Java CardGameApp

MSC DDB - Team International
D2 - Objektorientierte Programmierung

16.12.2022

Team

Eric Langer > Dokumentation
Felix Ossmann > Testing
Hannes Brottrager > Logging/Data
Markus Hilbert > Lead/Architekt

Card Game App

Ziele

1. *Prozeduren*
2. *OOP-Konzepte*
3. *Java kennenlernen*
4. *Frameworks nutzen*

Nicht-Ziele

1. *GUI-Development*
2. *Spiele KI*
3. *Design Patterns*
4. *Produktive App*

Setup

1. Docker
2. VS-Code
3. Dev-Container
4. GIT

Card Game App

Architektur

1. Vom Prototypen
2. durch Sackgassen
3. zum Monolithen
4. zur Abstraktion

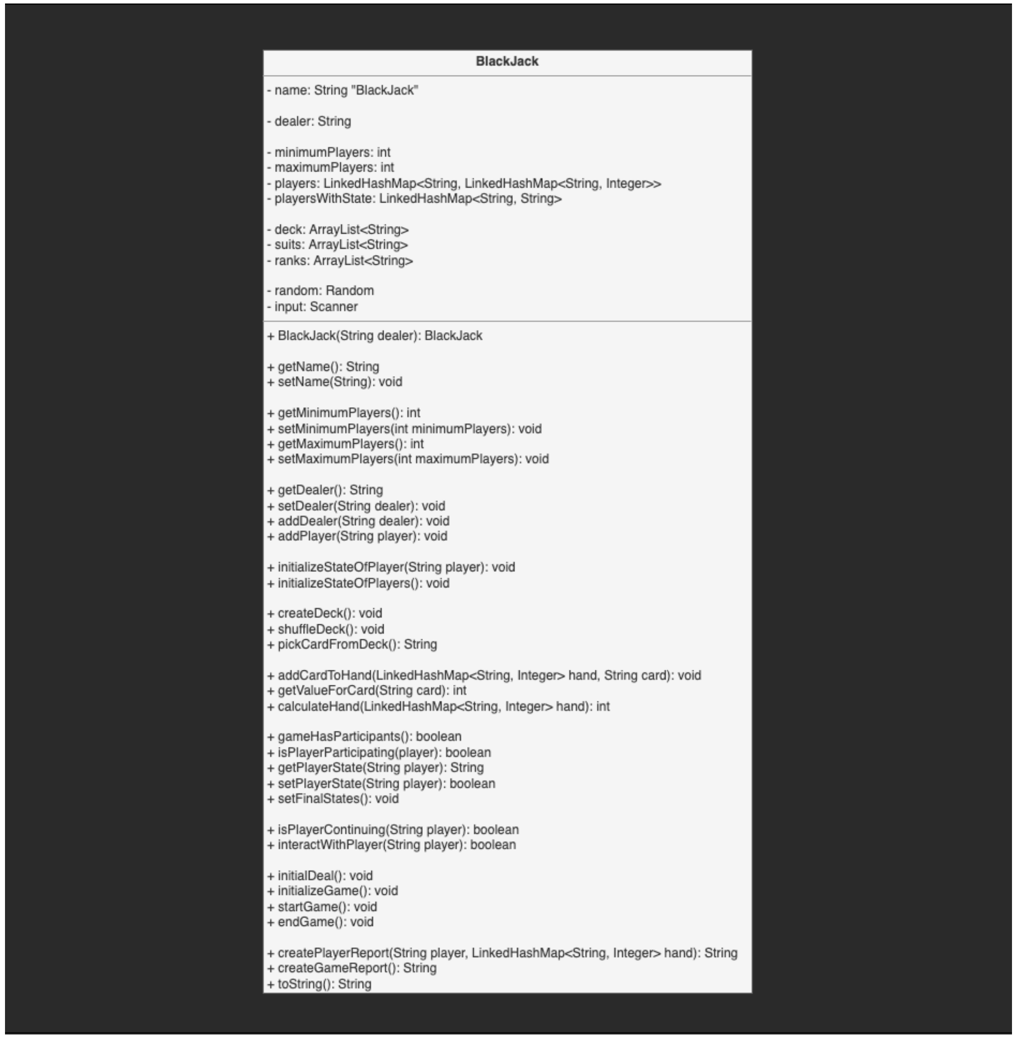
Card Game App

Prototyp & Sackgassen

```
> git checkout v1.0
> git checkout v2.0
> git checkout enum-xp
> git checkout another-xp
> git checkout extensive-ideas
```

