Protokoll für das Monster Trading Card Game

Student Name: Mohamed Nasr
Project: Monster Trading Card Game
GitHub Link: MTCG GitHub Repository

1. Einleitung:

Das Monster Trading Card Game ist ein HTTP/REST-basierter Server, der es Spielern ermöglicht, Karten zu erwerben, Decks zu verwalten und gegen andere Spieler zu kämpfen. Die Implementierung verwendet Java, und es wurden keine HTTP-Hilfsbibliotheken verwendet. Alle HTTP-Protokolle und REST-Endpunkte wurden manuell erstellt.

2. Design-Entscheidungen:

1. **Server-Implementierung**: Der Server basiert auf Java-Sockets und verarbeitet HTTP-Anfragen manuell. Es wurde kein HTTP-Framework verwendet, um die REST-API zu implementieren.

2. **Routing-Funktionalität**: Die Routen werden manuell über eine zentrale `Router`-Klasse verwaltet. Jeder Endpunkt ist einem spezifischen Service zugeordnet, der die entsprechende Funktionalität bereitstellt.

3. **Token-basierte Authentifizierung**: Bei erfolgreichem Login wird ein Token generiert, das in zukünftigen Anfragen verwendet wird, um den Benutzer zu authentifizieren. Das Token wird in-memory im Benutzerobjekt gespeichert.

4. **In-Memory-Datenhaltung**: Es wurde keine Datenbank verwendet. Alle Benutzer und deren Kartenstapel werden in-memory gespeichert, was die Anforderungen an die Registrierung und das Login erfüllt.

3. Struktur der Anwendung:

Die Anwendung folgt einer klaren Aufteilung der Verantwortlichkeiten:

- **Server-Package (`httpserver`)**: Hier werden die Kernkomponenten des Servers wie `Request`, `Response`, `Router` und der eigentliche `Server` selbst verwaltet.
- **Model-Package (`cardgame.model`)**: Enthält die zentralen Modelle für das Spiel, wie z.B. `User` und `Card`. Die `Card`-Klasse speichert Informationen über den Namen, den Schaden und den Elementtyp (Feuer, Wasser, Normal).
- **Service-Package (`cardgame.service`)**: Hier sind die Hauptlogik und -services wie `UserService` (für Registrierung und Login) und `CardService` (für die Kartenverwaltung) implementiert.

4. REST-Endpunkte:

Benutzerregistrierung (POST /users):

- Dieser Endpunkt registriert einen neuen Benutzer. Der Benutzername muss eindeutig sein, und das Passwort wird ebenfalls gespeichert. Bei erfolgreicher Registrierung wird ein `201 Created`-Status zurückgegeben.

Benutzer-Login (POST /sessions):

- Der Endpunkt ermöglicht es einem registrierten Benutzer, sich einzuloggen. Bei erfolgreichem Login wird ein Token generiert und an den Benutzer zurückgegeben. Dieses Token wird für zukünftige Anfragen verwendet.

5. Token-basierte Sicherheit:

Nach dem Login erhält der Benutzer ein Token, das für geschützte Endpunkte benötigt wird. Das Token wird bei jeder Anfrage in den HTTP-Headern übergeben und vom Server validiert.

6. Integrationstests:

Die Integrationstests wurden mithilfe des mitgegebenen CURL-Skripts durchgeführt. Diese Tests decken die Benutzerregistrierung, das Login und die Token-Validierung ab.

7. Klassen-Diagramm: