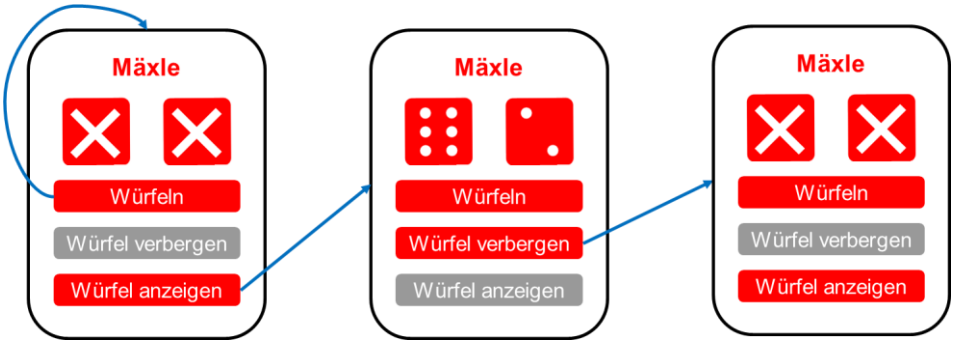
Hintergrund

Die *Mäxle Android-App* besteht aus einer Oberfläche und stellt in der Version 1 zwei Würfel sowie Funktionen zum Würfeln, Würfel verbergen und Würfel anzeigen bereit. Beim Start der App sowie beim Würfeln werden die Würfel verborgen. Version 2 erweitert die App um eine Model-Klasse sowie eine Datenklasse und erhöht dadurch die Softwarequalität, Version 3 ermöglicht das Würfeln durch Schütteln des Smartphones.

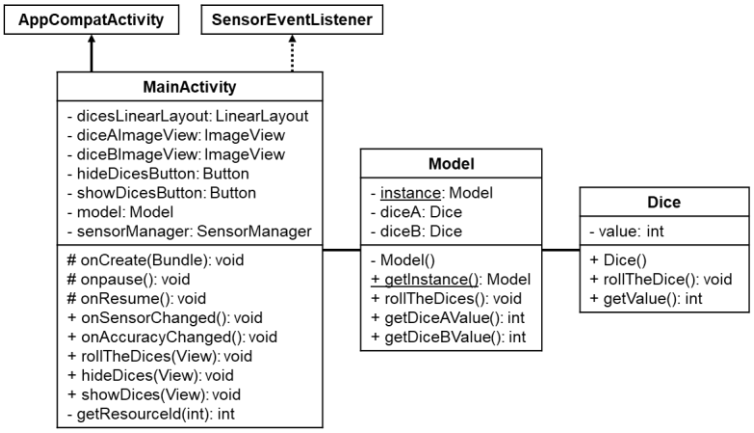
Das Spielprinzip

https://de.wikipedia.org/wiki/M%C3%A4xchen

Grafische Darstellung



Klassendiagramm (Version 3)



Hinweis zur Klasse Model

Die Klasse Model wird als sogenannter Singleton implementiert. Diese Implementierung stellt sich, dass zur Laufzeit genau ein Objekt der Klasse Model existiert.

Anleitung Version 1

1. Projekt und Activity anlegen und anpassen

1.1. Android Studio starten

1.2. Neues Projekt mit *Empty Activity* anlegen

- Name: Maexle
- Package name: com.sap.maexle
- Language: Java
- Minimum SDK: API 16

1.3. Würfel-Grafiken in Ordner *app – res – drawable* kopieren

1.4. Datei *strings.xml* öffnen

1.5. String-Definition *app_name* anpassen

- Wert: Mäxle

1.6. App-Icon anpassen (*File – New – Image Asset*)

2. Layout *activity_main.xml* anpassen

2.1. Datei *activity_main.xml* öffnen

2.2. Bildschirmelement *TextView* löschen

2.3. Layout in ein *LinearLayout (vertical)* ändern

- layout_width: match_parent
- layout_height: match_parent
- layout_margin: 50dp
- gravity: center
- orientation: vertical

