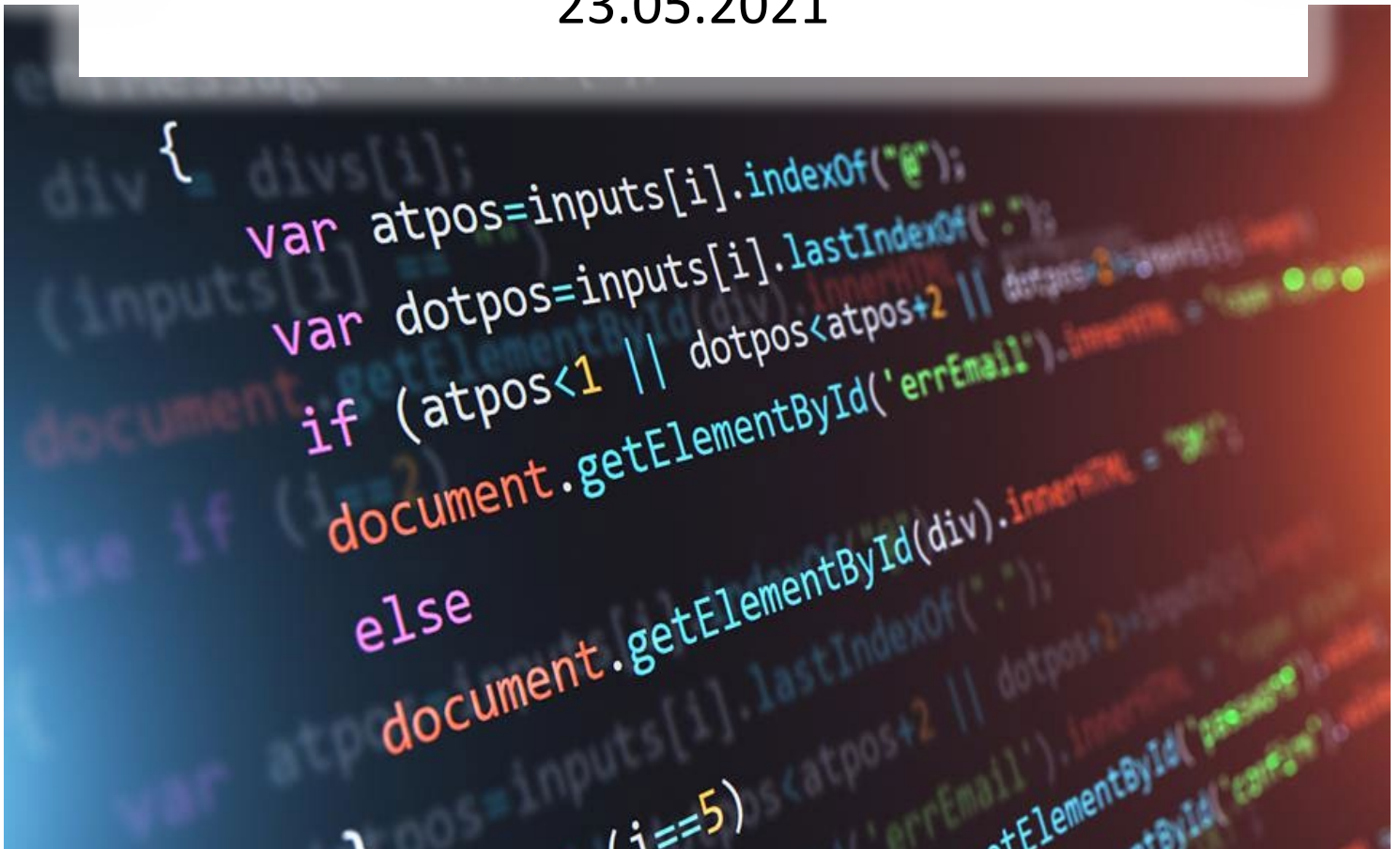


# Minesweeper V226V426

History

23.05.2021



# 1 Inhaltsverzeichnis

1	Inhaltsverzeichnis .....	1
2	Abbildungsverzeichnis .....	2
3	Tabellenverzeichnis .....	2
4	Vorwort .....	3
4.1	Funktionale Anforderungen und Rahmenbedingungen .....	3
4.2	Vorgehensmodell .....	3
4.3	Erwartete Resultate .....	4
4.4	Zeitliche Vorgaben und Arbeitsaufteilung .....	4
5	Diagramme .....	5
5.1	UseCase-Diagramm .....	5
5.2	Klassendiagramm .....	7
5.3	Sequenzdiagramm.....	9
6	DevOps .....	10
6.1	Produktvision .....	10
6.2	Product Backlogs / User-Stories.....	10
6.3	Sprintplanung.....	11
6.4	Kanban Board.....	11
6.5	Sprint Reviews.....	11
6.5.1	Sprint 1 Review.....	11
6.5.2	Sprint 2 Review.....	12
6.6	Sprint Retrospektives .....	13
6.6.1	Sprint 1 Retrospektive.....	13
6.6.2	Sprint 2 Retroperspektive .....	14
7	Git-Repository .....	16
8	Testkonzept.....	17
8.1	Testziele.....	17
8.2	Teststrategien und Stufen.....	18

8.2.1	Teststrategien.....	18
8.2.2	Teststufen.....	18
8.2.3	Testarten .....	18
8.3	Testbericht .....	19
9	Programmcode .....	22
9.1	Introduktion .....	22
9.2	Code-Ausschnitte .....	22
10	Schlusswort .....	27
10.1	Reflexion.....	27
10.2	Fazit .....	27

## 2 Abbildungsverzeichnis

Abbildung 1 Vorlage für Minesweeper Quelle: minesweeper.online.....	4
Abbildung 2 UseCase-Diagramm .....	5
Abbildung 3 Klassendiagramm .....	7
Abbildung 4 Sequenzdiagramm .....	9
Abbildung 5 Eingeklappte Product-Backlogs.....	10
Abbildung 6 Burn-Down-Chart Sprint 1 .....	11
Abbildung 7 Kanban Board mit Swimlanes .....	11
Abbildung 8 Github-Repository .....	16
Abbildung 9 V-Modell.....	18

## 3 Tabellenverzeichnis

Tabelle 1 Positiv-Retroperspektive 1.....	13
Tabelle 2 Negativ-Retroperspektive 1.....	14
Tabelle Positiv-Retroperspektive 1.....	15
Tabelle Negativ-Retroperspektive 1.....	15
Tabelle Übergeordnete Testziele .....	17
Tabelle TestartenTestbericht .....	18
Tabelle Testprotokoll des Minesweeper Projektes.....	21

## 4 Vorwort

Diese Projektarbeit wurde uns (Sven, Manuel und Lewin) in dem zusammengesetzten Modul V226&426 als Zusammenarbeit aufgetragen. Das Ziel ist, am Schluss ein funktionsfähiges Minesweeper Spiel implementiert zu haben. Gefordert wird auch, dass eine Teamplanung erstellt wird und schlussendlich alles dokumentiert ist. Dabei soll das Gelernte aus den beiden Modulen V226 und V426 eingebunden werden. Im folgenden Unterkapitel haben wir noch die genaue Aufgabenstellung festgehalten.

### 4.1 Funktionale Anforderungen und Rahmenbedingungen

- Der Spieler wählt ein Feld, das er aufdecken möchte.
- Trifft der Spieler eine Bombe, hat er verloren und das Spiel ist vorbei.
- Ist auf dem Feld keine Bombe versteckt, so zeigt das Feld die Anzahl der Bomben in den 8 umliegenden Feldern an.
- Das Spiel ist gewonnen, wenn der Spieler alle Felder aufgedeckt hat, die keine Bombe enthalten.

Weitere funktionale Anforderungen:

- Der Spieler soll vorgängig den Schwierigkeitsgrad wählen können. Es sollen zumindest folgende drei Schwierigkeitsgrade zur Verfügung stehen: Einfach, Mittel, Schwer. Diese unterscheiden sich in der Grösse des Spielfeldes sowie in der Anzahl der Minen.
  - Einfach: Feld 8x8 mit 10 Minen
  - Mittel: Feld 16x16 mit 40 Minen
  - Schwer: Feld 30x16 mit 99 Minen
- Die Spielzüge sollen rückgängig gemacht werden können
- Die Stoppuhr, welche die benötigte Zeit für ein Spiel ausweist, ist optional
- In der bekannten Version des Spiels kann man nicht nur blind Felder aufdecken, sondern auch solche markieren, wo sicher eine Mine liegt. Diese Anforderung ist ebenfalls optional.

Zudem sollen in das Programm einige der gelernten Design-Patterns einfließen. Beim Vorgehen wird verlangt, dass alles genaustens geplant, mit Diagrammen visualisiert und vollständig dokumentiert wird, darunter befinden sich Diagramme wie das Sequenzdiagramm und das UseCase-Diagramm. Es soll eine JavaDoc-Dokumentation zum Programm erstellt werden. Zudem muss die ganze Logik mit Unit-Tests getestet werden.

