Softwarekonzept des Kartenspiels *Mau-Mau*

## Modul: Komponentenbasierte Entwicklung

## Gruppe 6: Jasmin Quast, Maria Hallmann, Philipp Schalau, Richard Frost

Inhalt

[Modul: Komponentenbasierte Entwicklung 1](#_Toc108287623)

[Gruppe 6: Jasmin Quast, Maria Hallmann, Philipp Schalau, Richard Frost 1](#_Toc108287624)

[2 Komponentenschnitt 5](#_Toc108287625)

[3 Schnittstellenbeschreibung 6](#_Toc108287626)

[3.1 Interface AppController 6](#_Toc108287627)

[***Method Details*** 6](#_Toc108287628)

[**play** 6](#_Toc108287629)

[3.2 Interface ViewService 6](#_Toc108287630)

[3.3 Interface CardService 9](#_Toc108287644)

[3.4 Interface DeckService 9](#_Toc108287649)

[3.5 Interface GameService 11](#_Toc108287655)

[3.6 Interface PlayerService 13](#_Toc108287667)

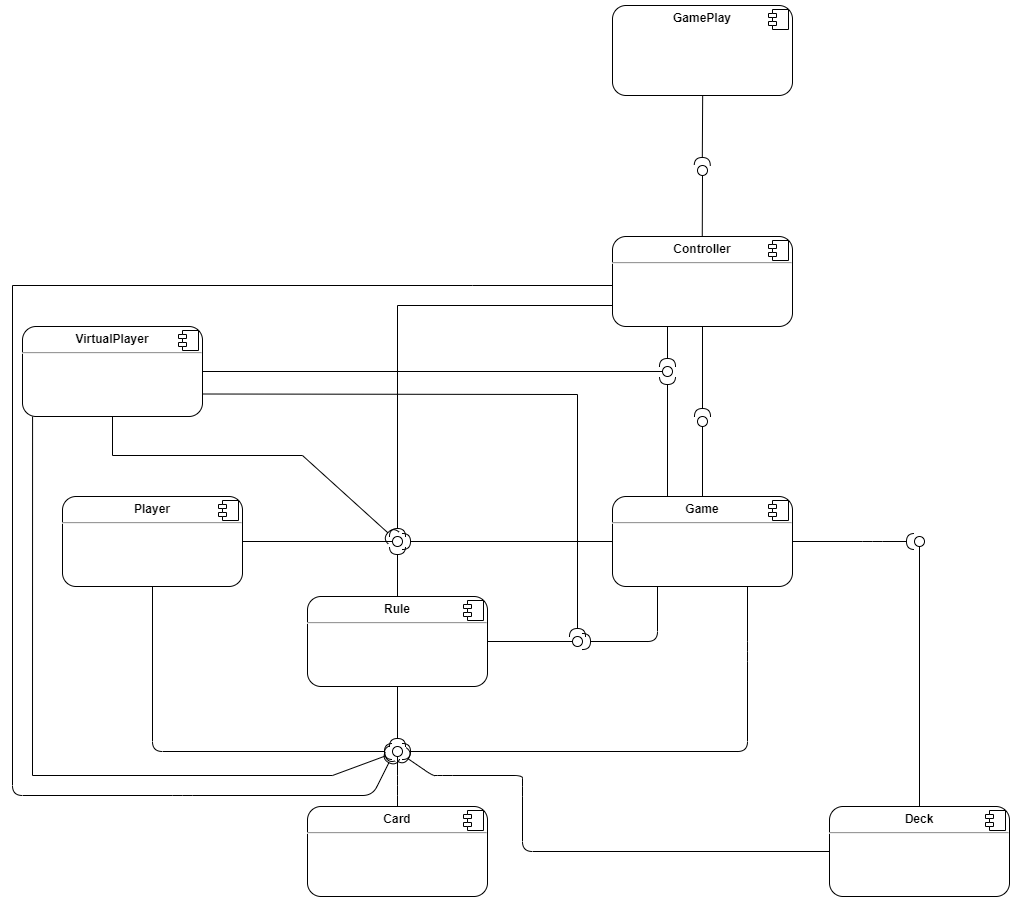
[4 Konzeptionelles Datenmodell 16](#_Toc108287683)

[5 Präsentationsschicht 16](#_Toc108287684)

[6 Frameworks 16](#_Toc108287685)

[7 Ablaufumgebung 20](#_Toc108287686)

# Komponentenschnitt



Unsere Software ist ein Multimodulprojekt. Sie besteht aus insgesamt 8 Modulen:

**Allgemein Schnittstellen in der Kurzbeschreibung nochmal updaten und ergänze, wenn was fehlt**

***Player***

Die Komponente Player erstellt neue Spieler und verwaltet ihre Handkarten. Sie stellt eine Schnittstelle zur Verfügung, die von den Komponenten Rule, Game und Controller benutzt werden.

