



Chess of Duty

Advanced Software-Engineering

im Rahmen der Prüfung zum
Bachelor of Science (B.Sc.)

des Studienganges Informatik

an der Dualen Hochschule Baden-Württemberg Karlsruhe

von

Clemens Richter & Johannes Peters

Abgabedatum:	30. April 2023
Bearbeitungszeitraum:	04.10.2022 - 30.04.2023
Matrikelnummer, Kurs:	xxxxxxx & 5802185, TINF20B1
Betreuer der Arbeit:	Herr Daniel Lindner

Inhaltsverzeichnis

1	Einleitung	1
2	Domain Driven Design	2
2.1	Ubiquitous Language	2
2.2	Domainenbausteine	2
2.3	Domainenevents	5
3	Clean Architecture	6

1 Einleitung

Im Rahmen des Moduls „Advanced Software Engineering“ wurde ein Schachspiel als Projektgrundlage ausgewählt. Chess of Duty ist ein Offline-Multiplayer-Schachspiel für zwei Personen. Das Hauptziel des Programmentwurfs besteht darin, Schach gemäß den Standardregeln zu implementieren. Dabei soll ein Leaderboard für die Spieler eingeführt werden, das auf einem Elo-System basiert.

Der Nutzen des Schachspiels für unsere Kunden entspricht dem von anderen Videospielen. Die Anwendung dient ausschließlich der Unterhaltung der Nutzer. Zusätzlich können die Anwender ihre strategischen Fähigkeiten und logisches Denken trainieren.

Das Schachspiel wird objektorientiert konzipiert und in Processing programmiert. Processing ist eine Open-Source-Programmiersprache, die auf Java basiert und einen besonderen Schwerpunkt auf die einfache Erstellung von Grafiken und Animationen setzt. Dadurch eignet sich Processing besonders für die Gestaltung interaktiver Benutzeroberflächen.

2 Domain Driven Design

2.1 Ubiquitous Language

Der Auftraggeber des Programmentwurfs ist ein deutscher Kunde. Obwohl in der Programmierung Englisch als die Standardsprache angenommen wird, wird aufgrund der Kundenlokalität die Projektsprache als „Deutsch“ festgelegt. Dadurch wird versucht die meisten Ausdrücke aus der Domäne ins Deutsche zu übernehmen. Auch wenn dies für den ungeübten Programmierer, der das Programmieren nur in Englisch ausübt, eine zusätzliche Herausforderung darstellt und teilweise zu Namensfindungsschwierigkeiten oder längeren Funktionsnamen führen kann, wird an der Domänensprache festgehalten, um dem Kunden den Code so übersichtlich wie möglich übergeben zu können.

2.2 Domainenbausteine

Value Objects

Ein Value Object ist ein unveränderliches Objekt, das einen bestimmten Wert repräsentiert und keine eigene Identität besitzt. Es definiert sich durch seine Eigenschaften und nicht durch eine eindeutige Identität, wodurch es austauschbar und vergleichbar wird.

Im Standardschach repräsentiert ein **Feld** eine Position, die durch eine Kombination aus Spalte und Zeile und Farbe definiert ist. Da Schachfelder keinen Lebenszyklus haben und ihre Gleichheit durch ihre Position bestimmt wird, werden sie als Value Objects geführt. Weiterhin ist es besonders wichtig für die grundlegende Funktion eines Schachspiels, dass Schachfelder als unveränderliche Objekte behandelt werden, da ein Schachspiel auf einer „festen Unterlage“ spielt wird.

Ein **Schachbrett** kann ebenfalls als Value Object betrachtet werden, weil es einen unveränderlichen Zustand repräsentiert. Ein Schachbrett ist in der domainspezifischen Abbildung eines Schachspiels von der Realität in Programmcode eine übergeordnete

Verwaltungsstruktur der einzelnen Felder. Dabei werden 64 Schachfelder definiert und als zweidimensionale Spielebene dargestellt. Die Gleichheit von zwei Schachbrettern basiert auf der Gleichheit ihrer Felder, unabhängig der tatsächlichen Instanz. Da die Felder jedoch als Value Object geführt werden, kann dies auch auf das Schachbrett übertragen werden. Das Schachbrett weist keine Verhaltensänderungen auf, sodass die Integrität des Objekts die ganze Zeit über gewahrt werden muss.

Ein **Schachzug** setzt sich aus einer ausgewählten Figur, der Startposition, der Endposition und weiteren Informationen über den Einfluss der Bewegung der betreffenden Figur auf umliegende Figuren zusammen. Dazu zählen Informationen darüber, ob eine andere Figur geschlagen wird, ob durch den Zug Schach geboten wird, ob eine Bauernumwandlung stattfindet oder ob andere Figuren ebenfalls bewegt werden müssen, wie es bei der Rochade der Fall ist. Die Implementierung eines Schachzugs dient der Protokollierung einer gespielten Schachpartie. Ein Schachzug wurde als Value Object klassifiziert, da er keinen erkennbaren Lebenszyklus besitzt und seine Eigenschaften nach der Erstellung nicht mehr verändert werden.