2.4. Bildschirmelement *TextView* zu *LinearLayout (vertical)* hinzufügen

- layout_width: match_parent
- layout_height: wrap_content
- gravity: center_horizontal
- id: textView
- text: Mäxle
- textSize: 60sp
- textStyle: bold

2.5. Bildschirmelement *LinearLayout (horizontal)* zu *LinearLayout (vertical)* hinzufügen

- layout_width: match_parent
- layout_height: wrap_content
- id: dicesLinearLayout
- gravity: center
- orientation: horizontal

2.6. Bildschirmelement *ImageView* hinzufügen zu *LinearLayout (horizontal)* hinzufügen

- layout_width: wrap_content
- layout_height: wrap_content
- layout_weight: 1
- id: diceAImageView
- padding: 15dp

- srcCompat: @drawable/zero

2.7. Bildschirmelement *ImageView* zu *LinearLayout (horizontal)* hinzufügen

- layout_width: wrap_content
- layout_height: wrap_content
- layout_weight: 1
- id: diceBImageView
- padding: 15dp
- srcCompat: @drawable/zero

2.8. Bildschirmelement *Button* zu *LinearLayout (vertical)* hinzufügen

- layout_width: match_parent
- layout_height: wrap_content
- id: rollTheDicesButton
- text: Würfeln

2.9. Bildschirmelement *Button* zu *LinearrollTheDicesButtonLayout (vertical)* hinzufügen

- layout_width: match_parent
- layout_height: wrap_content
- id: hideDicesButton
- text: Würfel Verbergen

2.10. Bildschirmelement *Button* zu *LinearLayout (vertical)* hinzufügen

- layout_width: match_parent
- layout_height: wrap_content
- id: showDicesButton
- text: Würfel Anzeigen

3. Klasse *ActivityMain* anpassen

3.1. Klasse *ActivityMain* öffnen

3.2. Attribut *dicesLinearLayout* deklarieren

- Sichtbarkeit: privat
- Datentyp: LinearLayout

3.3. Attribut *diceAImageView* deklarieren

- Sichtbarkeit: privat
- Datentyp: ImageView

3.4. Attribut *diceBImageView* deklarieren

- Sichtbarkeit: privat
- Datentyp: ImageView

3.5. Attribut *hideDicesButton* deklarieren

- Sichtbarkeit: privat
- Datentyp: Button

3.6. Attribut *showDicesButton* deklarieren

- Sichtbarkeit: privat
- Datentyp: Button

3.7. Attribut *diceAResourceId* deklarieren

- Sichtbarkeit: privat
- Datentyp: int

3.8. Attribut *diceBResourceId* deklarieren

- Sichtbarkeit: privat
- Datentyp: int

3.9. Methode *onCreate(Bundle)* ergänzen

- Bildschirmelement *dicesLinearLayout* mit Attribut *dicesLinearLayout* verbinden
  - Referenz: this
  - Methode: findViewByid(int)
  - Parameterwert: R.id.dicesLinearLayout
- Bildschirmelement *diceAImageView* mit Attribut *diceAImageView* verbinden
  - Referenz: this
  - Methode findViewByid(int)
  - Parameterwert: R.id.diceAImageView
- Bildschirmelement *diceBImageView* mit Attribut *diceBImageView* verbinden
  - Referenz: this
  - Methode findViewByid(int)
  - Parameterwert: R.id.diceBImageView
- Bildschirmelement *hideDicesButton* mit Attribut *hideDicesButton* verbinden
  - Referenz: this
  - Methode findViewByid(int)
  - Parameterwert: R.id.hideDicesButton
- Bildschirmelement *showDicesButton* mit Attribut *showDicesButton* verbinden
  - Referenz: this
  - Methode findViewByid(int)
  - Parameterwert: R.id.showDicesButton

