pmh philipp-matthäus-hahn Technische Schule Nürtingen	**Arbeitsblatt**	Datum:	pmhs Nürtingen Kanalstraße 29 72622 Nürtingen bar@pmhs.de
	Klassen	Name:	

Aufgabe 1: Klassen und Objekte in ES6

a.) Recherchiere z.B. auf https://javascript.info/class *, was es mit Klassen in JavaScript (ES6) auf sich hat und wie damit Objekte erzeugt werden können.*

Notiere dir wichtige Informationen zu folgenden Stichworten.

class	
constructor	
new Object	
prototype	
Methode	
Attribut/ Property	
this	

Um das Projekt gut organisieren zu können ist es sinnvoll für Klassen neue Dateien zu verwenden.

Klassennamen sollten immer mit großen Buchstaben beginnen.

Aufgabe 2: Stein-Schere-Papier Duell beim Client

a.) Deklariere im client.js eine neue Variable "challengedPlayer" und initialisiere sie mit einem leeren string. In der Funktion "showHighscore" müssen dem ListItem für einen Spieler zusätzliche Attribute zur Steuerung des modalen Dialogs (aus Bootstrap 4) "challengeMenu" und EventListener beim "click" auf einen Spieler hinzugefügt werden. Dazu muss folgender Code eingefügt werden:

```
    el.setAttribute("data-toggle", "modal");
    el.setAttribute("data-target", "#challengeMenu");
    el.setAttribute("id", highscore[i].name);
    el.addEventListener('click', onChallengePlayer);
```

b.) Definiere die Funktion "onChallengePlayer", die ein "event" als Parameter erhält. Sie speichert die "id" des Elements, von dem das Event kommt (`event.target.id`) in der Variable "challengedPlayer" und setzt den Text des HTML-Elements "challengedPlayerName" auf den Namen des herausgeforderten Spielers.

c.) Füge den Buttons mit den IDs "rock", "paper" und "scissors" je einen "click"-EventListener hinzu, der die Funktion "sendRPSTurn" aufruft.

d.) Definiere die Funktion "sendRPSTurn". Sie prüft zunächst ob "challengedPlayer" einen Wert enthält und nicht den Namen des Players. Dann definiert sie ein neues Objekt "myturn" mit folgendem Aufbau:

myturn:Turn
player1 = player.name player2 = challengedPlayer turn = event.target.id game = „rps“

Und schickt es als JSON-String über den Socket an den Server.

Aufgabe 3: Stein-Schere-Papier Duell beim Server

a.) Erzeuge eine Datei auf Ebene des Servers mit dem Namen "rps-duel.js". Die Datei soll eine neue Klasse "RPSDuel" enthalten. Die Klasse benötigt einen Konstruktor, der die Spieler-Objekte aus der Lobby (Socket und Spielerinformationen) der beiden Spieler "p1" und "p2" als Parameter erwartet. Nach der Definition der Klasse muss über Node.js die Klasse von außen zugänglich gemacht werden, dass geschieht über `module.exports = RpsDuel;` (wobei der Name der Klasse für den Esport angegeben werden muss).

Hilfe-karte 1

Hilfekarte 1 enthält wichtige Informationen und Erklärungen zur Klasse RPSDuel und ihrem Konstruktor.

b.) Binde die neue Klasse mit `const RPSDuel = require('./rps-duel');` in "server.js" ein. Die Dateiendung lässt man dabei weg.

c.) Deklariere in der "server.js" ein neues Objekt mit dem Namen "challenges" und initialisiere es mit einem Array. Verwende dazu den "constructor" von Array mit dem Signalwort "new".

```
let challenges = new Array();
```

d.) Definiere in der Funktion "connected" einen neuen Event-Handler für den Socket, der auf das Event mit dem Namen "challenge" reagiert und die Funktion "onChallenge" mit den Daten des Events als Übergabeparameter aufruft.

e.)Die Funktion "onChallenge" muss als nächstes definiert werden. Sie enthält die Logik für das sog. "Matchmaking" also das Herausfordern und annehmen einer Herausforderung.

