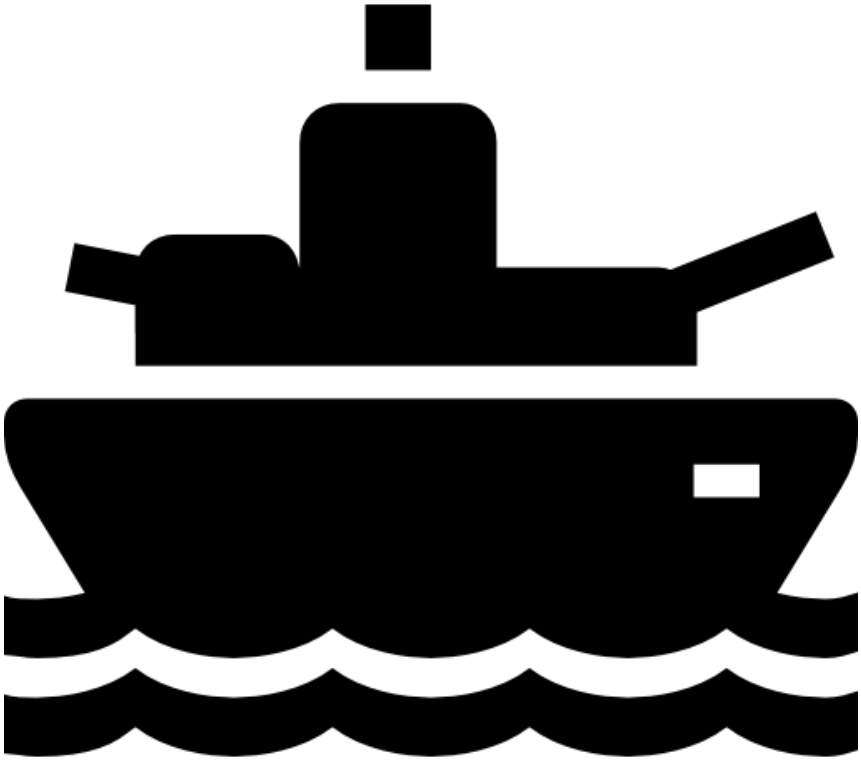
# BATTLESHIP

## Software-Entwicklung 2

**Projekt**
**https://gitlab.mi.hdm-stuttgart.de/lb092/Battleship**




**Projektmitglieder**

| | |
|---|---|
| **Celine Wichmann** | **cw089** |
| **Lea Baumgärtner** | **lb092** |

# INHALT



## 1) ABSTRACT

Die Battleship-Dyade beschäftigte sich mit dem Programmieren des Spieles „Schiffe versenken" jedoch mit einigen Innovationen. Dabei ist das Hauptziel des Spielers die Schiffe auf dem Koordinatensystem des Gegners allesamt zu vernichten.

Die Innovationen bestehen darin, dass nach dem Schiffe setzen auf das eigene Koordinatensystem jeweils eine bestimmte Anzahl an besonderen Objekten generiert und auf dem Koordinatensystem zufällig gesetzt werden.

Das Spiel beginnt damit, indem man auf den Button „New Game“ drückt. Daraufhin erhält man die Wahl, ob man im Singleplayer Modus, also gegen den Computer, oder im Multiplayer Modus, also gegen einen realen Gegner, spielen möchte.

Je nachdem welcher Modus ausgewählt wurde kommt anschließend das Fenster, in welchem man seinen Namen bzw. die Namen der Spieler einträgt. Nachdem man dies mithilfe der „Enter“ Taste bestätigt, erhält man die Wahl über den Schwierigkeitsgrad des Spiels, welches nicht nur die Größe der Koordinatensysteme, sondern auch die Anzahl an Schiffen und speziellen Elementen, bestimmt.

Daraufhin wird das Koordinatensystem des ersten Spielers generiert, und er erhält die Möglichkeit seine Schiffe zu setzen. Im Singleplayer Modus wird anschließend gleich zum Spiel übergegangen, während im Hintergrund die Schiffe des Computerspielers sowie bei beiden jeweils die speziellen Elemente zufällig gesetzt werden.