Card Game App

Schrittweise zum Ziel

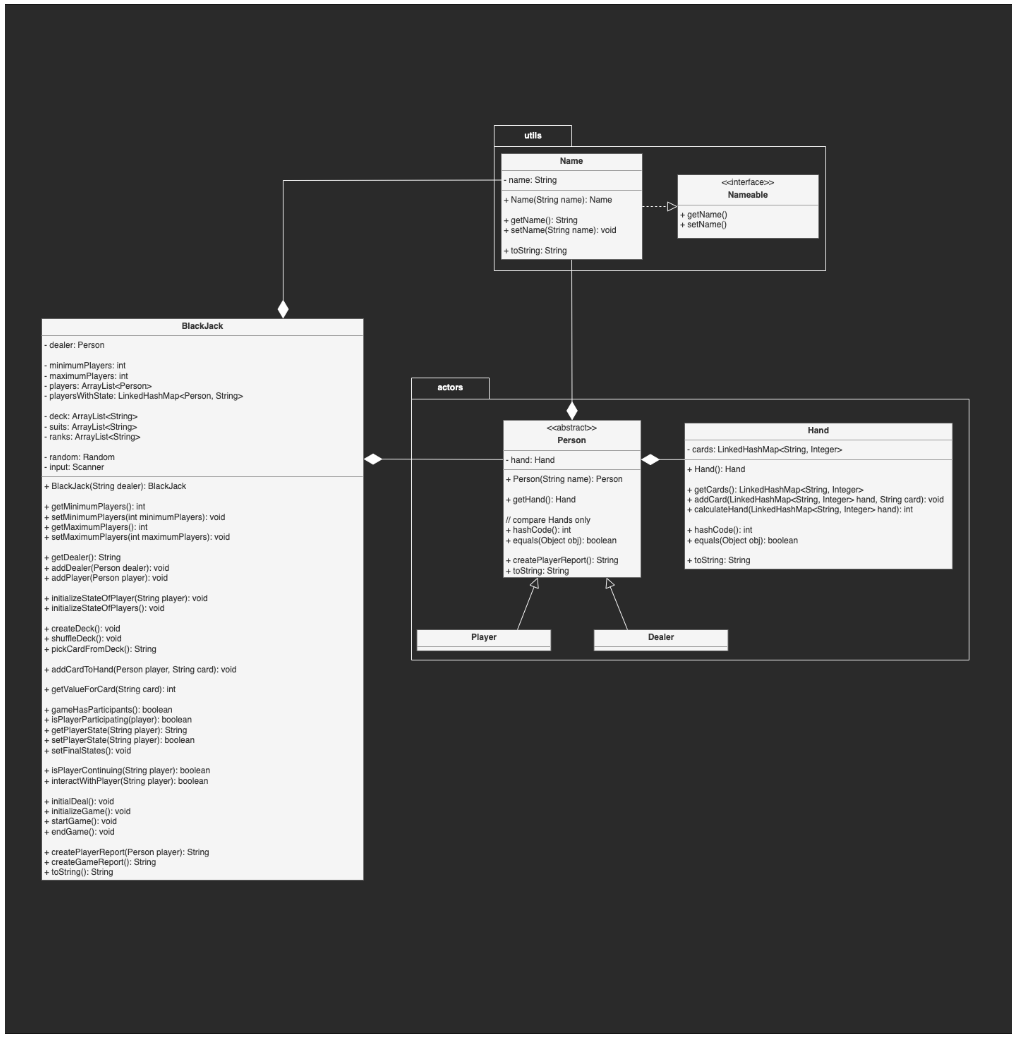


```
BlackJack

- name: String "BlackJack"

- dealer: String

- minimumPlayers: int
- maximumPlayers: int
- players: LinkedHashMap<String, LinkedHashMap<String, Integer>>
- playersWithState: LinkedHashMap<String, String>

- deck: ArrayList<String>
- suits: ArrayList<String>
- ranks: ArrayList<String>

- random: Random
- input: Scanner

+ BlackJack(String dealer): BlackJack

+ getName(): String
+ setName(String): void

+ getMinimumPlayers(): int
+ setMinimumPlayers(int minimumPlayers): void
+ getMaximumPlayers(): int
+ setMaximumPlayers(int maximumPlayers): void

+ getDealer(): String
+ setDealer(String dealer): void
+ addDealer(String dealer): void
+ addPlayer(String player): void

+ initializeStateOfPlayer(String player): void
+ initializeStateOfPlayers(): void

+ createDeck(): void
+ shuffleDeck(): void
+ pickCardFromDeck(): String

+ addCardToHand(LinkedHashMap<String, Integer> hand, String card): void
+ getValueForCard(String card): int
+ calculateHand(LinkedHashMap<String, Integer> hand): int

+ gameHasParticipants(): boolean
+ isPlayerParticipating(player): boolean
+ getPlayerState(String player): String
+ setPlayerState(String player): boolean
+ setFinalStates(): void

+ isPlayerContinuing(String player): boolean
+ interactWithPlayer(String player): boolean

+ initialDeal(): void
+ initializeGame(): void
+ startGame(): void
+ endGame(): void

+ createPlayerReport(String player, LinkedHashMap<String, Integer> hand): String
+ createGameReport(): String
+ toString(): String
```

```
> git checkout step1
```