**Virtueller Spieler updaten!**

***Card***

Die Komponente Card erstellt einzelne Karten (bestehend aus zwei Enums) und Kartenlisten. Ihre Schnittstelle wird von den Komponenten Deck, Game, Player, Rule und Controller genutzt.

***Deck***

Die Komponente Deck verwaltet die Kartenliste und stellt somit das Kartendeck des Spiels zur Verfügung, die sie von der Komponente Card erhalten hat. Es werden Karten herausgegeben und hinzugefügt. Die Komponente Game nutzt die Schnittstelle von Deck.

***Rule***

Die Komponente Rule prüft die Anwendung von Spielregeln und ob gespielte Karten zulässig sein. Ihre Schnittstelle wird ebenfalls von der Komponente Game benutzt.

***Game***

**Persistence muss noch beschrieben werden!**

Die Komponente Game vereint Spieler und das Kartendeck zu einem neuem Gameobjekt. Zusätzlich werden gespielte oder zuziehende Karten weitergereicht an Spieler und Deck. Außerdem behandelt sie die Ergebnisse aus der Komponente Rule und kümmert sich um die Konsequenzen für anzuwendende Spielregeln für den Spieler oder des Spiels generell.

Die Komponente Controller und greift auf diese Schnittstelle zu.

***Controller***

Der Controller besteht aus dem Appcontroller und der View. Die View übernimmt alle sichtbaren Ausgaben des Spiels an den Spieler und nimmt ebenfalls die Eingaben des Spielers entgegen. Der Appcontroller übernimmt den gesamten Spielablauf und geht mit der Auswahl, die der Spieler der View übermittelt hat, entsprechend um. Rundenbasiert wird gespielt bis der ein Spieler keine Karten mehr auf der Hand besitzt.

Hinzukommend können Spiele gespeichert und geladen werden.

Die Schnittstelle der Controller Komponente wird von GamePlay verwendet.

***GamePlay***

Diese Komponente beinhalt die Mainmethode und startet damit das gesamte Spiel.

# Schnittstellenbeschreibung

Welche Schnittstellen bestehen zwischen den Komponenten? Was leisten diese Schnittstellen? (JavaDoc)

Die unten aufgeführten Tabellen zeigen alle Methoden, die zur jeweiligen Schnittstelle gehören und was sie leisten.

**Virtueller Spieler und RuleService müssen dringend eingefügt/ergänzt werden.**

**Wie will Kempa das Interface dargestellt haben? E-Mail?**

## Interface AppController

|  |
| --- |
| ***Method Details***  **play**  void play()  runs the entire game |