Im Multiplayer Modus erhält nach der Bestätigung des ersten Spielers auch der zweite Spieler die Möglichkeit seine Schiffe zu setzen. Erst nach dessen Bestätigung werden im Hintergrund auf beiden Koordinatensystemen die speziellen Elemente zufällig gesetzt und das Spiel kann beginnen.

Während dem Spiel dürfen die Spieler abwechselnd auf das Feld des jeweiligen Gegners schießen. Dabei sieht jeder nur seine eigene Angriffsfläche sowie die des Gegners, um die Position der Schiffe des jeweiligen Gegners geheim zu halten und dadurch den Multiplayer Spaß zu garantieren. Außerdem besteht während dem Spiel die Möglichkeit ins Menü zu gehen und dort, entweder fortzufahren, ein neues Spiel zu beginnen oder das Spiel zu beenden.

Je nachdem ob ein Spieler ein Schiff des Gegners oder ein spezielles Element auf dem Feld trifft, erhält dieser unterschiedliche Punkte.

Derjenige, der zuerst alle Schiffe getroffen hat, gewinnt das Spiel.

Daraufhin wird in einem Fenster jeweils die Anzahl an Punkten sowie die Anzahl an Schüssen des Gewinners als auch die des Verlierers angezeigt. Am Ende hat man die Wahl, ob man erneut spielen oder zurück zum Opening-Screen möchte.

## 2) STARTKLASSE

Die Startklasse der Anwendung befindet sich in der Klasse „Opening-Screen.java" im Package battleshipGUI.

## 3) BESONDERHEITEN

Während des Projektes haben wir uns hauptsächlich darauf fokussiert, alle notwendigen und geforderten Themen abzudecken.

Dadurch kam es leider dazu, dass wir es aus Zeitgründen nicht geschafft haben das Spiel fertig zu stellen. So ist die Beschreibung des Projektes im Abstract dementsprechend formuliert worden, wie es wäre, wenn es vollständig funktionsfähig wäre.

Folgende Klassen wurden nicht eingebaut, doch dazu aufbewahrt, um mögliche Erweiterungen für später gewährleisten zu können.

- Package ships: *TwoFieldBoat, ThreeFieldBoat, FourFieldBoat und QuatrupleFieldBoat*
- Package game: *Artificial Intelligence, Menu*
- Package com: *CustomButtonTest, CustomButtonTest2*
- Package battleshipGUI: *EndingScreen, MenuScreen, OpeningSceneCSS.css*

Folgende Funktionen des Spiels sind noch nicht verfügbar:

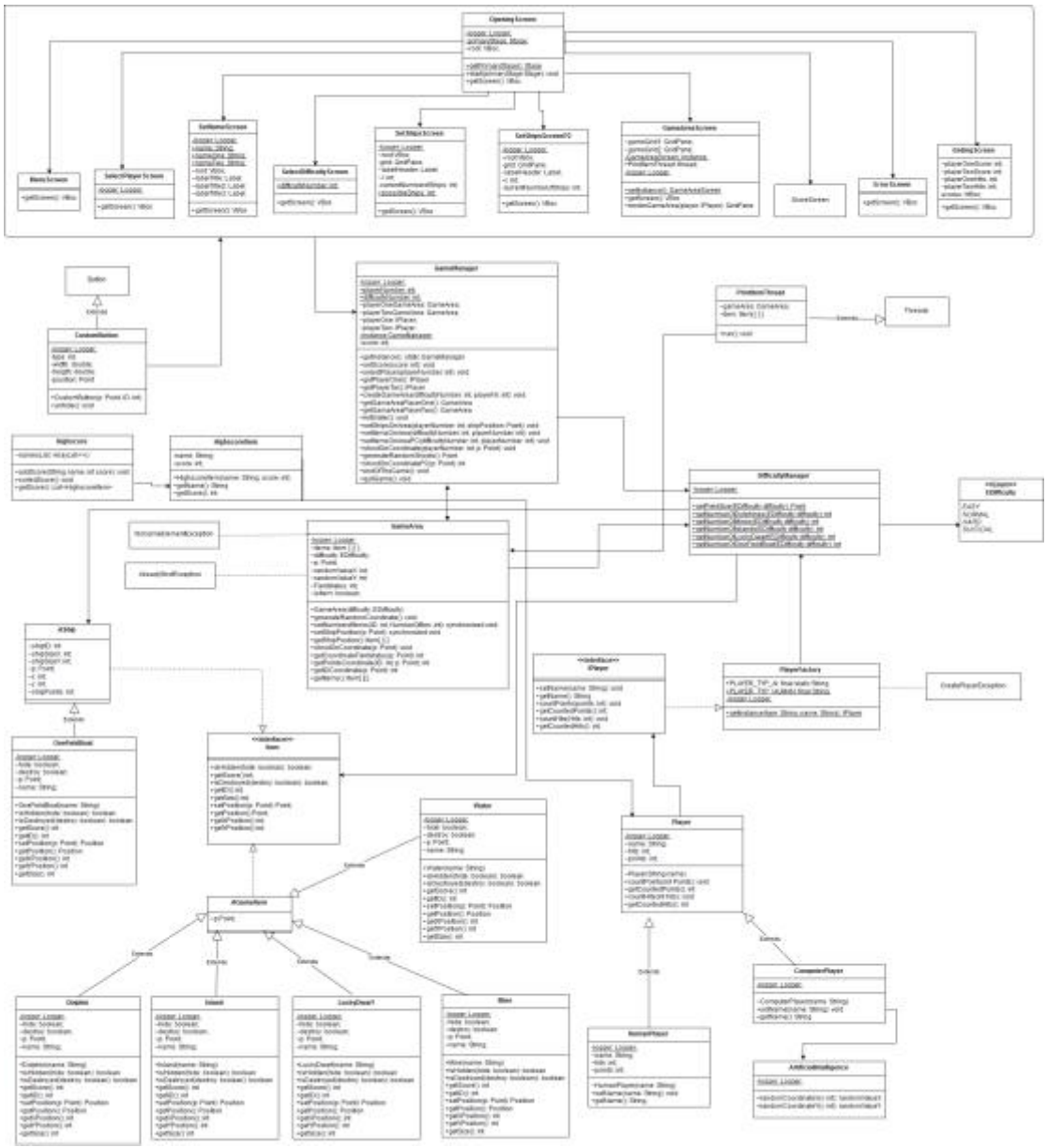
- *Abwechselndes Schießen der Player*
- *Automatisches Wechseln zum EndingScreen, wenn alle Schiffe eines Spielers zerstört wurden.*

- *Anzeigen des EndingScreens – alternative ist ScoreScreen*

Das besondere an unserem Projekt ist jedoch, dass wir die GUI und die Logik vor allem schön getrennt haben.

Dadurch haben wir eine besonders wichtige Klasse namens GameManager, welche alle GUI Klassen und die Logik verbindet und den Ablauf zwischen diesen großen Strukturen regelt.

# 4) UML-KLASSENDIAGRAMM



# 5) STELLUNGNAHMEN

→ Kurze Beschreibungen, wie die Bewertungskategorien implementiert wurden.

| Kategorie | Zu finden in... Package \| Klasse | Stellungnahme |
|---|---|---|
| Interfaces / Vererbung | gameConfigurations<br>- IPlayer (Interface)<br>- Item (Interface)<br>- Player (Abstract)<br>- AGameElement (Abstract)<br>- AShip (Abstract) | Interfaces:<br><br>Interface für die Implementierung der Spieler HumanPlayer und ComputerPlayer.<br><br>Interface für die Implementierung der Items allgemein, welches die besonderen Spielelemente als auch die Schiffe betrifft.<br><br>Vererbung:<br>Interface IPlayer wird in der Abstrakten Klasse Player implementiert. Von hier aus erben HumanPlayer und ComputerPlayer ihre Attribute und Methoden.<br><br>Interface Item wird sowohl in der AGameElement implementiert, als auch in der AShip. Von hier aus erben einerseits die Spielelemente, wie Dolphin oder Mine als auch die Schiffe bzw. das Schiff OneFieldBoat. |
| Package-Struktur | Projekt Battleship<br>Packages:<br>- battleshipGUI<br>- com.hdm-stuttgart.Battleship<br>- game<br>- gameConfigurations<br>- gameElements<br>- ships | Wir haben die Klassen entsprechend ihrer Funktionen und Stellungen sortiert.<br><br>Im Package battleshipGUI befindet sich alles, was in Verbindung mit der allgemeinen GUI der Applikation steht.<br><br>In com.hdm-stuttgart.Battleship befinden sich Kernelemente wie beispielsweise der GameManager, |

