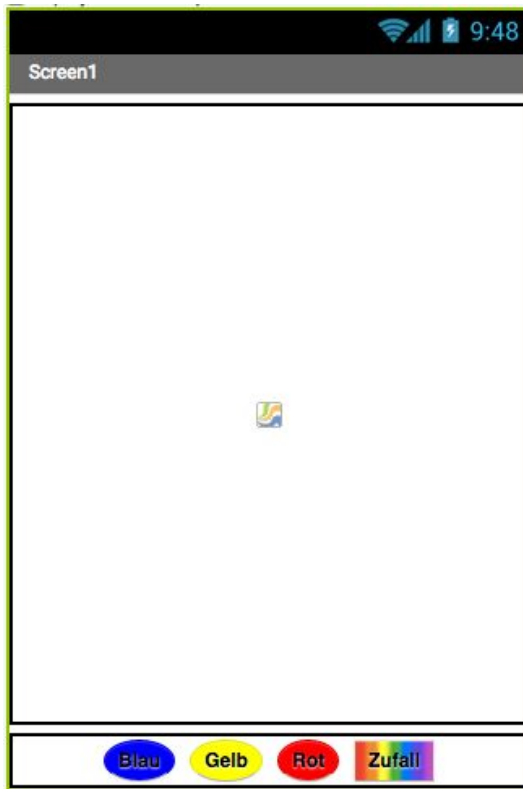




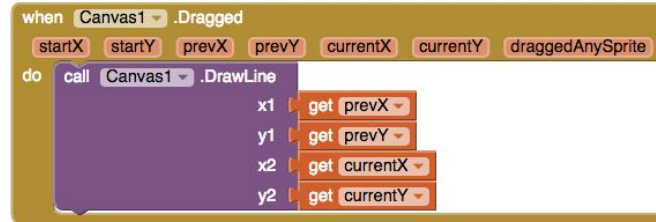
Du brauchst diese Komponenten:

- 1 Canvas (Width und Height = Fill Parent)
- 1 HorizontalArrangements
- 3 Buttons (Shape = oval)
- 1 Button (Shape = default)

Bild Dateien für diese App findest du hier:
appcamps.link/zeichenapp

Was passiert hier?

Das *Canvas* ist deine Zeichenfläche. Du findest das Canvas in der *Palette* unter *Drawing & Animation*.



Um mit dem Finger in dem Canvas zu zeichnen, brauchst du diesen Block.



So kannst du die Farb-Buttons programmieren.

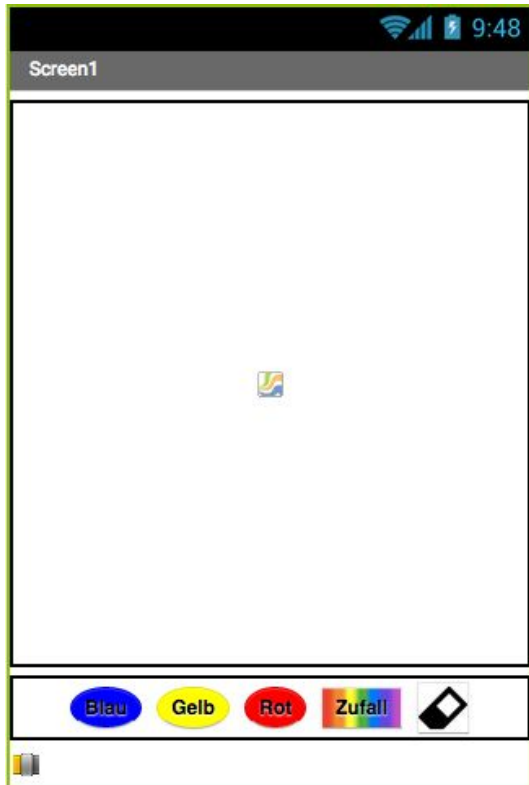
Eine Zufallsfarbe erstellst du mit **make color** (→ in Colors) und drei zufälligen Werten zwischen 0 und 255 (**random integer** → in Math)



Die **Zufallsfarbe** kannst du selbst „mischen“ und zwar mit drei Werten für Rot, Grün und Blau (=RGB). Jede Farbe kann dabei einen Wert von 0 bis 255 zugewiesen bekommen. **Lila** ist eine Mischung aus Rot und Blau und hat z.B. folgende RGB-Werte: R=100, G=0, B=100.

