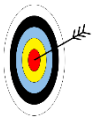
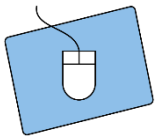
(3) Button-Klick & Texteingabe



Nachdem ihr jetzt Buttons und ein Eingabefeld hinzugefügt habt, wollt ihr sie sicherlich auch benutzen können und festlegen, was bei einem Klick oder Eintrag passieren soll! Dazu taucht ihr nun in die **Java-Programmierung** ein und lernt …

- …das Konzept von **Activities**,
- …die **Besonderheiten** im Java-Code bezüglich eines Android-Projekts und
- …die **Verknüpfung** zwischen XML-Files und Java kennen.



1. Öffnet als Erstes im ersten Unterordner des **Java-Quellcode-Ordners** die Datei *Startbildschirm.java*, die Android Studio bei der Erstellung eures Projektes direkt angelegt hat (Abbildung 1)!
2. Seht euch den Java-Code an und versucht, zu identifizieren, was euch bereits aus der Schule oder allgemein bekannt ist und welche Programmteile neu für euch sind! Die folgende Tabelle kann euch bei der **Gegenüberstellung** helfen:

BEKANNTES	NEUES

Activities

Die bereits angelegte Klasse Startbildschirm **erbt** von der Klasse Activity. **Activities** sind so etwas wie das **Grundkonzept** von Android: eine App besteht aus mehreren, lose miteinander verbundenen Activities, die verschiedene Bildschirme realisieren und sich untereinander aufrufen können.

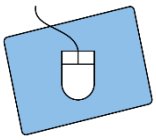
```
1   package de.sli.schiffeversenken;
2
3   import android.support.v7.app.AppCompatActivity;
4   import android.os.Bundle;
5
6   public class Startbildschirm extends AppCompatActivity {
7
8       @Override
9       protected void onCreate(Bundle savedInstanceState) {
10          super.onCreate(savedInstanceState);
11
12          setContentView(R.layout.activity_startbildschirm);
13      }
14  }
15
```

Abbildung 1: Die automatisch generierte Klasse Startbildschirm.java

(3) Button-Klick & Texteingabe

Jede App hat eine **Haupt-Activity**, die beim Starten als Erstes geöffnet wird – in eurem Fall also die von Beginn an erstellte *activity_startbildschirm.xml*. Die onCreate()-Methode, die ihr schon in der Java-Datei zur Activity seht (**Abb. 1, S. 1**), sind dazu da, alles zu laden, was beim App-Start benötigt wird.

Diese Methoden belassen wir für sich und kümmern uns jetzt erst einmal wieder um die XML-Datei zu unserem Startbildschirm.



1. Wechselt über den Reiter unter eurem Vorschaufenster in die **XML-Ansicht** eures Startbildschirms!
2. Sucht nach eurem Button, der das Spiel starten soll. Die nebenstehende Abbildung zeigt euch, wie der Eintrag ungefähr aussehen könnte.
   [**Hinweis:** *Ihr könnt den Button identifizieren, indem ihr nach der rot markierten Zeile sucht und dort den Namen eures Buttons findet!*]

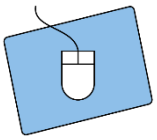
```
<Button
    android:id="@+id/button_start"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:onClick="starteSpiel"
    android:text="@string/startbutton_text" />
```

Abbildung 2: XML-Code für den Start-Button

Irgendetwas fehlt aber noch bei euch… Richtig! Die vorletzte Zeile, die ihr in Abb. 2 seht! Schauen wir uns die einmal etwas genauer an:

```
android:onClick="starteSpiel"
```

- `android:` — *sagt, dass es sich um einen Android-spezifischen Befehl handelt*
- `onClick` — *bedeutet, dass der Button eine Klick-Funktion hat*
- `"starteSpiel"` — *der Name der Funktion (wie sie auch im Java-Code heißen wird)*



3. Okay, jetzt fügt diese Zeile eurem Button in der XML-Datei hinzu! Den Namen der Funktion könnt ihr dabei frei, aber bitte sinnvoll, wählen. ;-)
4. Achtet auf **Doppelpunkt**, **Anführungszeichen** etc. und darauf, dass ihr sonst nichts verändert, und speichert die Datei!

