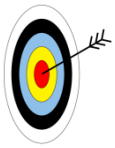
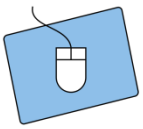
(5) Das Spielfeld



Mit Erreichen von Arbeitsblatt 5 habt ihr es bereits geschafft euer Spiel zu starten. Das ist schon einmal super. Allerdings kann noch nicht viel gespielt werden. Das sollt ihr jetzt ändern. Arbeitsblatt 5 legt dazu die Grundlage, indem ihr das komplette Layout eurer **Spielansicht** designen werdet. Dazu zählen...

- ... ein *Weiter*-Button um nach dem Zug des Computers weiter zu spielen.
- ...eine Anzeige für den Namen des aktiven Spielers.
- ...eine Anzeige für Informationen über den Spielverlauf (z.B. Treffer).
- ... ein 8x8 großes Spielfeld.



Öffnet zunächst das XML-Layout eurer Spielansicht. In diesem werdet ihr die meiste Zeit arbeiten. Wechselt von der „Drag'n'drop"-Ansicht in die XML-Ansicht.

Anordnung einzelner Elemente

Ihr habt nun ein leeres XML-Layout vor euch. Dieses Layout wird eure Spielansicht, welche die vier oben genannten Elemente enthalten soll. Bevor ihr euch diesen Elementen widmen könnt, braucht ihr allerdings ein paar Informationen über den Aufbau eines XML-Dokuments bzw. wie ihr die Elemente in diesem übersichtlich anordnen könnt. Ihr benötigt dafür ein **LinearLayout**.

Lasst euch nicht von dem Namen verwirren. So genannte LinearLayouts sind XML-Elemente die ihr in eurem eigentlichen Layout verwenden könnt um Objekte anzuordnen.

Diese LinearLayouts geben euch zwei Optionen wie ihr Buttons etc. anordnen könnt. Ihr könnt zwischen der Option *horizontal* und *vertical* wählen. Was diese bedeuten, werdet ihr auf der nächsten Seite erfahren.

(5) Das Spielfeld

LinearLayout (Horizontal)	LinearLayout (Vertical)
Alle Elemente, die ihr in ein horizontales LinearLayout einfügt, werden **nebeneinander** angeordnet.	Alle Elemente, die ihr in ein vertikales LinearLayout einfügt, werden **untereinander** angeordnet.
XML-Code	**XML-Code**
<LinearLayout android:layout_width="match_parent" android:layout_height="wrap_content" android:orientation="horizontal" > HIER WERDEN ELEMENTE HINZUGEFÜGT </LinearLayout>	<LinearLayout android:layout_width="wrap_content" android:layout_height="match_parent" android:orientation="vertical" > HIER WERDEN ELEMENTE HINZUGEFÜGT </LinearLayout>

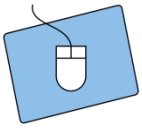
Zusätzlich muss in jedem XML-Dokument ein so genannter **namespace** deklariert werden. Dieser muss immer im ERSTEN Element des XML-Dokuments hinzugefügt werden. Er erlaubt es euch Anweisungen wie „android:layout_width“ zu benutzen. Hier ein kleines Beispiel:

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <android.widget.LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
3      xmlns:app="http://schemas.android.com/apk/res-auto"
4      xmlns:tools="http://schemas.android.com/tools"
5      android:layout_width="match_parent"
6      android:layout_height="match_parent"
7      android:orientation="vertical"
8      tools:context="de.sli.schiffeversenken.Spielansicht">
9
10     <LinearLayout
11         android:layout_width="match_parent"
12         android:layout_height="wrap_content"
13         android:orientation="horizontal" >
14
15         <Button
16             android:id="@+id/button1"
17             android:layout_width="wrap_content"
18             android:layout_height="wrap_content" />
19
20         <Button
21             android:id="@+id/button2"
22             android:layout_width="wrap_content"
23             android:layout_height="wrap_content" />
24     </LinearLayout>
25
26     <Button
27         android:id="@+id/button3"
28         android:layout_width="wrap_content"
29         android:layout_height="wrap_content" />
30
31 </android.widget.LinearLayout>
```

Hier wird der namespace deklariert. Diese Zeile ist immer identisch.

Ein horizontales LinearLayout mit 2 Buttons in einem vertikalen Layout.

Ein dritter Button. Dieser ist im äußeren (vertikalen) LinearLayout.



Schreibt das Beispiel **NICHT** ab, sondern überlegt euch welche Anordnung der Button 1, 2 und 3 theoretisch erzeugt werden würde.

Auf der nächsten Seite findet ihr die Lösung.

(5) Das Spielfeld

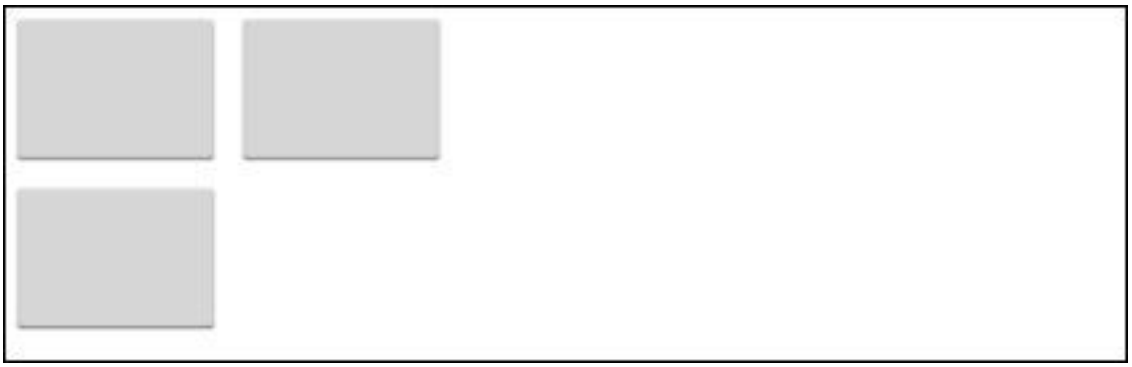
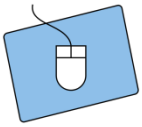


Abbildung 1: Durch den Beispielcode erzeugte Anordnung dreier Buttons



Als nächstes müsst ihr euer eigenes Layout erzeugen. Durch das vorige Beispiel wisst ihr schon wie das geht. Überlegt euch wie ihr die Elemente *Weiter-Button*, *Label(aktiverSpieler)*, *Label(Spielinformation* und *das Spielfeld* anordnen wollt.

1. Erzeugt nur das Layout (kombinieren von LinearLayouts)
2. Fügt mit Hilfe der Informationen im nächsten Abschnitt die Elemente ein

Der Weiter-Button

Der Weiter-Button soll, wie der Name schon sagt, ein Button werden. Als *android:id* soll er den Wert „weiterButton“ bekommen. Als Text soll „Weiter“ auf dem Button stehen. Zusätzlich sollt ihr eine onClick-Funktion für den Button anlegen, deren Namen ihr euch aussuchen dürft.

Denkt daran den String „Weiter“ eurer Strings.xml hinzuzufügen.

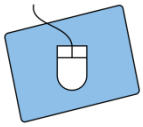
Der aktive Spieler

Die Anzeige des aktiven Spielers soll ein reines Textfeld sein. Was könntet ihr zur Darstellung von Text benutzen? Die *android:id* des gewählten XML-Elements soll „aktiverSpieler“ heißen. Der Name soll noch nicht angezeigt werden, also soll der Text ein leerer Text („“) sein.

(5) Das Spielfeld

Die Spielinformationen

Zur Anzeige von Spielinformationen, wie Anzahl der Treffer o.ä., braucht ihr eine weitere Textanzeige. Die *android:id* soll „spielinformation“ heißen.



Nachdem ihr diese drei Elemente in euer Layout eingefügt habt, könnt ihr direkt einmal die App testen und schauen was passiert, wenn ihr ein neues Spiel startet.
Überlegt euch im Vorfeld wie es aussehen soll und überprüft, ob es auch tatsächlich so ist.

Das 8x8-Spielfeld

Bevor ihr mit dem Spielfeld weiter macht ist es wichtig, dass der vorige Teil funktioniert. Da 64 neue Elemente (keine Angst die meisten sind Copy/Paste) hinzugefügt werden, kann das XML-Dokument sonst leicht unübersichtlich werden.

Für das Spielfeld werdet ihr eine neue Form von XML-Elementen kennenlernen. Die so genannten **TableLayouts**. Diese dienen euch als einfache Tabelle mit einzelnen Reihen. Bei einem 8x8 Spielfeld braucht ihr also 8 Reihen mit jeweils 8 Elementen. Der XML-Code für ein TableLayout sieht folgendermaßen aus (diesen dürft ihr für euer Spielfeld abschreiben):

```
<TableLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content" >

        <TableRow
                android:id="@+id/tableRow1"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:weightSum="8">

                HIER DIE ELEMENTE DER EINZELNEN REIHEN EINTRAGEN

        </TableRow>

</TableLayout>
```

In dem Beispiel kommt ein, euch unbekannter, XML-Befehl vor: *Android:weightSum* werdet ihr benutzen damit alle Elemente in eurem Spielfeld gleich groß sind. Der Wert 8 sagt der Reihe daher, dass sie in 8 gleich große Teile aufgespalten werden soll.

(5) Das Spielfeld

Nun geht es weiter mit dem Copy/Paste-Teil.
Jedes Element im Spielfeld soll ein Button werden. Diese Buttons sollen keinen Text aber dafür verschiedene andere Parameter erhalten.

android:id
Als Id werden die Buttons wie eine Matrix durchnummeriert. **Der erste Button in der ersten Reihe kriegt die android:id="@+id/spiel00". Oder anders formuliert: „spiel(Reihe)(Spalte)"**
Wie sehen dann die weiteren ids aus? Wenn ihr euch nicht sicher seid, fragt einen Betreuer.

android:onClick
Die *android:onClick-Funktion* wird diesmal nicht benutzt.

android:layout_weight
Der letzte wichtige Parameter ist *android:layout_weight*. Dieser soll folgenden Wert erhalten: android:layout_weight="1".

Dieser Parameter muss dann genutzt werden wenn ihr eine weight_sum deklariert habt. Er wird einem Element zugewiesen und gibt an, wie groß das Element sein soll. Haben alle Elemente den Wert 1, dann sind sie gleich groß. Hat ein Element den Wert 2 ist es doppelt so groß wie die anderen.
Achtung: Die Summe aller *layout_weight* muss genau gleich der *weight_sum* sein.

android:layout_width/height
Diese dürft ihr nicht vergessen. Überlegt euch einfach selber was Sinn ergibt.

Jetzt habt ihr schon die gesamte Spielansicht fertig. Fehlt nur noch die Verknüpfung mit dem eigentlichen Spiel.