Hilfe-karte 2

Hilfekarte 2 enthält zusätzliche Informationen zur Funktion "onChallenge".

Aufgabe 4: Spiellogik RPSDuel

a.) Vereinfache den Code deiner Klasse "RPSDuel", indem du die namen der Spieler in den Attributen "name1" und "name2" des aktuellen Objektes speicherst. Außerdem benötigst du ein Array als Attribut des erzeugten Objekts, dass die Auswahl "selections" der Spieler enthält. Als Indizes verwenden wir diesmal ebenfalls keine Zahlen, sondern wie beim Array "players" die namen der Spieler. Initialisiere jede Stelle des Arrays mit "null", da zu Beginn keine Auswahl getroffen ist.

b.) Füge der Klasse "RPSDuel" eine Methode "sendToPlayer" hinzu, die als Übergabeparameter "data" und den "playerName" (Name des Spielers im Array "players") bekommt. Die Methode ruft über den Socket des durch playerName gewählten Spielers die Methode "emit" auf und sendet ein neues "msg"-Event mit Absender (from) "Server" und "data" als Text (text). (siehe Start-Message in "connected"-Methode bei server.js).

c.) Definiere eine Methode "sendToPlayers" in der Klasse "RPSDuel", die als Übergabeparameter "data" erhält und diese mit Hilfe der Methode "sendToPlayer" des aktuellen Objekts (this) an alle Spieler des aktuellen Objekts übermittelt. Da das Array keine Zahlen als Indizes hat, muss eine `for ... in ...` Schleife zum durchlaufen des Arrays verwendet werden. Recherchiere dazu im Internet.

```
for (let p in this.players) {
      this.sendToPlayer(this.players[p].player.name, data);
    }
```

d.) Definiere eine Methode "makeTurn" in der Klasse "RPSDuel", die als Übergabeparameter einen "playerName" und mit "selection" die Auswahl des jeweiligen Spielers als string erhält. Die Funktion prüft, ob im Array "selections" des aktuellen Objekts noch "null" steht. In diesem Fall wird die Auswahl dort gespeichert und mit "sendToPlayer" eine Bestätigung der Auswahl an den Client (playerName) geschickt.

e.) Am Ende der "makeTurn" Funktion wird die Funktion "checkGameOver" aufgerufen. Sie prüft ob alle Elemente in "selections" einen Wert haben und schickt dann an alle Spieler über "sendToPlayers" eine GameOver-Meldung, in der die Spielernamen und deren Auswahl steht.

Zusatzaufgabe: Optimiere deinen Code

a.) Recherchiere im Internet, wie du einen Modal in Bootstrap 4 in deinem client.js ausblenden kannst, sobald der Spieler seinen Zug gemacht hat.

b.) Suche die for-Schleifen in deinem Code und überlege dir ob und wie du sie durch forEach-Methoden oder for ... in-Schleifen ersetzen kannst.

c.) Überlege dir wie du deinen Code durch die Verwendung von Klassen vereinfachen könntest und definiere weitere Klassen.

d.) Versuche den Code so zu erweitern, dass geprüft wird, wie das Spiel ausgegangen ist. "checkGameOver" informiert dann die Spieler durch entsprechende Meldungen über Gewonnen, Verloren oder Unentschieden.Außerdem kann über die Methode "checkGameOver" an die "makeTurn" und von dort zum Hauptprogramm des Servers ein Rückgabewert (z.B. 0:Unentschieden, 1:Spieler 1 gewinnt, 2: Spieler 2 gewinnt) übermittelt werden. Im Hauptprogramm kann dann entsprechend der Highscore angepasst werden (jeweils 1 Punkt für Unentschieden oder 3 Punkte für einen Sieg).