Prof. Dr.-Ing. R. Roosmann, H. Plitzner

Aufgabenblatt 05

Ziel dieses Aufgabenblatts ist es, Ihnen einen ersten Eindruck zur Programmierung mobiler Anwendungen mit Java unter Android Studio zu geben.

Abgabe: Gr. 3: 01.12.2021, Gr. 1+2: 06.12.2021; Max. Punktzahl: 10; Min. Punktzahl: 6 Punkte

Aufgabe 5.1: Hello World (3 Punkte)

Hello World! ist für viele Entwickler das erste Programm, das sie in einer Programmiersprache schreiben. Also, schreiben Sie in dieser Aufgabe ein Hello-World Programm für Android mit Java unter Verwendung von Android Studio.

In den letzten Aufgabenblättern haben Sie das Interaktionsbrett genutzt, um grafische Elemente zu zeichnen. In dieser Aufgabe wollen wir zum Zeichnen die vorhandenen Android-Klassen einsetzen.

Android verwendet Views zur Darstellung von Inhalten für die Mensch-Maschine Interaktion. Es existierten verschiedene View-Typen. Sie verwenden in dieser Aufgabe die Klasse `ImageView`. Diese Klasse ist in der Lage grafische Elemente zu visualisieren. Aber wo kommen diese Inhalte her? Zum einen gibt es die Klasse `Canvas`. Diese kann verglichen werden mit einem Maler (Künstler). Ein Maler kann zwar zeichnen, benötigt allerdings ein Blatt, um seine Kunst sichtbar zu machen. Unter Android ist die Klasse `Bitmap` vergleichbar zu dem Zeichenblatt eines Malers. Eines fehlt noch. Wir haben den Maler (`Canvas`), das Blatt (`Bitmap`) und benötigen jetzt noch Pinsel + Farbpalette. Dies liefert uns in Android die Klasse `Paint`.

Los geht's. Starten Sie Android Studio und legen ein neues Android Projekt an. Klicken Sie hierzu den Menüpunkt `File` an und wählen `New Project`. Es öffnet sich ein Popup-Fenster, in dem Sie als Projekttyp *Empty Activity* unter dem Reiter *Phone and Tablet* auswählen. Im nächsten Schritt ist das Projekt zu konfigurieren. Wählen Sie einen sinnvollen Namen aus, definieren das Package und den Speicherort. Des Weiteren ist als Programmiersprache Java und als Minimum API Level die *API19 Android 4.4 (KitKat)* zu wählen. Klicken sie anschließend auf den `Finish`-Button. Android Studio legt nun das Projekt an und öffnet die Klasse `MainActivity`.

Activities werden zur Verwaltung von Views eingesetzt. Soll ein Android-Projekt mit einer View gestartet werden, so ist die Main-Activity in der `AndroidManifest.xml` zu definieren. Standardmäßig übernimmt Android-Studio alles für Sie und trägt dort die generierte Klasse `MainActivity` ein.

Da wir Instanzen der Klassen `ImageView`, `Canvas`, `Bitmap` und `Paint` benötigen, legen Sie hierfür private Instanzvariablen in der Klasse `MainActivity` an. Fügen Sie zwei zusätzliche `int`-Variablen für die Breite und Höhe der Bitmap an und weisen diesen bspw. jeweils den Wert 800 zu.

In der `onCreate`-Methode der Klasse `MainActivity`, die direkt nach dem Start der Android-App ausgeführt wird, initialisieren Sie die Instanzvariablen nach der Anweisung **super**.onCreate(savedInstanceState) wie folgt:

```
this.bitmap = Bitmap.createBitmap(this.breite, this.hoehe,
    Bitmap.Config.ARGB_8888);
this.canvas = new Canvas(this.bitmap);
this.imageView = new ImageView(this);
this.imageView.setImageBitmap(this.bitmap);
this.paint = new Paint();
```

Da eine Activity eine View steuert, muss die zugehörige View der `MainActivity` explizit bekannt gemacht werden. Fügen Sie deshalb dem zuletzt eingefügten Source-Code der `onCreate`-Methode folgende Anweisung hinzu:

```
setContentView(imageView);
```

