Verteilte Systeme

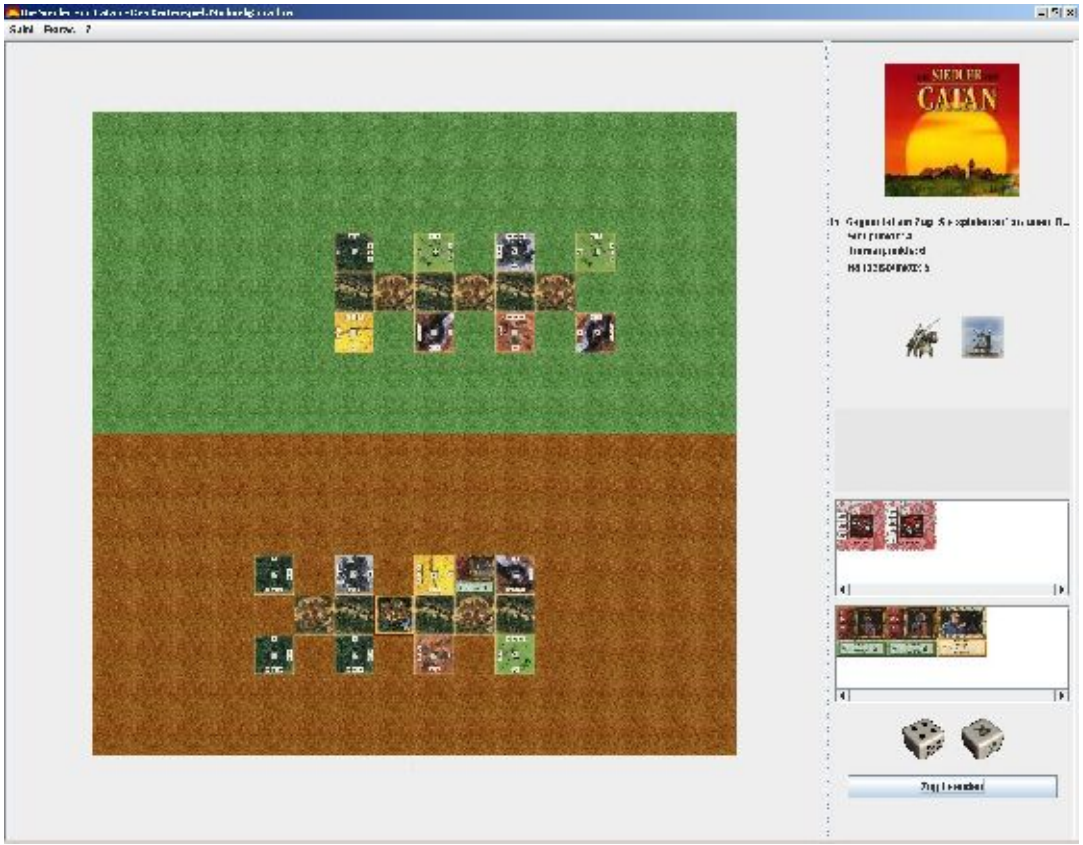
Die Siedler von Catan - Das Kartenspiel

Kartenspiel für 2 Personen

Verteilte Anwendung mittels Java und RMI

<u>Programmierer:</u>

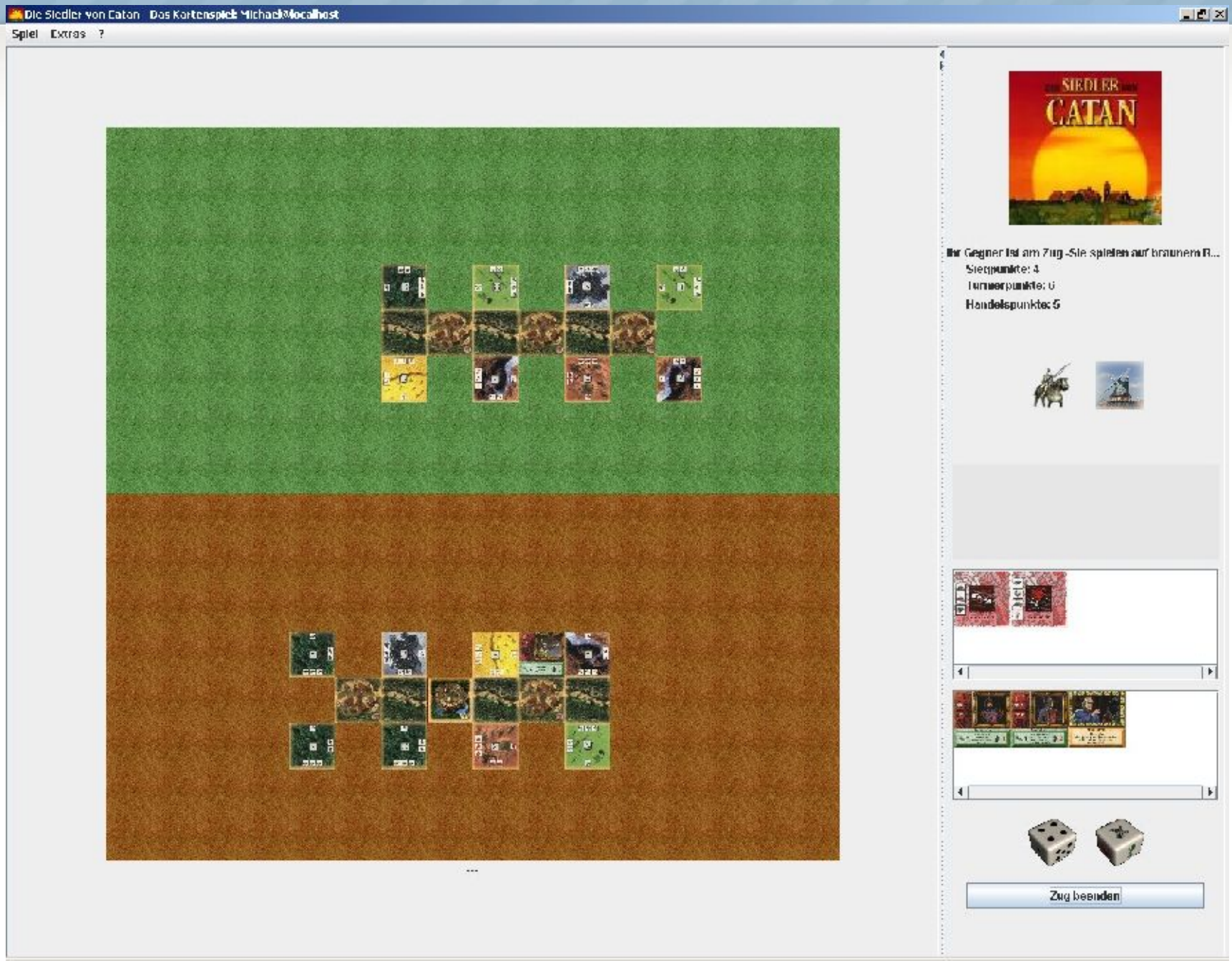
- Michael Körner:
  - GUI
  - RMI
- Marcus Zelend:
  - Kartenverwaltung
  - Spiellogik
- Danny Christl:
  - Spiellogik
  - GUI