3.10. Methode *rollTheDices(View)* definieren

- Sichtbarkeit: öffentlich
- Rückgabewert: void
- Parametername: view

3.11. Methode *hideDices(View)* definieren

- Sichtbarkeit: öffentlich
- Rückgabewert: void
- Parametername: view

3.12. Methode *showDices(View)* definieren

- Sichtbarkeit: öffentlich
- Rückgabewert: void
- Parametername: view

3.13. Methode *getResourceId(int)* definieren

- Sichtbarkeit: privat
- Rückgabewert: int
- Parametername: value

3.14. Methode *getResourceId(int)* implementieren

- Eingehenden Würfelwert prüfen
  - Würfelwert = 1: Datenquellen-Id *R.drawable.one* zurückgeben
  - Würfelwert = 2: Datenquellen-Id *R.drawable.two* zurückgeben
  - Würfelwert = 3: Datenquellen-Id *R.drawable.three* zurückgeben

- Würfelwert = 4: Datenquellen-Id *R.drawable.four* zurückgeben
- Würfelwert = 5: Datenquellen-Id *R.drawable.five* zurückgeben
- Würfelwert = 6: Datenquellen-Id *R.drawable.six* zurückgeben
- Sonst: Datenquellen-Id *R.drawable.zero* zurückgeben

3.15. Methode *hideDices(View)* implementieren

- Datenquellen-Id zum Würfelwert 0 lesen und zwischenspeichern
  - Bezeichner: zeroResourceId
  - Datentyp: int
  - Referenz: this
  - Methode: getResourceId(int)
  - Parameterwert: 0
- Bild-Datenquelle für Attribut *diceAImageView* setzen
  - Referenz: diceAImageView
  - Methode: setImageResource(int)
  - Parameterwert: zeroResourceId
- Bild-Datenquelle für Attribut *diceBImageView* setzen
  - Referenz: diceBImageView
  - Methode: setImageResource(int)
  - Parameterwert: zeroResourceId
- Drucktaste *Würfel verbergen* deaktivieren
  - Referenz: hideDicesButton
  - Methode: setEnabled(boolean)
  - Parameterwert: false
- Drucktaste *Würfel anzeigen* aktivieren
  - Referenz: showDicesButton
  - Methode: setEnabled(boolean)
  - Parameterwert: true

3.16. Methode *rollTheDices(View)* implementieren

- Würfel verbergen
  - Referenz: this
  - Methode: hideDices(view)
  - Parameterwert: view
- Zufallsobjekt erzeugen und zwischenspeichern
  - Bezeichner: random
  - Datentyp: Random
- Zufallszahl zwischen 1 und 6 erzeugen und zwischenspeichern
  - Bezeichner: diceAValue
  - Datentyp: int
  - Referenz: random
  - Methode: nextInt(int)
  - Parameterwert: 6
  - Formel: Zufallszahl + 1
- Zufallszahl zwischen 1 und 6 erzeugen und zwischenspeichern
  - Bezeichner: diceBValue
  - Datentyp: int

    - Referenz: random
    - Methode: nextInt(int)
    - Parameterwert: 6
    - Formel: Zufallszahl + 1
- Datenquellen-Id zum Würfelwert der Variablen *diceAValue* lesen und dem Attribut *diceAResourceId* zuweisen
    - Referenz: this
    - Methode: getResourceId(int)
    - Parameterwert: diceAValue
- Datenquellen-Id zum Würfelwert der Variablen *diceBValue* lesen und dem Attribut *diceBResourceId* zuweisen
    - Referenz: this
    - Methode: getResourceId(int)
    - Parameterwert: diceBValue

3.17. Methode *showDices(View)* implementieren

- Bild-Datenquelle für Attribut *diceAImageView* setzen
    - Referenz: diceAImageView
    - Methode: setImageResource(int)
    - Parameterwert: diceAResourceId
- Bild-Datenquelle für Attribut *diceBImageView* setzen
    - Referenz: diceBImageView
    - Methode: setImageResource(int)
    - Parameterwert: diceBResourceId
- Drucktaste *Würfel verbergen* aktivieren
    - Referenz: hideDicesButton
    - Methode: setEnabled(boolean)
    - Parameterwert: true
- Drucktaste *Würfel anzeigen* deaktivieren
    - Referenz: showDicesButton
    - Methode: setEnabled(boolean)
    - Parameterwert: false