## Interface ViewService

|  |
| --- |
| *Method Details*  **getNumberOfPlayer**  int getNumberOfPlayer()  shows gaming instructions and asks the player for the desired number of players  **Returns:**  number of players  **getPlayerNames**  [List](https://docs.oracle.com/en/java/javase/17/docs/api/java.base/java/util/List.html)<[String](https://docs.oracle.com/en/java/javase/17/docs/api/java.base/java/lang/String.html)> getPlayerNames(int numberOfPlayer)  asks for player names depending on the number of players  **Parameters:**  numberOfPlayer - number of players in the game  **Returns:**  list of player names  **showStartGameMessage**  void showStartGameMessage()  lets the player know that the game has started  **showTopCard**  void showTopCard([Card](file:///C:\Users\MariaH\Desktop\javaDoc\htw\kbe\maumau\card\export\Card.html) topCard)  lets the player know which card is on top of the pile  **Parameters:**  topCard - card that is in top of the discard pile  **showHandCards**   * + void showHandCards([Player](file:///C:\Users\MariaH\Desktop\javaDoc\htw\kbe\maumau\player\export\Player.html) player, [Suit](file:///C:\Users\MariaH\Desktop\javaDoc\htw\kbe\maumau\card\export\Suit.html) suit)   shows the player his hand cards and if there is a suit wish, it is also shown  **Parameters:**  player - player who is in turn  suit - suit that was requested (optional)  **getPlayedCard**  [Card](file:///C:\Users\MariaH\Desktop\javaDoc\htw\kbe\maumau\card\export\Card.html) getPlayedCard([Player](file:///C:\Users\MariaH\Desktop\javaDoc\htw\kbe\maumau\player\export\Player.html) player)  asks the player which card he wants to discard or if he wants to draw card(s)  **Parameters:**  player - player who is in turn  **Returns:**  card - chosen card that the player wants to play  **getChosenSuit**   * + [Suit](file:///C:\Users\MariaH\Desktop\javaDoc\htw\kbe\maumau\card\export\Suit.html) getChosenSuit([Player](file:///C:\Users\MariaH\Desktop\javaDoc\htw\kbe\maumau\player\export\Player.html) player, [List](https://docs.oracle.com/en/java/javase/17/docs/api/java.base/java/util/List.html)<[Suit](file:///C:\Users\MariaH\Desktop\javaDoc\htw\kbe\maumau\card\export\Suit.html)> suits)   asks the player for the suit wish  **Parameters:**  player - player who is in turn  suits - list of suits to choose from  **Returns:**  desired suit wish  **saidMau**  boolean saidMau([Player](file:///C:\Users\MariaH\Desktop\javaDoc\htw\kbe\maumau\player\export\Player.html) player)  asks the player if he wants say 'mau'  **Parameters:**  player - player who is in turn  **Returns:**  true if player said 'mau', false if not  **showDrawnCardMessage**   * + void showDrawnCardMessage([Player](file:///C:\Users\MariaH\Desktop\javaDoc\htw\kbe\maumau\player\export\Player.html) player, int numberOfDrawnCards)   lets the player know how many cards he has drawn  **Parameters:**  player - player who is in turn  numberOfDrawnCards - number of cards to be drawn  **showErrorMessage**  void showErrorMessage([String](https://docs.oracle.com/en/java/javase/17/docs/api/java.base/java/lang/String.html) exceptionMessage)  lets the player know want went wrong in the game  **Parameters:**  exceptionMessage - message that was thrown by exception  **showWinnerMessage**  void showWinnerMessage([Player](file:///C:\Users\MariaH\Desktop\javaDoc\htw\kbe\maumau\player\export\Player.html) player)  lets players know who has won  **Parameters:**  player - player who won  **hasNextRound**  boolean hasNextRound()  aks for next round to play  **Returns:**  true if player wants to play again, false if game should quit |

## Interface CardService

|  |
| --- |
| ***Method Details***  **getSuits**  [List](https://docs.oracle.com/en/java/javase/17/docs/api/java.base/java/util/List.html)<[Suit](file:///C:\Users\MariaH\Desktop\javaDoc\htw\kbe\maumau\card\export\Suit.html)> getSuits()  all available Suits are put into a list  **Returns:**  List of Suits  **getLabels**  [List](https://docs.oracle.com/en/java/javase/17/docs/api/java.base/java/util/List.html)<[Label](file:///C:\Users\MariaH\Desktop\javaDoc\htw\kbe\maumau\card\export\Label.html)> getLabels()  all available Labels are put into a list  **Returns:**  List of Labels  **getCards**  [List](https://docs.oracle.com/en/java/javase/17/docs/api/java.base/java/util/List.html)<[Card](file:///C:\Users\MariaH\Desktop\javaDoc\htw\kbe\maumau\card\export\Card.html)> getCards()  32 cards are initialized. The card stack consists of 4 suits with 8 labels each.  **Returns:**  List of Cards |

