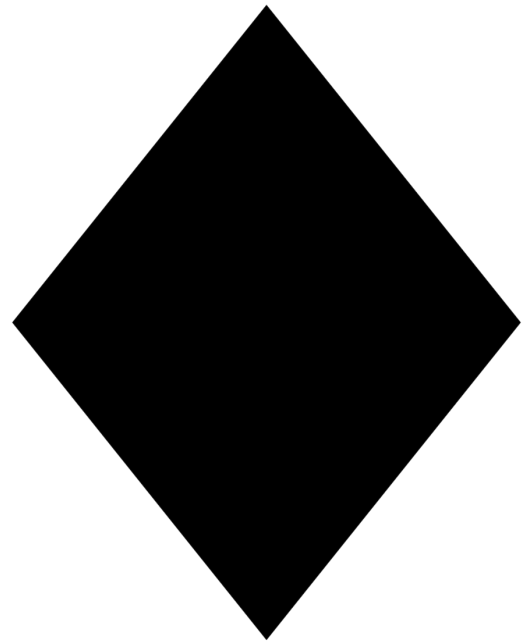


Casono

Lars, Jona, Mathis, Julian

Aktueller Status – Projektverlauf & Zeitplan

Zeitplan

- Verzögerungen im Ablauf
- Verschiebung von internen Meilensteinen
- Mangelnde Parallelisierung

Verantwortlichkeiten

- Mangelnde Eigeninitiative
- Compliance-problem

Herausforderungen & Probleme

Interne Kommunikation

- Fehlende Transparenz

Strukturelle Mängel

- Interne Kommunikation
- Fehlende Disziplin bei Richtlinien

Lösungsansätze

- **verbindliche Reporting-Zyklen** ✓
- **Optimierung der Kommunikationskanäle** ✓
- **Strikte Einhaltung der Checklisten** ✓

QS Maßnahmen

Konstruktives Qualitätsmanagement

- JavaDoc
- Logging (Log4J2)
- CONTRIBUTORS-Datei mit Guidelines
- GitLab Tasks und Issues mit Vorlagen

QS Maßnahmen

Analytisches Qualitätsmanagement

- GitLab Pipeline
  - Linting-Phase durch Checkstyle
  - Build-Phase prüft ob das Projekt kompiliert werden kann
  - Report-Phase erzeugt einen GitLab Code Quality Report
  - JavaDoc-Phase prüft den JavaDoc auf Syntaktische Fehler
  - Test-Phase für Unit-Tests aus
- Tests durch JUnit 5 und Report durch JaCoCo