Gliederung

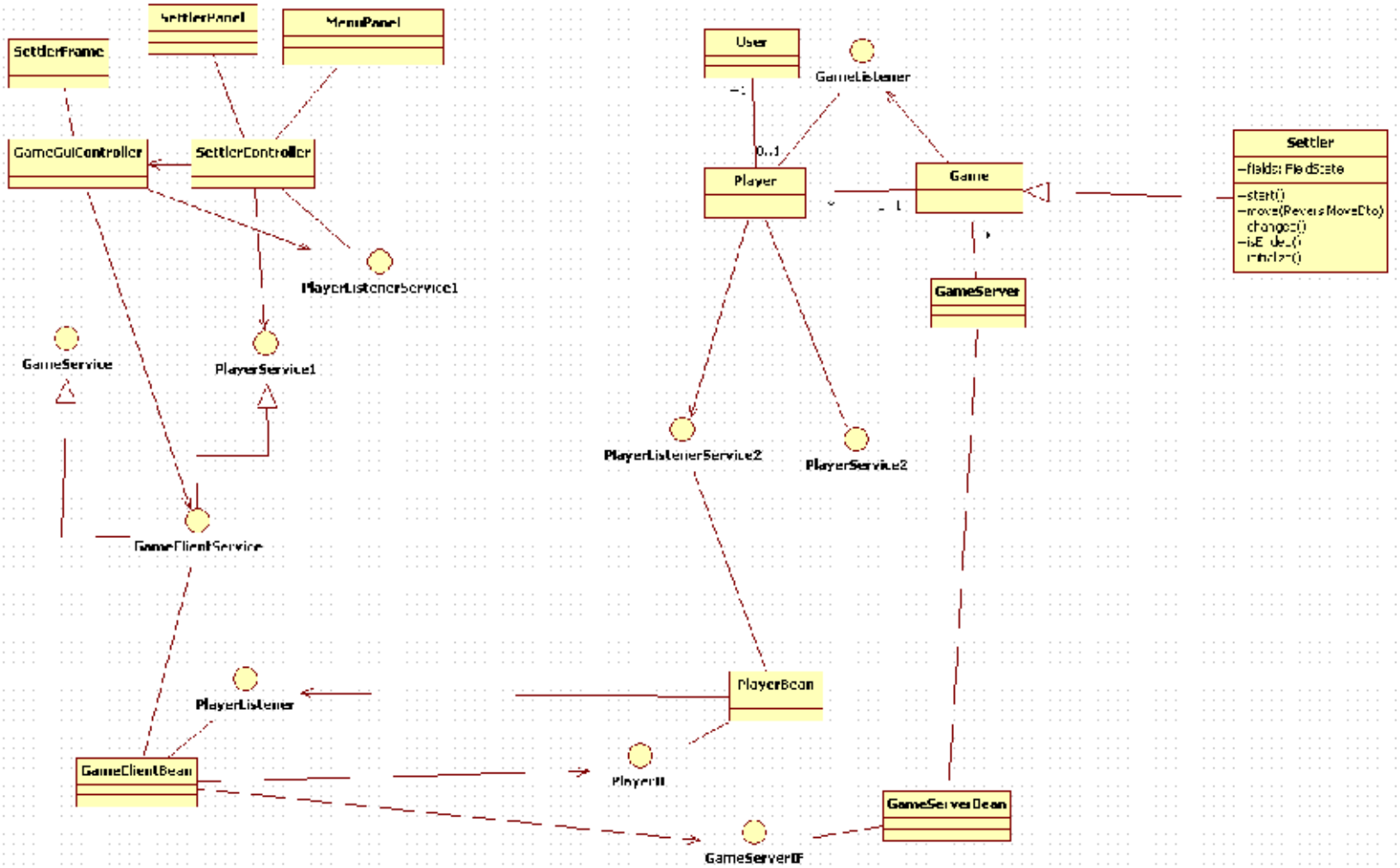
1. Einführung in das Spiel und die GUI
2. Konzept der Dtos und des Sendens am Beispiel
3. Aufbau und Typen der Karten im Spiel
4. Prinzipielle Spiellogik
5. Aufbau und Methoden des Fürstentums
6. Ausblick und Verbesserungsmöglichkeiten

