Institut für Informatik
Lehr- und Forschungseinheit für Theoretische Informatik
Dr. Thorsten Altenkirch, Dr. Ralph Matthes
Thomas Anderl, Alessandro Macri

Programmierpraktikum
WS 1999/00

---

4. Milestone (Siedler von Catan)

12. 1. 2000

Deadline: 26.1.2000

Dieser *Milestone* gliedert sich in zwei Schritte:

1. **Graphik.** Es soll eine Klasse `CatanView` implementiert werden, die eine graphische Repräsentation der Insel anzeigt. Es soll möglich sein, mehrere *Views* einer Insel zu erzeugen (für jeden Spieler eine).

2. ***Remote Method Application.*** Die Klasse `CatanView` soll in das spielerseitige Klientprogramm integriert werden, das mit Hilfe von Java RMI auf den im Server abgelegten Zustand der Insel zugreift.

Das Ziel des *Milestones* ist, Klienten zu realisieren, die über das Netz auf den Spielzustand zugreifen. Diese Version kann zusammen mit der ASCII-Schnittstelle vom vorigen *Milestone* getestet werden. Eine graphische Eingabe der Spielzüge wird in diesem *Milestone* noch nicht verlangt.

Hinweise zum 1.Schritt

Wir folgen hier dem Konzept *Model-View-Control*, das auch Java *Swing* zugrundeliegt. Hierbei werden die verschiedenen Aspekte einer graphischen Komponente voneinander getrennt:

Model Das Datenmodell, das wir im 2. Milestone implementiert haben.

View Die graphische Anzeige, die in diesem Milestone mittels `CatanView` realisiert wird.

Control Die Spielkontrolle, die im 3. Milestone zunächst mittels einer ASCII-Schnittstelle realisiert wurde.

Die Realisierung dieses Schrittes benötigt natürlich auch Erweiterungen in der Klasse `Catan`, z.B. durch Implementierung bzw. Modifikation folgender Methoden:

- `void addView(CatanView cv)`
  Es wird eine neue View eingetragen, d.h. ein weiterer Klient kann den Spielzustand sehen. In diesem Schritt muß der Klient **noch** von derselben *Java Virtual Machine* ausgeführt werden.

- `setSiedlung`, `setStadt`, `setStrasse`
  Diese Methoden müssen alle *Views* von der Veränderung des Inselzustandes informieren.

Der Konstruktor von `CatanView` sollte als Argument eine Instanz von `Catan` erhalten und `addView` für dieses Objekt aufrufen. Dann soll eine Methode `update` aufgerufen werden, mit der die View den gegenwärtigen Zustand der Insel, also sowohl statische Informationen, wie Typ und Wert der Felder, als auch dynamische Informationen, wie Siedlungen und Straßen abfragt. Die View soll diese Daten intern abspeichern, d.h. bei einem *repaint* muß der Klient **nicht** auf die Insel zugreifen!

Hinweise zum 2.Schritt

Java RMI erlaubt einem Objekt, das in einer *Java Virtual Machine* (JVM) erzeugt wurde, Methoden von Objekten, die sich auf anderen JVMs befinden, aufzurufen. Dabei erspart man sich, selber ein Protokoll zu entwerfen. Die Argumente der *remote method invocations* werden serialisiert und über das Netz übertragen. Zur Einführung lese man sich den *Trail RMI* des *Java Tutorials* durch. Weitere Beispiele zur Verwendung von RMI aus Kapitel 15 des Buches *Java Examples in a Nutshell* findet man lokal unter *Examples (15-1 bis 15-7).*

Nach Rücksprache mit dem *Coach* kann in begründeten Fällen von der Spezifikation abgewichen werden. Sinnvolle Erweiterungen sind natürlich immer willkommen.