### 4.2 Vorgehensmodell

Bei der Erstellung der Teamplanung wird gefordert, dass dies mit Hilfe von DevOps umgesetzt wird. Was heisst, dass das Projekt agil mit Scrum entwickelt wird. Es soll auch ein Kanban-Board zur Übersicht erstellt und mit passenden WIP-Limits versehen werden. Die Sprintplanung, die Sprintziele, Sprintbacklogs und alle User-Stories sollen jeweils vor jedem Sprint erstellt sein. Bei Scrum gibt es Daily-Scrums und Sprint-

Meetings die selbstverständlich protokolliert werden müssen. Zudem soll auch jeweils eine Sprint-Retrospektive am Ende des Sprints erfolgen.

### 4.3 Erwartete Resultate

- Link zu Ihrem Azure DevOps-Projekt.
- Protokolle zu den Scrum-Meetings
  - Daily-Scrum (Protokoll optional)
  - Sprint-Reviews
  - Sprint-Retrospektiven
- UseCase-Diagramm
- Klassendiagramm
- Sequenzdiagramm
- Dokumentation der Tests
- Java Projekt als JAR-Archivdatei mit
  - JavaDoc
  - Unittests

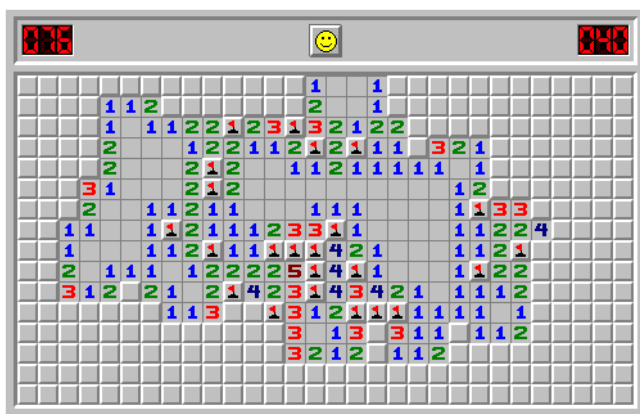


Abbildung 1 Vorlage für Minesweeper Quelle: minesweeper.online

### 4.4 Zeitliche Vorgaben und Arbeitsaufteilung

Die wichtigsten Termine sind: Kickoff am 28.04.2021 und Abgabe am 26.05.21. Somit ergibt sich ein Zeitraum von vier Wochen für das ganze Projekt, welches in zwei Sprints aufgeteilt wird. Sprint 1 eine Woche, Sprint 2 drei Wochen. Im Gesamten soll der Zeitaufwand ca. 36 Stunden betragen. Dies ergibt sich aus den drei Teammitgliedern:

5 MB à 3 Lektionen: 15 Lektionen à 3 Mitglieder: 45 Lektionen = 33.75h

5 Wochen à 1h Hausaufgabe pro Woche = 5h à 3 Teammitglieder = 15h

**Total: 48.75h**

## 5 Diagramme

### 5.1 UseCase-Diagramm

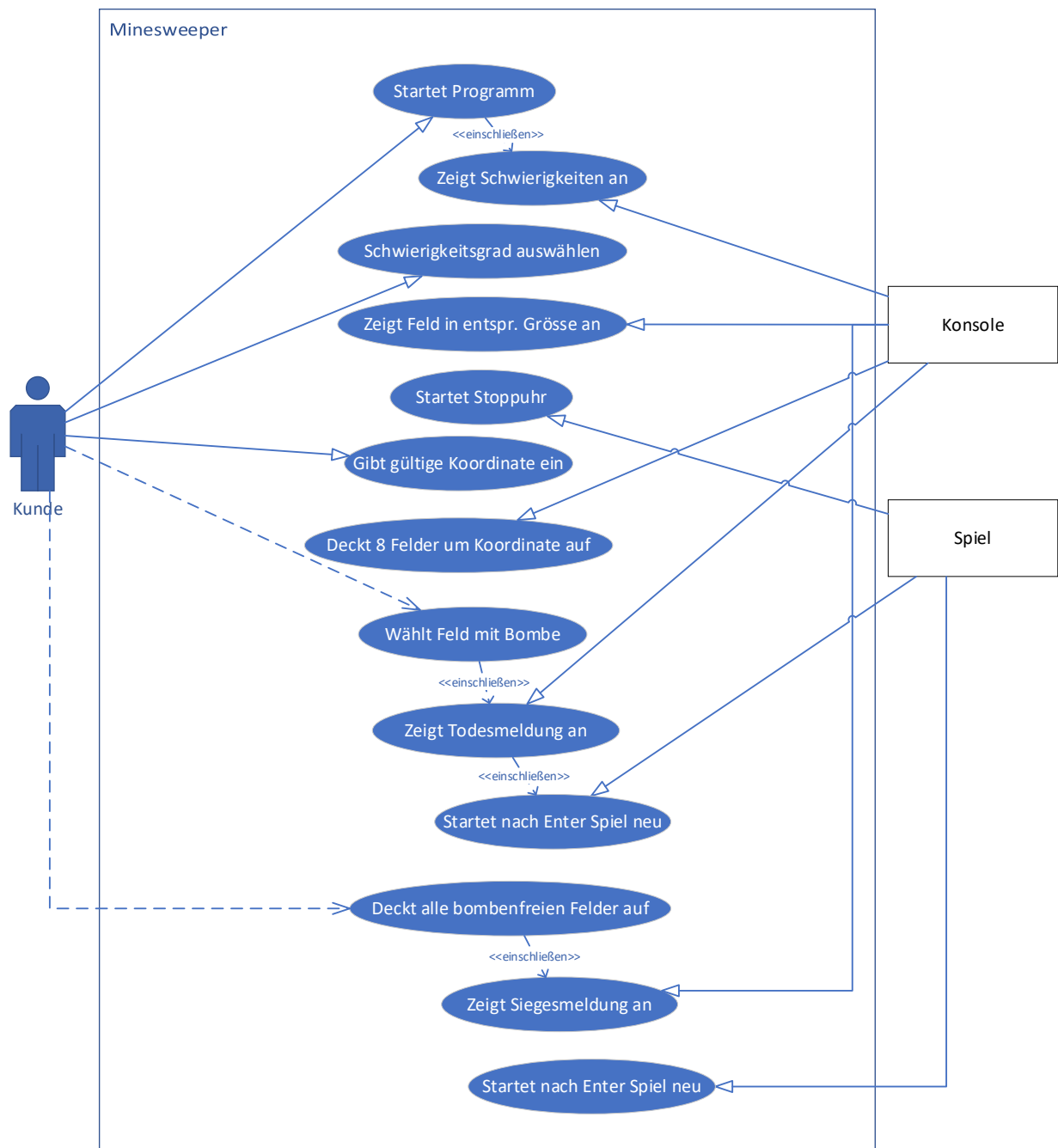


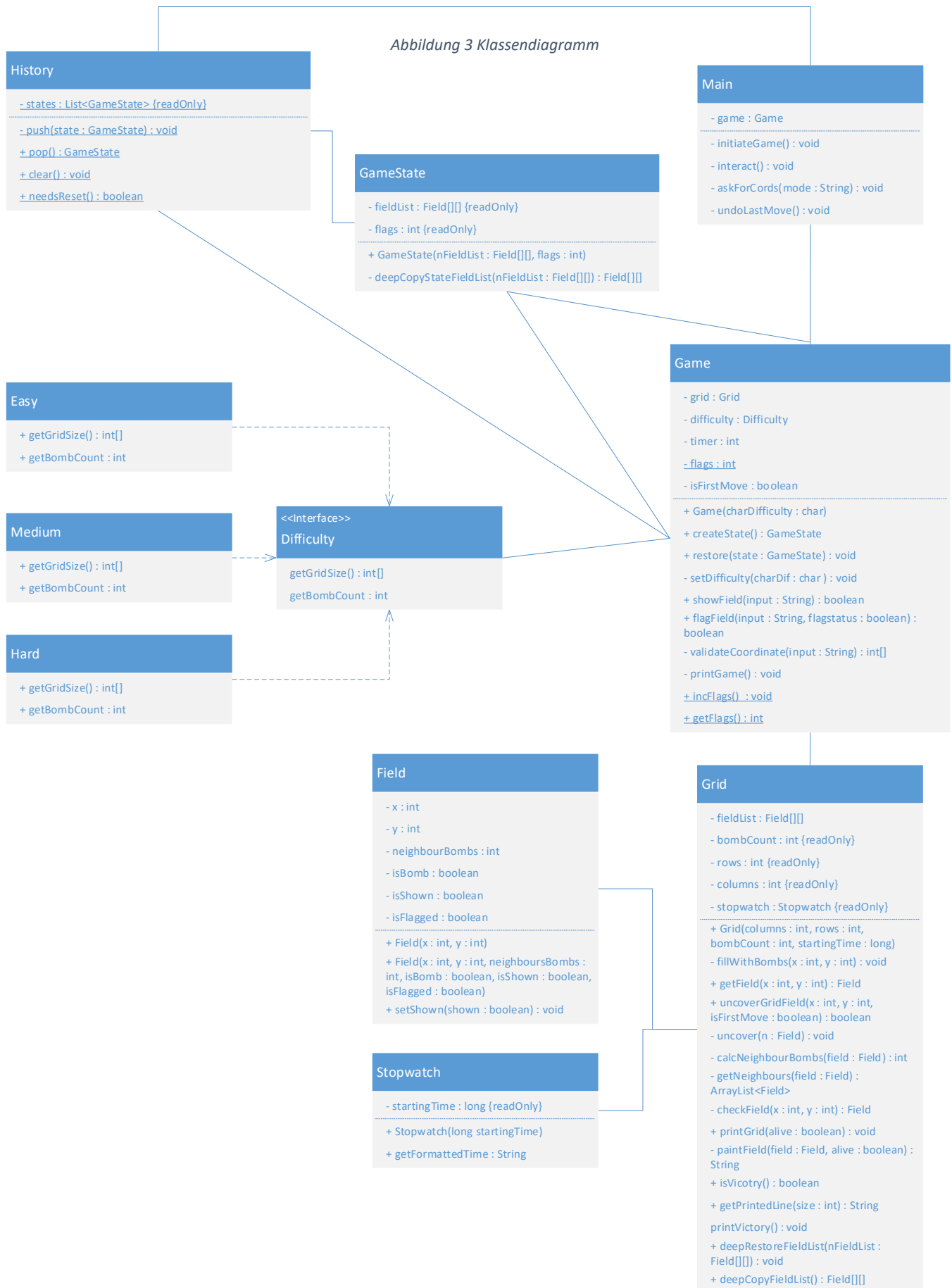
Abbildung 2 UseCase-Diagramm

Das UseCase-Diagramm war das erste der Diagramme, die wir erstellt haben. Es beinhaltet einen menschlichen Akteur, den Spieler und 2 Systeme. Diese sind das Spiel und die Konsole. Der Spieler startet das Programm und die Konsole lässt ihn die Schwierigkeit wählen. Der Spieler trifft nun eine Wahl und die Konsole gibt ihm ein entsprechendes Feld aus. Sobald das passiert ist, startet das Spiel eine Stoppuhr, um

die Zeit zu messen. Nun gibt der Spieler die Koordinate für das Feld ein, das er bearbeiten will. Wenn um das Feld herum keine Bomben sind, werden die 8 benachbarten Felder aufgedeckt. Sollte der Spieler ein Feld mit einer Bombe auswählen, zeigt ihm die Konsole eine Todesmeldung an und er kann mit Enter das Spiel neustarten. Wenn der Spieler alle Felder ohne Bomben aufgedeckt hat, zeigt die Konsole eine Siegesmeldung an und das Spiel kann mit Enter neu gestartet werden.

## 5.2 Klassendiagramm

Abbildung 3 Klassendiagramm





In unserem Klassendiagramm haben wir elf verschiedene Klassen dargestellt. Für die Schwierigkeitsstufen Hard, Medium und Easy haben wir das Strategy-Pattern verwendet. Die drei Klassen implementieren das Interface «Difficulty», welches sie dazu zwingt, die zwei Methoden darin zu überschreiben. Das Field ist eine Klasse, die ein einzelnes Feld im Spielfeld des Grids darstellen soll und entsprechende Funktionalität bietet. Für das Rückgängigmachen von Spielzügen haben wir das Memento-Pattern implementiert. Die Game-Klasse fungiert als Originator, von ihr sollen also alle benötigten Werte gespeichert werden. Das Memento ist die GameState-Klasse, die diese für das Spiel kritischen Werte verpackt, um dann im Caretaker, der History-Klasse, in den Stack aufgenommen werden zu können.

## 5.3 Sequenzdiagramm

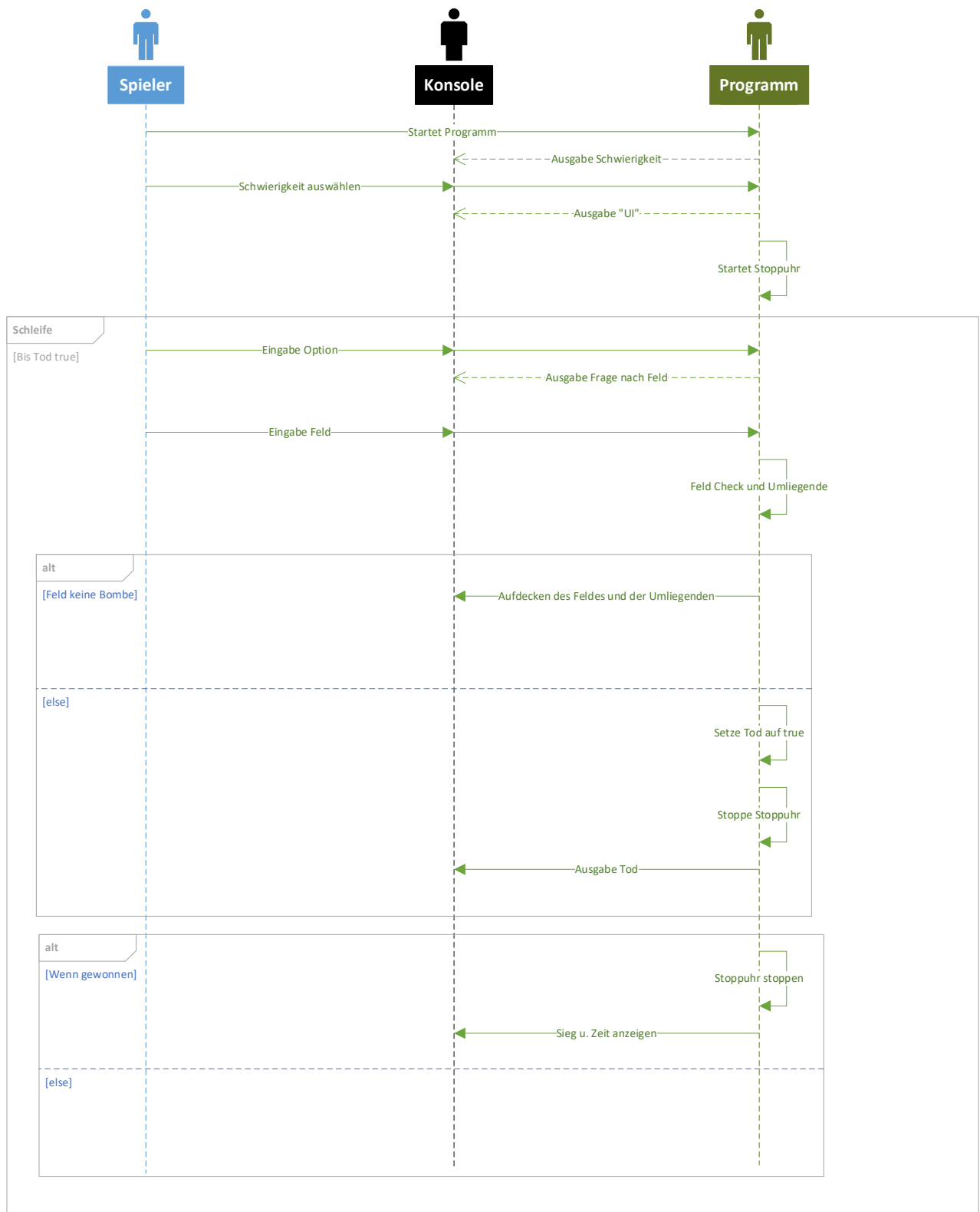


Abbildung 4 Sequenzdiagramm

Auch das Sequenzdiagramm hat drei Akteure, den Spieler, die Konsole und das Spiel. Als Erstes gibt die Konsole dem Spieler die Schwierigkeitsgrade aus und er entscheidet sich für einen, anschliessend wird das UI auf die Konsole ausgegeben und die Stoppuhr wird gestartet. Der Spieler gibt jetzt ein, was er machen möchte und anschliessend auf welches Feld der Befehl ausgeführt werden soll. Wenn das Feld keine Bombe ist, wird es und die umliegenden Felder aufgedeckt. Sollte es eine Bombe sein, wird die Stoppuhr angehalten und eine Todesmeldung ausgegeben. Falls alle Felder ausser die Bomben aufgedeckt wurden, wird die Stoppuhr angehalten und die Siegesanzeige ausgegeben.

## 6 DevOps

Dieses Projekt wurde basierend auf der agilen Methode Scrum erstellt. Scrum bietet Entwicklerteams eine sehr flexible Vorgehensweise für ihre Projekte. Vorgegeben ist, dass diese Planung mit dem Tool DevOps von Microsoft durchgeführt wird.

### 6.1 Produktvision

Wir wollen ein Minesweeper-Spiel mit Benutzeroberfläche in der Konsole programmieren, dass exakt den Vorgaben des Auftraggebers entspricht. Die Planung soll vollständig dokumentiert und nach dem agilen Vorgehensmodell Scrum gearbeitet werden. Unser Produkt soll alle verlangten Funktionen erfüllen und durch mögliche Erweiterungen herausragen. Benutzer sollen sich im Programm trotz der Konsolenbedienung gut zurechtfinden, es soll also übersichtlich und einfach zu bedienen sein. Da es sich um ein Schulprojekt handelt, wollen und werden wir mit diesem keinen finanziellen Erlös erzielen.