Nächste Aufgaben:

- a) Überlege dir wie du einen Radierer programmieren kannst.
- b) Füge einen „Slider“ ein, über den du die Zeichenstärke einstellen kannst.



Du brauchst diese Komponenten:

- 1 Canvas (Width und Height = Fill Parent)
- 1 HorizontalArrangements
- 3 Buttons (Shape = oval)
- 2 Buttons** (Shape = default)
- 1 Slider** (**Wichtig**: Diese Werte unter Properties einsetzen)



Bild Dateien für diese App findest du hier:

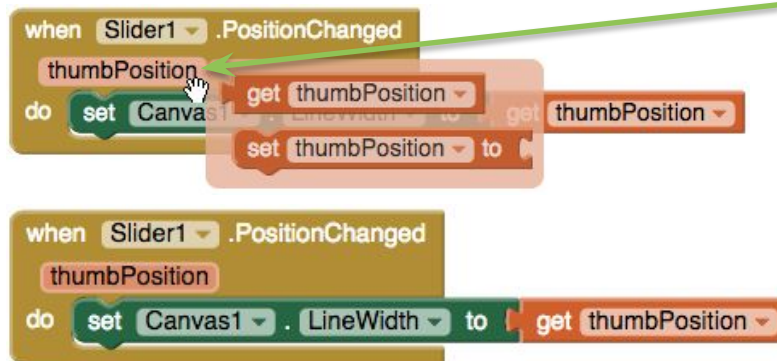
appcamps.link/zeichnapp

Was passiert hier?

a) Mit dem **Radierer** kannst du mit deinem Finger Dinge „wegradieren“. Hierzu bitte als **PaintColor** des Canvas die Hintergrundfarbe des Canvas auswählen.



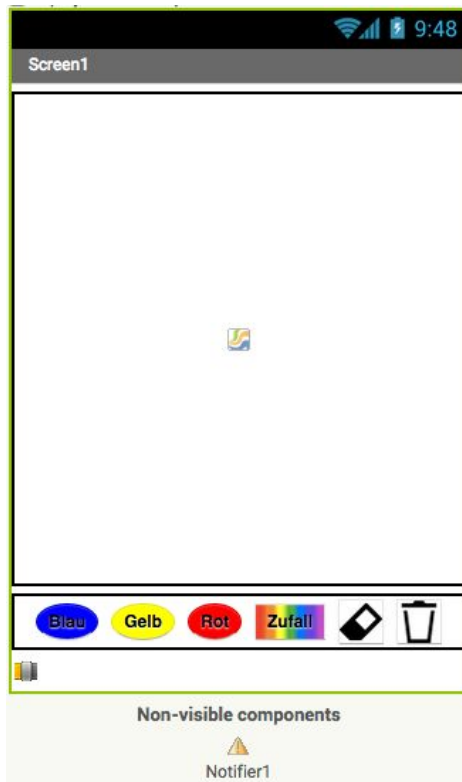
b) Jedes Mal wenn der **Slider** verändert wird (*Slider.PositionChanged*) weisen wir dem Canvas eine neue Zeichenstärke zu (set *Canvas.LineWidth*).



Tipp: Mit der Maus über **thumbPosition** gehen, dann wird dir **get thumbPosition** angeboten. Das fügst du dann hier ein.

Nächste Aufgaben:

Füge einen Lösch-Button ein, der das gesamte Canvas „bereinigt“. Frage den Benutzer beim Klick auf den Button, ob er wirklich alles löschen möchte. Hierfür brauchst du einen **Notifier**. Diesen findest du im Design Bereich in der Palette User Interface.



Du brauchst diese Komponenten:

- 1 Canvas (Width und Height = Fill Parent)
- 1 HorizontalArrangements
- 3 Buttons (Shape = oval)
- 3 Buttons** (Shape = default)
- 1 Slider
- 1 Notifier**

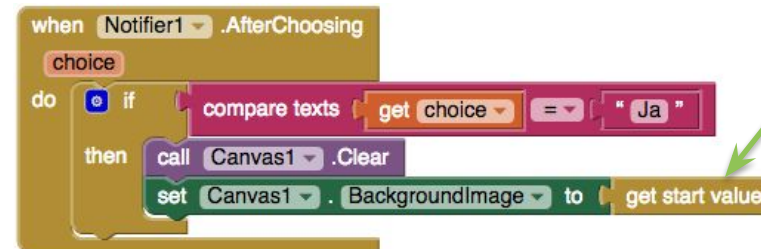
Bild Dateien für diese App findest du hier:
appcamps.link/zeichenapp

Was passiert hier?

Wenn der Benutzer auf den Löschen Button klickt, erscheint ein Warn-Dialog und der Benutzer muss bestätigen, dass er löschen möchte. Hierfür benutzt du den **Notifier**.



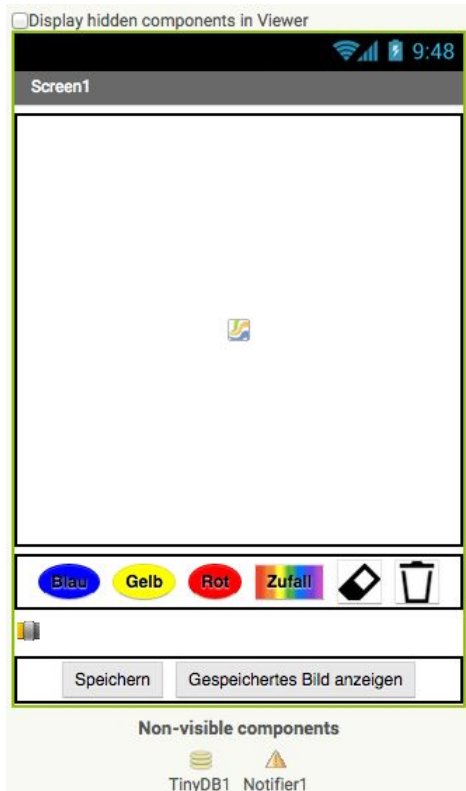
Wenn der Benutzer beim Warn-Dialog „Ja“ antwortet, wird das Bild gelöscht. Für den Fall, dass der Benutzer „Nein“ antwortet musst du nichts programmieren. Der Warn-Dialog wird geschlossen. Mehr passiert nicht.



Folgenden Block findest du unter
Built-in → **Control**

Nächste Aufgaben:

- a) Füge einen Button ein, um das Bild zu speichern. Für die Speicherung benötigst du eine **TinyDB**.
- b) Füge einen Button ein, um das zuletzt gespeicherte Bild anzuzeigen.



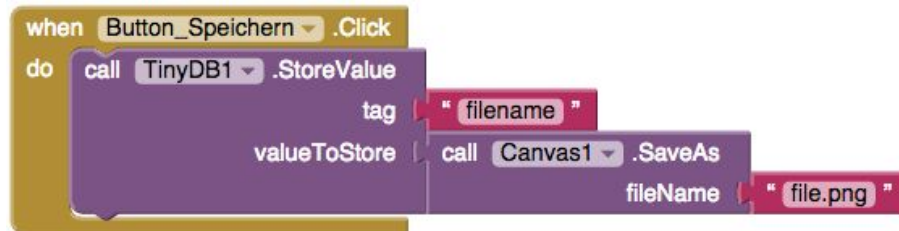
Du brauchst diese Komponenten:

- 1 Canvas (Width und Height = Fill Parent)
- 2 HorizontalArrangements
- 3 Buttons (Shape = oval)
- 4 Buttons (Shape = default)
- 1 Slider
- 1 Notifier
- 1 TinyDB

Bild Dateien für diese App findest du hier:
appcamps.link/zeichenapp

Was passiert hier?

Wenn der Benutzer auf den Speichern-Button klickt, wird das Bild in der **TinyDB** gespeichert. Die **TinyDB** ist eine Datenbank auf dem Handy bzw. dem Emulator. Über das „tag“ (hier: filename) definierst du, wie du später auf die Datei zugreifen kann.



Wenn der Benutzer auf den Anzeigen-Button klickt, wird das zuletzt gespeicherte Bild im Canvas angezeigt. (In diesem Beispiel kann immer nur ein Bild gespeichert werden.) Über das „tag“ rufst du die entsprechende Datei auf.



Weitere Ideen?

Hast du weitere Ideen für die Zeichen App? Passe die App entsprechend an! Viel Spaß!