Die Hausaufgaben müssen von jedem Studierenden einzeln bearbeitet und abgegeben werden. Für die Hausaufgabe sind die aktuellen Informationen vom Blog `https://seblog.cs.uni-kassel.de/ws1920/programming-methodologies/` zu berücksichtigen.

**Abgabefrist ist der 27.02.2020 - 23:59 Uhr**

## Abgabe

Wir benutzen für die Abgabe der Hausaufgaben Git. Jedes Repository ist nur für den Studierenden selbst, sowie für die Betreuer und Korrekteure sichtbar.

Für die Bonushausaufgabe benötigt ihr ein neues Repository, das über folgenden Link angelegt wird:

`https://classroom.github.com/a/BiVfk14X`

**Nicht, oder zu spät gepushte (Teil-)Abgaben werden mit 0 Punkten bewertet.**

## Vorbereitung

Zur Bearbeitung der Hausaufgabe sollte eine Entwicklungsumgebung verwendet werden. Wir empfehlen aufgrund der Nachvollziehbarkeit die Verwendung von IntelliJ (siehe Aufgabenblatt 3). Die Abgabe muss als **lauffähiges** Projekt abgegeben werden.

**Zur Abgabe der Hausaufgaben darf ein beliebiges Git-Tool genutzt werden**

### Gitignore

Füge eine Gitignore zu deinem Projekt hinzu. Diese muss im Root-Verzeichnis des Projektes liegen (also parallel zu dem .git-Verzeichnis). Ihr findet diese Gitignore zum Herunterladen auf unserem Blog neben dem Link zu dieser Hausaufgabe 3.

Erst nach erfolgreichem Hinzufügen der Gitignore sollt ihr den ersten Commit machen und das Projekt in das Repository pushen.

**Ein nicht-Einbinden der Gitignore führt zu Punktabzug.**

# Aufgabe 1 - Konzeption

In diesem Teil der Bonusaufgabe sollen alle Schritte zur Konzeption einer Anwendung durchlaufen werden. Zusammenfassend werden zunächst User-Szenarien erstellt, welche die Spielregeln beschreiben und anschließend Objekt- beziehungsweise Klassendiagramme daraus generiert.

**Achte darauf, das Repository der aktuellen Hausaufgabe zu verwenden.**

## 1.1 User-Szenarien

Für die erste Aufgabe sollen zwei textuelle Szenarien zu konkreten Spielsituationen des Spiels „Tic-Tac-Toe“ erstellt werden. Die Szenarien sollten jeweils in Englisch verfasst sein. Die Regeln des klassischen Spiels können hier noch einmal nachgelesen werden.

Für diese Hausaufgabe führen wir jedoch ein paar extra Regeln ein:

1. Zu Beginn einer Partie kann ausgewählt werden ob auf einem 3x3, 4x4 oder 5x5 Feld gespielt wird.
2. Um auf einem Feld zu Gewinnen muss eine gerade Linie aus Symbolen gelegt werden, welche die selbe Anzahl besitzen muss wie die Kantenlänge des Feldes. (Also bei 4x4 müssen es 4 Symbole in einer waagerechten, senkrechten oder diagonalen Linie sein)

Lege die 2 erstellten .txt Dateien in einem Ordner mit dem Namen ”taskOnePointOne” in deinem Repository ab.

## 1.2 Objektdiagramme

Leite für das folgende User-Szenario und für die beiden User-Szenarien aus der vorangegangenen Aufgabe passende Objektdiagramme ab. Erstelle dazu jeweils ein Objektdiagramm zur Start- und Endsituation. Für jedes der 3 Szenarien müssen somit zwei Diagramme entstehen. Benenne die Dateien eindeutig (beispielsweise ”<Szenariotitel><Start | End>”)

**Title: Win Game**

| | |
|---|---|
| **Start:** | Alice and Bob are playing Tik-Tak-Toe on a 3x3 Board. Alice uses circles, and Bob uses crosses. Alice has one circle in the upper left and middle left field. Bob has a cross in the upper right and middle right field. It’s Alice’s turn. |
| **Action:** | Alice puts a circle in the lower left corner. |
| **Result:** | Alice managed to create a line of 3 circles. Alice won the game. |

Lege die 6 erstellten Dateien in einem Ordner mit dem Namen ”taskOnePointTwo” in deinem Repository ab.