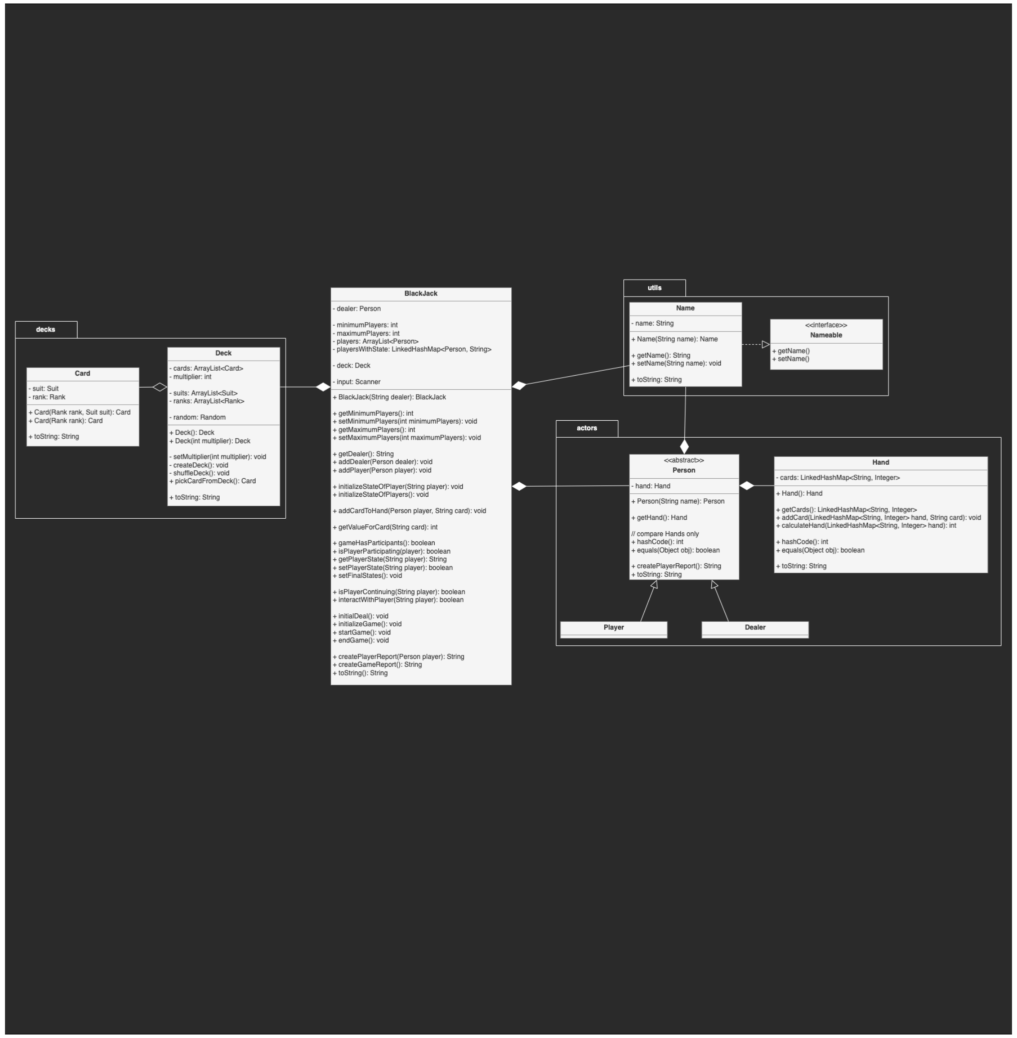
Schrittweise zum Ziel

```
> git checkout step2
```



