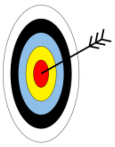
(6) Verknüpfen der GUI mit dem Spiel

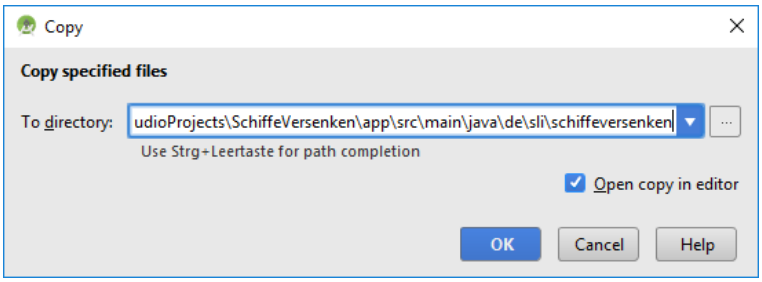


Das Einzige was eurer App jetzt noch fehlt, ist die Verknüpfung eurer GUI mit dem Spiel „Schiffe Versenken“. Damit auch alles perfekt funktioniert, werdet ihr…

- …als Vorbereitung einige neue Elemente und Variablen hinzufügen,
- …dafür sorgen, dass das Spielfeld korrekt gezeichnet wird,
- …den Wechsel zwischen drei verschiedenen Spielzuständen implementieren.

Vorbereitungen (1)

Als erstes werdet ihr die Java-Datei: „Spiel.java“ in euer Projekt importieren. Die Dateien findet ihr auf dem Desktop im Ordner „GUI-Programmierung“. Zieht sie per Drag’n’Drop in den Ordner src. Ihr bekommt folgende Benachrichtigung , die ihr einfach bestätigt:



Vorbereitungen (2)

Als nächstes werdet ihr das Spiel mit eurer GUI verknüpfen. Um euch diesen Teil etwas einfacher zu machen haben wir in der *spiel.java* einen Teil des Programmcodes im oberen Bereich auskommentiert.

Ihr erkennt auskommentierten Code in Java daran, dass er folgendermaßen umschlossen ist:

```
/*
        auskommentierter Programmcode
*/
```

Den Programmcode ohne die Symbole (/* und */) kopiert ihr in eure *Spielansicht.java.* Der erste Teil wird direkt in eure onCreate()-Methode kopiert. Der zweite Teil wird ans Ende des Dokuments kopiert. Beachtet dabei die geschweiften Klammern.

Als letztes müsst ihr noch folgende Variablen erstellen:

Im Kopf der Spielansicht.java

```
int spielZustand;
Spiel spiel;
```

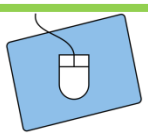
(6) Verknüpfen der GUI mit dem Spiel

Das Spielfeld zeichnen

Damit habt ihr alle Vorbereitungen getroffen um mit dem eigentlichen Spiel zu beginnen. Die Optik des Spielfeldes habt ihr auf dem letzten Arbeitsblatt bereits erstellt (8x8-Buttons). Nun müsst ihr dafür sorgen, dass euer Spielfeld auch wie ein „Schiffe Versenken"-Spielfeld aussieht. Erstellt dazu zunächst die Methode in eurer SpielansichtActivity:

```
zeichneSpielfeld (int spieler){}
```

Auf der nächsten Seite werdet ihr Informationen dazu bekommen was diese Methode machen soll und welche Hilfsfunktionen ihr zur Verfügung habt.



Implementiert mit den Hilfen auf der nächsten Seite die Methode *zeichneSpielfeld()*.

Was soll in zeichneSpielfeld() passieren?
Die Methode soll das komplette Spielfeld, also alle 64 Buttons, durchlaufen und für jeden Button eine Anfrage an das Spiel schicken. Diese Anfrage dient dazu herauszufinden, wie der aktuelle Zustand dieses Buttons ist. Mögliche Antworten wären ein Wasserfeld, ein Schiff, ein Wassertreffer oder ein Schiffstreffer. Sobald ihr wisst um welchen Typ es sich handelt, müsst ihr den entsprechenden Button entsprechend einfärben. Eine Hilfsfunktion dafür findet ihr im nächsten Abschnitt.

Jetzt könnt ihr euch vielleicht denken, warum der Parameter *int spieler* übergeben wird. Denn die Ansicht mit Wasserfeld, Schiffen und Treffern gibt es sowohl für den Benutzer also auch für den Computer.

Eure Hilfsfunktionen/Hilfen

- Überlegt euch zunächst wie ihr ein Spielfeld durchlaufen könnt. Tipp: for-Schleife
- Um den Status eines Buttons abzufragen ruft ihr

  ```
  spiel.getFeld(int x, int y, Spieler spieler)
  ```

  Achtung: Die beiden Integer beim Aufruf müssen einen Wert haben. Wo kriegt ihr den wohl her? (for-Schleifen?)
  //Diese Funktion liefert euch einen Wert zwischen 0-3. Mehr //dazu im nächsten Punkt.
- Den erhaltenen Wert könnt ihr mit folgenden (Integer)-Werten vergleichen und wisst dadurch um was für ein Feld es sich handelt:
  - spiel.WASSER
  - spiel.SCHIFF
  - spiel.WASSERTREFFER
  - spiel.SCHIFFTREFFER
  - spiel.FELDGROESSE

  spiel.X (mit X=Wasser etc.) sind Integer Variablen, die ihr abfragen könnt.