### 1.3 Klassendiagramm

Erstelle ein Klassendiagramm, welches sämtliche Objektdiagramme aus Aufgabe 1.2 vereinigt. Dies bedeutet, dass das Klassendiagramm eine klare Abstraktion der Objektdiagramme darstellt.

Lege die erstellte Datei in einem Ordner mit dem Namen ”taskOnePointThree” in deinem Repository ab.

## Aufgabe 2 - Initialisierung

Nach Abschluss der Konzeption der Anwendung soll das Projekt initialisiert werden. Dafür soll fublib.org genutzt werden.

`https://www.fulib.org/`

Schreibt eine Szenario, welches das Modell aus Aufgabe 1.3 erzeugt. Ladet im Anschluss wie in Hausaufgabe 04 das Projekt als .zip herunter und bewegt es in das Git-Repository. Das so erstellte Gradle-Projekt dient nun als Grundstein für die folgende Anwendung.

## Aufgabe 3 - Spiel

Nach dem Abschließen aller Vorbereitungen soll nun das Spiel umgesetzt werden. Abbildung 1 und 2 zeigen Beispiele wie die zwei Ansichten der Anwendung aussehen könnten.

Bei der Umsetzung ist zwingend darauf zu achten beide Ansichten in getrennten FXMLs zu erstellen. Jede dieser Dateien soll ebenfalls eine eigene Controllerklasse erhalten.

Wie in der Anwendung "Shroom Wars" sind alle schreibenden Datenmodell-Zugriffe durch eine Klasse mit dem Namen ModelBuilder sowie die bekannten build<Entity> und remove<Entity> Methoden zu behandeln.

