Protokoll zur Implementierung des Klassendiagramms

1. Zielsetzung

Das Ziel dieser Arbeit war die Entwicklung eines Klassendiagramms zur Modellierung einer Anwendung, die folgende Hauptaspekte umfasst:

- Verwaltung von Benutzern und Sitzungen
- Implementierung eines HTTP-Servers
- Abbildung von Spielkarten und einer Battle-Logik für ein Monster Trading Card Game (MCTG).

2. Vorgehensweise

2.1 Design der Hauptklassen

- **Card (Abstrakte Klasse)**:
  - Modelliert allgemeine Eigenschaften und Methoden von Karten, wie `Damage` und `CardElementType`.
  - Spezifische Kartenarten (`NormalCard`, `SpellCard`, `MonsterCard`) erben von der Basisklasse `Card`.
  - Methoden wie `PlayCard` und `AssignDamage` wurden definiert, um das Verhalten von Karten zu steuern.
- **User**:
  - Repräsentiert einen Benutzer mit Attributen wie `UserName`, `Deck`, `Coins` und `SessionToken`.
  - Enthält Methoden zur Verwaltung von Benutzerdaten wie `AddPackage`, `CreateCard` und `Logon`.
- **Battle**:
  - Implementiert die Logik für Kämpfe zwischen zwei Spielern (`Player1` und `Player2`).
  - Enthält Methoden wie `Start` und `Play` zur Durchführung der Kämpfe.

2.2 Implementierung des HTTP-Servers

- **HttpSvr**:
  - Implementiert grundlegende Serverfunktionen wie `Run` und `Stop`.
  - Verarbeitet eingehende Anfragen mit der Methode `Incoming`.
- **HttpHeader und HttpStatusCode**:
  - Dienen zur Verwaltung und Interpretation von HTTP-Headern und Statuscodes.

2.3 Session- und Benutzerverwaltung

- **Handler (Abstrakte Klasse)**:
  - Definiert die Basislogik für verschiedene Handler wie `SessionHandler` und `UserHandler`.
  - Methoden wie `HandleEvent` wurden implementiert, um eingehende Ereignisse zu verarbeiten.
- **SessionHandler**:
  - Verarbeitet aktive Benutzersitzungen und enthält Methoden wie `GetUsernameFromSession`.
- **UserHandler**:
  - Zuständig für die Benutzerverwaltung, einschließlich Methoden wie `HandleAddPackage` und `HandleGetUser`.

2.4 Token-Management

- **Token (Statische Klasse)**:
  - Verarbeitet die Authentifizierung von Benutzern mittels Tokens.
  - Enthält Methoden wie `CreateTokenFor` und `Authenticate`.

2.5 Erweiterung des Systems

- **Round**:
  - Ergänzt die Battle-Logik durch die Abbildung einzelner Runden.
- **Enumerationen und Delegates**:
  - `ElementType` zur Modellierung von Kartenelementtypen wie `Water`, `Fire`, `Normal`.
  - `HttpSvrEventHandler` zur Definition von Ereignisverarbeitern im HTTP-Server.

3. Herausforderungen

- **Komplexität der Vererbung**:
  - Die Vererbungshierarchie zwischen `Card` und deren Unterklassen musste klar definiert werden, um zukünftige Erweiterungen zu erleichtern.
- **Sitzungsmanagement**:
  - Die Synchronisation aktiver Sitzungen stellte sicher, dass Benutzer eindeutig identifiziert werden können.
- **HTTP-Kommunikation**:
  - Die Implementierung eines effizienten Request-Response-Mechanismus war entscheidend, um die Systemstabilität zu gewährleisten.

4. Ergebnisse

Das Klassendiagramm bildet eine solide Grundlage für die weitere Entwicklung der Anwendung. Es wurden folgende zentrale Funktionalitäten modelliert:

- Benutzerverwaltung

- Sitzungs- und Authentifizierungslogik
- HTTP-Server und Header-Management
- Spielkarten und Battle-Mechanik

5. Nächste Schritte

- Implementierung der Logik für spezielle Kartenfähigkeiten und deren Auswirkungen im Kampf.
- Integration von Sicherheitsmechanismen zur Sicherstellung der Datenintegrität.
- Unit-Tests für die einzelnen Komponenten, um die Stabilität und Funktionalität des Systems zu überprüfen.

6. Github Repository Link

Link zum Repository