Einführung und GUI



Realisation der Verteilten Applikation mittels RMI

Gesamtaufbau



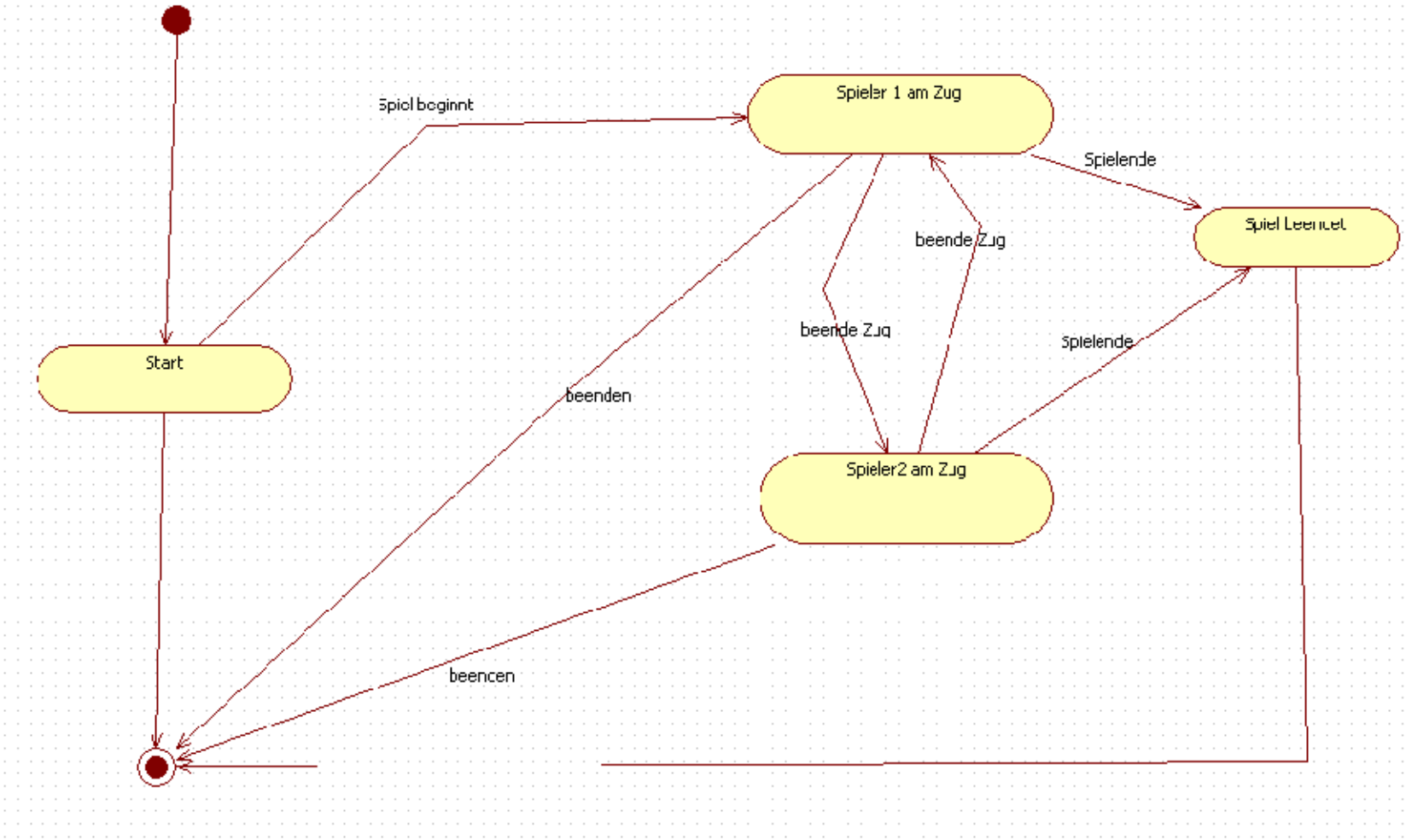
Sendevorgang

Gesendet werden:

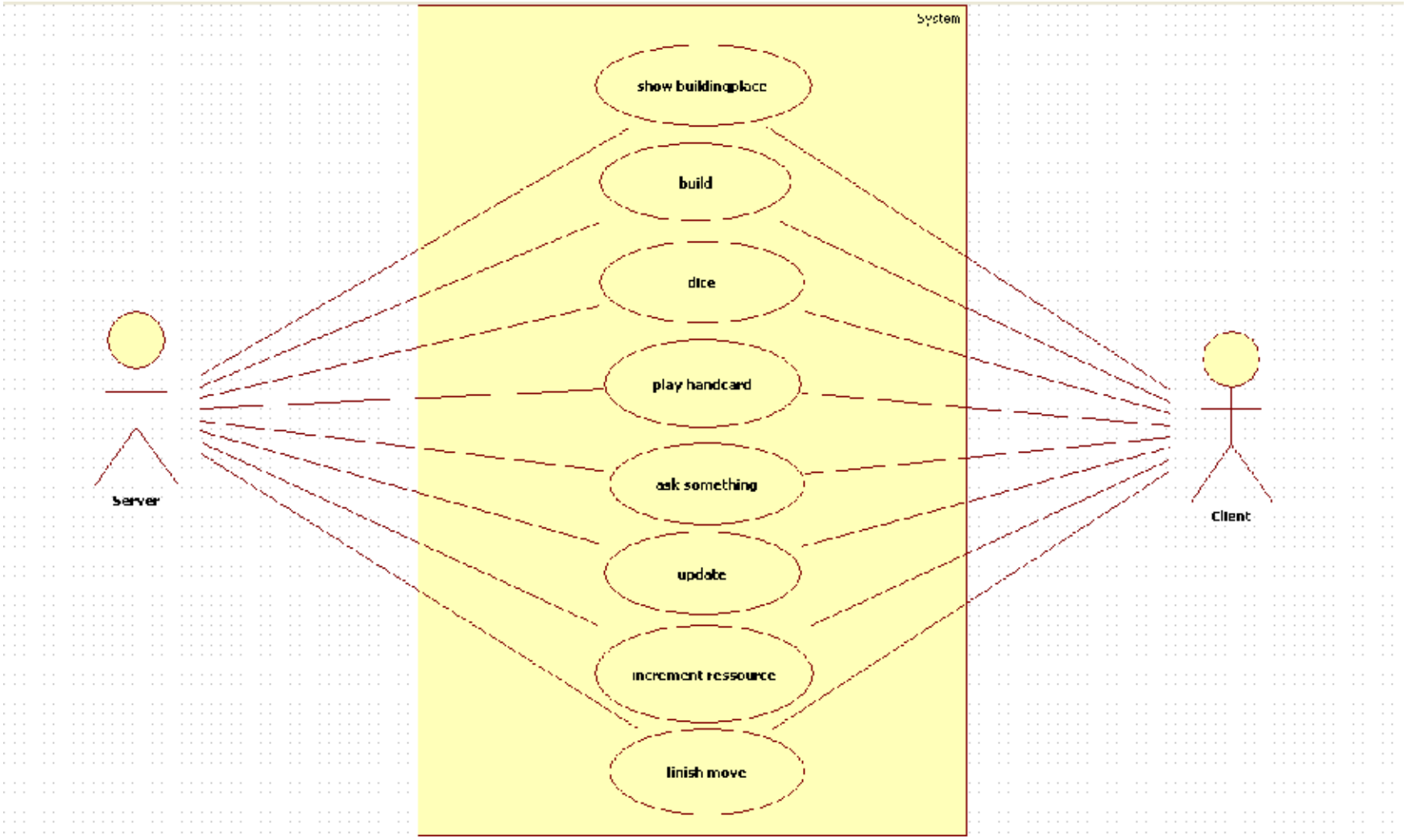
- SettlerMoveDto → (Client zum Server) → stellt den aktuellen Zug dar

- SettlerGameDto → (Server zum Client) → repräsentiert Aktionen des Games

Zustände


Spiel beginnt
Spieler 1 am Zug
Spielende
Spiel beendet
beende Zug
beende Zug
Spielende
Start
beenden
Spieler2 am Zug
beencen

Anwendungsfälle



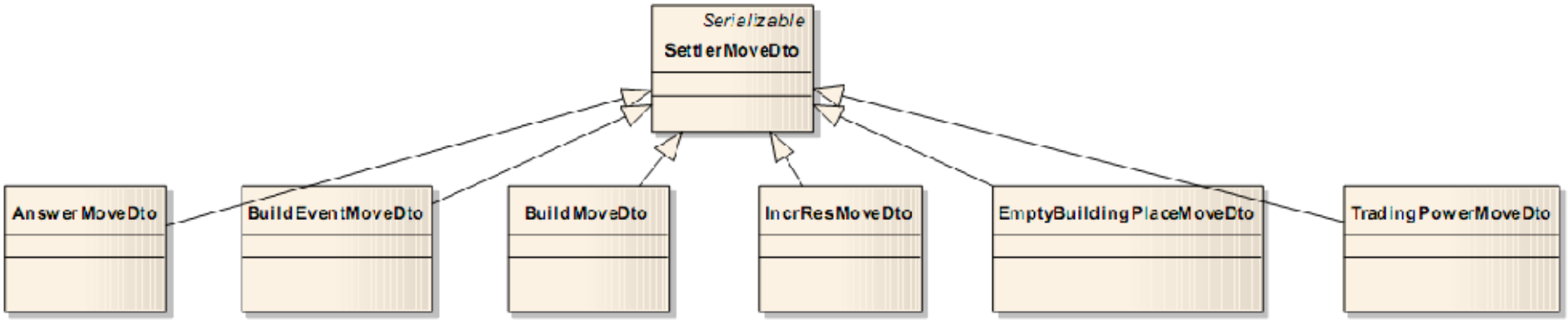
Sendevorgang

- Unterschiedliche Daten erfordern Vererbungskonzept innerhalb der DTOs

- Jede Dto erbt entweder von SettlerMoveDto oder SettlerGameDto und bekommt nur einen kind (Art der zu sendenden Information) vererbt