Eigenschaften des Serverseitigen Netzwerkes

- Grundlage für alle weiteren Funktionen

- Clean-Architecture
  - Wartbarkeit
  - Erweiterbarkeit
  - Benutzbarkeit

- Befolung von DRY und KISS

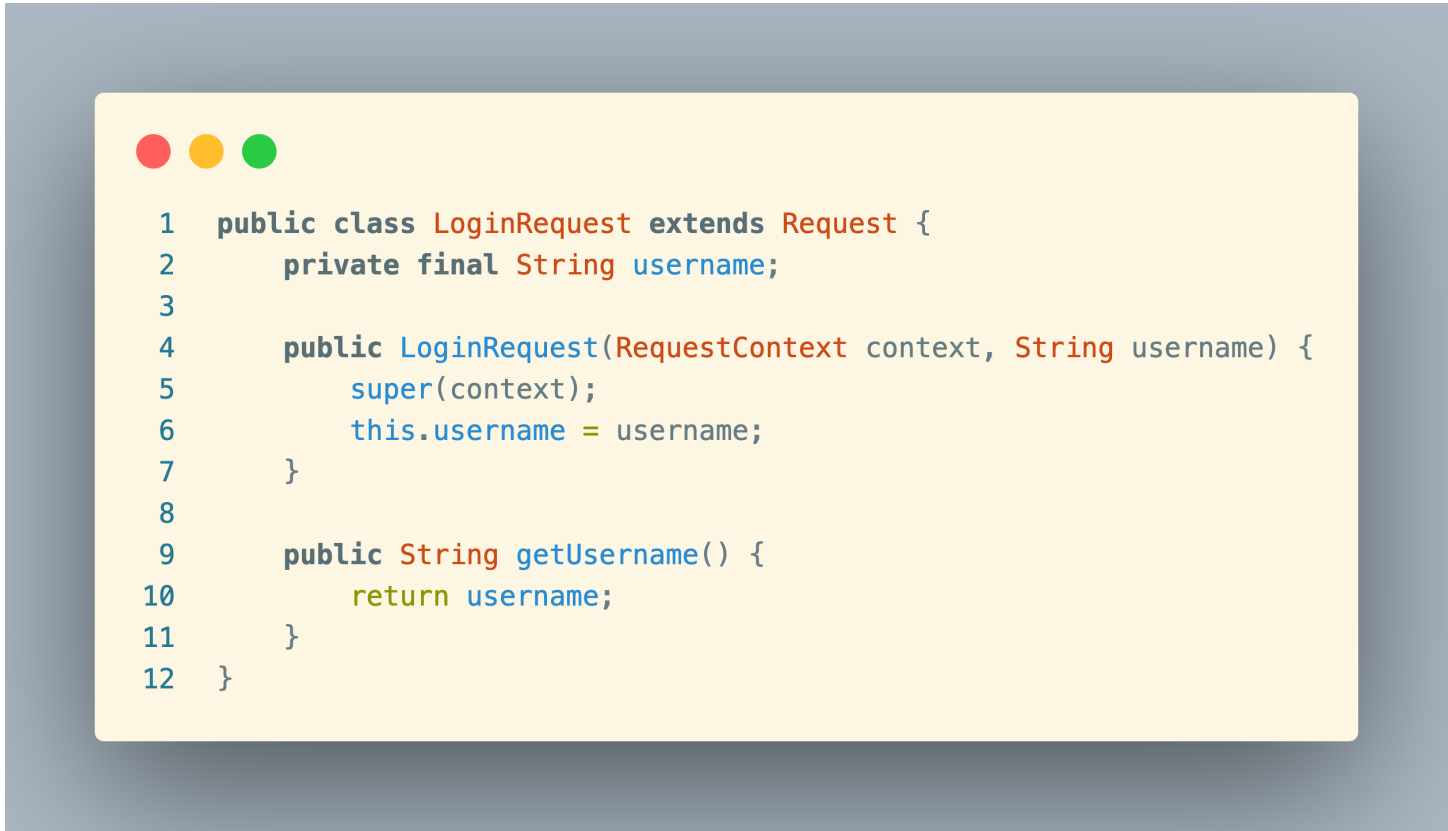
Eigenschaften des Netzwerkes

- Vollständige Abstrahierung des Netzwerkes
  - Interne Prüfung des Syntax
  - Parsing der Anfrageparameter in Datenstruktur
  - Implizite Konvertierung der Datentypen

- Vermeidung von Dupliziertem Code

Eigenschaften des Netzwerkes

- Hinzufügen eines neuen Commands
  - Erstellen der Request



```java
public class LoginRequest extends Request {
    private final String username;

    public LoginRequest(RequestContext context, String username) {
        super(context);
        this.username = username;
    }

    public String getUsername() {
        return username;
    }
}
```