## Interface DeckService

|  |
| --- |
| ***Method Details***  **createDeck**  [Deck](file:///C:\Users\MariaH\Desktop\javaDoc\htw\kbe\maumau\deck\export\Deck.html) createDeck([List](https://docs.oracle.com/en/java/javase/17/docs/api/java.base/java/util/List.html)<[Card](file:///C:\Users\MariaH\Desktop\javaDoc\htw\kbe\maumau\card\export\Card.html)> cards) throws [IllegalDeckSizeException](file:///C:\Users\MariaH\Desktop\javaDoc\htw\kbe\maumau\deck\exceptions\IllegalDeckSizeException.html)  initialise card deck  **Parameters:**  cards - Card stack that has to add to deck  **Returns:**  deck that include the required card stack  **Throws:**  [IllegalDeckSizeException](file:///C:\Users\MariaH\Desktop\javaDoc\htw\kbe\maumau\deck\exceptions\IllegalDeckSizeException.html) - when card stack is empty or has an invalid Suit Label ratio  **initialCardDealing**  [List](https://docs.oracle.com/en/java/javase/17/docs/api/java.base/java/util/List.html)<[Card](file:///C:\Users\MariaH\Desktop\javaDoc\htw\kbe\maumau\card\export\Card.html)> initialCardDealing([Deck](file:///C:\Users\MariaH\Desktop\javaDoc\htw\kbe\maumau\deck\export\Deck.html) deck)  at the beginning of a game: depending on the initial number of drawn cards, cards are dealt from the draw pile  **Parameters:**  deck - card deck  **Returns:**  initial cards for player  **getCardsFromDrawPile**  [List](https://docs.oracle.com/en/java/javase/17/docs/api/java.base/java/util/List.html)<[Card](file:///C:\Users\MariaH\Desktop\javaDoc\htw\kbe\maumau\card\export\Card.html)> getCardsFromDrawPile([Deck](file:///C:\Users\MariaH\Desktop\javaDoc\htw\kbe\maumau\deck\export\Deck.html) deck, int numberOfDrawCards)  depending on the number of cards to be drawn, the cards are dealt from the draw pile  **Parameters:**  deck - card deck  numberOfDrawCards - number of cards to be drawn  **Returns:**  card list of drawn cards  **setCardToTopCard**  [Card](file:///C:\Users\MariaH\Desktop\javaDoc\htw\kbe\maumau\card\export\Card.html) setCardToTopCard([Deck](file:///C:\Users\MariaH\Desktop\javaDoc\htw\kbe\maumau\deck\export\Deck.html) deck, [Card](file:///C:\Users\MariaH\Desktop\javaDoc\htw\kbe\maumau\card\export\Card.html) discardedCard)  discarded card is set to the new top card and previous top card is set to discard pile  **Parameters:**  deck - card deck  discardedCard - card that has to set to the new top card  **Returns:**  new top card |

## Interface GameService

|  |
| --- |
| ***Method Details***  **createGame**  [Game](file:///C:\Users\MariaH\Desktop\javaDoc\htw\kbe\maumau\game\export\Game.html) createGame([List](https://docs.oracle.com/en/java/javase/17/docs/api/java.base/java/util/List.html)<[Player](file:///C:\Users\MariaH\Desktop\javaDoc\htw\kbe\maumau\player\export\Player.html)> players) throws [IllegalDeckSizeException](file:///C:\Users\MariaH\Desktop\javaDoc\htw\kbe\maumau\deck\exceptions\IllegalDeckSizeException.html), [InvalidPlayerSizeException](file:///C:\Users\MariaH\Desktop\javaDoc\htw\kbe\maumau\game\exceptions\InvalidPlayerSizeException.html)  initializes a game with the desired number of players and the needed card deck  **Parameters:**  players - list of players participating in the game  **Returns:**  new game  **Throws:**  [IllegalDeckSizeException](file:///C:\Users\MariaH\Desktop\javaDoc\htw\kbe\maumau\deck\exceptions\IllegalDeckSizeException.html) - when deck has not the right size of cards  [InvalidPlayerSizeException](file:///C:\Users\MariaH\Desktop\javaDoc\htw\kbe\maumau\game\exceptions\InvalidPlayerSizeException.html) - when player list size is above four or below two  **switchToNextPlayer**  void switchToNextPlayer([Game](file:///C:\Users\MariaH\Desktop\javaDoc\htw\kbe\maumau\game\export\Game.html) game)  sets the player for the next round, noting whether the round is played clockwise or counterclockwise.  **Parameters:**  game -  **initialCardDealing**  void initialCardDealing([Game](file:///C:\Users\MariaH\Desktop\javaDoc\htw\kbe\maumau\game\export\Game.html) game)  at the start of the game, each player is dealt their hand cards  **Parameters:**  game -  **giveDrawnCardsToPlayer**  void giveDrawnCardsToPlayer(int numberOfDrawnCards, [Game](file:///C:\Users\MariaH\Desktop\javaDoc\htw\kbe\maumau\game\export\Game.html) game)  as many cards are drawn from the draw pile as are indicated and then the cards are dealt to the player  **Parameters:**  numberOfDrawnCards - number of cards the player must draw  game -  **mustPlayerDrawCards**  boolean mustPlayerDrawCards([Game](file:///C:\Users\MariaH\Desktop\javaDoc\htw\kbe\maumau\game\export\Game.html) game)  Checks whether the player may play a card in this round. He may not if he has to draw cards (e.g. because a seven is on top and the player has no seven in his hand)  **Parameters:**  game -  **Returns:**  true when player can play a card, false when player has to draw cards instead  **applyCardRule**  void applyCardRule([Game](file:///C:\Users\MariaH\Desktop\javaDoc\htw\kbe\maumau\game\export\Game.html) game)  checks if the played card match a card rule belonging to the card rule further methods will be executed  **Parameters:**  game -  **validateCard**  void validateCard([Card](file:///C:\Users\MariaH\Desktop\javaDoc\htw\kbe\maumau\card\export\Card.html) card, [Game](file:///C:\Users\MariaH\Desktop\javaDoc\htw\kbe\maumau\game\export\Game.html) game) throws [PlayedCardIsInvalidException](file:///C:\Users\MariaH\Desktop\javaDoc\htw\kbe\maumau\rule\exceptions\PlayedCardIsInvalidException.html)  validates the card to be played if the card is valid, it is added to the discard pile if a suit wish was played and valid then removes suit wish from game  **Parameters:**  card - card that wants to be played  game -  **Throws:**  [PlayedCardIsInvalidException](file:///C:\Users\MariaH\Desktop\javaDoc\htw\kbe\maumau\rule\exceptions\PlayedCardIsInvalidException.html) - when the played card is not valid for that round  **setPlayersSuitWish**  void setPlayersSuitWish([Suit](file:///C:\Users\MariaH\Desktop\javaDoc\htw\kbe\maumau\card\export\Suit.html) userWish, [Game](file:///C:\Users\MariaH\Desktop\javaDoc\htw\kbe\maumau\game\export\Game.html) game)  set a suit wish of a player into the game and set askForSuitWish state to false  **Parameters:**  userWish - suit wish of the player  game -  **isGameOver**  boolean isGameOver([Game](file:///C:\Users\MariaH\Desktop\javaDoc\htw\kbe\maumau\game\export\Game.html) game)  calls the game as won when the active player has no more hand cards  **Parameters:**  game -  **Returns:**  true, if active player has no hand cards, false if not  **resetPlayersMau**  void resetPlayersMau([Game](file:///C:\Users\MariaH\Desktop\javaDoc\htw\kbe\maumau\game\export\Game.html) game)  set 'mau' state to false of active player  **Parameters:**  game - |