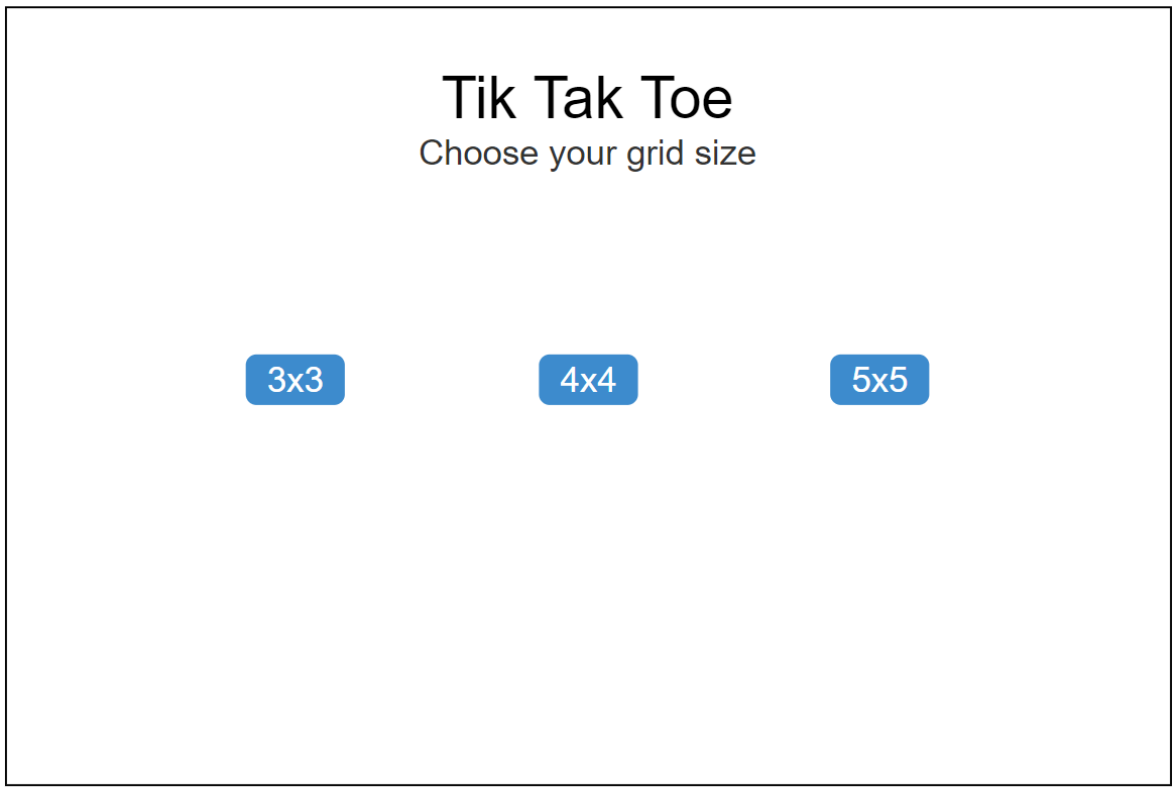


Abbildung 1: Setup View

Sobald ein Spiel gewonnen oder verloren wurde soll die Ansicht zurück zur Ansicht 1 wechseln, sodass sofort wieder in ein neues Spiel eingestiegen werden kann.

In der Aufgabenstellung sind bewusst Lücken gelassen. Die nicht beschriebene Funktionalität soll in sofern umgesetzt werden, dass das Spiel vollständig spielbar ist. Die Art der Umsetzung und die darin verwendeten Techniken und Methoden obliegt jedem selbst.

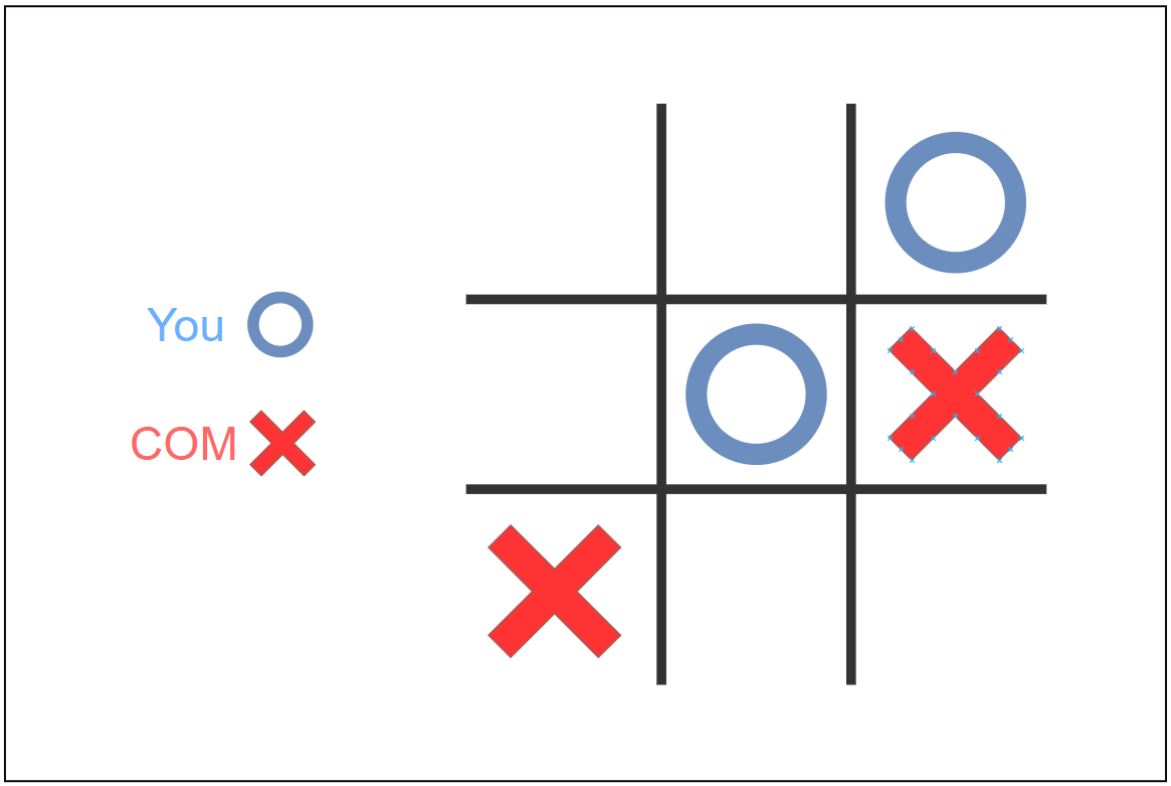


Abbildung 2: Ingame View