| | | |
|---|---|---|
| | | der sozusagen der Verwalter zwischen GUI und Logik darstellt. Außerdem befinden sich hier unsere Images.<br><br>Im Package game befinden sich die Kernklassen in Bezug auf die Logik des Spiels wie beispielsweise die GameArea oder der DifficultyManager.<br><br>Im gameConfigurations befinden sich hauptsächlich die Klassen in Bezug auf den Player, als auch das allgemeine Interface für alle Spielitems.<br><br>Im Package gameElements befinden sich alle Klassen der speziellen Elemente für das Spiel und im Package ships befinden sich alle Klassen für die Schiffe. Hierbei existieren noch unterschiedlich große Schiffe für eventuell spätere Bearbeitungen und Vergrößerung des Spiels. |
| Exceptions | game<br>- NoGameElement Exception.java<br>gameConfigurations<br>- CreatePlayer Exception.java | Hier befinden sich unsere selbst geschriebenen Exceptions.<br><br>Dabei wird die NoGameElementException geworfen, wenn man versucht auf jegliche Attribute oder Methoden eines Elements zuzugreifen, obwohl an der gegebenen Position keins gefunden wurde oder sonstiges in Bezug auf die Elemente schiefläuft.<br><br>Die CreatePlayerException wurde in der PlayerFactory eingebaut und wird geworfen, wenn es nicht möglich war, einen Player zu erstellen oder sonstige Probleme in Bezug auf die Player entstehen. |
| Grafische Oberfläche (JavaFX) | alles im Package battlesshipGUI | In diesem Package befinden sich alle Klassen in Bezug auf die GUI. Den Start der GUI bildet hierbei der OpeningScreen.java. Die anderen Screens wurden so gebaut, dass sie jeweils ineinander verschachtelt |

| | | |
|---|---|---|
| | | werden konnten, um den Ablauf des Spiels zu garantieren. |
| Logging | beinahe in jeder Klasse in jedem Package | Wir haben nahezu in jeder Klasse einen neuen Logger erzeugt um stets verfolgen zu können, was genau geschieht, und wann welche Methode aufgerufen wird.<br>Dabei haben wir je nach Auswirkung oder if-else-Struktur die die Loggingstufen .info, .debug und .error benutzt. |
| UML | befindet sich im Dokument | Bereits zu Beginn hatten wir ein UML-Diagramm ausgearbeitet. Doch je mehr man programmierte, umso mehr musste man das UML-Diagramm entsprechend verändern und bearbeiten, bis es voll und ganz konsistent in Bezug auf unsere Applikation war. |
| Threads | com.hdm_stuttgart.Battleship<br>-PrintItemsThread | Den Thread nutzen wir, um in einem Intervall von 60 Sekunden einmal alle Items in der Konsole auszugeben. |
| Streams | Com.hdm_stuttgart.Battleship<br>-Highscore<br>-HighscoreItem | Mithilfe von Streams haben wir eine Highscore List angelegt, in der die Player mit ihren während des Spiels erhaltenen Punkten aufgelistet werden. Die List wird sortiert, sodass der Spieler mit den meisten Punkten an erster Stelle steht. |
| Lambda-Funktionen | battleshipGUI | Wir haben Lambda-Funktionen hauptsächlich in Bezug auf das Event-Handling von UI-Elementen genutzt. |
| Dokumentation | in jeder Klasse | Wir haben im obersten Teil jeder Klasse nach den Imports und vor dem Source-Code jeweils eine Dokumentation erstellt, welche beschreibt, wozu die Klasse existiert und was ihre Aufgabe ist. Des Weiteren werden die einzelnen Methoden und ihre Aufgaben beschrieben. |
| Test-Fälle | Battleship/src/test/java/<br>com.hdm_stuttgart.<br>Battleship<br>- CreateGameArea Test.java<br>- PlayerFactoryTest. java | Hier wird das Erschaffen von GameAreas oder Player (HumanPlayer und ComputerPlayer) getestet. |
| Factories | gameConfigurations<br>- PlayerFactory.java | In dieser Factory wird je nach Parametereingabe bestimmt, |

| | | |
|---|---|---|
| | | welche Arten von Player erschaffen werden sollen. Konnte aufgrund der Eingabe kein Player erschaffen werden, wird die CreatePlayerException geworfen. |