utils
Name
- name: String
+ Name(String name): Name
+ getName(): String
+ setName(String name): void
+ toString: String
<<interface>>
Nameable
+ getName()
+ setName()
BlackJack
- dealer: Person
- minimumPlayers: int
- maximumPlayers: int
- players: ArrayList<Person>
- playersWithState: LinkedHashMap<Person, String>
- deck: ArrayList<String>
- suits: ArrayList<String>
- ranks: ArrayList<String>
- random: Random
- input: Scanner
+ BlackJack(String dealer): BlackJack
+ getMinimumPlayers(): int
+ setMinimumPlayers(int minimumPlayers): void
+ getMaximumPlayers(): int
+ setMaximumPlayers(int maximumPlayers): void
+ getDealer(): String
+ addDealer(Person dealer): void
+ addPlayer(Person player): void
+ initializeStateOfPlayer(String player): void
+ initializeStateOfPlayers(): void
+ createDeck(): void
+ shuffleDeck(): void
+ pickCardFromDeck(): String
+ addCardToHand(Person player, String card): void
+ getValueForCard(String card): int
+ gameHasParticipants(): boolean
+ isPlayerParticipating(player): boolean
+ getPlayerState(String player): String
+ setPlayerState(String player): boolean
+ setFinalStates(): void
+ isPlayerContinuing(String player): boolean
+ interactWithPlayer(String player): boolean
+ initialDeal(): void
+ initializeGame(): void
+ startGame(): void
+ endGame(): void
+ createPlayerReport(Person player): String
+ createGameReport(): String
+ toString(): String
actors
<<abstract>>
Person
- hand: Hand
+ Person(String name): Person
+ getHand(): Hand
// compare Hands only
+ hashCode(): int
+ equals(Object obj): boolean
+ createPlayerReport(): String
+ toString: String
Hand
- cards: LinkedHashMap<String, Integer>
+ Hand(): Hand
+ getCards(): LinkedHashMap<String, Integer>
+ addCard(LinkedHashMap<String, Integer> hand, String card): void
+ calculateHand(LinkedHashMap<String, Integer> hand): int
+ hashCode(): int
+ equals(Object obj): boolean
+ toString: String
Player
Dealer

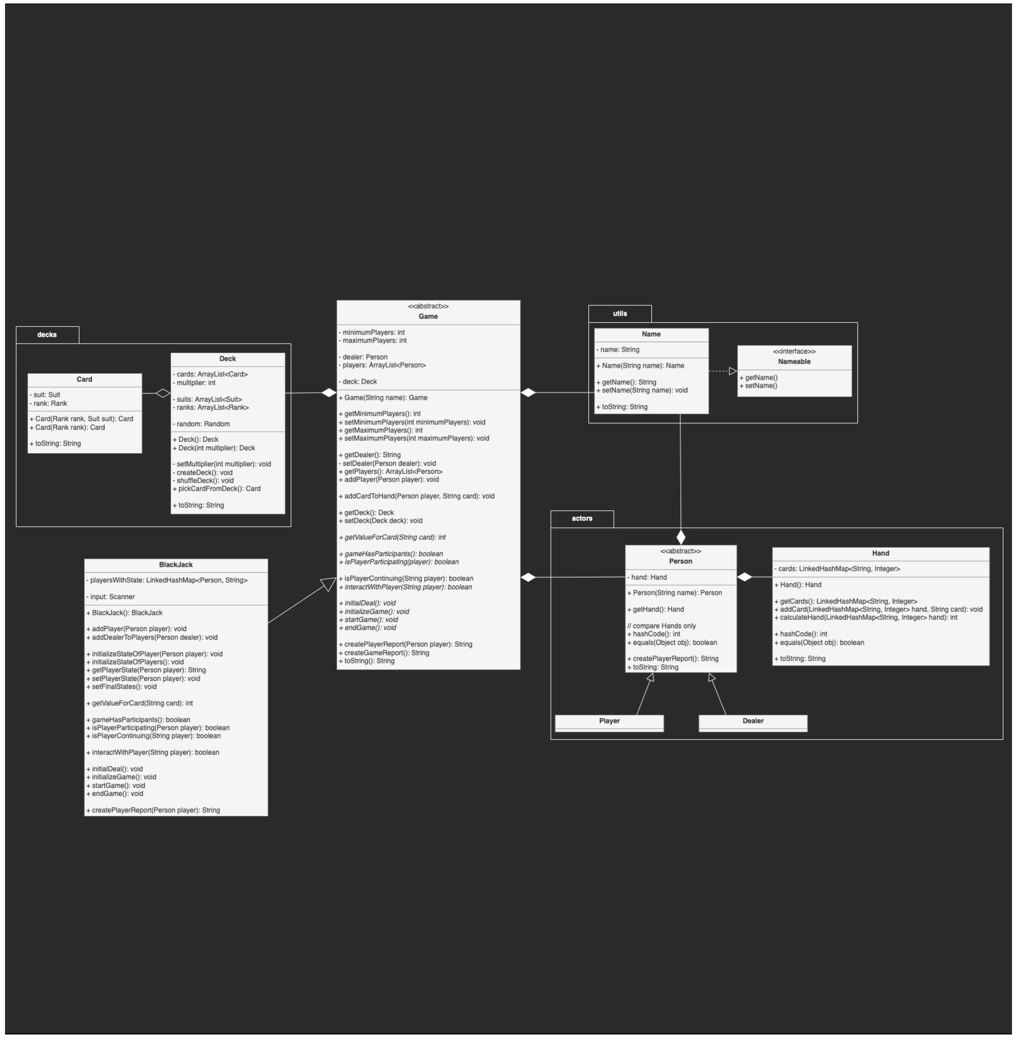
Schrittweise zum Ziel


decks
Card
- suit: Suit
- rank: Rank
+ Card(Rank rank, Suit suit): Card
+ Card(Rank rank): Card
+ toString: String
Deck
- cards: ArrayList<Card>
- multiplier: int
- suits: ArrayList<Suit>
- ranks: ArrayList<Rank>
- random: Random
+ Deck(): Deck
+ Deck(int multiplier): Deck
- setMultiplier(int multiplier): void
- createDeck(): void
- shuffleDeck(): void
+ pickCardFromDeck(): Card
+ toString: String
BlackJack
- dealer: Person
- minimumPlayers: int
- maximumPlayers: int
- players: ArrayList<Person>
- playersWithState: LinkedHashMap<Person, String>
- deck: Deck
- input: Scanner
+ BlackJack(String dealer): BlackJack
+ getMinimumPlayers(): int
+ setMinimumPlayers(int minimumPlayers): void
+ getMaximumPlayers(): int
+ setMaximumPlayers(int maximumPlayers): void
+ getDealer(): String
+ addDealer(Person dealer): void
+ addPlayer(Person player): void
+ initializeStateOfPlayer(String player): void
+ initializeStateOfPlayers(): void
+ addCardToHand(Person player, String card): void
+ getValueForCard(String card): int
+ gameHasParticipants(): boolean
+ isPlayerParticipating(player): boolean
+ getPlayerState(String player): String
+ setPlayerState(String player): boolean
+ setFinalStates(): void
+ isPlayerContinuing(String player): boolean
+ interactWithPlayer(String player): boolean
+ initialDeal(): void
+ initializeGame(): void
+ startGame(): void
+ endGame(): void
+ createPlayerReport(Person player): String
+ createGameReport(): String
+ toString(): String
utils
Name
- name: String
+ Name(String name): Name
+ getName(): String
+ setName(String name): void
+ toString: String
<<interface>>
Nameable
+ getName()
+ setName()
actors
<<abstract>>
Person
- hand: Hand
+ Person(String name): Person
+ getHand(): Hand
// compare Hands only
+ hashCode(): int
+ equals(Object obj): boolean
+ createPlayerReport(): String
+ toString: String
Hand
- cards: LinkedHashMap<String, Integer>
+ Hand(): Hand
+ getCards(): LinkedHashMap<String, Integer>
+ addCard(LinkedHashMap<String, Integer> hand, String card): void
+ calculateHand(LinkedHashMap<String, Integer> hand): int
+ hashCode(): int
+ equals(Object obj): boolean
+ toString: String
Player
Dealer

```
> git checkout step3
```