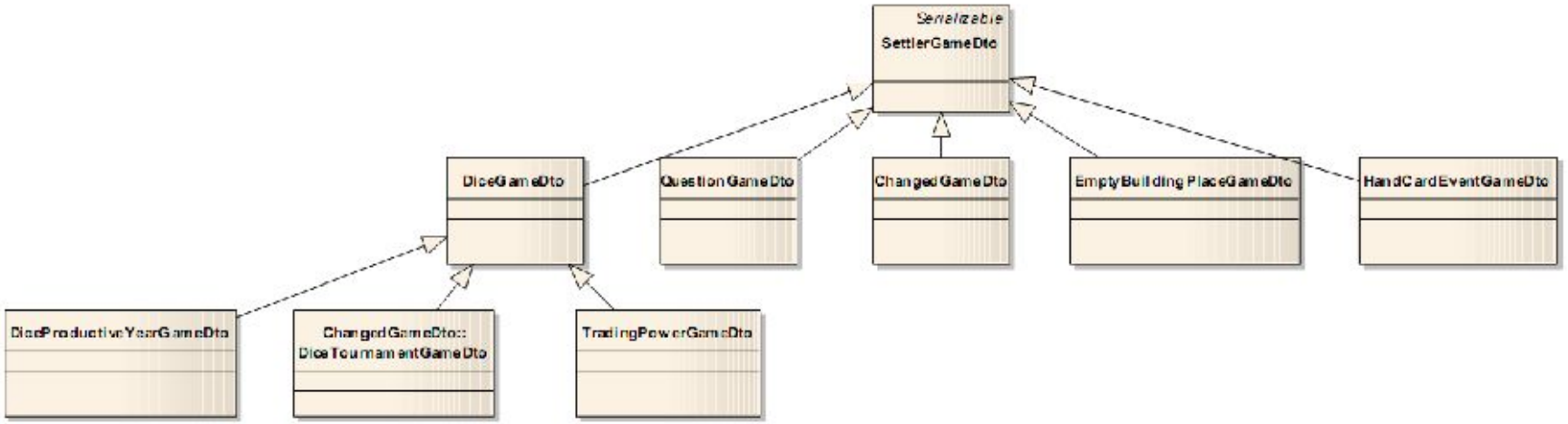
SettlerMoveDto

- Das Objekt der jeweiligen Klasse repräsentiert den Zug bzw. die Anfrage des Spielers an den Server