### 6.2 Product Backlogs / User-Stories

Nach dem Erstellen des DevOps haben wir als Team begonnen, die User-Stories zu erstellen, diese geben aus verschiedenen Perspektiven bekannt, welche Kriterien gefordert sind. Falls diese zu gross ausfallen, werden sie aufgeteilt. Schlussendlich betrug die Anzahl von User-Stories 13, welche wiederum eigene unterschiedliche Tasks beinhalten. User-Stories haben eine Vorgabe, wie sie geschrieben werden sollen, diese sieht wie folgt aus: Ich als (Zielgruppe) möchte (Kriterium). So kann gewährleistet werden, dass die einzelnen Stories auch qualitativ hochwertig und einfach zu verstehen sind. Bei jedem dieser Backlog-Items wurde durch das gesamte Team, mittels Abstimmung, geschätzt, wie viele Stunden benötigt werden, um die Aufgabe vollständig umzusetzen.

	Order	Work Item Type	Title
+	1	Product Backl...	> Schwierigkeitsgrade einstellbar
	2	Product Backl...	> Protokolle von Teamsitzungen verfassen
	3	Product Backl...	> Spielzug rückgängig machbar
	4	Product Backl...	> Stoppuhr im Spiel (Optional)
	5	Product Backl...	> Testkonzept erstellen
	6	Product Backl...	> Flagging (Optional)
	7	Product Backl...	> JavaDoc-Dokumentation
	8	Product Backl...	> JUnittests erstellen
	9	Product Backl...	> Basisfunktionalität
	10	Product Backl...	> Sprint Reviews & -Retrospektives
	11	Product Backl...	> Projektabgabe
	12	Product Backl...	> Gesamtdokumentation
	13	Product Backl...	> GitHub-Repository erstellen und testen

Abbildung 5 Eingeklappte Product-Backlogs

## 6.3 Sprintplanung

Folgend wurde die Sprintplanung eingeleitet. Die Product-Backlogs wurden in zwei Sprints eingeteilt. Der Erste Sprint dauert eine, der Zweite drei Wochen. Bei den Sprints werden die einzelnen Aufgaben eines Backlogs verschoben, um festzuhalten, ob diese Aufgaben noch zu machen, in Arbeit oder beendet sind. Da jeder dieser Aufgaben ein geschätzter Aufwand in Stunden zugeteilt worden ist, kann DevOps automatisch einen sogenannten Burn-Down-Chart erstellen.

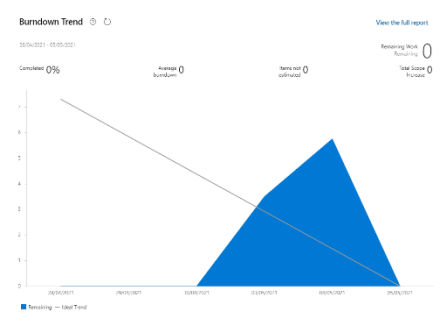


Abbildung 6 Burn-Down-Chart Sprint 1

## 6.4 Kanban Board

Das Kanban Board wurde erstellt und mit folgenden Swimlanes erweitert: Planung, Programm und Dokumentation. Diese Swimlanes sind Teil ein spezielles Add-ons. Die einzelnen Spalten des Boardes wurden mit WIP-Limits begrenzt. Somit dürfen nicht mehr als fünf Aufgaben gleichzeitig bearbeitet werden, ansonsten würde das Team überfordert werden. Dadurch, dass beide Spalten maximal fünf Aufgaben gleichzeitig beinhalten können, gibt es keinen «Stau», denn beide haben den gleich grossen Durchsatz.

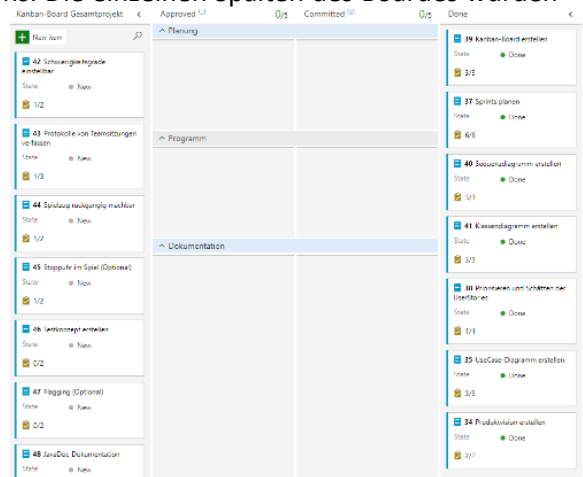


Abbildung 7 Kanban Board mit Swimlanes

## 6.5 Sprint Reviews

### 6.5.1 Sprint 1 Review

#### Protokoll Scrum-Meeting: Sprint 1 Review, 19.05.2021

Das Entwicklungsteam, bei uns wir alle drei, haben über unsere geleistete Arbeit gesprochen und die Ergebnisse vorgestellt. Wir haben den gesamten bisherigen Projektfortschritt im DevOps gemeinsam angeschaut, um einen Überblick zu erhalten. Da wir keinen direkten Product Owner haben, bewerteten wir gemeinsam die Ergebnisse und besprachen dann, ob wir sie so lassen, oder noch verbessern sollten. Bis auf eine kleine Verbesserung im Klassendiagramm wurden alle Ergebnisse vom Team akzeptiert und abgeschlossen. Unser Sprintziel für Sprint 1 lautete wörtlich: «Planung des Projektes zu weiten Teilen abschliessen und Diagramme dazu erstellen.» Jedes Teammitglied erhielt dann die Möglichkeit, ein Feedback zum bisherigen Projektverlauf zu äussern. Dabei kamen wir vor allem zum Schluss, dass wir den Sprint 1 erfolgreich absolviert und das Sprintziel erreicht hatten. Allerdings ist uns aufgefallen, dass wir wohl zu

wenig in den ersten Sprint eingeplant hatten, denn der Arbeitsaufwand zum zweiten Sprint ist nicht proportional verteilt und wir vermuten, dass wir uns im nächsten Sprint ziemlich beeilen müssen, um das ganze Projekt rechtzeitig abschliessen zu können.

Ins Zentrum unseres Sprint-Review-Meetings stellten wir die folgenden zwei Fragen, zu denen wir unsere Diskussionserkenntnisse auch gleich schriftlich zusammengefasst haben:

1. Welchen Wert hat der Sprint geschaffen?
  - Die vollständige Planung beider Sprints wurde erfolgreich abgeschlossen und zahlreiche Diagramme und anderweitige Implementierungshilfen erstellt.
2. Was hält uns davon ab, diesen Wert jetzt sofort zu nutzen?
  - Nichts, wir werden für den weiteren Projektverlauf stets die Planung und die Sprint-Backlogs im Auge behalten und uns strikt an die festgelegten Vorgehensweisen halten. Die Diagramme werden wir in der Implementierung einsetzen, um bei der Entwicklung Zeit und Verwirrungen sparen zu können.

Anpassungen im Product-Backlog haben wir keine vorgenommen. Im Rückblick auf den Verlauf würden wir allerdings noch mehr Aufgaben in Sprint 1 packen, damit Sprint 2 weniger überfüllt ist. Dies ist allerdings nun nicht mehr möglich und wir müssen uns im Sprint 2 einfach mehr beeilen. Dies ist allerdings nach Inspektion des Sprint-Backlogs und Konsultierung des Teams durchaus im Bereich des Möglichen. Zum Stand unserer Zwischenpräsentation waren wir mit der Planungsphase fast fertig. Aus diesem Grund haben wir uns entschieden, die Produktvision, die Diagramme, das Burn-Down-Chart sowie Sequenz-, Klassen- und Use-Case Diagramm einzubinden. Ausser dem Burn-Down-Chart sind diese Ergebnisse nicht auf dem DevOps einsehbar. Dieses Burn-Down-Chart ist, da wir die Planung mit in den Sprint einbezogen haben, leider etwas komisch geraten, weil dadurch immer mehr Aufwand dazu kam. Anschliessend haben wir das DevOps selbst noch der Klasse präsentiert, um unseren Fortschritt zu visualisieren.

## 6.5.2 Sprint 2 Review

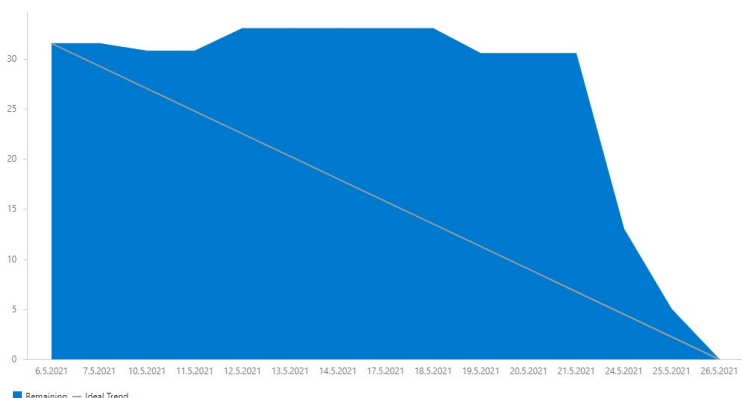
Die zweite Besprechung im Team war um einiges umfassender als die erste, dies liegt daran, dass der zweite Sprint drei Wochen lang ging, was auch heisst, dass mehr Arbeit erledigt werden musste und konnte.

