Hintergrund

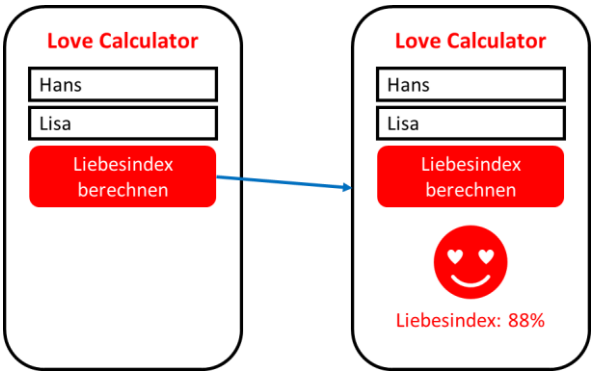
Die *Love Calculator Android-App* besteht aus einer Activity, welche die Eingabe zweier Namen sowie die anschließende Berechnung und Ausgabe des Liebesindexes als Grafik und in Textform ermöglicht. Der Liebesindex berechnet sich nach einer wissenschaftlich belegten Formel.

Formel

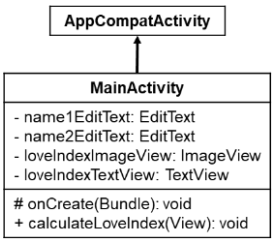
$L = (A + B)\ mod\ 100 + 1$

Legende: L = Liebesindex, A = Dezimalwert von Name 1, B = Dezimalwert von Name 2

Grafische Darstellung



Klassendiagramm



Anleitung

1. Projekt und Activity anlegen und anpassen

1.1. Android Studio starten

1.2. Neues Projekt mit *Empty Activity* anlegen

- Name: LoveCalculator
- Package name: com.sap.lovecalculator
- Language: Java
- Minimum SDK: API 16

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

1.4. Datei *strings.xml* öffnen

1.5. String-Definition *app_name* anpassen

- Wert: Love Calculator

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
- orientation: vertical

2.4. Bildschirmelement *Plain Text* zu *LinearLayout (vertical)* hinzufügen

- layout_width: match_parent
- layout_height: wrap_content
- ems: 10
- hint: Dein Name
- id: name1EditText
- inputType: txtPersonName
- text: *leer*

2.5. Bildschirmelement *Plain Text* zu *LinearLayout (vertical)* hinzufügen

- layout_width: match_parent
- layout_height: wrap_content
- ems: 10
- hint: Sein/Ihr Name
- id: name2EditText
- inputType: txtPersonName
- text: leer

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

- layout_width: match_parent
- layout_height: wrap_content
- id: calculateLoveindexButton
- text: Liebesindex berechnen

2.7. Bildschirmelement *ImageView* hinzufügen zu *LinearLayout (verticall)* hinzufügen
- layout_width: match_parent
- layout_height: wrap_content
- id: loveindexImageView
- srcCompat: *leer*

2.8. Bildschirmelement *TextView* zu *LinearLayout (vertical)* hinzufügen
- layout_width: match_parent
- layout_height: wrap_content
- gravity: center_horizontal
- id: loveindexTextView
- text: *leer*
- textSize: 24sp

3. Klasse *ActivityMain* anpassen

3.1. Klasse *ActivityMain* öffnen

3.2. Attribut *name1EditText* deklarieren
- Sichtbarkeit: privat
- Datentyp: EditText

3.3. Attribut *name2EditText* deklarieren
- Sichtbarkeit: privat
- Datentyp: EditText

3.4. Attribut *loveindexImageView* deklarieren
- Sichtbarkeit: privat
- Datentyp: ImageView

3.5. Attribut *loveindexTextView* deklarieren
- Sichtbarkeit: privat
- Datentyp: TextView

3.6. Methode *onCreate(Bundle)* ergänzen
- Bildschirmelement *name1EditText* mit Attribut *name1EditText* verbinden
  - Referenz: this
  - Methode: findViewByid(int)
  - Parameterwert: R.id.name1EditText
- Bildschirmelement *name2EditText* mit Attribut *name2EditText* verbinden
  - Referenz: this
  - Methode: findViewByid(int)
  - Parameterwert: R.id.name2EditText
- Bildschirmelement *loveindexImageView* mit Attribut *loveindexImageView* verbinden
  - Referenz: this
  - Methode: findViewByid(int)
  - Parameterwert: R.id.loveindexImageView
- Bildschirmelement *loveindexTextView* mit Attribut *loveindexTextView* verbinden
  - Referenz: this
  - Methode: findViewByid(int)
  - Parameterwert: R.id.loveindexTextView

