**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

**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 entnehmen welches Schiffteil bereits vom Gegner zerstört worden ist.
* Folgende Zeichen werden verwendet:
  + Schiff -> #
  + Zerstörtes Schiff -> X
  + Wassertreffer -> O
* Nach dem wechseln des Spielzuges bleiben 5 Sekunden bis das Spielfeld des jeweils anderen angezeigt wird
* Beispiel eins ausgegebenen Spielfeldes ist im Anhang

**Beschreibung der Architektur:**

* Die Master-Klasse ist das Herzstück und steuert den ganzen Spielablauf
* Weitere und Detaillierte Informationen können dem UML-Klassendiagramm entnommen werden

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

* ~~Welcher Ansatz wird verfolgt~~

**Dokumentation von einem kompletten Spielablauf:**

* Kopie der kompletten Ausgabe auf der Konsole. Evtl. eigene Datei

**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 absichtlichen Einfügen von Logikfehler die entsprechenden Tests fehlschlagen und bei reine Optimierungen des Codes mit gleichem Ergebnis die Tests weiterhin erfolgreich bestanden werden.

**Code-Coverage Ausgabe:**

* Tests mit Coverage ausführen und Ausgabe hier einfügen

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

* Gewünscht ist eine Coverage von min 75%
  + Wurde dies erreicht? Wenn nein, warum nicht?
* Begründet warum ihr euch sicher seid alles mit Tests abgedeckt zu haben
* Fehlerfälle in den Tests überprüft?
* 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

**Bewertung der Fehlersicherheit:**

* Werden Fehleingaben korrekt abgefangen?
  + Nenne 2 Beispiele wo ein Fehler abgefangen wird
* Fehleingaben vom User (ungültige Koordinate, platzieren oder Richtung) werden abgefangen und fordern den User anschließend zur erneuten Eingabe
* Beim speichern und laden von Spielständen werden Exceptions abgefangen und dem User in verständlicher Schreibweise gezeigt

**Pylint Ausgabe:**

* Pylint auf der Konsole Ausführen und Ergebnis einfügen

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

* Begründet noch vorhandene Pylint Warnungen
* Begründet warum euer Code gut lesbar ist
* Die Warnung „Too many instance attributes (10/7)” haben wir in 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
* 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.

**Bewertungstabelle:**

* Ausgefüllte Bewertungstabelle