## Interface PlayerService

|  |
| --- |
| ***Method Details***  **createPlayers**  [List](https://docs.oracle.com/en/java/javase/17/docs/api/java.base/java/util/List.html)<[Player](file:///C:\Users\MariaH\Desktop\javaDoc\htw\kbe\maumau\player\export\Player.html)> createPlayers([List](https://docs.oracle.com/en/java/javase/17/docs/api/java.base/java/util/List.html)<[String](https://docs.oracle.com/en/java/javase/17/docs/api/java.base/java/lang/String.html)> names) throws [InvalidPlayerNameException](file:///C:\Users\MariaH\Desktop\javaDoc\htw\kbe\maumau\player\exceptions\InvalidPlayerNameException.html)  initializes new players  **Parameters:**  names - list of player names  **Returns:**  list of players  **Throws:**  [InvalidPlayerNameException](file:///C:\Users\MariaH\Desktop\javaDoc\htw\kbe\maumau\player\exceptions\InvalidPlayerNameException.html) - when the name is empty, blanks only or longer then 15 symbols  **addDrawnCards**  void addDrawnCards([Player](file:///C:\Users\MariaH\Desktop\javaDoc\htw\kbe\maumau\player\export\Player.html) player, [List](https://docs.oracle.com/en/java/javase/17/docs/api/java.base/java/util/List.html)<[Card](file:///C:\Users\MariaH\Desktop\javaDoc\htw\kbe\maumau\card\export\Card.html)> card)  adds drawn card(s) to hand cards to given player and hand cards are sorted by suit (from black to red) and label (from ASS to SEVEN).  **Parameters:**  player - who gets drawn cards  card - card that was drawn  **removePlayedCard**  void removePlayedCard([Player](file:///C:\Users\MariaH\Desktop\javaDoc\htw\kbe\maumau\player\export\Player.html) player, [Card](file:///C:\Users\MariaH\Desktop\javaDoc\htw\kbe\maumau\card\export\Card.html) card)  removes card that was played from hand card of player  **Parameters:**  player - player who played the given card  card - played card to be removed  **validateName**  void validateName([String](https://docs.oracle.com/en/java/javase/17/docs/api/java.base/java/lang/String.html) name, [List](https://docs.oracle.com/en/java/javase/17/docs/api/java.base/java/util/List.html)<[String](https://docs.oracle.com/en/java/javase/17/docs/api/java.base/java/lang/String.html)> names) throws [InvalidPlayerNameException](file:///C:\Users\MariaH\Desktop\javaDoc\htw\kbe\maumau\player\exceptions\InvalidPlayerNameException.html)  validates name  **Parameters:**  name - that has to be validated  names - list of names that is needed to check if names duplicate  **Throws:**  [InvalidPlayerNameException](file:///C:\Users\MariaH\Desktop\javaDoc\htw\kbe\maumau\player\exceptions\InvalidPlayerNameException.html) - when String name is empty, has whitespaces, is too long or names duplicate |