Gut, damit hättet ihr dem Android-Projekt schon einmal gesagt, dass der Start-Button bitteschön **klickbar** sein soll und wie die Funktion heißen soll. Was bei dem Klick passieren soll, muss jetzt im nächsten Schritt im Java-Code implementiert werden…

Schaut euch vorher nochmal den XML-Code zum Button an: Welche Einstellungen aus den Properties erkennt ihr hier wieder?

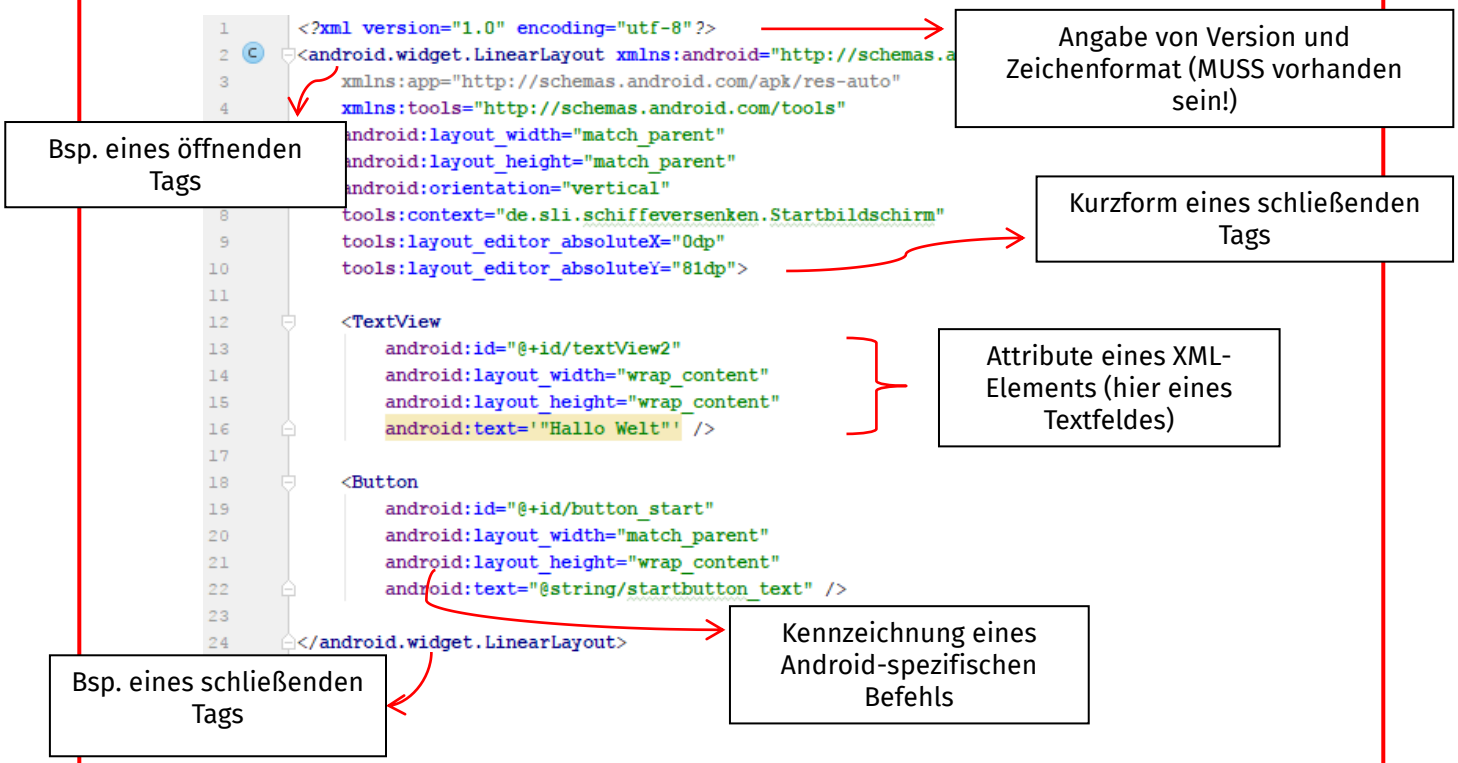
(3) Button-Klick & Texteingabe

XML

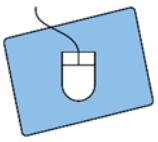
Jetzt habt ihr schon die ganze Zeit von diesem **XML** gelesen und sogar schon Codezeilen verändert und geschrieben! Da wird es doch Zeit, einmal grundlegend zu erzählen, was das denn für eine Sprache ist…