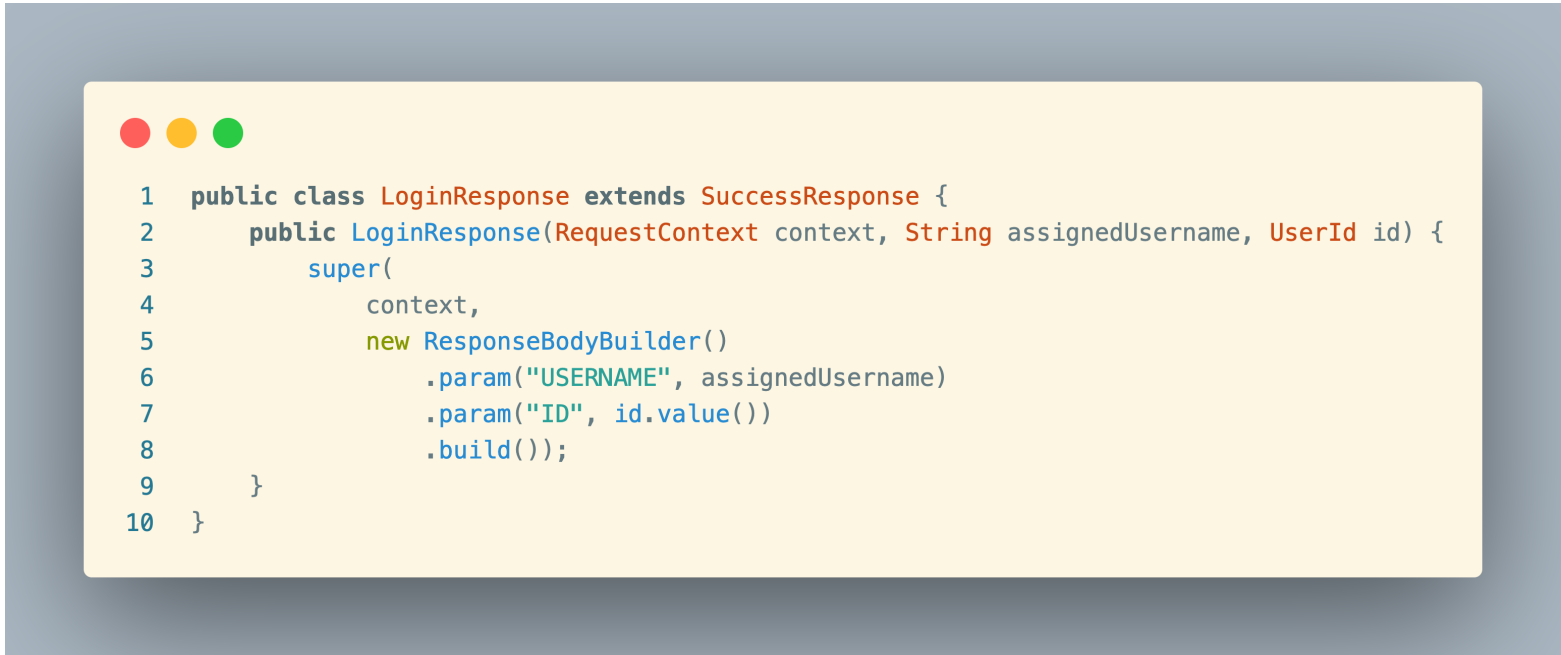
Eigenschaften des Netzwerkes

- Hinzufügen eines neuen Commands
  - Erstellen der Request
  - Erstellen des Parsers

```java
public class LoginParser implements CommandParser<LoginRequest> {
    @Override
    public LoginRequest parse(PrimitiveRequest primitiveRequest) {
        RequestParameterAccessor accessor =
                new RequestParameterAccessor(primitiveRequest.parameters());
        return new LoginRequest(primitiveRequest.context(), accessor.require("USERNAME"));
    }
}
```

Eigenschaften des Netzwerkes

- Hinzufügen eines neuen Commands
  - Erstellen der Request
  - Erstellen des Parsers
  - Erstellen der Response



```java
public class LoginResponse extends SuccessResponse {
    public LoginResponse(RequestContext context, String assignedUsername, UserId id) {
        super(
            context,
            new ResponseBodyBuilder()
                .param("USERNAME", assignedUsername)
                .param("ID", id.value())
                .build());
    }
}
```