## Interface RuleService

|  |
| --- |
| ***Method Details***  **validateCard**  void validateCard([Card](file:///C:\Users\MariaH\Desktop\javaDoc\htw\kbe\maumau\card\export\Card.html) playedCard,[Card](file:///C:\Users\MariaH\Desktop\javaDoc\htw\kbe\maumau\card\export\Card.html) topCard,[Suit](file:///C:\Users\MariaH\Desktop\javaDoc\htw\kbe\maumau\card\export\Suit.html) userWish, int drawCounter) throws [PlayedCardIsInvalidException](file:///C:\Users\MariaH\Desktop\javaDoc\htw\kbe\maumau\rule\exceptions\PlayedCardIsInvalidException.html)  validates if a card can be played, also if card was JACK, SEVEN and if there is a suit wish  **Parameters:**  playedCard - card user wants to play  topCard - card on top of discard pile  userWish - suit wish after JACK was played  drawCounter - number of drawn cards  **Throws:**  [PlayedCardIsInvalidException](file:///C:\Users\MariaH\Desktop\javaDoc\htw\kbe\maumau\rule\exceptions\PlayedCardIsInvalidException.html) - throw exception when card cannot be played  **mustDrawCards**  boolean mustDrawCards([Card](file:///C:\Users\MariaH\Desktop\javaDoc\htw\kbe\maumau\card\export\Card.html) topCard)  Checks if a player needs to draw cards  **Parameters:**  topCard - card on top of discard pile  **Returns:**  true if card has label SEVEN, false when not  **getDefaultNumberOfDrawnCards**  int getDefaultNumberOfDrawnCards()  gets default number of drawn cards  **Returns:**  number of cards to be drawn  **mustSuspend**  boolean mustSuspend([Card](file:///C:\Users\MariaH\Desktop\javaDoc\htw\kbe\maumau\card\export\Card.html) topCard)  checks if next player is suspended for one round  **Parameters:**  topCard - card on top of discard pile  **Returns:**  true if card is ASS, false if not  **mustDrawCards**  boolean mustDrawCards([Player](file:///C:\Users\MariaH\Desktop\javaDoc\htw\kbe\maumau\player\export\Player.html) player, [Card](file:///C:\Users\MariaH\Desktop\javaDoc\htw\kbe\maumau\card\export\Card.html) topCard, int drawCounter)  checks if at least one hand card of the player is a SEVEN and top card is a SEVEN and draw counter is greater or equal default number of drawn cards, decides if a player has to draw cards  **Parameters:**  player - player who is in turn  topCard - card on top of discard pile  drawCounter - number of drawn cards  **Returns:**  true if player has to draw cards, false if player has not to draw  **isCardJack**  boolean isCardJack([Card](file:///C:\Users\MariaH\Desktop\javaDoc\htw\kbe\maumau\card\export\Card.html) topCard)  Checks if last card played was a JACK and player can make a wish  **Parameters:**  topCard - card on top of discard pile  **Returns:**  true if card was JACK, false if not  **changeGameDirection**  boolean changeGameDirection([Card](file:///C:\Users\MariaH\Desktop\javaDoc\htw\kbe\maumau\card\export\Card.html) topCard)  Checks if card was NINE and game direction has to change  **Parameters:**  topCard - card on top of discard pile  **Returns:**  true if card was NINE, false if not  **isPlayersMauInvalid**  boolean isPlayersMauInvalid([Player](file:///C:\Users\MariaH\Desktop\javaDoc\htw\kbe\maumau\player\export\Player.html) player)  Checks if player said 'mau' and if it's valid  **Parameters:**  player - player in turn  **Returns:**  false, if players 'mau' is valid, true if it's invalid |

# Konzeptionelles Datenmodell

Welches Datenmodell wählen Sie? Wie wird das konzeptionell Modell auf das physische abgebildet? Wird denormalisiert? Welche Komponente ist für welchen Teil des Datenmodells zuständig? (Datenhohheit)

**Datenbankmodell einfügen**

# Präsentationsschicht

Wie ist die Architektur Ihrer GUI? Beschreiben Sie Ihre Dialoge! (Skizzen, Screenshots) Welche Services (aus den Schnittstellen) werden aufgerufen?

Das Spiel ist eine interaktive Konsolenanwendung. Demzufolge werden die Benutzereingaben über die Konsole eingetragen und die Ausgaben des Spiels werden ebenfalls darüber angezeigt.

