# BWINF Aufgabe 3: Tobis Turnier

## Einleitung:

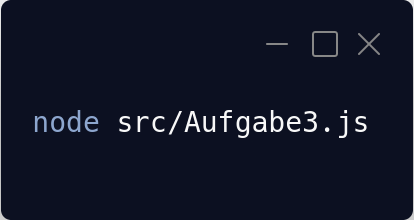
Zum Lösen der Aufgabe nutze ich Node JS.

Die meisten Funktionen des Programms lagert ich in Klassen aus.

Diese verteilte ich gruppiert in mehrere Dateien.

Diese Dateien können als Node JS Module geladen werden und somit auf die Klassen zugegriffen werden.

Das Programm kann mit folgendem Befehl ausgeführt werden:



## Umsetzung:

### Spielerdateien einlesen:



Als erstes Lese ich die Spielerdatei ein. Dazu lese ich die Datei als String, den ich in die einzelnen Zeilen unterteile und erstelle für alle Zeilen nach der 2. Zeile ein Spieler Objekt mit einer Id und die eingelesenen Stärke.

Für die 2 Turnierformen entwickelte ich 2 Klassen, die den Spielablauf und die Bewertung implementieren. Sie sind beide Unterklassen der Competition Klasse.

Das (mehrfache) Ziehen aus der Urne habe ich in die Klassen DefaultMatch und MatchMultipleRounds ausgelagert.

### Liga Wettbewerb:

Für ein Ligaspiel wird erst mit zwei for Schleifen über jedes Spielerpaar iteriert und für sie eine Urnenziehung durchgeführt.

Alle Gewinne werden in einer Map gespeichert.

Zur Auswertung wird die Map danach in ein Array Aus Arrays umsortiert.

Dieses Array kann dann mit der sort Funktion sortiert werden. Als erste Priorität wird dabei die Anzahl der gewonnenen Spiele getrachtet und bei einem Gleichstand wird der Spieler mit der geringeren Id bevorzugt.

Die Spieler der ganz vorne im Array ist ist der Gewinner.

### KO Wettbewerb:

Für das KO Spiel sollten 2 Varianten implementiert werden: K.O. und K.O.×5.

Ich beschloss mich diese in einer Klasse zusammenzufassen und die Anzahl der Urnenziehungen pro Spielerpaar in eine Variable auszulagern.

Das macht z.B. auch die Variante K.O.×7 sehr einfach einstellbar.

Als Erstes wird das Spieler Array je nach Spielplan sortiert.

Ich implementierte 3 Spielpläne:

* Zufall
* Nach Id
* Nach Id (rückwärts sortiert)

Dann Wird die erste Runde gestartet in der alle Spieler teilnehmen.

Die Gewinner dieser Runde werden in die nächste Runde gelassen.

Durch eine Rekursion wird das solange wiederholt bis nur ein Spieler übrig ist.

Dieser ist der Gewinner des Wettbewerbs.

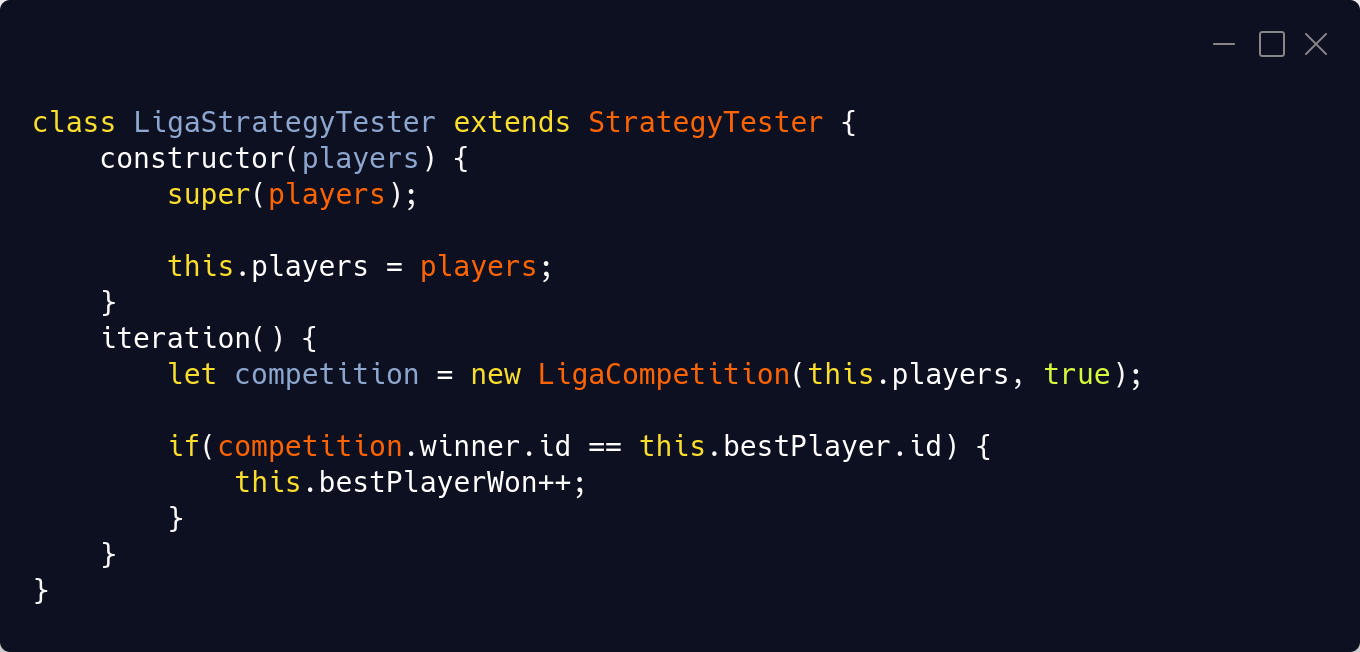
### Strategie Testen:



Ich implementierte eine Klasse, die für jede Strategie als Elternklasse benutzt werden kann.

Sie ermittelt den besten Spieler und führt die Funktion iteration so oft aus wie per iterations Variable angegebe.

Die Iteration Funktion muss in einer der Strategie spezifischen Klassen implementiert werden.



Die Strategie spezifische Klasse muss also nur noch das Starten der Wettbewerbe mit den richtigen Parametern implementieren.

## Ergebnisse:

# 

# 

## Fazit:

Je nach Spieler Datei ist die Wahrscheinlichkeit, dass der beste Spieler gewinnt sehr unterschiedlich.

Bei Datei 4 unterscheiden sich die Spielstärken kaum, was dazu führt, dass der beste Spieler nur selten in K.O. und K.O. x 5 gewinnt. Erst bei Strategien wie K.O. x 501 gewinnt der beste Spieler oft. Diese sind allerdings nicht in der Praxis realisierbar.

Anders in der Datei 3, denn die Spielerstärken dort sind deutlich unterschiedlicher.

Dort gewinnt der Beste Spieler in den meisten Strategien sehr oft.

Für ein spannendes Spiel sollte wahrscheinlich jeder die Chance haben zu gewinnen, aber Stärke trotzdem eine Rolle spielen.

Für sehr ähnlich starke Gruppen würde ich also eher die Liga Strategie empfehlen, da der beste Spieler zwar keine schlechten Chancen hat, aber den anderen auch nicht die Freude nimmt.

Für diverse Gruppen würde ich die K.O. Strategie emnpfehlen, da sie alle mal gewinnen lässt.

Für Gruppen die durch eine Person sehr dominiert wird (siehe Datei 1) empfehle ich die Liga Strategie.