Eigenschaften des Netzwerkes

- Hinzufügen eines neuen Commands
  - Erstellen der Request
  - Erstellen des Parsers
  - Erstellen der Response
  - Erstellen des Handlers

```java
public class LoginHandler extends CommandHandler<LoginRequest> {
    private final UserRegistry userRegistry;
    private final UserFactory userFactory;

    public LoginHandler(ResponseDispatcher responseDispatcher, UserRegistry userRegistry) {
        super(responseDispatcher);
        this.userRegistry = userRegistry;
        this.userFactory = new UserFactory(userRegistry);
    }

    @Override
    public void execute(LoginRequest request) {
        Optional<User> existingUser = userRegistry.getBySessionId(request.getSessionId());
        if  existingUser.isPresent()) {
          ( responseDispatcher.dispatch(
                    new ErrorResponse(
                            request.getContext(),
                            "ALREADY_LOGGED_IN",
                            "This session is already associated with an active user."));
            return;
        }

        User user = userFactory.create(request.getUsername(), request.getSessionId());
        responseDispatcher.dispatch(
                new LoginResponse(request.getContext(), user.getName(), user.getId()));
    }
}
```

Eigenschaften des Netzwerkes

- Hinzufügen eines neuen Commands
  - Erstellen der Request
  - Erstellen des Parsers
  - Erstellen der Response
  - Erstellen des Handlers
  - Registrierung des Commands

Goals

Goals

- Den Pot gewinnen
- Täuschung anderer Spieler
- Gewinn maximieren
- Verlust minimieren
- Informationsvorteil erlangen
- Positionsspiel

Goal: Täuschung anderer Spieler

- "Bluffen"
- Das eigene Spielverhalten verschlüsseln
- "Mundbluff"

Goal: Gewinn maximieren

- "Outs"
- Die eigenen Karten schützen
- Andere Spieler dazu bringen, zu "Folden"

Goal: Verlust minimieren

- "Risk of Ruin"
- Bluffs erkennen
- Die "Stärke" der eigenen Hand

Goal: Informationsvorteil erlangen

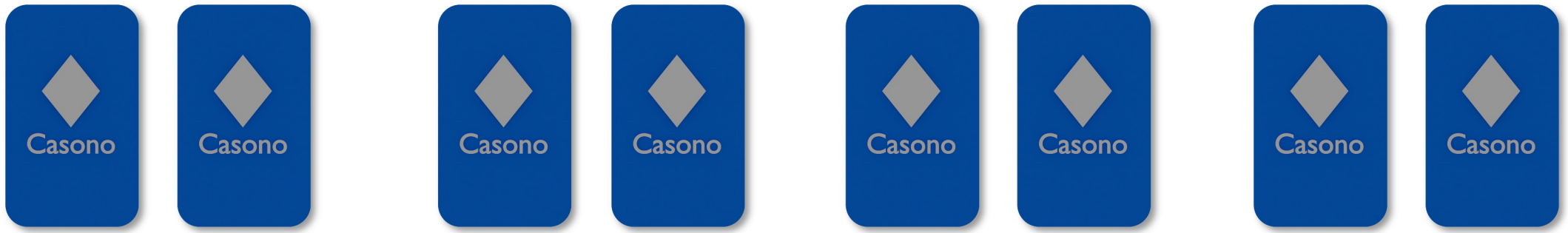
- Verteilung möglicher Kartenkombinationen
- Je mehr Spieler vor einem an der Runde waren, desto mehr Informationen hat man
- "Pot Odds"

Goal: Positionsspiel

- Die eigene Position einschätzen und ausnutzen können
- "späte Positionen"

Casono Rules

TDA Rules version 1 2024



Jona

(Small Blind)

Mathis

Julian

Lars

(Big Blind)