XML (engl.: ***E**xtensible **M**arkup **L**anguage* – dt.: *Erweiterbare Auszeichnungssprache*) ist eine Sprache, mit der Daten – Websites, Layouts etc. - **strukturiert** werden können. Anders als in einem Editor (z.B. dem Drag-'n'-Drop Editor in Android Studio oder auch Word) ist das Ergebnis dabei **nicht sofort sichtbar**, sondern man „programmiert" quasi das Aussehen. Möchte man z.B. beim Gestalten einer Website mit **HTML** – auch eine Auszeichnungssprache – ein Wort fett markieren, würde man nicht auf eine Schaltfläche klicken, sondern das zu markierende Wort zwischen ein öffnendes und schließendes **Tag** schreiben: Beispiel

Schauen wir uns ein XML-Dokument bei der Android-Programmierung einmal genauer an:



Wie ihr seht, ist ein XML-Dokument **hierarchisch** aufgebaut. Elemente (hier: **Textfeld und Button**) können innerhalb anderer Elemente angeordnet werden, sodass dadurch ein Design erstellt werden kann.

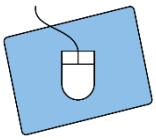


Wie sieht der Bildschirm unseres kleinen Beispiels also aus? Zeichnet ihn kurz auf ein Blatt Papier und versucht, daran den XML-Code nachzuvollziehen (Erklärungen zu Layouts folgen auf dem nächsten AB.)!

(3) Button-Klick & Texteingabe

Button-Klick implementieren

Wechselt jetzt wieder in die Datei *Startbildschirm.java* und setzt euren Cursor unter die beiden vorhandenen `onCreate()`-Methoden. Hier wollen wir die Methode schreiben, die beim Klick auf den Start-Button aufgerufen wird!



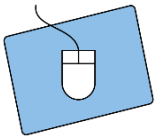
1. Implementiert hier eine Methode, die **öffentlich** und **ohne Rückgabewert** ist und als Namen den Eintrag aus eurer XML-Datei erhält!
2. Wählt als **Parameter** den Eintrag View v.
   Warum wird hier wohl ein View-Element übergeben? Erinnert euch noch einmal daran, woher ihr den Button im Drag-'n'-Drop Editor genommen habt…

   _____
3. Lasst den Methodenrumpf noch **leer**! Zum Starten des Spiels muss die aktuelle Activity gewechselt werden – und was dahintersteckt, erfahrt ihr erst auf dem nächsten Arbeitsblatt.

Durch den Parameter weiß Android, dass die Methode zu einem **View-Element** (hier: einem Button) gehört und durch den Namen weiß Android auch, dass sie genau zum Start-Button gehört.

Finish!

Jetzt müsst ihr nur noch implementieren, was genau beim **Button-Klick** geschehen soll. Beim Start-Button soll erstmal nichts passieren, damit ihr aber auch einmal ein Ergebnis sehen könnt, sollt ihr jetzt den Button zum Beenden des Spiels programmieren!



Übt nun an dem zweiten Button alles, was ihr bis hierher gelernt habt.

1. Sucht und identifiziert den Button in der XML-Datei.
2. Fügt den Eintrag zur Definition eines Button-Klicks hinzu. Denkt an einen guten und sinnvollen Namen!
3. Implementiert die Methode im Java-Code…
4. …und fügt dieses Mal den Aufruf finish() in den Methodenrumpf ein. Dieser dient schlichtweg dazu, die App beim Klick sofort zu beenden.

Und wie geht's weiter?!

Testet die App und überprüft, ob sie sich auch wirklich beenden lässt!!!

Auf den nächsten Arbeitsblättern wollen wir uns dann damit beschäftigen, wie ihr das **Texteingabefeld vorbereiten** müsst, damit der Name des Benutzers auch in die neue Ansicht des Spielbildschirms übernommen wird und während des Spiels auf dem Bildschirm erscheint.
Jetzt habt ihr alle Vorbereitungen getroffen, um euer Projekt auf die nächste Stufe zu heben! Die nächsten Blätter führen euch durch die **Erstellung einer neuen Activity** und ihr könnt den Spielbildschirm mit den Funktionen des Schiffe-Versenken-Spiels erstellen.

Quelle:  ,  ,  ,  sowie die Screenshots aus Android Studio angefertigt vom InfoSphere-Team