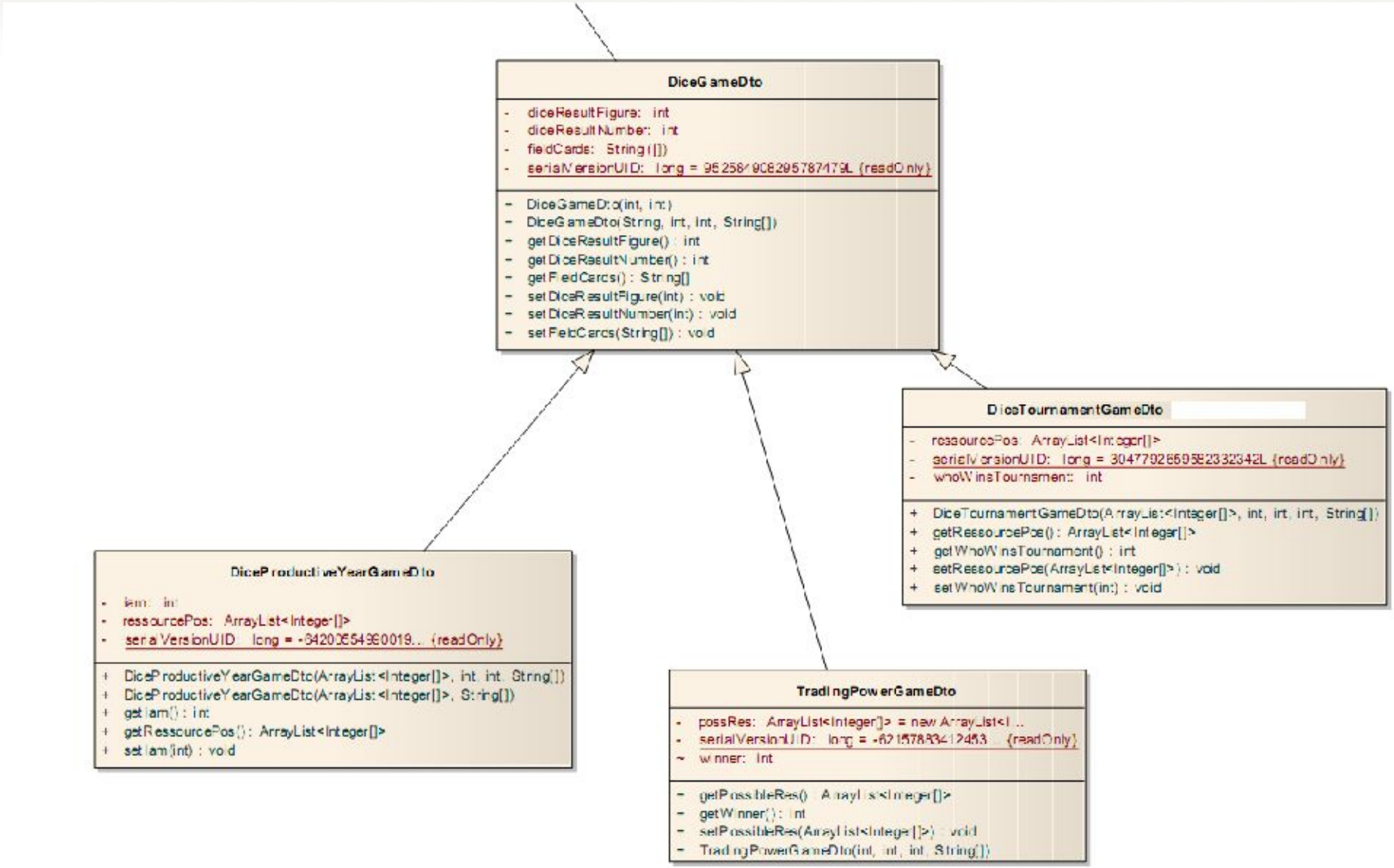


SettlerGameDto

-Das Objekt der jeweiligen Klassen repräsentiert die Anfrage bzw. Darstellung der Daten vom Spiel



Beispiel Würfeln

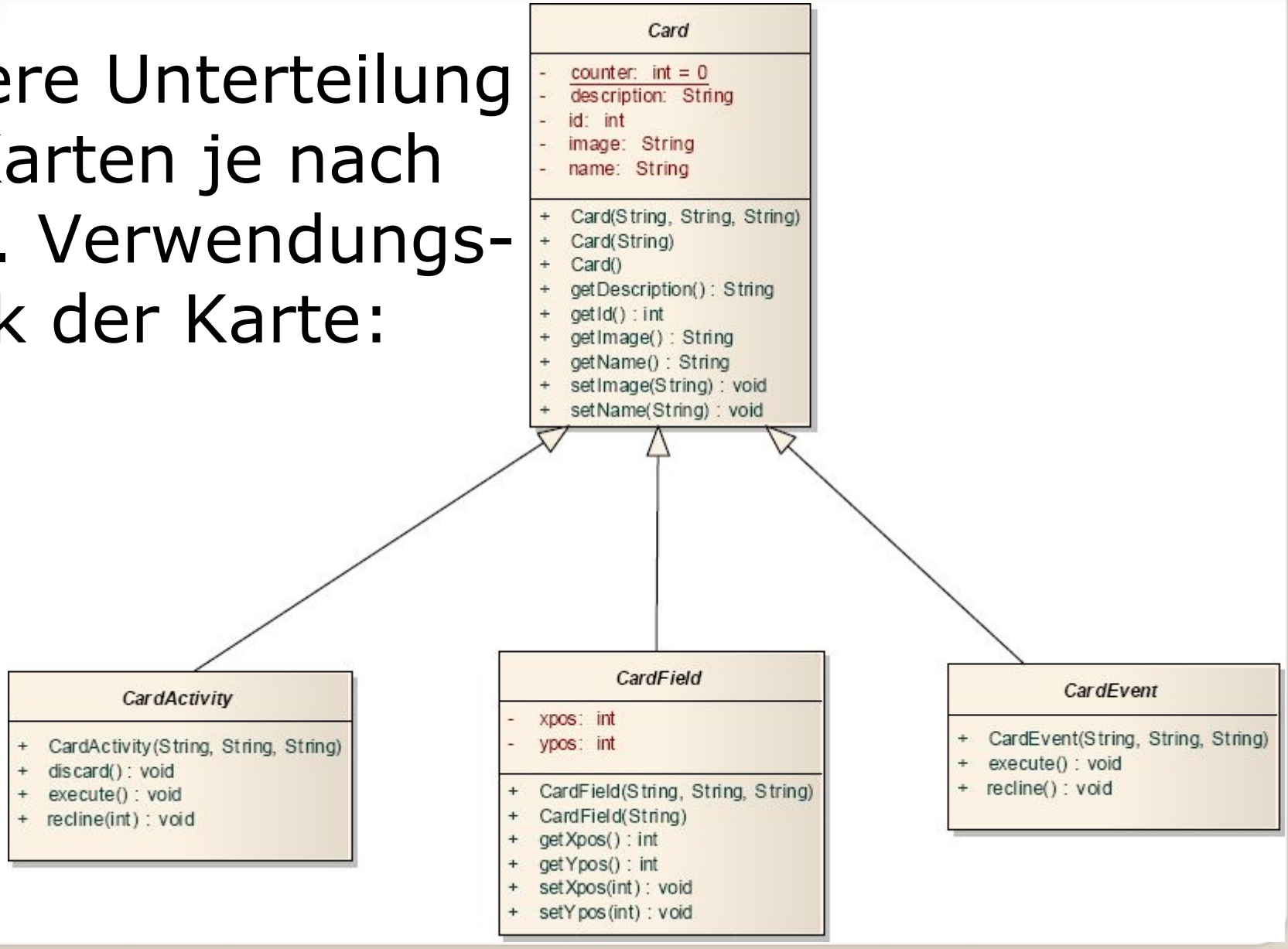


Die Spielkarten

- Die Spielkarten befinden sich im Package „game.Cards“
- Alle Spielkarten werden von der gemeinsamen Basisklasse „Card“ abgeleitet
- Diese stellt Name, Kartentext, das Bild der Karte sowie eine fortlaufende ID bereit
- Klasse „Deck“ verwaltet die Kartenstapel

Karten: Klassenstruktur

- Weitere Unterteilung der Karten je nach Art u. Verwendungs-zweck der Karte:



Karten: Klassenstruktur

- Weitere Unterteilung der Karten je nach Art und Verwendungszweck der Karte:

Karten: Klassenstruktur

- Weitere Unterteilung der Karten je nach Art und Verwendungszweck der Karte:

Karten: Klassenstruktur

- Weitere Unterteilung der Karten je nach Art und Verwendungszweck der Karte:

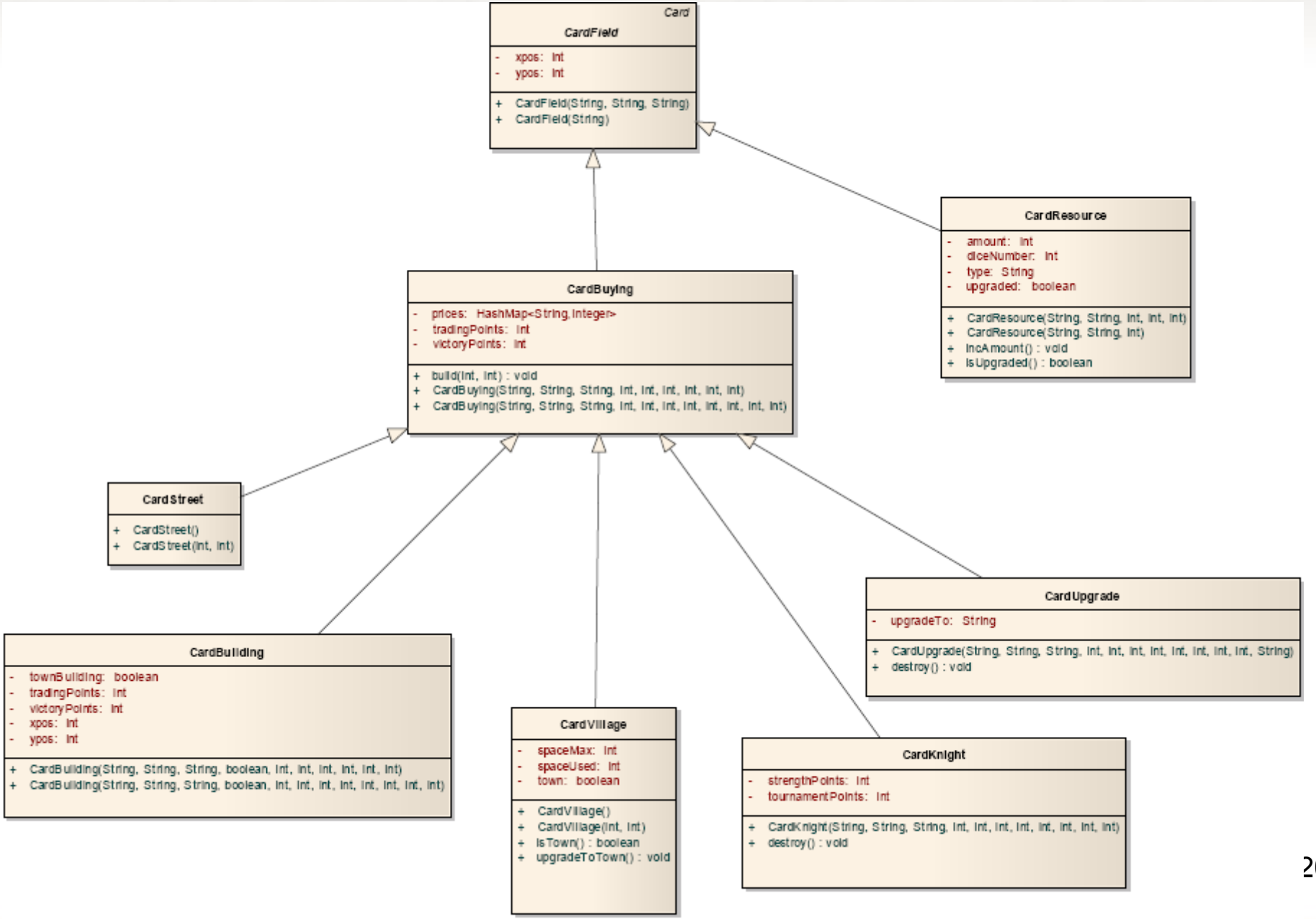
Karten: Ereigniskarten

- ursprünglich geplant war, in jeder Karte eine Methode bereitzustellen die das Ereignis der Karte ausführt
- später verworfen, da nicht mit Spiellogik kompatibel
  → in Spielklasse „Settler“ ausgelagert
- Unterklassen nur beibehalten um Karten leichter anlegen zu können (Name, Beschreibung und Kartengrafik werden im Konstruktor angegeben)

Karten: Feldkarten

- Basisklasse „CardField“ (stellt Position der Karte auf dem Spielplan bereit)
- weitere Spezialisierung je nach Art der Karte

