tutsplus.com/tutorials/android-sensors-in-depth-proximity-and-gyroscope--cms-28084

Sensoren – Beispiel – MainActivity (2/3)

```
@Override
public final void onAccuracyChanged(Sensor sensor, int accuracy) {}
@Override
public final void onSensorChanged(SensorEvent event) {
    //check sensor type matches current sensor type set by button click
    if( event.sensor.getType() == sensorInd){
        //light sensor
        if(event.sensor.getType() ==  Sensor.TYPE_LIGHT){
            float valueZ = event.values[0];
            Toast.makeText(this, "luminescence "+valueZ,Toast.LENGTH_LONG).show();
            if(valueZ > 40){
                getWindow().getDecorView().setBackgroundColor(Color.RED);
            }else{
                getWindow().getDecorView().setBackgroundColor(Color.GREEN);
            }
        }else if(event.sensor.getType() ==  Sensor.TYPE_GYROSCOPE){
            if(event.values[2] > 0.5f) { // anticlockwise
                getWindow().getDecorView().setBackgroundColor(Color.BLUE);
            } else if(event.values[2] < -0.5f) { // clockwise
                getWindow().getDecorView().setBackgroundColor(Color.YELLOW);
            }
        }
```

Quellen: http://www.zoftino.com/android-sensors-examples
https://code.tutsplus.com/tutorials/android-sensors-in-depth-proximity-and-gyroscope--cms-28084

Sensoren – Beispiel – MainActivity (3/3)

```
else if(event.sensor.getType() ==  Sensor.TYPE_STEP_DETECTOR){
            float steps = event.values[0];
            textView.setText("steps : "+steps);
        }else if(event.sensor.getType() ==  Sensor.TYPE_GAME_ROTATION_VECTOR){
            float[] rotMatrix = new float[9];
            float[] rotVals = new float[3];
            SensorManager.getRotationMatrixFromVector(rotMatrix, event.values);
            SensorManager.remapCoordinateSystem(rotMatrix,
            SensorManager.AXIS_X, SensorManager.AXIS_Y, rotMatrix);
            SensorManager.getOrientation(rotMatrix, rotVals);
            float azimuth = (float) Math.toDegrees(rotVals[0]);
            float pitch = (float) Math.toDegrees(rotVals[1]);
            float roll = (float) Math.toDegrees(rotVals[2]);
            if(azimuth > 60 && azimuth < 90){
                getWindow().getDecorView().setBackgroundColor(Color.GREEN);
            }else if(pitch > 10){
                getWindow().getDecorView().setBackgroundColor(Color.BLUE);
            }else if(roll > 15){
                getWindow().getDecorView().setBackgroundColor(Color.YELLOW);
            }else{
                getWindow().getDecorView().setBackgroundColor(Color.RED);
            }
        }
...
```

Quellen: http://www.zoftino.com/android-sensors-examples
https://code.tutsplus.com/tutorials/android-sensors-in-depth-proximity-and-gyroscope--cms-28084

Sensoren – Beispiel – activity_main.xml (1/2)

```
<?xml version="1.0" encoding="utf-8"?>
<ScrollView
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent">
    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:id="@+id/sensors_layout"
        android:layout_margin="8dp"
        android:orientation="vertical">
        <TextView
            android:id="@+id/sensors"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:textColor="@android:color/holo_red_dark"/>
        <Button
            android:id="@+id/sensors_list"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            style="@style/Widget.AppCompat.Button.Colored"
            android:onClick="sensorsList"
            android:text="Sensors List"/>
```

Quellen: http://www.zoftino.com/android-sensors-examples
https://code.tutsplus.com/tutorials/android-sensors-in-depth-proximity-and-gyroscope--cms-28084

Sensoren – Beispiel – activity_main.xml (2/2)

```
<Button
            android:id="@+id/light_sensor"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            style="@style/Widget.AppCompat.Button.Colored"
            android:onClick="lightSensor"
            android:text="Light Sensor"/>
        <Button
            android:id="@+id/gyroscopeSensor"
            style="@style/Widget.AppCompat.Button.Colored"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:onClick="gyroscopeSensor"
            android:text="Gyroscope Sensor" />
        <Button
            android:id="@+id/rotation_sensor"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            style="@style/Widget.AppCompat.Button.Colored"
            android:onClick="rotationSensor"
            android:text="Rotation Sensor"/>
        <Button
            android:id="@+id/steps_sensor"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            style="@style/Widget.AppCompat.Button.Colored"
            android:onClick="stepCounterSensor"
            android:text="Step Counter Sensor"/>
            </LinearLayout></ScrollView>
```

Quellen: http://www.zoftino.com/android-sensors-examples
https://code.tutsplus.com/tutorials/android-sensors-in-depth-proximity-and-gyroscope--cms-28084

Hinweise zum Beispiel – MainActivity (1-3)

- Sie können die MainActivity.java im folgenden Link übernehmen
  - https://pastebin.com/AfnJRG6m
- Sie können die activity_main.xml im folgenden Link übernehmen
  - https://pastebin.com/h6Q0MUte
- Bei dem Beispiel handelt es sich um eine Zusammensetzung aus
  - http://www.zoftino.com/android-sensors-examples
  - https://code.tutsplus.com/tutorials/android-sensors-in-depth-proximity-and-gyroscope--cms-28084

@LET‘S TRY

- Implementieren Sie das Beispiel zu den Sensoren in einem neuen Android Projekt

Quellen

- https://www.itwissen.info/Framework-framework.html
- https://developer.android.com/guide/topics/sensors/sensors_overview
- https://developer.android.com/training/data-storage/room/accessing-data
- https://developer.android.com/training/data-storage/room/defining-data
- https://developer.android.com/training/data-storage/room/accessing-data#java
- http://www.zoftino.com/android-sensors-examples
- https://code.tutsplus.com/tutorials/android-sensors-in-depth-proximity-and-gyroscope--cms-28084
- https://medium.freecodecamp.org/room-sqlite-beginner-tutorial-2e725e47bfab