**Python - Programmentwurf**

**Team Mitglieder:**

|  |  |
| --- | --- |
| **Name** | **e-Mail (DHBW)** |
| Florian Herkommer | herkommer.flor@dhbw-ravensburg.de |
| Florian Glaser | glaser.florian-it20@it.dhbw-ravensburg.de |
| Martin Heimpel | heimpel.martin-it20@it.dhbw-ravensburg.de |
|  |  |

**Verwendete Python Version: 3.9**

**Verwendete Pylint Version: 2.7.0**

**Verwendete nachinstallierte Bibliotheken: keyboard**

**Beschreibung des Dateiformats zum Speichern des Spiels:**

* JSON-Format siehe Beispieldatei im Anhang
  + Beispieldatei: *test\_safe\_game.json*

**Beschreibung des Benutzerinterfaces:**

* Eingabe / GUI erfolgt über die Konsole
* Jeder Spieler sieht ein „Spickzettel“, wo die bereits getätigten Schüsse markiert werden. Darunter sieht der jeweilige Spieler sein eigenes Spielfeld und kann dort sehen welches Schiffteil bereits vom Gegner zerstört worden ist.
* Folgende Zeichen werden verwendet:
  + Schiff -> #
  + Zerstörtes Schiff -> X
  + Wassertreffer -> O
* Nach Wechseln des Spielzuges bleiben 5 Sekunden, bis das Spielfeld des jeweils anderen angezeigt wird
* Beispiel eines ausgegebenen Spielfeldes ist im Anhang: *beispiel\_GUI.jpg*

**Beschreibung der Architektur:**

* Die Master-Klasse ist das Herzstück und steuert den ganzen Spielablauf
* Weitere und Detaillierte Informationen können dem UML-Klassendiagramm entnommen werden, siehe Datei *UML\_Klassendiagramm.png*

**~~Beschreibung des Computergegners (4er Gruppen):~~**

* ~~Welcher Ansatz wird verfolgt~~

**Dokumentation von einem kompletten Spielablauf:**

* Kopie der kompletten Ausgabe auf der Konsole. Evtl. eigene Datei
  + Siehe *schiffeversenken.html* (am besten im Microsoft Edge anschauen)

**Log von den Tests:**

* Alle Tests über Konsole ausführen und Ausgabe hier einfügen (oder extra Datei)

**Bewertung der Testergebnisse:**

* Alle Tests werden erfolgreich abgeschlossen
* Die Tests sind unserer Meinung nach sinnvoll, da bei absichtlichem Einfügen von Logikfehlern die entsprechenden Tests fehlschlagen und bei reinen Optimierungen des Codes die Tests mit gleichem Ergebnis weiterhin erfolgreich bestanden wurden.

**Code-Coverage Ausgabe:**

* Tests mit Coverage ausführen und Ausgabe hier einfügen
  + Siehe Ordner *htmlcov*

**Bewertung der Coverage und Sinnvollheit der Tests:**

* Gewünscht ist eine Coverage von min 75%
  + Wurde dies erreicht? Wenn nein, warum nicht?
    - Wir haben eine insgesamte Coverage von 93%
      * Bestimmte Methoden konnten wir nicht Testen, da wir mehrere Eingaben hintereinander simulieren müssen. Wir wissen jedoch nur, wie man ein Input() versorgt und nicht weiter folgende
      * Einzelne Methoden, die in so einer oben erwähnten Methode aufgerufen werden, werden einzeln getestet
* Begründet warum ihr euch sicher seid alles mit Tests abgedeckt zu haben
* Fehlerfälle in den Tests überprüft?
  + Wir testen Eingaben, zum Beispiel zu schießen, mit gültiger und ungültiger Eingabe.
  + Da wir für jede User-Eingabe eine eigene Funktion geschrieben haben, welche das ganze Fehlerhandling übernimmt und diese auch getestet ist, sind wir uns sicher alle uns in den Sinn gekommenen Fehlerfälle getestet zu haben.

**Bewertung der Fehlersicherheit:**

* Werden Fehleingaben korrekt abgefangen?
  + Nenne 2 Beispiele wo ein Fehler abgefangen wird
    - Beim Speichern und Laden von Spielständen werden Exceptions abgefangen und dem User in verständlicher Schreibweise gezeigt
    - Fehleingaben vom User (ungültige Koordinate, platzieren oder Richtung) werden abgefangen und fordern den User anschließend zur erneuten Eingabe.
* Zweimaliges Schießen auf dieselbe Koordinate wird nicht unterbunden, da wir die Auffassung verfolgen, dass, wenn ein Spieler zweimal auf dasselbe Feld schießt, er selbst schuld ist. Außerdem ist es keine ungültige Eingabe.

**Pylint Ausgabe:**

* Pylint-Ausgabe (ausgeführt in der Windows PowerShell) in Datei *pylint-Ausgabe.txt*

**Bewertung der Code-Qualität/Lesbarkeit:**

* Begründet noch vorhandene Pylint Warnungen
  + Die Warnung „Too many instance attributes (10/7)” haben wir zur Kenntnis genommen, haben jedoch nur 6 Attribute in unserer \_\_init\_\_ vordefiniert und haben in der Master Klasse auch keine weiteren Attribute gefunden als diese 6. Daher gehen wir davon aus, dass es sich hierbei um eine Falschmeldung handelt.
* Begründet warum euer Code gut lesbar ist
  + Durch das Strukturieren und Gruppieren von zusammenhängenden Methoden und den zwar längeren, aber dafür eindeutigen Variablennamen, kann man unseren Code ohne Vorkenntnisse über Abkürzungen gut lesen.
  + Wir haben uns weitestgehend an das Prinzip KISS und „Single Responsibility“ gehalten.
  + Während des Programmierprozesses ist uns aufgefallen, dass man Ausgabe und Logik noch weiter hätte trennen können, wodurch die master.py übersichtlicher geworden wäre.

**Bewertungstabelle:**

* Ausgefüllte Bewertungstabelle