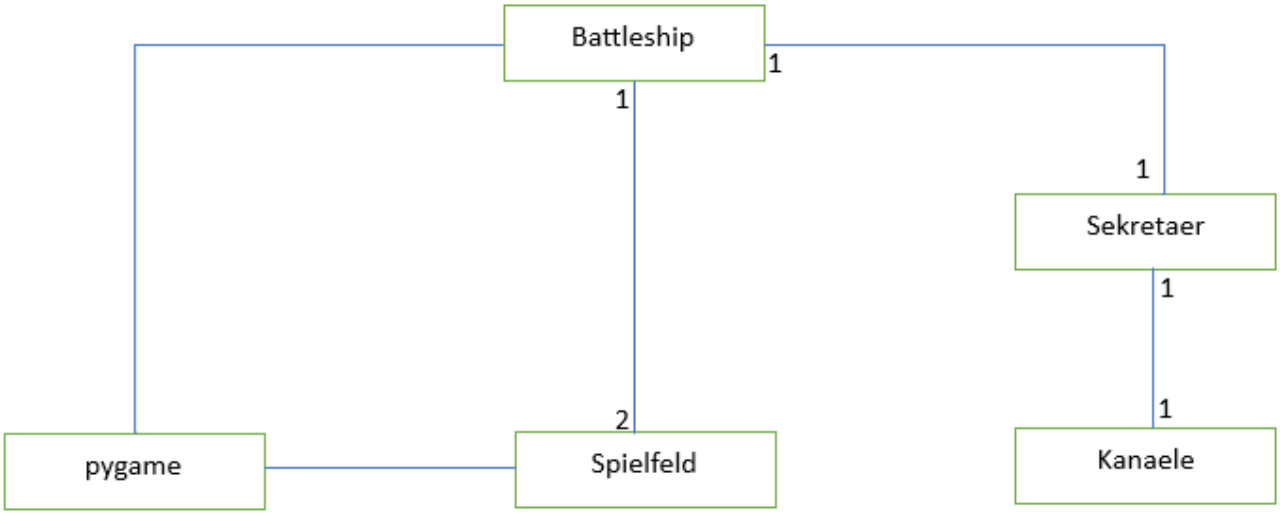
Gustav Freitag, Ivaylo Staykov 16.02.2024

Systemarchitektur

Entsprechend der objektorientierten Analyse fragt man sich, welche Objekte bei einem solchen Brettspiel beteiligt sind und welche Klassen sich daraus ableiten.
Im Spiel Schiffe versenken (engl. **Battleship**) gibt es 2 **Spielfelder.** Das eigene und das des Gegners. Beide davon haben jeweils eine Menge von Schiffen, welche auf ihnen verteilt sind. Um die Schiffaufstellung des Gegners zu erfahren und um seine Schüsse zu kommunizieren, ist ein **Sekretaer** notwendig. Dieser benötigt eine Möglichkeit, Nachrichten zu versenden, er braucht also (Nachrichten-)**Kanaele**. Für alle graphischen Darstellungen und die ereignisgesteuerte Programmierung stützt sich das Programm auf das Modul **pygame** ab.
Man erhält damit folgendes, einfaches UML-Klassendiagramm, was die Struktur des gesamten Softwaresystems darstellt.



Man beachte, dass bei den Assoziationen zum Pygame-Modul pygame keine Kardinalitäten stehen, weil hier nur bereitgestellte Funktionen des Moduls genutzt werden und nicht Objektinstanzen in Beziehung stehen. Außerdem sind kleinere, in Python enthaltene Module wie time und random nicht aufgeführt.

Nachdem die Grobstruktur des Softwaresystems damit geklärt ist, folgen nun die einzelnen **UML-Klassendiagramme.**
Auf der obersten (Anwendungs-)Ebene steht die Klasse Battleship. Die Initialisierung eines Objekts bewirkt das Öffnen des Spielfensters. Das Attribut Surface ist hierbei ein Bestandteil des Pygame-Moduls. Es ist ein eigener Datentyp, welcher mit dem screen-Attribut bei der turtle vergleichbar ist. Die Klasse Spielfeld hat als wesentliches Attribut eine Liste von Listen von Tupeln, welche symbolisieren ob auf ihnen ein Schiff steht ob dieses Feld bereits angeschossen wurde. Darüber hinaus wird jedes Schiff als Liste von Tupeln in einer weiteren Liste gespeichert. Hierbei repräsentieren die Tupel jedoch die Koordinaten der Felder auf welchen die einzelnen Schiffe stehen. Zur Kommunikation wird ein Sekretaer benutzt, welcher den Aufbau der Schiffe beider Spieler, sowie Schüsse empfangen und versenden kann. Dabei erfolgt die eigentliche, lokale Netzwerkkommunikation über die bekannte Klasse Kanaele.
Für alle genannten Klassen gilt, dass weitere Methoden- und Attributnamen selbstsprechend sind. Für genauere Angaben ist die Spezifikation innerhalb des Quelltextes heranzuziehen.

Gustav Freitag, Ivaylo Staykov 16.02.2024

Battleship
surface: Surface msf: Spielfeld gsf: Spielfeld ships: [pygame.rect] ship_images: [pygame.image] sek: Sekretaer pcnummer: str
init(surface: Surface)->Battleship aufbauPhase()

Sekretaer
gegnerIP: str port: int erster: bool
init(pcnummer: int, heimspiel: bool) ->Sekretaer sendeSchiffe(schiffe: [[(int, int)]]) empfangeSchiffe()→[[(int, int)]] kommuniziereSchiffe(schiffe: [[(int, int)]]) ->[[(int, int)]] sendeZug(feld: (int,int)) empfangeZug()->(int,int) gibErster()→bool quit()

Spielfeld
surface: Surface groeße: int pos: (int,int) felder: [[(int,int)]] schiffe: [[(int,int)]] sunken_ship_images: [pygame.image]
init(surface: Surface) -> Spielfeld gibPos()→(int,int) beschieße(feld: (int,int))) istFrei(feld: (int, int))->bool zeichneBrett() setzteSchiffe(gegnerSchiffe: [[(int, int)]]) setzeSchiff(alt_schiff, neu_schiff: [(int,int)]) gibSchiffe() -> [[(int, int)]] sindVersenkt()→bool istVersenkt(id: int)->bool

Kanaele
zielIP: str zielPort: int
init(IP: str, port: int)->Kanal empfangen()->str senden(nachricht: str) erster()->bool schließen()