Schrittweise zum Ziel

```
> git checkout step4
```


decks
Card
- suit: Suit
- rank: Rank
+ Card(Rank rank, Suit suit): Card
+ Card(Rank rank): Card
+ toString: String
Deck
- cards: ArrayList<Card>
- multiplier: int
- suits: ArrayList<Suit>
- ranks: ArrayList<Rank>
- random: Random
+ Deck(): Deck
+ Deck(int multiplier): Deck
- setMultiplier(int multiplier): void
- createDeck(): void
- shuffleDeck(): void
+ pickCardFromDeck(): Card
+ toString: String
<<abstract>>
Game
- minimumPlayers: int
- maximumPlayers: int
- dealer: Person
- players: ArrayList<Person>
- deck: Deck
+ Game(String name): Game
+ getMinimumPlayers(): int
+ setMinimumPlayers(int minimumPlayers): void
+ getMaximumPlayers(): int
+ setMaximumPlayers(int maximumPlayers): void
+ getDealer(): String
- setDealer(Person dealer): void
+ getPlayers(): ArrayList<Person>
+ addPlayer(Person player): void
+ addCardToHand(Person player, String card): void
+ getDeck(): Deck
+ setDeck(Deck deck): void
+ getValueForCard(String card): int
+ gameHasParticipants(): boolean
+ isPlayerParticipating(player): boolean
+ isPlayerContinuing(String player): boolean
+ interactWithPlayer(String player): boolean
+ initialDeal(): void
+ initializeGame(): void
+ startGame(): void
+ endGame(): void
+ createPlayerReport(Person player): String
+ createGameReport(): String
+ toString(): String
BlackJack
- playersWithState: LinkedHashMap<Person, String>
- input: Scanner
+ BlackJack(): BlackJack
+ addPlayer(Person player): void
+ addDealerToPlayers(Person dealer): void
+ initializeStateOfPlayer(Person player): void
+ initializeStateOfPlayers(): void
+ getPlayerState(Person player): String
+ setPlayerState(Person player): void
+ setFinalStates(): void
+ getValueForCard(String card): int
+ gameHasParticipants(): boolean
+ isPlayerParticipating(Person player): boolean
+ isPlayerContinuing(String player): boolean
+ interactWithPlayer(String player): boolean
+ initialDeal(): void
+ initializeGame(): void
+ startGame(): void
+ endGame(): void
+ createPlayerReport(Person player): String
utils
Name
- name: String
+ Name(String name): Name
+ getName(): String
+ setName(String name): void
+ toString: String
<<interface>>
Nameable
+ getName()
+ setName()
actors
<<abstract>>
Person
- hand: Hand
+ Person(String name): Person
+ getHand(): Hand
// compare Hands only
+ hashCode(): int
+ equals(Object obj): boolean
+ createPlayerReport(): String
+ toString: String
Hand
- cards: LinkedHashMap<String, Integer>
+ Hand(): Hand
+ getCards(): LinkedHashMap<String, Integer>
+ addCard(LinkedHashMap<String, Integer> hand, String card): void
+ calculateHand(LinkedHashMap<String, Integer> hand): int
+ hashCode(): int
+ equals(Object obj): boolean
+ toString: String
Player
Dealer

Demo?

Card Game App

Gelernt

Card Game App

1. *OOP & Lambda*
2. *GIT, VS-Code, UML & MD*
3. *Bottom Up & Top Down*
4. *Kartenspiele ;)*

Erkenntnisse

1. *Lange Wege*
2. *Datentypen*
3. *Komplexe Spielregeln*
4. *Zeit & Backlog*

Card Game App

Danke