Die folgenden Punkte wurden angesprochen:

Arbeitsaufteilung, Git-Repository, Sprintziel, Planung, Dokumentation, Zusammenarbeit im Team.

Das Sprint Ziel lautet: «Projekt vollständig implementieren, testen, dokumentieren und abgeben.»

Das Team gab schlussendlich ein Feedback zum Projektverlauf mithilfe der Retrospektiven. Diese gaben genaue Einblicke,



was gut und schlecht lief, zudem hat das Team versucht Lösungen für diese Probleme, bzw. den Erhalt der guten Dinge, zu finden.

Im Nachhinein wurde klar, dass die Arbeitseinteilung nicht gelungen ist, denn das war ein Anliegen aller. Das Dev-Ops gab einen sehr genauen Überblick der herrschenden Situation, denn Sprint zwei, obwohl er länger dauert, hatte er immens viele Aufgaben, welche lange dauerten. Das Team fand als Lösung, dass bereits Anfangs bei der Planung alle zustimmen müssen, welche Stories wo eingetragen werden, zudem muss eine klarere Einteilung der Aufgaben geschehen. Die Erstellung des Programmes verlief fast wie geplant, das Einzige was nicht richtig funktionierte, war das Git-Repository. Es ist uns ein Fehler unterlaufen, wodurch es leider gelöscht werden musste, was dazu führte, dass jetzt nur noch ein einziger Commit getätigt wurde. Zum Glück konnte das gesamte Projekt trotzdem noch gerettet werden. Um solche Dinge später zu beheben, schlug das Team vor, klarere Rollen zu vergeben. Positiv verlief jedoch die Dokumentation. Dies lag daran, dass die Aufteilung klar und strukturiert war. Alle im Team kamen zum Schluss, dass die Arbeit im Gesamten ein Erfolg war und somit Sprint 2 erfolgreich absolviert wurde.

Welchen Wert hat der Sprint geschaffen?

Die Beendigung des gesamten Projektes, darunter: dem Programm, Die Dokumentation, Alle Unit-Test, Dev-Ops, Git-Repository, usw.

## 6.6 Sprint Retrospektives

### 6.6.1 Sprint 1 Retrospektive

#### Protokoll Scrum-Meeting: Sprint 1 Retrospektive 19.05.2020

Im Scrum-Meeting vom 19.05.2020 hat das Team die Retrospektive zu Sprint 1 gemacht. Jedes Teammitglied hatte die Möglichkeit, Dinge aufzuführen, die während des ersten Sprints gut oder schlecht gelaufen sind. Im Anschluss darauf, wurden diese Dinge bewertet, jeder hatte im Gesamten drei Stimmen zur Verfügung. Die Rückmeldungen, die besonders stark vertreten waren, sind folgende:

#### Positiv:

Objekt	Votes
Erstellung des DevOps	2
Git-Repository erstellen	1
Userstories schätzen & priorisieren	1
Erstellung der verschiedenen Diagramme	1

Tabelle 1 Positiv-Retroperspektive 1

**Negativ:**

Objekt	Votes
Kanban-Board erstellen	1
Termineinhaltung	1

Tabelle 2 Negativ-Retroperspektive 1

Die Restlichen haben keine Stimmen bekommen, weshalb sie nicht aufgeführt werden.

**Negative und positive Aspekte des Sprints 1****Positiv:**

*Erstellung des DevOps:* Die Absprache im Team ist sehr gut gelaufen, zudem hat sich jedes Mitglied engagiert. Die einzelnen Product-Backlogs haben eine gute Qualität.

*Git-Repository erstellen:* Alle im Team haben beim Erstellen des Repositories Neues lernen können. Zudem ist der Vorgang und die kleine Schulung des Projektteams reibungslos verlaufen.

*Userstories schätzen & priorisieren:* Dies hat sehr gut funktioniert, jedes Teammitglied hat seinen Teil beigetragen, weshalb diese Aufgabe sehr schnell erledigt werden konnte.

*Erstellung der verschiedenen Diagramme:* Jedes Teammitglied übernahm einen Diagrammtyp, dadurch konnte die Kapazität enorm erhöht werden, was dazu führte, dass diese Aufgabe innerhalb von 40 Minuten erledigt war.

**Negativ:**

*Kanban-Board erstellen:* Die Erstellung des Boards war etwas schwierig, da das Team die Erfahrung mit Kanban noch nicht hatte, was zu Verzögerungen führte, da sich alle noch informieren mussten. Schliesslich konnte aber auch dies gelöst werden.

*Termineinhaltung:* Der Aufwand für die zwei Sprints hätte gleichmässiger verteilt werden müssen. Wir rechnen nun damit, dass im Sprint Überstunden geleistet werden müssen. Damit Sprint 2 besser läuft werden wir eine strikte Arbeitskontrolle einbauen.

**6.6.2 Sprint 2 Retroperspektive****Protokoll Scrum-Meeting: Sprint 2 Retrospektive 26.05.2020**

Im Scrum-Meeting vom 26.05.2020 hat das Team die Retrospektive zu Sprint 2 durchgeführt. Jedes Teammitglied hatte die Möglichkeit, Dinge aufzuführen, die während des ersten Sprints gut und schlecht gelaufen sind. Im Anschluss darauf, wurden diese Dinge bewertet, jeder hatte im Gesamten fünf Stimmen zur Verfügung. Die Rückmeldungen, die besonders stark vertreten waren, sind Folgende:

**Positiv:**

Objekt	Votes
Das Erstellen von JavaDoc im gesamten Team	2
Die Zusammenarbeit, bei den JUnit-Tests erstellen.	2
Code-Schnipsel einfügen und beschreiben	1
Inhalt der Dokumentation schreiben	1

*Tabelle 3 Positiv-Retroperspektive 1***Negativ:**

Objekt	Votes
Arbeitsaufteilung über den ganzen Sprint 2 hinweg	4
Die Einstellung des Dokumentes war von Fehlern geplagt.	3

*Tabelle 4 Negativ-Retroperspektive 1*

Die Restlichen haben keine Stimmen bekommen, weshalb sie nicht aufgeführt werden.



## Negative und positive Aspekte des Sprints 2

### Positiv:

*Erstellen von JavaDoc: Alle Teammitglieder haben tüchtig zusammengearbeitet, dass die Java Dokumentation des Codes entsteht. Über die Aufteilung wurde abgestimmt und jede Person konnte ihren Beitrag dazu leisten. Die Qualität ist stets auf dem gleichen Niveau, da eine einheitliche Vorgabe für JavaDoc benutzt wurde.*

*Zusammenarbeit, JUnit-Tests: Die Unit-Tests wurden im Team erarbeitet, dazu wurden jedem Teammitglied bestimmte Methoden zugewiesen, welche sie mit den notwendigen Tests prüfen sollten. Die Kommunikation war anfangs noch etwas haperig, aber nach einiger Zeit lief alles wie geplant.*

*Code-Schnipsel: Die Code Schnipsel wurden spezifisch ausgewählt und in der Dokumentation erklärt, dies konnte sehr effizient im Team erledigt werden.*

*Inhalt der Dokumentation: Beim Inhalt der Dokumentation war das gesamte Team einverstanden. Jedes Teammitglied schrieb zwar bestimmte Teile der Dokumentation, jedoch haben wir zur Qualitätssicherung unsere Texte gegenseitig verbessert.*

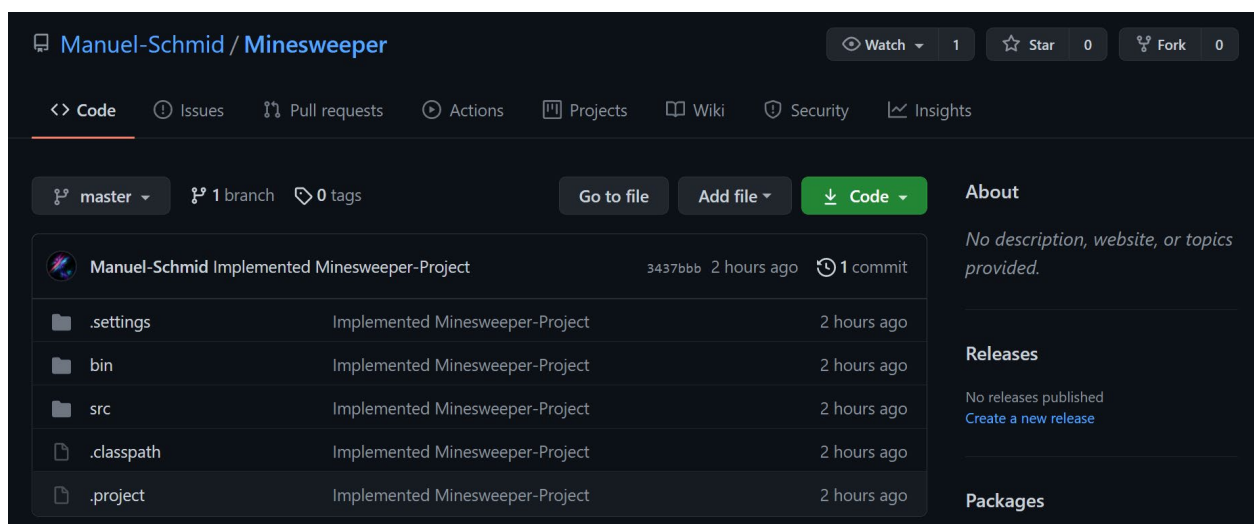
### Negativ:

*Arbeitsaufteilung: Die Arbeitsaufteilung funktionierte anfangs bei vielen der Aufgaben nicht gut. Doch nach einer Weile gelang es uns, alles in Einklang zu bringen.*

*Erstellung des Dokumentes: Das Dokument der Dokumentation hatte einige Fehler. Durch manche Bugs in Word wurde die Formatierung zerstört oder geändert.*

## 7 Git-Repository

Um auch auf ältere Versionen des Codes zuzugreifen und diesen allen Teammitgliedern verfügbar zu machen, haben wir uns entschieden Git in unserem Projekt zu verwenden. Zu diesem Zweck hat einer von uns ein Repository auf Github mit dem Namen Minesweeper erstellt und den anderen Zugriff darauf gegeben. In der CMD war es dann möglich das Repository und das Eclipse-Projekt zu verknüpfen. Um es zu



verwenden, mussten wir immer am Anfang einer Work-Session den Code herunterladen und ihn anschliessend auf einem eigenen Branch bearbeiten. Danach wird dieser Code hochgeladen, kontrolliert und anschliessend in den master-Branch eingeführt. Leider hatten wir Probleme mit dem alten Repository, weshalb wir ein neues erstellen mussten. Dadurch hat das Neue leider nur wenige Commits.

## 8 Testkonzept

### Beschreibung

Das Testkonzept beschreibt die Testziele, Testobjekte, Testarten, Testinfrastruktur sowie die Testorganisation. Es umfasst ebenfalls die Testplanung und die Testfallbeschreibungen. Für jeden Testfall wird eine detaillierte Testfallbeschreibung erstellt. Diese stellt die Spezifikation des Tests dar. Die Testplanung legt den logischen und zeitlichen Ablauf der Tests fest. Das Testkonzept bildet die Grundlage, auf der die Testorganisation und die Testinfrastruktur bereitgestellt und die Tests durchgeführt werden. Es wird bei neuen Erkenntnissen stets nachgeführt.

### 8.1 Testziele

#### Global messbare Testziele über alle Testfälle hinweg:

Nr.	Beschreibung	Messgrösse	Priorität*
1	<i>Unit-Tests sind alle erfolgreich</i>	<i>Keine falschen Tests</i>	<i>M</i>
2	<i>Unit-Test sind alle einzigartig</i>	<i>Keine testarten doppelt</i>	<i>1</i>
3	Unit-Test enthalten edge-cases	<i>Mindestens einer pro Methode</i>	<i>M</i>
4	<i>Alle Methoden werden mit Unit-Tests getestet</i>	<i>Jede Methode hat min. 1 Test</i>	<i>2</i>
5	<i>Keine Fehler bei den Integrationstests</i>	<i>Keine Integritätsfehler</i>	<i>M</i>
6	<i>Systemtests werden eingehalten</i>	<i>Entspricht den Anforderungen</i>	<i>M</i>
7	<i>Abnahmetests sind erfolgreich</i>	<i>Keine Mängel durch diesen Test</i>	<i>M</i>
* <b>Priorität: M = Muss / 1 = hoch, 2 = mittel, 3 = tief</b>			

Tabelle 5 Übergeordnete Testziele

## 8.2 Teststrategien und Stufen

Hier werden die Strategien und Stufen beschrieben und kurz erläutert.

### 8.2.1 Teststrategien

Die Erste Stufe der Tests, also die Unit-Tests werden mithilfe von JUnit-Tests bewältigt.

Die Integrationstests werden automatisch vom Repository-System Git durchgeführt.

Die Systemtests werden mittels Anforderungen überprüft. Schlussendlich werden die

Abnahmetests mittels Vorlagen durch den Nutzer und Kunde überprüft. Mit Unit-Tests testen wir die privaten Methoden nicht direkt, sondern den Effekt, die sie auf das System und die öffentlichen Methoden haben.

### 8.2.2 Teststufen

In diesem Projekt wird mithilfe von vier verschiedenen Tests geprüft, ob es die gewünschten Rahmenbedingungen einhält. Die einzelnen Teststufen sind folgende: Unit-Test, Integrations-Test, Systemtests und Abnahmetests. Mithilfe all dieser wird die Qualität gewährleistet.

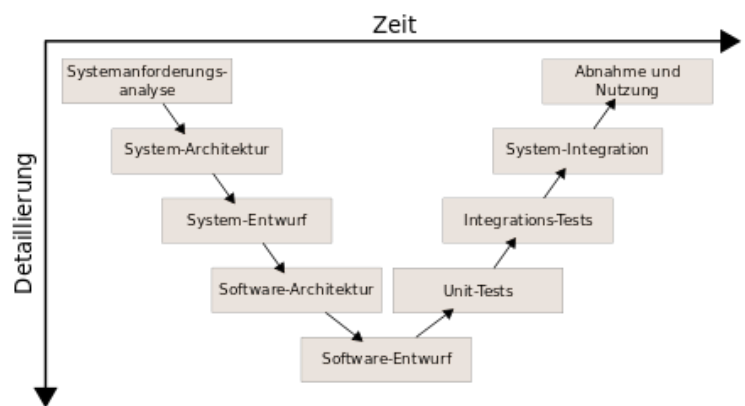


Abbildung 9 V-Modell

### 8.2.3 Testarten

Nr.	Testart	Beschreibung
1	JUnit-Tests	Einzelne Module werden auf ihre Richtigkeit mit Hilfe eines Java Tools getestet.
2	Integration-Tests	Test von Zusammenarbeit voneinander abhängigen Komponenten.
3	Systemtests	Gesamte Anforderungen (funktionale und nicht funktionale) werden gegen das System getestet.
4	Abnahmetests	Test durch den Kunden

Tabelle 6 Testarten Testbericht

### 8.3 Testbericht

Nr.	Testname	Erwartetes Ergebnis	Tatsächliches Ergebnis	Bestanden Ja/Nein
1	testEasyDifficultyBombCount	10, da die getBomb() Methode das zurück gibt.	10	Ja
2	testMediumDifficultyBombCount	40, da die getBomb() Methode das zurück gibt.	40	Ja
3	testHardDifficultyBombCount	99, da die getBomb() Methode das zurück gibt.	99	Ja
4	testEasyDifficultyFlagCount	10, da die getBomb() Methode das zurück gibt.	10	Ja
5	testMediumDifficultyFlagCount	40, da die getBomb() Methode das zurück gibt.	40	Ja
6	testHardDifficultyFlagCount	99, da die getBomb() Methode das zurück gibt.	99	Ja
7	testFlagIncrease	11, da die getBomb() Methode das zurück gibt.	11	Ja
8	testShowFieldInvalid-Coordinates	True, da x20 nicht existiert.	True	Ja
9	testShowFieldInvalid-CoordinatesSpace	True, da a 6 ein falsches Format ist.	True	Ja
10	testShowFieldIsAlreadyShown	True, da C4 bereits aufgedeckt ist.	True	Ja
11	testFlagFieldStandard	True, A2 wird normal aufgedeckt.	True	Ja
12	testFlagFieldEdgeCoordinates	True, da A003 ein falsches Format ist.	True	Ja
13	testFlagFieldInvalid-CoordinatesSpace	True, da a 6 ein falsches Format ist.	True	Ja
14	testFlagFieldInvalid-Coordinates	True, da x30 nicht existiert.	True	Ja
15	testFlagFieldIsAlreadyFlagged	True, da das Feld C4 bereits geflagged ist.	True	Ja
16	testFlagFieldNoFlagsLeft	True, da alle Flaggen aufgebraucht sind.	True	Ja
17	testUnFlagFieldStandard	True, da A2 normal von einer Flagge befreit wird.	True	Ja
18	testUnFlagFieldEdgeCoordinates	True, A003 ein falsches Format ist.	True	Ja

Nr.	Testname	Erwartetes Ergebnis	Tatsächliches Ergebnis	Bestanden Ja/Nein
19	testUnFlagFieldInvalidCoordinatesSpace	True, da a 6 ein falsches Format ist.	True	Ja
20	testUnFlagFieldInvalidCoordinates	True, da x30 nicht existiert.	True	Ja
21	testUnFlagFieldIsAlreadyUnFlagged	True, da C4 noch keine Flagge hat.	True	Ja
22	testFlagFieldCount	8, da zwei von zehn Flaggen gesetzt werden.	8	Ja
23	testFlagFieldFlagsCount	10, da alle Flaggen gesetzt werden und sie danach aber wieder entflagged werden.	10	Ja
24	testGetField	67, da beim Grid 6,7 abgefragt wird.	67	Ja
25	testEasyActualBombCount	10, die Bomben werden aus den Arrays ausgelesen.	10	Ja
26	testMediumActualBombCount	40, die Bomben werden aus den Arrays ausgelesen.	40	Ja
27	testHardActualBombCount	99, die Bomben werden aus den Arrays ausgelesen.	99	Ja
28	testDeepCopyFieldList	False	False	Ja
29	testGetBombCountEasy	10, die Bomben werden mithilfe der Methode ausgelesen.	10	Ja
30	testGetBombCountMedium	40, die Bomben werden mithilfe der Methode ausgelesen.	40	Ja
31	testGetBombCountHard	99, die Bomben werden mithilfe der Methode ausgelesen.	99	Ja
32	testGetRowsEasy	8, da die Zeilen aus dem Grid ausgelesen werden.	8	Ja
33	testGetRowsMedium	16, da die Zeilen aus dem Grid ausgelesen werden.	16	Ja
34	testGetRowsHard	16, da die Zeilen aus dem Grid ausgelesen werden.	16	Ja
35	testGetColumnsEasy	8, da die Spalten aus dem Grid ausgelesen werden.	8	Ja

Nr.	Testname	Erwartetes Ergebnis	Tatsächliches Ergebnis	Bestanden Ja/Nein
36	testGetColumnsMedium	16, da die Spalten aus dem Grid ausgelesen werden.	16	Ja
37	testGetColumnsHard	30, da die Spalten aus dem Grid ausgelesen werden.	30	Ja
38	testLinePrinter	-----, da getPrintedLine() die anzahl Chars berechnet.	-----	Ja
39	testFieldConstructor	False, da actualShown, actualBomb und actualFlagged automatisch false sind. actualX und actualY sind 2 und 4.	False & 2 & 4	Ja
40	testStopwatch	Aktuelle Zeit, da die aktuelle Zeit gelesen wird.	Aktuelle Zeit	Ja
41	testUncoverGridFieldFirstMove	True, da Feld 0,0 als erstes Feld aufgedeckt wird.	True	Ja
42	testUncoverGridFieldNormal	True, da Feld 1,5 aufgedeckt wird.	True	Ja
43	testUncoverGridFieldVictory	False, da alle Bomben mit einer Flagge bedeckt werden.	False	Ja
44	testSetShown	True, da das Feld 6,7 aufgedeckt wird.	True	Ja


Tabelle 7 Testprotokoll des Minesweeper Projektes

Alle Unit-Test laufen nun erfolgreich, ein Paar Fehler mussten wir noch beheben, womit wir allerdings keine grossen Probleme hatten, da der Ursprung der Fehler schnell geklärt war.

```

Minesweeper_UnitTests
Runs: 44/44 * Errors: 0 • Failures: 0

```



## 9 Programmcode

### 9.1 Introduction

In diesem Kapitel werden wir ein Paar Code-Ausschnitte aus unserem Programmcode, die wir als wichtig empfinden, als Bilder einfügen und erklären, was darin passiert. Dabei werden wir mehr auf den Effekt, den die Ausschnitte auf das ganze Programm haben, eingehen, als konkret auf die Art, wie dies implementiert wurde.

### 9.2 Code-Ausschnitte

Wenn ein neues Spiel erstellt wird, wird mit der setDifficulty-Methode unter Anwendung des Strategy-Patterns die Schwierigkeitsstufe gesetzt. Die Daten werden der gewählten Schwierigkeit entsprechend genutzt, um die Anzahl Flags und die Ausmasse des neu erstellten Grids zu bestimmen.

```
public Game(char charDifficulty) {
    setDifficulty(charDifficulty);
    flags = difficulty.getBombCount();
    grid = new Grid(difficulty.getGridSize()[0], difficulty.getGridSize()[1], difficulty.getBombCount(), new Date().getTime());
}

private void setDifficulty(char charDif) {
    switch (charDif) {
        case 'E' → difficulty = new Easy();
        case 'M' → difficulty = new Medium();
        case 'H' → difficulty = new Hard();
    }
}
```

Der Konstruktor der Grid-Klasse erstellt das neue Grid mit den übergebenen Massen, die alle als final Konstanten gespeichert werden, sie können also danach nicht mehr verändert werden. Danach wird unser zweidimensionales Array mit neuen Field-Objekten befüllt.

```
public Grid(int columns, int rows, int bombCount, long startingTime) {
    this.rows = rows;
    this.columns = columns;
    this.bombCount = bombCount;
    this.stopwatch = new Stopwatch(startingTime);
    fieldList = new Field[rows][columns];

    for (int i = 0; i < fieldList.length; i++) {
        for (int j = 0; j < fieldList[0].length; j++) {
            fieldList[i][j] = new Field(i, j);
        }
    }
}
```

Es war uns ein Anliegen, als zusätzliche Funktionalität einzubauen, dass der Spieler beim ersten Spielzug nicht sterben, also nicht auf eine Bombe treten kann. Um dies zu gewährleisten, wird das Feld erst nach dem ersten Spielzug, mit der Methode `fillWithBombs()`, mit Bomben befüllt. Den Ausschnitt der `uncoverGridField`-Methode, in der dieser Fall abgefragt wird, ist im Bild unten zu sehen. Der Fall, das mit einem Undo dieser erste Spielzug erneut eingegeben werden könnte, wurde bedacht und abgefangen. Die übergebenen Koordinaten sind die des Feldes, das der Spieler als erstes aufdecken will. Der obere Loop im unteren Bild füllt das Feld mit der, der Schwierigkeit entsprechenden, Anzahl Bomben während darauf geschaut wird, dass kein Feld zweimal zur Bombe gemacht und das gewünschte Startfeld ebenfalls nicht verändert wird.

```
public boolean uncoverGridField(int x, int y, boolean isFirstMove) {  
    if (isFirstMove) {  
        fillWithBombs(x,y);  
    }  
}
```

```
private void fillWithBombs(int x, int y) {  
    int bx, by;  
    Random r = new Random();  
    for (int i = 0; i < bombCount; i++) {  
        do {  
            bx = r.nextInt(rows);  
            by = r.nextInt(columns);  
        } while (getField(bx,by).isBomb() || (bx == x && by == y));  
        getField(bx,by).setBomb(true);  
    }  
  
    for (int i = 0; i < fieldList.length; i++) {  
        for (int j = 0; j < fieldList[0].length; j++) {  
            fieldList[i][j].setNeighbourBombs(calcNeighbourBombs(fieldList[i][j]));  
        }  
    }  
}
```



In der Aufgabenstellung ist nur festgelegt, dass die 8 benachbarten Felder des aufzudeckenden Feldes auch aufgedeckt werden sollen, falls dieses Feld keine benachbarten Bomben hat. Wir wollten uns aber die persönliche kleine Herausforderung stellen, dies wie im echten Minesweeper-Spiel umzusetzen, also dass auch die Nachbarn der Nachbarn usw. abgefragt und aufgedeckt werden, falls diese wieder keine benachbarten Bomben haben. Diese Methode ist von der Logik her wahrscheinlich die anspruchsvollste, also sozusagen unsere Königsmethode, da der positive Effekt auf das Spielerlebnis unserer Meinung nach sehr gross ist. Zu diesem Zweck haben wir eine Methode geschrieben, welche auf Rekursion basiert:

```
private void uncover(Field n) {  
    n.setShown(true);  
    if(n.getNeighbourBombs() == 0) {  
        ArrayList<Field> neighbours = getNeighbours(n);  
        for (Field f : neighbours) {  
            if (!f.isBomb() && !f.isShown()) {  
                uncover(f);  
            }  
        }  
    }  
}
```

In der getNeighbours-Methode, die essenziell für die oben erwähnte Funktion ist, werden die benachbarten Felder des übergebenen Feldes als Liste zurückgegeben, von der aus sie, da sie als Pointer zu den Feldern im echten Spielfeld fungieren, direkt verändert werden können. Mit checkField() wird verhindert, dass bei Feldern am Rand allenfalls nicht existente Objekte zur List hinzugefügt werden, da sie als null zurückgegeben und beim return rausgefiltert werden. Ansonsten könnte hier sehr einfach ein Error ausgelöst werden. Also wird diese Methode im Fall eines Eckfeldes selbständig die nicht-existenten Felder ausfiltern und eine Liste mit nur drei Feldern zurückgegeben.

```
private ArrayList<Field> getNeighbours(Field field) {  
    int x = field.getX();  
    int y = field.getY();  
    ArrayList<Field> n = new ArrayList<>();  
    n.add(checkField(x, y+1));  
    n.add(checkField(x, y-1));  
    n.add(checkField(x-1, y));  
    n.add(checkField(x-1, y+1));  
    n.add(checkField(x-1, y-1));  
    n.add(checkField(x+1, y));  
    n.add(checkField(x+1, y+1));  
    n.add(checkField(x+1, y-1));  
    return n.stream().filter(Objects::nonNull).collect(Collectors.toCollection(ArrayList::new));  
}
```