Jona

Mathis

Julian

Lars

Jona

Mathis

Julian

Lars

Jona

Mathis

Julian

Lars


Casono
Casono
Casono
Casono
Casono
200
CASONO
200
CASONO
500
CASONO
100
CASONO
Casono
Casono
Casono
Casono
Casono
Casono
Casono
Casono
Jona
Mathis
DEALER
CASONO
Julian
Lars
20000
CASONO
100
CASONO
200
CASONO
100
CASONO

Jona

Mathis

Julian

Lars

Jona

Mathis

DEALER
CASONO

Julian

Lars

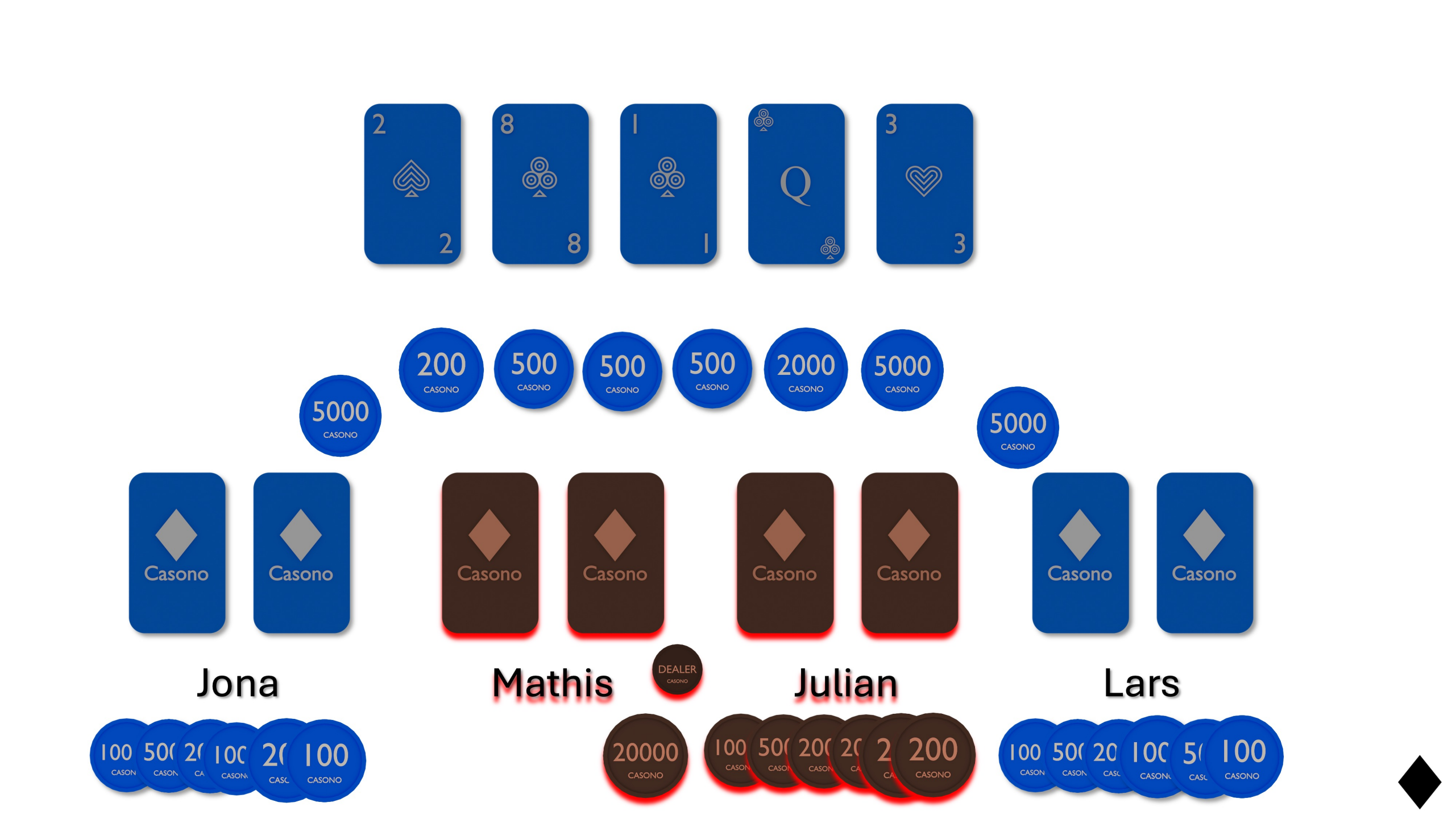

Casono
Casono
1
1
Q
3
3


200
CASONO
500
CASONO
500
CASONO
500
CASONO
1000
CASONO


1000
CASONO
1000
CASONO
Casono
Casono
Casono
Casono
Casono
Casono
Casono
Casono
Jona
Mathis
DEALER
CASONO
Julian
Lars
100
CASON
50(
CASON
2(
CA
100
CASON
2(
CASC
100
CASONO
20000
CASONO
100
CASON
50(
CASON
20(
CASON
2(
CA
2
CA
200
CASONO
100
CASON
50(
CASON
20
CASC
100
CASONO