3.18. Methode *onCreate(Bundle)* ergänzen

- Würfeln
    - Referenz: this
    - Methode: rollTheDices(View)
    - Parameterwert: dicesLinearLayout

4. Layout *activity_main.xml* anpassen

4.1. Bildschirmelement *rollTheDicesButton* ergänzen

- onClick: rollTheDices

4.2. Bildschirmelement *hideDicesButton* ergänzen

- onClick: hideDices

4.3. Bildschirmelement *showDicesButton* ergänzen

- onClick: showDices

5. Datei *colors.xml* und Datei *themes.xml* anpassen
   5.1. Datei *colors.xml* öffnen
   5.2. Color-Definition *red* ergänzen
      - Wert: #FF0000
   5.3. Datei *themes.xml* öffnen
   5.4. Style-Definition *colorPrimary* anpassen
      - Wert: @color/red
   5.5. Style-Definition *colorPrimaryVariant* anpassen
      - Wert: @color/red
   5.6. Style-Definition *colorOnPrimary* anpassen
      - Wert: @color/white
   5.7. Style-Definition *colorSecondary* anpassen
      - Wert: @color/white
   5.8. Style-Definition *colorSecondaryVariant* anpassen
      - Wert: @color/white
   5.9. Style-Definition *colorOnSecondary* anpassen
      - Wert: @color/black
   5.10. Style-Definition *android:textColor* ergänzen
      - Wert: @color/red
   5.11. Datei *themes.xml (night)* öffnen
   5.12. Style-Definition *colorPrimary* anpassen
      - Wert: @color/red
   5.13. Style-Definition *colorPrimaryVariant* anpassen
      - Wert: @color/red
   5.14. Style-Definition *colorOnPrimary* anpassen
      - Wert: @color/white
   5.15. Style-Definition *colorSecondary* anpassen
      - Wert: @color/white
   5.16. Style-Definition *colorSecondaryVariant* anpassen
      - Wert: @color/white
   5.17. Style-Definition *colorOnSecondary* anpassen
      - Wert: @color/black
   5.18. Style-Definition *android:textColor* ergänzen
      - Wert: @color/red

Anleitung Version 2

1. Klasse *Dice* erstellen und implementieren
   1.1. Klasse *Dice* erstellen und öffnen
   1.2. Attribut *value* deklarieren
      - Sichtbarkeit: privat
      - Datentyp: int
   1.3. Konstruktor *Dice()* definieren
      - Sichtbarkeit: öffentlich

1.4. Methode *rollTheDice()* definieren
- Sichtbarkeit: öffentlich
- Rückgabewert: void

1.5. Methode *getValue()* definieren und implementieren
- Sichtbarkeit: öffentlich
- Rückgabewert: int

1.6. Konstruktor *Dice()* implementieren
- Würfeln
  - Referenz: this
  - Methode: rollTheDice()

1.7. Methode *rollTheDice()* implementieren
- Zufallszahl zwischen 1 und 6 erstellen und dem Attribut *value* zuweisen (siehe Schritt-für-Schritt-Anleitung Version 1 - 3.13)

1.8. Methode *getValue()*implementieren
- Attribut *value* zurückgeben

2. Klasse *Model* erstellen und implementieren

2.1. Klasse *Model* erstellen und öffnen

2.2. Statisches Attribut *instance* deklarieren
- Sichtbarkeit: privat
- Datentyp: Model

2.3. Attribut *diceA* deklarieren
- Sichtbarkeit: privat
- Datentyp: Dice

2.4. Attribut *diceB* deklarieren
- Sichtbarkeit: privat
- Datentyp: Dice

2.5. Konstruktor *Model()* definieren
- Sichtbarkeit: privat

2.6. Statische Methode *getInstance()* definieren
- Sichtbarkeit: public
- Rückgabewert: Model

2.7. Methode *rollTheDices()* definieren
- Sichtbarkeit: privat
- Rückgabewert: void

2.8. Methode *getDiceAValue()* definieren
- Sichtbarkeit: privat
- Rückgabewert: int

2.9. Methode *getDiceBValue()* definieren
- Sichtbarkeit: privat
- Rückgabewert: int

2.10. Konstruktor *Model()* implementieren
- Würfelobjekt erzeugen um dem Attribut *diceA* zuweisen
- Würfelobjekt erzeugen um dem Attribut *diceB* zuweisen

