**Aufgabenverteilung**

|  |  |
| --- | --- |
| Aufgabenbereiche | Umsetzung |
| Player | Lilli |
| MyBot | Jan |
| Board |  |
| Game |  |

**Programm Aufbau**

Unser Programm setzt sich zusammen aus den vier Klassen Player, Bot, Board, und Game.

Ein Objekt der **Klasse Player** verwaltet lediglich seine Information, wie den Namen des Spielers und seine Nummer. Außerdem kann er mit einem ihm übergebenen Spielfeld interagieren und darauf einen Zug machen, indem er seine Signatur (Spielernummer) an ausgewählter Position setzt.

Die **Klasse Bot**, erbt die Fähigkeiten vom Spieler und erhält noch die Möglichkeit dazu in 3 verschiedenen Komplexitätsgraden auf das Spiel zu reagieren.

Die **Klasse Board** verwaltet das Spielfeld, bietet die Möglichkeit dieses auf der Konsole zu visualisieren und kontrolliert, wenn es in Benutzung ist, seinen Zustand. Es kann erkennen welcher Spieler gewonnen hat oder ob es zu einem Unentschieden gekommen ist. Es wird vertikal, horizontal und diagonal geprüft.   
Zudem ist das Board dynamisch anpassbar und lässt sich in Verschiedenen Größen erzeugen und kann auf verschiedene Gewinnbedingungen initialisiert werden.   
Zusätzlich bietet das Board die Möglichkeit die Züge der Spieler auf ihre Gültigkeit zu kontrollieren.

Die letzte Klasse ist die **Klasse Game**. Sie fügt alle Module zusammen und ermöglicht durch das Nutzen der ihr zu Verfügung gestellten Klassen und deren Methoden einen Spielablauf.   
Dafür hat sie einmal eine Methode „start“, die vor Beginn des eigentlichen Spiels die notwendigen Informationen vom Benutzer erfragt. Hier werden die beiden Spieler initialisiert und falls sich unter diesen ein Bot befindet, sein Schwierigkeitsgrad vom Benutzer gewählt.  
Zusätzlich erstellt die Funktion ein Board-Objekt aufgrund der vom Benutzer gewählten Gewinnbedingung, Spalten- und Zeilenanzahl.  
Ist das erfolgreich geschehen, startet die Methode, die zweite Methode der Klasse Game, die Gameloop.

Im Falle das zwei Spieler gegeneinander antreten oder ein Spieler gegen einen Bot, wird die Gameloop so lange laufen, bis eine der beiden Parteien gewonnen hat oder es zu einem Unentschieden gekommen ist.   
Dazu ruft sie einen der beiden Spieler auf und fordert diesen auf sein Zug zu machen. Wer beginnt, wird hierbei zufällig entschieden. Nach dem erfolgreichen Setzen ist der nächste Spieler dran. Ist die Runde beendet, wird dem Nutzer die Wahl gegeben erneut zu spielen oder das Spiel zu beenden.

Lässt der Benutzer zwei Bots gegeneinander spielen, kommt noch die Option hinzu, zu wählen wie häufig man die beiden gegeneinander antreten lassen möchte. Der Spielablauf bleibt sonst unverändert. Die Bots wählen ihre Züge aufgrund dem zu Beginn gewählten Schwierigkeitsgrad.

**KI-Konzept:**

Es wurden 3 Versionen des Bots erstellt, die aufgrund von unterschiedlicher Priorisierung mit den gleichen Methoden unterschiedliche Spielmuster verfolgen. Dadurch ist zum einen ein defensiv ausgerichteter Bot entstanden, der erst schaut, ob er sich verteidigen muss. Wenn das an keiner Position notwendig ist, greift er an und setzt zu seinen Gunsten.  
Zum anderen ist ein offensiv ausgerichteter Bot entstanden, der zuerst schaut, ob es ihm möglich ist eine Reihe zu vervollständigen und wenn das an keiner Stelle sinnvoll ist, prüft, ob er verteidigen kann. Ist auch das an keiner Stelle notwendig wird er an eine bereits existierende Reihe anbauen.

* Zukünftig abwägen möglich -> Zukunftsaussicht

Die Bots nutzen 3 Methoden. Die erste Methode ist das Verteidigen (check\_in\_danger). Diese schaut nacheinander, ob der Gegner horizontal, vertikal oder diagonal eine Reihe hat die mindestens zwei Elemente lang ist. Wenn das der Fall ist, so berechnet sie für den spezifischen Fall ein Feld, welches den Gegner blockiert. Damit verteidigt sich der Bot frühzeitig gegen seinen Gegner.

Die zweite Methode ist das Angreifen (check\_finish). Diese ähnelt dem Verteidigen sehr, da sie die gleichen Methoden nutzt. Jedoch wird in der Methode statt wie beim Verteidigen die Nummer des Gegners hier die eigene Nummer übergeben. Die Methode schaut nun, ob der Bot selbst horizontal, vertikal oder diagonal eine Reihe hat, die mindestens zwei Elemente lang ist. Trifft dieser Fall ein so berechnet sie ein Feld, welches die Reihe verlängert und den Bot somit näher an den Sieg heranführt.

Die letzte Methode ist zum Start eines Angriffs des Bots gedacht (start\_attack). Sie schaut nur nach einem eigenen Feld und prüft dann, ob die Felder daneben frei sind. So macht der Bot in dieses seinen Zug und kann damit in der nächsten Runde einen Angriff starten.

Der defensive Bot schaut zuerst, ob er verteidigen kann (check\_in\_danger) und greift danach erst an (check\_finish). Der offensive Bot schaut zuerst ob das mittlere Feld frei ist, danach ob er angreifen kann (check\_finish), dann ob er verteidigen muss (check\_in\_danger) und wenn er dies auch nicht muss, so startet er einen Angriff (start\_attack).

Die Bots sind nach Schwierigkeit sortiert. Der Level 1 Bot setzt in ein zufällig gewähltes Feld. Er ist dadurch einfach zu schlagen. Der Level 2 Bot kann verteidigen und auch bei einer sich ergebenen Möglichkeit einen Angriff zu Ende bringen. Jedoch wird er selbst nie einen Angriff starten. Der Bot ist dadurch schwieriger zuschlagen, auch wenn er selbst selten gefährlich wird. Der Level 3 Bot wiederum ist der stärkste aller drei Bots, da er nicht nur passiv agiert, sondern auch aktiv angreifen kann. Er macht dies durch seine Befähigung zum Starten eines Angriffs er ist nun nicht mehr auf den Zufall angewiesen, um anzugreifen. Dies unterscheidet ihn vom Level 2 Bot.