**Screenshots mit Dialogen einfügen und kurz beschreiben. Geht erst, wenn die View auch wirklich fertig ist, sonst hat man doppelt Arbeit**

# Frameworks

Welche Frameworks und/oder Produkte kommen zum Einsatz? (Technische Architektur) Wie werden diese konfiguriert?

**Text sollte so pasen, aber vllt muss nochmal die pom hier geupdatet werden**

Die Software basiert auf der Programmiersprache Java in der Version 17 und wird von Apache Maven, einem Build Management System, verwaltet und gebaut. Das Projekt ist als ein Multimodulprojekt konzipiert.

Um die Erstellung und Durchführung von Tests zu erleichtern, wurden die Frameworks JUnit 5 und Mockito in Maven eingebunden. Letzteres wird speziell für das Mocken von Objekten verwendetet.

Um mit der Oracle Datenbank zu kommunizieren wird auf den Hibernate EntityManager und der Oracle Java Database Connectivity (ojdbc) zurückgegriffen. Die EntityManager-Api wird für den Zugriff auf die Datenbank verwendetet, d.h. sie kann persistente Entitätsinstanzen erstellen und entfernen, aber auch Entitäten anhand ihrer Primärschlüsselidentität finden und Abfragen über alle Entitäten durchführen. Zweiteres ist eine Java-API, mit dem sich SQL-Anweisungen in Datenbanken ausführen lassen.

Der Spielablauf wird mittels Log4j protokolliert und die Dependency Injection wird mit Hilfe von Spring Context realisiert.

Im Parent pom sind alle Konfigurationen eingetragen, die den Kind-Komponenten vererbt werden:

|  |
| --- |
| <?xml version="1.0" encoding="UTF-8"?> <project xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns="http://maven.apache.org/POM/4.0.0"  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">   <modelVersion>4.0.0</modelVersion>  <groupId>htw.kbe</groupId>  <artifactId>maumau</artifactId>  <version>0.0.1-SNAPSHOT</version>   <packaging>pom</packaging>  <name>mau-mau</name>  <description>KBE Mau-Mau Project</description>   <modules>  <module>game</module>  <module>card</module>  <module>deck</module>  <module>rule</module>  <module>player</module>  <module>JaCoCo</module>  <module>controller</module>  <module>gamePlay</module>  </modules>   <dependencies>  <dependency>  <groupId>org.springframework</groupId>  <artifactId>spring-context</artifactId>  <version>5.3.20</version>  </dependency>  <dependency>  <groupId>junit</groupId>  <artifactId>junit</artifactId>  <version>4.13.2</version>  <scope>test</scope>  </dependency>  <dependency>  <groupId>org.junit.jupiter</groupId>  <artifactId>junit-jupiter</artifactId>  <version>5.8.2</version>  <scope>test</scope>  </dependency>  <dependency>  <groupId>org.mockito</groupId>  <artifactId>mockito-junit-jupiter</artifactId>  <version>4.5.1</version>  <scope>test</scope>  </dependency>  <dependency>  <groupId>org.apache.logging.log4j</groupId>  <artifactId>log4j-api</artifactId>  <version>2.17.2</version>  </dependency>  <dependency>  <groupId>org.apache.logging.log4j</groupId>  <artifactId>log4j-core</artifactId>  <version>2.17.2</version>  </dependency>  <dependency>  <groupId>org.hibernate</groupId>  <artifactId>hibernate-entitymanager</artifactId>  <version>5.6.1.Final</version>  </dependency>  <dependency>  <groupId>com.oracle.ojdbc</groupId>  <artifactId>ojdbc8</artifactId>  <version>19.3.0.0</version>  </dependency>  </dependencies>   <build>  <plugins>  <plugin>  <groupId>org.apache.maven.plugins</groupId>  <artifactId>maven-compiler-plugin</artifactId>  <version>3.10.1</version>  <configuration>  <release>17</release>  </configuration>  </plugin>  <plugin>  <artifactId>maven-surefire-plugin</artifactId>  <version>2.22.2</version>  </plugin>  </plugins>  </build>   <properties>  <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>  <project.reporting.outputEncoding>UTF-8</project.reporting.outputEncoding>  </properties> </project> |