5(
CASC
100
CASONO


Casono
8
8
1
1
Q
3
3


200 CASONO
500 CASONO
500 CASONO
500 CASONO
1000 CASONO
2000 CASONO
1000 CASONO
2000 CASONO
1000 CASONO
2000 CASONO
Casono
Casono
Casono
Casono
Casono
Casono
Casono
Casono
Jona
Mathis
DEALER CASONO
Julian
Lars
20000 CASONO
100 CASONO
200 CASONO
100 CASONO


2
2
8
8
1
1
Q
3
3
200
CASONO
500
CASONO
500
CASONO
500
CASONO
2000
CASONO
5000
CASONO
5000
CASONO
5000
CASONO
Casono
Casono
Casono
Casono
Casono
Casono
Casono
Casono
Jona
Mathis
DEALER
CASONO
Julian
Lars
100
CASONO
100
CASONO
20000
CASONO
200
CASONO
100
CASONO

Jona

Mathis

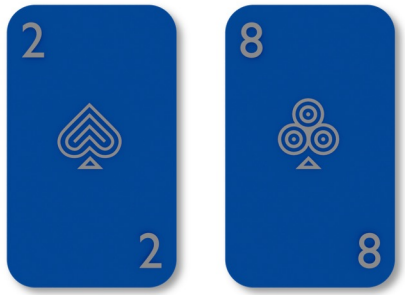
DEALER
CASONO

Julian

Lars

https://m.media-amazon.com/images/I/61WOOlDLfTL._AC_UF894,1000_QL80_.jpg

Je höher die Kombination,
desto stärker die Hand.