Das wohl grösste und bei weitem frustrierendste Problem, auf das wir bei der Implementierung gestossen sind, war die Umsetzung des Memento-Pattern. Dieses verstehen wir zwar gut und haben es auch fehlerfrei implementiert, nur haben wir nicht berechnet, das Java zweidimensionale Arrays von Objekten als Referenzen übergibt und somit sämtliche unserer BackUps auf dem Stack stets synchron mit dem Spielfeld geändert wurden, was natürlich das genaue Gegenteil von dem ist, was ein BackUp tun soll. Gelöst haben wir das Ganze, indem wir nicht die Liste, sondern die Inhalte der Liste kopiert und in eine ganz neue Liste gespeichert haben, bevor sie übergeben werden. Die Lösung ist recht simpel, aber der Prozess, das eigentliche Problem zu finden hat uns viel Zeit und Nerven gekostet. Umso grösser war dann die Freude, als die Funktion schlussendlich voll-Funktionsfähig war. Die Zwei Methoden zum Kopieren und Wiederherstellen des Spielfelds sind unten abgebildet.

```
public void deepRestoreFieldList(Field[][] nFieldList) {
    fieldList = new Field[rows][columns];
    for (int i = 0; i < fieldList.length; i++) {
        for (int j = 0; j < fieldList[0].length; j++) {
            Field f = nFieldList[i][j];
            fieldList[i][j] = new Field(f.getX(), f.getY(), f.getNeighbourBombs(), f.isBomb(), f.isShown(), f.isFlagged());
        }
    }
}

public Field[][] deepCopyFieldList() {
    final Field[][] copy = new Field[rows][columns];
    for (int i = 0; i < copy.length; i++) {
        for (int j = 0; j < copy[0].length; j++) {
            Field f = getField(i, j);
            copy[i][j] = new Field(f.getX(), f.getY(), f.getNeighbourBombs(), f.isBomb(), f.isShown(), f.isFlagged());
        }
    }
    return copy;
}
```

Wir haben das Programm in zwei Versionen geschrieben, eine für Eclipse und eine für IntelliJ, da wir bei Eclipse nicht herausgefunden haben, wie man farbigen Text in die Konsole ausgeben kann. Auch die restliche Formatierung der Ausgabe haben wir aufgrund der unterschiedlichen Fonts ein wenig angepasst. Da wir nach Vorgabe nur das Eclipse-Projekt abgeben müssen, wollen wir hier noch ein paar Bilder, die zeigen, wie das Spiel in Farbe aussieht, einfügen. Als Erklärung für das unterste Bild: Wir haben noch eine weitere freiwillige Erweiterung implementiert, nämlich werden beim Tod des Spielers alle Bomben aufgedeckt, genau wie beim Original.

```

----- { Minesweeper } -----

Enter Difficulty ('E' = Easy, 'M' = Medium, 'H' = Hard) :
E

   | 01 | 02 | 03 | 04 | 05 | 06 | 07 | 08 | |
| A | ?? | ?? | ?? | ?? | ?? | ?? | ?? | ?? |
| B | ?? | ?? | ?? | ?? | ?? | ?? | ?? | ?? |
| C | ?? | ?? | ?? | ?? | ?? | ?? | ?? | ?? |
| D | ?? | ?? | ?? | ?? | ?? | ?? | ?? | ?? |
| E | ?? | ?? | ?? | ?? | ?? | ?? | ?? | ?? |
| F | ?? | ?? | ?? | ?? | ?? | ?? | ?? | ?? |
| G | ?? | ?? | ?? | ?? | ?? | ?? | ?? | ?? |
| H | ?? | ?? | ?? | ?? | ?? | ?? | ?? | ?? |

Flags: 10

```

```

   | 01 | 02 | 03 | 04 | 05 | 06 | 07 | 08 |
| A | ?? | 01 | ?? | ?? | ?? | ?? | ?? |
| B | ?? | ?? | ?? | ?? | ?? | ?? | ?? |
| C | ?? | ?? | ?? | ?? | ?? | ?? | ?? |
| D | ?? | ?? | ?? | ?? | 03 | ?? | ?? |
| E | 01 | 01 | 01 | ?? | ?? | ?? | ?? |
| F |   |   | 01 | 01 | ?? | ?? | ?? |
| G |   |   |   | 01 | ?? | ?? | ?? |
| H |   |   |   | 01 | ?? | ?? | ?? |

```

```

-----
YOU WON !!!
Your time: 31 seconds
-----

```

```

-----
YOU WON !!!
Your time: 1 minutes, 51 seconds
-----

```

```

   | 01 | 02 | 03 | 04 | 05 | 06 | 07 | 08 | 09 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | |
| A | ?? | ?? | ?? | ?? | 01 |   |   |   |   |   |   | 01 | ## | 01 |   |
| B | ?? | ?? | ?? | ?? | 02 | 01 | 02 | 01 | 01 |   |   | 01 | 01 | 01 |   |
| C | ?? | ?? | ?? | ?? | ?? | ?? | ?? | ?? | 02 |   |   |   |   |   |   |
| D | ?? | ?? | ?? | ?? | 03 | ?? | ?? | ?? | 02 |   |   | 01 | 02 | 03 | 02 | 01 |
| E | ?? | ?? | ?? | ?? | ?? | ?? | ?? | ?? | 02 |   |   | 01 | ?? | ?? | ?? |
| F | ?? | ?? | ?? | ?? | ?? | ?? | ?? | ?? | 01 | 01 | 01 | 02 | ?? | ?? | ?? |
| G | ?? | ## | ?? | ?? | ?? | ?? | ?? | ?? | ?? | ?? | ?? | ?? | ?? | ?? |
| H | ?? | ?? | ?? | ?? | ?? | ?? | ?? | ?? | ?? | ?? | ?? | ?? | ?? | ?? |
| I | ?? | ?? | ?? | ?? | ?? | ?? | ?? | ?? | ?? | ?? | ?? | ?? | ?? | ?? |
| J | ?? | ?? | ?? | ?? | ?? | ?? | ?? | ?? | ?? | ?? | ?? | ?? | ?? | ?? |
| K | ?? | ?? | ?? | ?? | ?? | ?? | ?? | ?? | ?? | ?? | ?? | ?? | ?? | ?? |
| L | ?? | ?? | ?? | ?? | ?? | ?? | ?? | ?? | ?? | 01 | ?? | ?? | ?? | ?? | ?? |
| M | ?? | ?? | ?? | ?? | ?? | ?? | ?? | ?? | ?? | ?? | ?? | ?? | ?? | ?? |
| N | ?? | ?? | ?? | ?? | ?? | ?? | ?? | ?? | ?? | ?? | ?? | ?? | ?? | ?? |
| O | ?? | ?? | ?? | ?? | ?? | ?? | ?? | ?? | ?? | ?? | ?? | ?? | XX | ?? |
| P | ?? | ?? | ?? | ?? | ?? | ?? | ?? | ?? | ?? | ?? | ?? | ?? | ?? | ?? |

Flags: 38

-----
YOU DIED !!!
-----

```

## 10 Schlusswort

### 10.1 Reflexion

Dieses Projekt hat uns, vor allem planungstechnisch, sehr viel abverlangt. Wir haben entgegen unserer Erwartung sehr viel Zeit mit dem Einrichten des DevOps, also der Planung verbracht. Zuerst war es sehr ungewohnt für uns, weil wir bis zu diesem Zeitpunkt noch nie mit Scrum als Vorgehensmodell gearbeitet haben, aber wir haben uns schnell daran gewöhnt und den enormen Mehrwert für dynamische Gruppenprojekte erkannt. Überraschenderweise haben uns die Akzeptanzkriterien der Userstories ein wenig Probleme bereitet, welche aber nach kurzem Durchlesen der Scrum-Dokumentation wieder geklärt wurden. Als wir die Sprints geplant hatten, haben wir realisiert das unser Burn-Down-Chart, während dem Arbeiten an Aufwand zunahm, weil wir das Erstellen des DevOps in den Sprint 1 miteinbezogen haben. Dies werden wir auf jeden Fall bei einem zukünftigen Projekt anders machen. Ausserdem haben wir unser Wissen und unsere Kompetenz im Bereich Scrum und agiles Arbeiten festigen können, da wir es an einem praxisorientierten Beispiel ausprobieren konnten. Wir haben auch zum ersten Mal für ein Projekt Git in Zusammenarbeit verwendet. Bei bisherigen Projekten haben wir meistens einfach einer verantwortlichen Person den Code geschickt, welche dann die Commits und Pushes durchgeführt hat. Also haben wir anfangs etwas Zeit gebraucht, um die Versionsverwaltung mit verschiedenen Kollaborateuren richtig zu verstehen. Was das Programm an sich angeht, haben wir uns ein wenig Zeit gelassen und es trat ein unerwartetes Problem auf, aufgrund dessen wir uns gezwungen sahen, ein neues Git-Repository zu erstellen. So sind leider die Originalen Commits und Pushes verloren gegangen und wir mussten im neuen Repository den gesamten Code in nur wenigen Commits neu hochladen.

### 10.2 Fazit

Alles in allem war das Projekt also sehr lehrreich, besonders was die Planungsphase anging. Wir haben das Vorgehensmodell Scrum benutzt und auch zum ersten Mal mit einem agilen Projektmanagement Tool gearbeitet. Ausserdem war es eine sehr gute Idee, Git zu verwenden, da es das Teilen und gemeinsame Bearbeiten des Programmcodes sehr erleichtert hat. Die Aufgabenstellung hat uns auch die Möglichkeit gegeben, mit 2-dimensionalen Arrays zu arbeiten, um auch so eine neue Erfahrung zu machen und unser Repertoire zu erweitern. Somit war dieses Projekt sehr wertvoll für unser Wissen und unsere Fachkompetenzen.