Entities

Eine Entity ist ein Objekt, das innerhalb der Domäne eine einzigartige Identität besitzt und sich im Laufe der Zeit verändern kann. Entities werden durch ihre Identität und nicht durch ihre Attribute definiert und weisen einen Lebenszyklus auf.

Im Domaincode gibt es die Klasse **Figur**. Diese dient als abstrakte Oberklasse für alle Figuren in einem Schachspiel. Da die Klasse Figur als *abstract class* implementiert ist, kann sie nicht instanziiert werden. Daher sind alle Spielfiguren, die von Figur erben, als Entitäten zu betrachten. Zu diesen Spielfiguren gehören **König**, **Dame**, **Turm**, **Läufer**, **Springer** und **Bauer**. Jede Figur kann über das Attribut `position` eindeutig identifiziert werden. Die Position der Figuren fungiert als natürlicher Schlüssel. Auf einem Feld kann immer nur eine Figur stehen, wodurch die Eindeutigkeit gewährleistet ist. Wenn eine Figur auf ein Feld mit einer anderen Figur zieht, wird die dort stehende Figur geschlagen und aus dem aktiven Spiel entfernt. Auch in diesem Fall steht letztendlich nur eine Figur auf dem Feld. Innerhalb der Domäne „Schach“ ist dieser Schlüssel veränderlich, da sich die Schachfiguren auf dem Spielfeld bewegen können. Die Verwendung eines Surrogatschlüssels wäre grundsätzlich auch möglich, aber in diesem Fall wird sich an

der Domäne orientiert. Im analogen Schach wird die Farbe einer Figur, der Figurentyp und die Position zur eindeutigen Identifizierung verwendet. Eine Schachfigur besitzt einen eigenen Lebenszyklus. Sobald ein Schachspiel startet, werden alle Figuren in ihrer Startposition instanziiert. Während des Spiels können sie nahezu unbegrenzt bewegt werden. Wenn eine Figur geschlagen wird, wird sie gelöscht.

Auch die **Spieler** können als Entität betrachtet werden. Ein Spieler kann durch seinen Namen als natürlichen Schlüssel identifiziert werden. Während eines Schachspiels kann ein Spieler in jedem Zug eine Figur bewegen, weiterhin kann er aber auch das Spiel aufgeben. Weiterhin ist der Spieler für das Elosystem nötig und ist veränderbar in seinem Rang.

Obwohl ein Schachzug ein Value Object ist, wird in diesem Projekt der **Spielzug** als Entity gewertet, obwohl nur zwei Schachzüge mit einer ID zusammengefasst werden. Der Spielzug lässt sich eindeutig durch einen `zugNummer` identifizieren, die mit jedem Spielzug so lange um eins hochzählt wie das Schachspiel geht. Durch den inkrementellen Schlüssel liegt die Vermutung nahe, dass es sich dabei um einen Surrogatschlüssel handelt, aber da diese Art der Zählung und Zugidentifizierung ebenfalls in der Domäne verwendet wird, ist die Einordnung als natürlicher Schlüssel passender. Der Grund, warum der Spielzug als Entity und nicht als Value Object gehandelt wird, liegt am Lebenszyklus des Objektes. Auch wenn es kein besonders umfangreicher Lebenszyklus ist, so wird nach dem Erstellen eines Schachzugs zuerst der weiße Zug gesetzt. In einem zweiten Schritt wird der schwarze Zug hinzugefügt, bevor der Schachzug zur weiteren Ausgabe verarbeitet und final wieder gelöscht wird.

Ein **Schachspiel** sollte ebenfalls als Entität behandelt werden. Es besitzt eine eigene Identität und eine eigene Lebenszeit über eine Partie Schach hinweg, in dem es verschiedene Zustände annehmen kann. Das Schachspiel ist eine übergeordnete logische Ebene, welche Spieler, Schachbrett und Figuren zusammenlegt. Auch wenn beim Schach die Ausgangslage der Figuren immer gleich ist, so unterscheidet sich eine Partie Schach in ihrem Verlauf.

Aggregate

- Figur -> Dame, König, Turm, Läufer, Springer, Bauer
- Spielzug -> Schachzug

- Schachbrett -> Feld

Repositories

- Persistierung der gesamten Schachpartie als Logdatei
- Persistierung der Spieler mit Elo für das Elosystem

2.3 Domainenevents

domainenrelevante Ereignisse

- Figur wird umgesetzt
- Rochade wird ausgeführt
- Bauer wird umgewandelt
- Figur wird geschlagen
- König steht im Schach
- König wird Matt gesetzt
- Spieler gibt auf
- Spieler gewinnt das Spiel

3 Clean Architecture

Schicht 4 - Abstraction Code

- Bewegungsmatrizen

Schicht 3 - Domain Code

- Dame
- König
- Turm
- Läufer
- Springer
- Bauer
- Schachbrett
- Feld
- ...

Schicht 2 - Application Code

Schicht 1 - Adapters

Schicht 0 - Plugins