2.11. Statische Methode *getInstance()* implementieren

- Statisches Attribut *instance* überprüfen
  - Wert = null: Modelobjekt erzeugen und dem statischen Attribut *instance* zuweisen

2.12. Methode *rollTheDices()* implementieren

- Würfeln
  - Referenz: diceA und diceB
  - Methode: rollTheDice()

2.13. Methode *getDiceAValue()* implementieren

- Würfelwert von Attribut *diceA* zurückgeben
  - Referenz: diceA
  - Methode: getValue()

2.14. Methode *getDiceBValue()* implementieren

- Würfelwert von Attribut *diceB* zurückgeben
  - Referenz: diceB
  - Methode: getValue()

3. Klasse *MainActivity* anpassen

3.1. Klasse *MainActivity* öffnen

3.2. Attribut *model* deklarieren

- Sichtbarkeit: privat
- Datentyp: Model

3.3. Methode *onCreate(Bundle)* ergänzen

- Modelobjekt lesen und dem Attribut *model* zuweisen
  - Klasse: Model
  - Methode: getInstance()

3.4. Methode *rollTheDices(View)* anpassen

- Erzeugen und Zwischenspeichern des Zufallsobjektes durch Würfeln ersetzen
  - Referenz: model
  - Methode: rollTheDices()
- Erzeugen der Zufallszahlen durch Lesen der Würfelwerte ersetzen
  - Referenz: model
  - Methode: getDiceAValue() und getDiceBValue()

Anleitung Version 3

1. Klasse *MainActivity* anpassen

1.1. Klasse *MainActivity* öffnen

1.2. Attribut *sensorManager* deklarieren

- Sichtbarkeit: privat
- Datentyp: SensorManager

1.3. Methode *onCreate(Bundle)* ergänzen

- Sensoren-Manager auslesen und Attribut *sensorManager* zuweisen
  - Referenz: this
  - Methode: getSystemService(String)
  - Parameterwert: SENSOR_SERVICE

1.4. Schnittstelle *SensorEventListener* realisieren

1.5. Methode *onSensorChanged(SensorEvent)* überschreiben

- X-Wert auslesen und zwischenspeichern
  - Bezeichner: x
  - Datentyp: float
  - Referenz: sensorEvent
  - Attribut: values
  - Index: 0
- Y-Wert auslesen und zwischenspeichern
  - Bezeichner: y
  - Datentyp: float
  - Referenz: sensorEvent
  - Attribut: values
  - Index: 1
- Z-Wert auslesen und zwischenspeichern
  - Bezeichner: z
  - Datentyp: float
  - Referenz: sensorEvent
  - Attribut: values
  - Index: 2
- Beschleunigung berechnen und zwischenspeichern
  - Bezeichner: acceleration
  - Datentyp: float
  - Formel: (x * x + y * y + z * z) / (SensorManager.GRAVITY_EARTH * SensorManager.GRAVITY_EARTH)
- Beschleunigung prüfen
  - Wert > 3: Würfeln
    - Referenz: this
    - Methode: rollTheDices(View)
    - Parameterwert: dicesLinearLayout

1.6. Methode *onAccuracyChanged(Sensor, int)* überschreiben

1.7. Methode *onResume()* überschreiben

- Klasse als Ereignisbehandler anmelden
  - Referenz: sensorManager
  - Methode: registerListener(SensorEventListener, Sensor, int)
  - Parameterwerte: this, sensorManager.getDefaultSensor(Sensor.TYPE_ACCELEROMETER), SensorManager.SENSOR_DELAY_NORMAL

1.8. Methode *onPause()* überschreiben

- Klasse als Ereignisbehandler abmelden
  - Referenz: sensorManager
  - Methode: unregisterListener(SensorEventListener)
  - Parameterwert: this