Um die Testabdeckung des gesamten Projektes zu überblicken wurde JaCoCo (als separated Modul) eingeführt und zeigt folgende Konfigurationen in der pom:

|  |
| --- |
| <?xml version="1.0" encoding="UTF-8"?> <project xmlns="http://maven.apache.org/POM/4.0.0"  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">  <parent>  <artifactId>maumau</artifactId>  <groupId>htw.kbe</groupId>  <version>0.0.1-SNAPSHOT</version>  </parent>   <modelVersion>4.0.0</modelVersion>   <artifactId>JaCoCo</artifactId>  <version>0.0.1-SNAPSHOT</version>  <name>JaCoCo</name>   <description>Compute aggregated test code coverage</description>   <properties>  <maven.deploy.skip>true</maven.deploy.skip>  </properties>   <dependencies>  <dependency>  <groupId>htw.kbe</groupId>  <artifactId>deck</artifactId>  <version>0.0.1-SNAPSHOT</version>  <scope>compile</scope>  </dependency>  <dependency>  <groupId>htw.kbe</groupId>  <artifactId>player</artifactId>  <version>0.0.1-SNAPSHOT</version>  <scope>compile</scope>  </dependency>  <dependency>  <groupId>htw.kbe</groupId>  <artifactId>card</artifactId>  <version>0.0.1-SNAPSHOT</version>  <scope>compile</scope>  </dependency>  <dependency>  <groupId>htw.kbe</groupId>  <artifactId>rule</artifactId>  <version>0.0.1-SNAPSHOT</version>  <scope>compile</scope>  </dependency>  <dependency>  <groupId>htw.kbe</groupId>  <artifactId>game</artifactId>  <version>0.0.1-SNAPSHOT</version>  <scope>compile</scope>  </dependency>  <dependency>  <groupId>htw.kbe</groupId>  <artifactId>controller</artifactId>  <version>0.0.1-SNAPSHOT</version>  <scope>compile</scope>  </dependency>  </dependencies>   <build>  <plugins>  <plugin>  <groupId>org.jacoco</groupId>  <artifactId>jacoco-maven-plugin</artifactId>  <version>0.8.8</version>  <executions>  <execution>  <id>report-aggregate</id>  <phase>verify</phase>  <goals>  <goal>report-aggregate</goal>  </goals>  </execution>  </executions>  </plugin>  </plugins>  </build> </project> |

Die Konfigurationsdatei (log4j2.xml) wurde folgender Maßen für das Protokollieren des Spielverlaufs aufgesetzt:

|  |
| --- |
| <?xml version="1.0" encoding="UTF-8"?>  <Configuration status="WARN">  <Appenders>  <File name="fout" fileName="logs/app.log" append="true">  <PatternLayout>  <Pattern>%d{yyyy-MM-dd HH:mm:ss} [%c{1}]: %-5p %m%n</Pattern>  </PatternLayout>  </File>  </Appenders>  <Loggers>  <Root level="INFO">  <AppenderRef ref="fout"/>  </Root>  </Loggers> </Configuration> |

Die Konfiguration der persistence.xml sehen wie folgt aus:

|  |
| --- |
| <?xml version="1.0" encoding="UTF-8"?> <persistence version="2.1"  xmlns="http://xmlns.jcp.org/xml/ns/persistence" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/persistence http://xmlns.jcp.org/xml/ns/persistence/persistence\_2\_1.xsd">   <persistence-unit name="MauMau" transaction-type="RESOURCE\_LOCAL">  <provider>org.hibernate.jpa.HibernatePersistenceProvider</provider>  <class>htw.kbe.maumau.game.export.Game</class>  <properties>  <property name="javax.persistence.jdbc.driver" value="oracle.jdbc.OracleDriver" />  <property name="javax.persistence.jdbc.url" value="jdbc:oracle:thin:@oradbs03.f4.htw-berlin.de:1521:oradb1" />  <property name="javax.persistence.jdbc.user" value="u572897" />  <property name="javax.persistence.jdbc.password" value="p572897" />  <property name="hibernate.dialect"  value="org.hibernate.dialect.SQLServer2012Dialect" />  <property name="hibernate.show\_sql" value="true" />  <property name="hibernate.hbm2ddl.auto" value="update" />  </properties>  </persistence-unit> </persistence> |

# Ablaufumgebung

Auf welchen Rechnern in welcher Umgebung wird Ihr System eingesetzt? (Architektur der technischen Infrastruktur)

**Hier kann bestimmt noch was ergänzt werden**

Das Spiel kann auf jedem Rechner laufen, der eine Internetanbindung hat. Damit kann das Spiel gespeichert oder geladen werden. Betriebssysteme wie Windows, macOS und Linux werden unterstützt.

Zusätzlich muss auf dem Rechner Java 17 installiert sein sowie eine Entwicklungsumgebung wie beispielsweise IntelliJ IDEA oder Visual Studio Code.