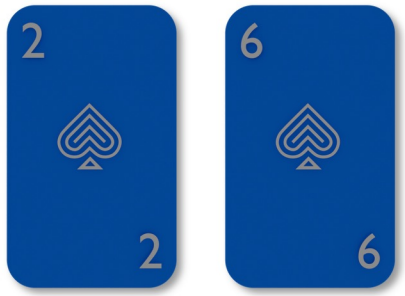


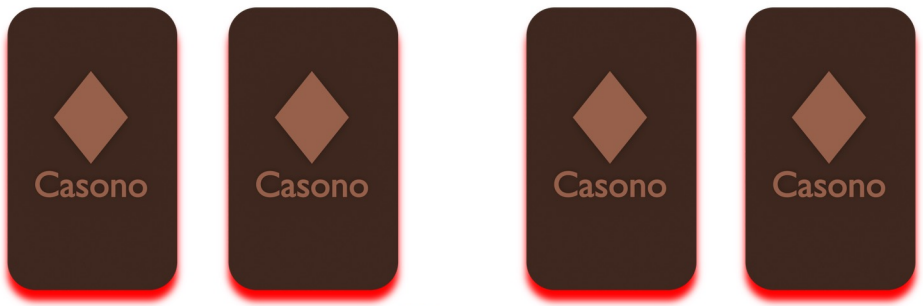
Jona

Mathis

Julian

Lars





Jona

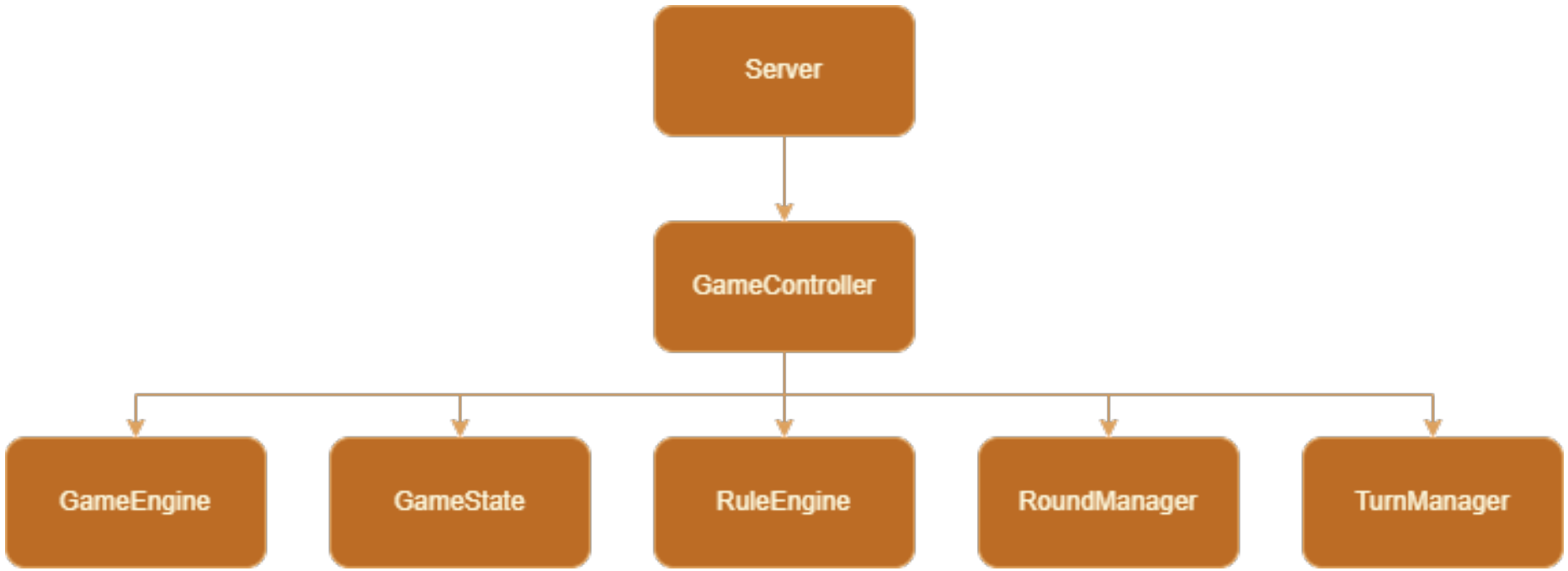
Mathis

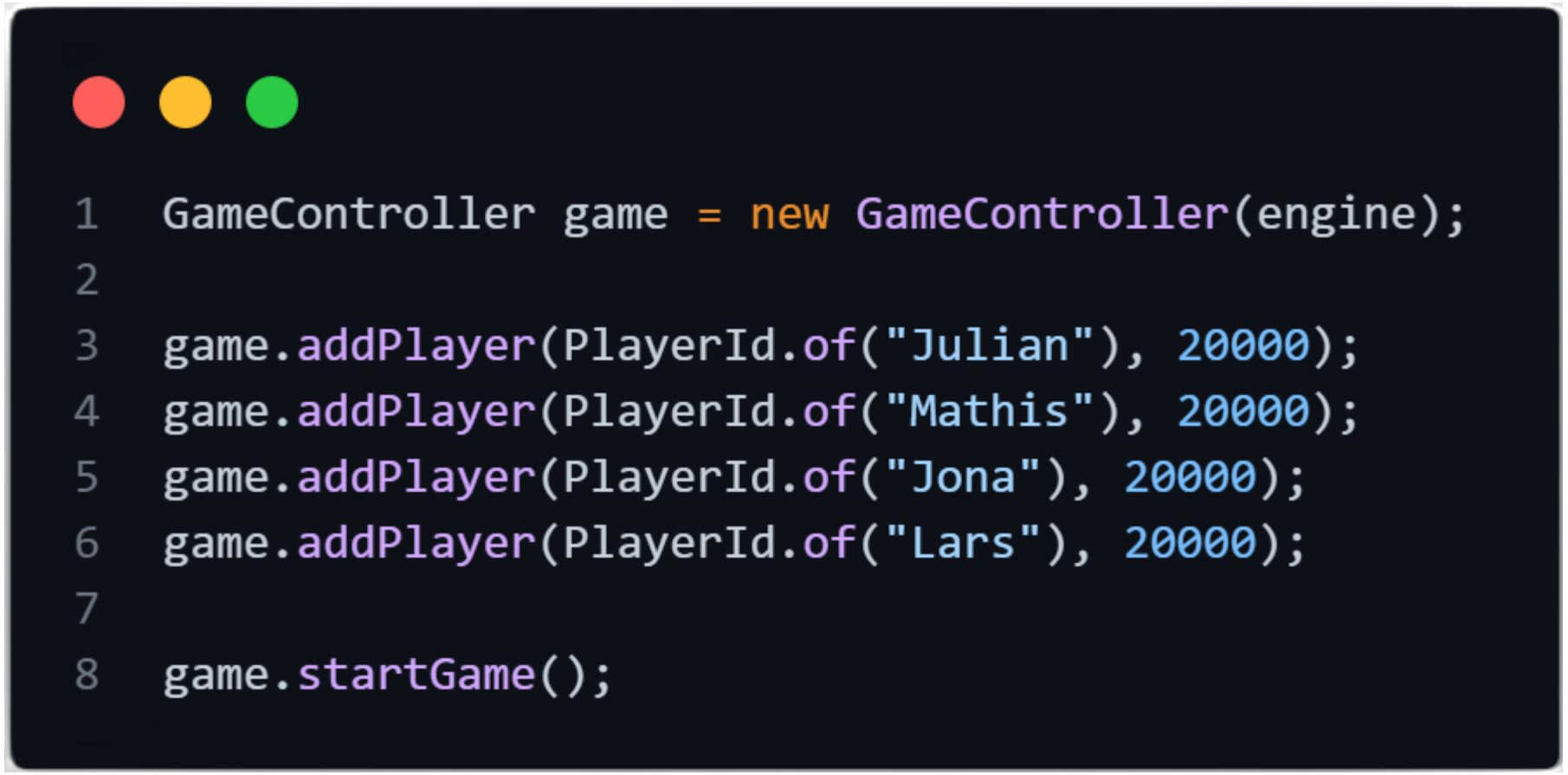
DEALER
CASONO

Julian

Lars

Casono Game Engine


Server
GameController
GameEngine
GameState
RuleEngine
RoundManager
TurnManager


GameController game = new GameController(engine);
game.addPlayer(PlayerId.of("Julian"), 20000);
game.addPlayer(PlayerId.of("Mathis"), 20000);
game.addPlayer(PlayerId.of("Jona"), 20000);
game.addPlayer(PlayerId.of("Lars"), 20000);
game.startGame();

```
1  game.playerCall(PlayerId.of("Julian"));
2  game.playerFold(PlayerId.of("Mathis"));
3  game.playerCall(PlayerId.of("Jona"));
4  game.playerRaise(PlayerId.of("Lars"), 1200);
```

```
1  String winner = game.showdown();
```

```
1  game.dealTurn();
2  game.dealFlop();
3  game.dealRiver();
```