3.7. Methode *calculateLoveindex(View)* definieren

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

3.8. Methode *calculateLoveindex(View)* implementieren

- Werte der Attribute *name1EditText* und *name2EditText* auslesen und zwischenspeichern
  - Bezeichner: name1 und name2
  - Datentyp: String
  - Referenz: name1EditText und name2EditText
  - Methode: getText().toString()
- Variablen *name1* und *name2* prüfen
  - name1 = *leer* oder name2 = *leer*
    - Toast erzeugen
      - Klasse: Toast
      - Methode: makeText(Context, CharSequence, int).show()
      - Eingabeparameter: this, „Bitte beide Namen eingeben“, Toast.LENGTH_LONG
    - Bild-Datenquelle für Attribut *loveindexImageView* setzen
      - Referenz: loveindexImageView
      - Methode: setImageResource(int)
      - Parameterwert: 0
    - Wert für Attribut *loveindexTextView* setzen
      - Referenz: loveindexTextView
      - Methode: setText(CharSequence)
      - Parameterwert: „“
    - Methode beenden
- Variablen *name1* und *name2* zusammenführen und zwischenspeichern
  - Bezeichner: text
  - Datentyp: String
- Attribut *textValue* deklarieren und initialisieren
  - Datentyp: int
  - Wert: 0
- Innerhalb einer Schleife das Attribut um den Zahlenwert jedes Zeichens der Zeichenkette *text* erhöhen
  - Indexzähler: int i = 0
  - Schleifenbedingung: i < text.length()
  - Indexmanipulation: i++
  - Referenz: text
  - Methode: charAt(int)
  - Parameterwert: i
- Liebesindex berechnen und zwischenspeichern
  - Bezeichner: loveindex
  - Datentyp: int
  - Formel: textValue % 100 + 1
- Variable resourceId deklarieren
  - Datentyp: int
- Liebesindex prüfen

- Wert < 20: Der Variablen *resourceId* den Wert *R.drawable.extreme_sad* zuweisen
- Wert < 40: Der Variablen *resourceId* den Wert *R.drawable.sad* zuweisen
- Wert < 60: Der Variablen *resourceId* den Wert *R.drawable.neutral* zuweisen
- Wert < 80: Der Variablen *resourceId* den Wert *R.drawable.happy* zuweisen
- Sonst: Der Variablen *resourceId* den Wert *R.drawable.extreme_happy* zuweisen

- Bild-Datenquelle für Attribut *loveindexImageView* setzen
  - Referenz: loveindexImageView
  - Methode: setImageResource(int)
  - Parameterwert: resourceId
- Wert für Attribut *loveindexTextView* setzen
  - Referenz: loveindexTextView
  - Methode: setText(CharSequence)
  - Parameterwert: „Liebesindex: “ + loveindex + „&“

4. Layout *activity_main.xml* anpassen

4.1. Bildschirmelement *calculateLoveindexButton* ergänzen

- onClick: calculateLoveindex

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/red

5.9. Style-Definition *colorOnSecondary* anpassen

- Wert: @color/black

5.10. Style-Definition *android:textColor* ergänzen

- Wert: @color/black

5.11. Style-Definition *android:accentColor* ergänzen

- Wert: @color/black

5.12. Datei *themes.xml (night)* öffnen

5.13. Style-Definition *colorPrimary* anpassen

- Wert: @color/red

5.14. Style-Definition *colorPrimaryVariant* anpassen

- Wert: @color/red

5.15. Style-Definition *colorOnPrimary* anpassen

- Wert: @color/white

5.16. Style-Definition *colorSecondary* anpassen

- Wert: @color/white

5.17. Style-Definition *colorSecondaryVariant* anpassen

- Wert: @color/red

5.18. Style-Definition *colorOnSecondary* anpassen

- Wert: @color/black

5.19. Style-Definition *android:textColor* ergänzen

- Wert: @color/white

5.20. Style-Definition *android:textColor* ergänzen

- Wert: @color/white