Karten: Feldkarten

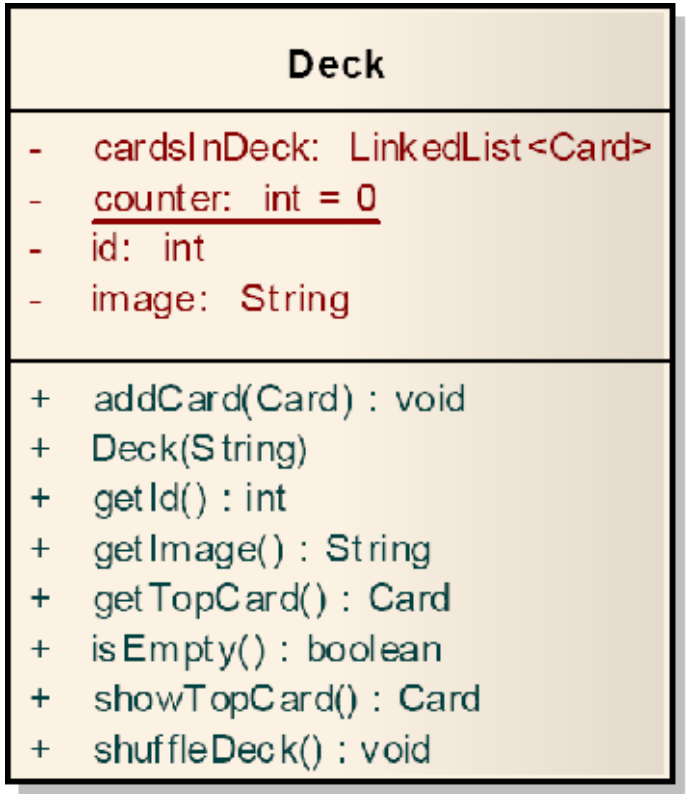


Karten: Feldkarten

- CardBuilding: Gebäude (kosten Rohstoffe, Gebiets- oder Stadtausbau, haben Einfluss auf Siegpunkte oder Handelsmacht)
- CardKnight: Ritter (Einfluss auf Rittermacht und Würfelereignis „Turnier“)
- CardVillage: Siedlungen (können zu Städten ausgebaut werden)
- CardResource: Landschaftsausbauten (es werden automatisch jeweils 2 beim Bau einer neuen Siedlung gebaut)
- CardStreet: Straßen (verbinden Siedlungen)

Karten: Kartenstapel

- Klasse „Deck“ verwaltet jeweils einen Zentralkartenstapel
- intern verwaltet als Liste von Karten (LinkedList)
- Methoden zum Ziehen der obersten Karte, Zurücklegen einer Karte unter den Stapel sowie Mischen (fehlt: Auswahl einer beliebigen Karte)



Karten: Kartenstapel

- Beim Initialisieren des Spiels werden Kartenstapel regelkonform angelegt:
  - Anlegen der Stapel u. Karten laut Regelheft
  - Ablegen der Karte auf („unter“) dem entsprechenden Stapel
  - zuletzt Mischen des Stapels
  - Ausbaukarten: werden nach Mischen gleichmäßig auf 5 Stapel verteilt
- Verwaltung aller 10 Zentralkartenstapel in einer HashMap „cardDecks“ in der Spielklasse

Spiellogik

- Spiellogik wird in der Klasse „Settler“ im Package „game.Settler“ festgelegt
- abgeleitet von „Game“
- steuert den gesamten Spielablauf, vom Initialisieren des Spiels über Ablauf der Spielzüge und Ereignisse bis hin zum Spielende
- sehr groß aufgrund des umfangreichen Regelwerks des Spiels

Spiellogik

- Im Konstruktor der Klasse wird das Spiel initialisiert. Dazu gehört:
  - Anlegen der Fürstentümer → „Princedom“
  - Bereitstellen der Zentralkartenstapel
  - Ziehen von Handkarten für beide Spieler

Spiellogik

- Es werden alle Spielregeln überwacht und ausgeführt. Dazu gehören u.a.:
  - Überwachung der Spielzüge (auch bei „Konteraktionen“ des Gegenspielers)
  - Ausführen von Würfelereignissen
  - Bau von Siedlungen, Gebäuden u. Einheiten
  - Ausführen von Aktions- und Ereigniskarten
  - Setzen von Ritter- und Handelsmacht
  - Auffüllen der Handkarten auf die erlaubte Anzahl

Spiellogik

- Spielaktionen, die ein Fürstentum verändern oder beeinflussen, sind in die Klasse „Princedom“ ausgelagert und werden über die Fürstentum-Objekte der beiden Spieler in der „Settler“-Klasse ausgeführt.

Das Fürstentum

- Klasse Princedom im Package game.settler
- Klasse Settler besitzt für jeden Spieler 1 Princedom als Member
- Umfangreichste Klasse im Projekt, da hier der größte Teil der Spiellogik stattfindet

Das Fürstentum

- Memberüberblick:
  - fieldCards: HashMap für alle Karten, die auf dem Spielfeld liegen
  - handCards: HashMap für die Karten, die der Spieler auf der Hand hat
  - Sieg-, Stärke, Handels- und Turnierpunkte
  - Handels- und der Rittermacht

Das Fürstentum

- Methodenüberblick:
  - Verwaltungsmethoden für das Hinzufügen und Entfernen der Feld- und Handkarten
  - Regeln der spielrelevanten Punkte
  - Verwaltung der Ressourcen (Kaufen von Karten, Erhöhen/Vermindern der Rohstoffe)
  - logische Methoden für das Auffinden von Karten (z.B. anhand Name oder Position)

Das Fürstentum

- Auffinden von Feldern abhängig vom Ereignis (z.B. leere Felder für Kartenanbau, erhöh-/ verminderbare Ressourcen, Positionen für Upgrades, etc…)
- Ausführen von Ereignissen, welche sich direkt auf Ressourcen,Punkte oder Karten auswirken (Raubüberfall, Upgrade)

Ausblick

- weitere Implementierung von Ereignissen
- Verbesserungen an der GUI
- Performanceverbesserungen