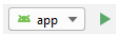
Somit ist jetzt alles vorbereitet, um den Text „Hallo Welt!!“ auf dem Canvas zu zeichnen. Fügen Sie dazu wieder hinter der Methode `this.setContentView(…)` folgenden Source-Code ein:

```
this.canvas.drawColor(Color.argb(255, 0, 0, 255));
this.paint.setTextSize(50);
this.halloWelt();
```

Fügen Sie anschließend der Klasse `MainActivity` die benötigte halloWelt-Methode als private Hilfsmethode hinzu.

```
private void halloWelt() {
    String text = "Hallo Welt!";
    float textWidth = this.paint.measureText(text);
    this.paint.setColor(Color.WHITE);
    this.canvas.drawText(text, breite /2 - textWidth/2, 100,
        this.paint);
}
```

Interessant: das `Paint`-Objekt bietet uns eine Methode, um die Breite eines bestimmten Textes zu ermitteln und somit den Text zentriert auf dem Canvas zu zeichnen. Kennen Sie ja schon vom Interaktionsbrett.



Aufgabe 5.2: Hallo Nachbar (1 Punkte)

Fügen Sie für die Textgröße eine finale Variable namens `textsize` in die `MainActivity` ein und weisen dieser den Wert `50` zu. Passen Sie Ihren aus Aufgabe 5.1 vorhandenen Source-Code an.

Erstellen Sie eine private Hilfsmethode `textZentrieren(String text, int y)`, über die ein übergebener String zentriert dem Canvas hinzugefügt wird. Neben dem String ist eine y-Position zu übergeben, um den Text auf dem Canvas in y-Richtung zu positionieren. Zur Zentrierung verwenden Sie die finale Instanzvariable breite (die ja die Breite der ImageView definiert) in Kombination mit der Methode `measureText(…)` von `this.paint`.

Zum Testen der neuen Methode erstellen Sie die private Hilfsmethode `halloNachbarn()`. Diese nutzt die eben erstellte Hilfsmethode und begrüßt Ihre Nachbarn. Der Text soll direkt unter dem Text „Hallo Welt!!“ dem Canvas hinzugefügt werden.

Passen Sie anschließend die Methode `onCreate(…)` an, so dass die Methode `halloNachbarn()` direkt nach der Methode `halloWelt()` ausgeführt wird.

Aufgabe 5.3: Smiley zeichnen (1 Punkte)

Als nächstes soll unter dem Text ein Smiley gezeichnet werden. Dieser muss nicht super toll aussehen (wäre allerdings auch nicht schlecht;-).

Erstellen Sie hierzu die Methode `public void zeichneSmiley(int radius)`.

Zur Information: die Farbe zum Zeichnen auf dem Canvas lässt sich über
`this.paint.setColor(Color.GREEN);` anpassen.

Das Canvas-Objekt stellt verschiedene Methoden zum Zeichnen von Linien, Rechtecken, Kreisen usw. bereit, die Sie verwenden sollen.

Passen Sie anschließend die Methode `onCreate(…)` an, so dass die Methode `zeichneSmiley(100)` direkt nach der Methode `halloNachbarn()` ausgeführt wird.

Es wäre gut, wenn Ihr Smiley etwas freundlicher aussehen würde, als meiner:-/

Aufgabe 5.4: der springende Punkt (5 Punkte)

Ähnlich, wie beim Pong-Spiel soll sich hier ein Ball, den Sie als Kreis auf dem Canvas zeichnen, in einem definierten Bereich bewegen. Dies soll in der privaten Methode `derSpringendePunkt()` der Klasse `MainActivity` umgesetzt werden.

Die Methode soll mit 60 FPS, also alle 17 Millisekunden, ausgeführt werden, um die `ImageView` zu aktualisieren. Dazu verwenden Sie die Klasse `java.util.Timer`, die als private Instanzvariable der Klasse `MainActivity` hinzuzufügen ist:

```
private Timer timer = new Timer();
```

Gehen Sie anschließend an das Ende der `onCreate`-Methode und fügen dort folgenden Source-Code ein:

```
this.timer.schedule(
        new TimerTask(){
            @Override
            public void run() {
                derSpringendePunkt();
            }
        }
        ,0,17 );
```

Das `Timer`-Objekt `this.timer` wird verwendet, um einen neuen `TimerTask` zu erstellen, der in seiner `run`-Methode die Methode `derSpringendePunkt()` aufruft. Dieser

`TimerTask` wird über `this.timer` mit einer Verzögerung (engl. *delay*) von 0 Millisekunden und einer Wiederholungsfrequenz (engl. *period*) von 17 Millisekunden bis zum Ende der Anwendung immer wieder aufgerufen. Um weitere Details zu erfahren, sehen Sie sich die Klassen `Timer` und `TimerTask` in der Referenzdokumentation zu Java SE 8 an.

Um zu testen, ob das `timer`-Objekt tatsächlich wie erwartet funktioniert, fügen Sie der Methode `derSpringendePunkt()` folgenden Source-Code hinzu:

```
public void derSpringendePunkt() {
    Log.i("MainActivity", LocalDateTime.now()
        + ": der springende Punkt");
}
```

Mit der Anweisung `Log.i(…)` wird über das Logcat-Fenster das aktuelle Datum, die aktuelle Zeit, sowie der definierte Text ausgegeben. Dies sollte nachdem Sie die Anwendung gestartet haben, alle 17 Millisekunden geschehen.

Nachdem sichergestellt ist, dass die Methode `derSpringendePunkt()` wiederholend aufgerufen wird, können Sie die Aufgabe umsetzen, dass ein Ball (als Kreis auf dem Canvas zu zeichnen) sich in einem definierten Bereich bewegt.

Der Bereich indem sich der Kreis bewegen darf, wird über folgende Instanzvariablen der Klasse `MainActivity` festgelegt:

```
int grenzeLinks = 30;
int grenzeRechts = 770;
int grenzeOben = 400;
int grenzeUnten = 770;
```

Um den Ball zu beschreiben fügen Sie der `MainActivity` folgende Instanzvariablen hinzu:

```
int ballRadius = 20;
float ballX = 100f;
float ballY = 700f;
float velociteX = 0.3f;
float velociteY = 4.5f;
```

Ein Canvas hat keine Methode `abwischen()`, wie Sie es vom Interaktionsbrett kennen. Wie können Sie jetzt aber sicherstellen, dass der Ball der vorangegangenen Iteration nicht mehr zu sehen ist? Machen Sie es sich möglichst einfach, indem der Kreis der vorangegangenen Iteration mit der Hintergrundfarbe des Canvas zu Beginn der Methode `derSpringendePunkt()` einfach erneut gezeichnet wird, bspw. über

```
this.paint.setColor(Color.BLUE);
```

Damit wird der Kreis der Vorgänger-Iteration zwar gezeichnet, ist aber nicht mehr zu sehen. Anschließend setzen Sie die Farbe von `this.paint` bspw. auf Rot, verschieben `this.ballX` um `this.velociteX` in x-Richtung und `this.ballY` um `this.velociteX` in y-Richtung und zeichnen den Kreis erneut. Falls der Kreis die Seiten des vorab definierten Bereichs berührt, ändern Sie das Vorzeichen von `this.velociteX` bzw. von `this.velociteY.`

Damit das ImageView neu gezeichnet wird, fügen Sie folgende Anweisung an das Ende der derSpringendePunkt-Methode ein:

```
this.imageView.invalidate();
```

Kommentieren Sie zum Abschluss das Logging in der ersten Zeile der Methode `derSpringendePunkt()` aus. Das Logging sollte ausschließlich verwendet werden, um die korrekte Funktionalität des Timers zu prüfen. Erweitern Sie dann die Methode `onCreate(…)`, so dass die Methode `derSpringedePunkt()` direkt nach der Methode `zeichneSmiley(100)` ausgeführt wird.

Führen Sie jetzt Ihre Android-App erneut aus und der Kreis sollte sich dynamisch im von Ihnen definierten Bereich bewegen.

Viel Erfolg!!