  Um euren Button anschließend zu verändern (einzufärben), müsst ihr zunächst den richtigen Button finden. Dazu benutzt ihr die zu Beginn kopierte Funktion:

  ```
  Button getButton(int x, int y)
  // gibt einen Button an der Position (x,y) zurück
  ```

  Um einen Button einzufärben, könnt ihr einfach folgende Funktion verwenden:

  ```
  button.setBackgroundColor(Color.BLUE)
  ```

  Es gibt noch mehr Farben als „BLUE". Probiert verschiedene aus oder schaut, was euch Android Studio vorschlägt wenn ihr „Color." hinschreibt.

(6) Verknüpfen der GUI mit dem Spiel

Die drei Spielzustände

Die Spielzustände beschreiben den Spielablauf. Dabei kennt das Spiel die folgenden drei Zustände:

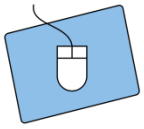
A.) Der Spieler sieht **sein** Spielfeld.
B.) Der Spieler sieht das Spielfeld des Gegners (zum Schießen).
C.) Der Spieler sieht das Spielfeld des Gegners um zu sehen, ob er getroffen hat.

Ihr sollt im Folgenden die Übergänge zwischen den Zuständen implementieren, das heißt den Übergang von A -> B, von B -> C und von C -> A. Diese Übergänge werden in zwei Methoden implementiert.

- C -> A und A -> B in der Methode eures *Weiter*-Buttons.
- B -> C in der zu Anfang kopierten Methode `buttonGedrueckt(int x, int y)`.

Eure Hilfsfunktionen
Um die Spielzüge zu implementieren braucht ihr die bereits angelegte Variable *spielZustand*. Ihr sollt über diese Variable die drei Zustände einführen, d.h. wenn die Variable == x ist, dann seid ihr in Zustand x. Überlegt euch also welchen Startwert die Variable bekommen soll und nutzt anschließend folgende Funktionen um die Zustände in den oben genannten Funktionen zu implementieren:

```
zeichneSpielfeld(spiel.COMPUTER);
```
Diese Funktion lädt die Felder der Computerspielers in das 8x8 Spielfeld.
```
zeichneSpielfeld(spiel.MENSCH);
```
Hier werden die Felder des Spielers geladen.
```
spiel.computerSchiesst();
```
Der Computer schießt auf das Spielfeld des Spielers.
```
spiel.spielerSchiesst(x,y);
```
Der Spieler schießt auf das Feld (x,y) des Computers.



Implementiert die Spielübergänge. Falls ihr eine Idee habt, aber nicht wisst, ob die Umsetzung so richtig ist, dürft ihr einfach die Betreuer fragen.

Denkt daran häufig zu testen!

(6) Verknüpfen der GUI mit dem Spiel

Wie geht's weiter

Ihr habt jetzt die Möglichkeit noch weitere Individualisierungen vorzunehmen. Zum Beispiel hattet ihr am Anfang einen Spielernamen eingegeben. Diesen könnt ihr in der Spielfeld-Activity folgendermaßen auslesen:

```
name = getIntent().getStringExtra("Name");
```

Anschließend könntet ihr den Namen verwenden um anzuzeigen welcher Spieler gerade am Zug ist. Über das Label Spielinformation kann ausgegeben werden, ob der Spieler getroffen hat. Diese Information findet ihr über

```
spiel.spielerSchiesst(int x, int y)
```

heraus. Denn diese Funktion liefert bei einem Treffer `true` zurück.
Außerdem könnt ihr noch folgendes machen:

- Eine Siegbedingung einfügen

  ```
  if(spiel.hatGewonnen(spiel.MENSCH))
  ```

  Abfrage im Übergang B -> C

  ```
  if(spiel.hatGewonnen(spiel.COMPUTER))
  ```

  Abfrage im Übergang C -> A

  ```
  finish()
  ```

  beendet das Spiel und bringt euch zurück auf den Startbildschirm. Wann soll das ausgeführt werden und wie könnte man diesen Zustand beschreiben? (Tipp: Neuen Zustand einführen und abfragen)
- Textuelle Ausgaben erweitern bzw. Rückmeldungen für einzelne Treffer einfügen.
- Den Weiter-Button entfernen und ein Spiel ohne diesen erstellen.
- Den Spieler selbst Schiffe setzen lassen.
- Ihr habt bisher für Farben die Klasse Color benutzt. Besser wäre die Farben über die Android-Ressourcen abzurufen mit:

  ```
  getResources().getColor();
  ```