PM1/Projekt 3

Aufgabe

Programmieren Sie eine Textversion des Spiels «Siedler von Catan» (Englisch: Catan).

Spielregeln und Anpassungen für Projekt 3

Spielaufbau

Das Regelbuch des Spiels ist online verfügbar:
https://www.catan.de/files/downloads/4002051693602_catan_-_das_spiel_0.pdf

Das Spiel ist grundsätzlich in drei Phasen eingeteilt:

1. Variabler Spielaufbau (Spielfeld)
2. Gründungsphase (2 Runden)
3. Eigentliches Spiel

Anpassungen am Spielumfang

Wie Sie beim Studium der Anleitung feststellen werden, ist das Spiel relativ umfangreich. Für die zu entwickelnde Version dürfen Sie deshalb alles weglassen, was die folgenden Spielelemente betrifft:

- Ereigniskarten
- Ritter
- Räuber
- Handeln mit anderen Spielern

Optional können Sie noch die folgenden Elemente weglassen, dies führt zu einer Reduktion der erreichbaren Punktzahl (siehe Abschnitt "Software: Erweiterte Funktionalität")

- Städte (d.h. Sie implementieren nur Siedlungen)
- Bestimmung der längsten Strasse

Anpassungen an den Spielregeln

Weitere Anpassungen betreffen die ersten Spielphasen. Anstelle der in der Spielanleitung beschriebenen Einsteigervariante soll es einen dynamischeren Einstieg geben, indem Sie eine Kombination aus Einsteigervariante und der Variante für Fortgeschrittene umsetzen, wie im Folgenden beschrieben.

Phase 1: Variabler Spielaufbau

Beim variablen Spielaufbau wird das Spielfeld zufällig aus Landfeldern zusammengestellt. Auf diesen werden dann Zahlenchips verteilt, die mögliche Augenzahlen beim Würfeln mit zwei Würfeln darstellen. Für die PM1-Version des Spiels dürfen Sie für diese Phase den fixen Spielaufbau auf Seite 4 der Anleitung verwenden. Diesen fixen Spielaufbau finden Sie auch am Ende der Aufgabenstellung abgedruckt. Dies jedoch als Textversion mit überlagertem Koordinatennetz.

Phase 2: Gründungsphase

In der Gründungsphase werden erste Siedlungen und Strassen platziert (zwei pro Spieler) und es erfolgt eine erste Auszahlung von Rohstoffen. In Ihrer Version soll diese Phase wie folgt ablaufen:

1. Eingabe der Anzahl Spieler [2-4]
2. Jeder Spieler kann der Reihe nach die eine seiner Siedlungen platzieren, sowie eine Strasse an die Siedlung anlegen. Die Regeln für das Setzen von Siedlungen und Strassen (z.B. Abstandsregel) müssen auch in der Gründungsphase eingehalten werden.
3. Nun platzieren nochmals alle Spieler der Reihe nach eine Siedlung plus Strasse, diesmal jedoch in umgekehrter Reihenfolge. Der Spieler, der vorhin zuletzt die Siedlung plus Strasse platziert hat, darf nun zuerst ziehen.
4. Jeder Spieler erhält sofort nach der Gründung seiner zweiten Siedlung seine ersten Rohstofferträge: Für jedes Landfeld, das an diese zweite Siedlung angrenzt, erhält er einen entsprechenden Rohstoff vom Vorrat der Bank.

Phase 3: Spielphase

Der Startspieler (derjenige, der wie oben beschrieben als Letzter seine zweite Siedlung gegründet hat) kommt nun als erster zum Zug. Danach folgen die anderen Spieler in fixer Reihenfolge.

Ein Zug läuft gemäss dem nebenstehend gezeigten Fluss-diagramm (Abbildung 1) ab.

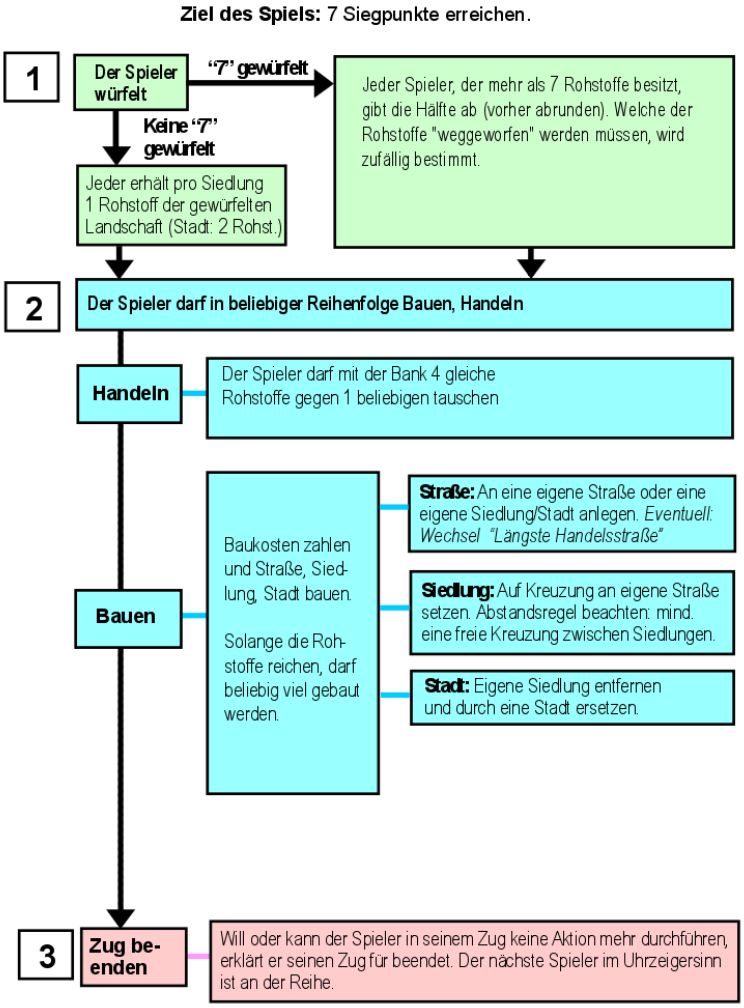

Abbildung 1: Spielzug

Codebasis

Da die Aufgabe trotz Vereinfachungen immer noch sehr umfangreich ist, müssen Sie für dieses Projekt nicht bei Null beginnen. Wir liefern Ihnen bereits einiges an Code mit. Das Klassendiagramm dieses Grundgerüsts sehen Sie in Abbildung 2 Klassendiagram der Ausgangslage.

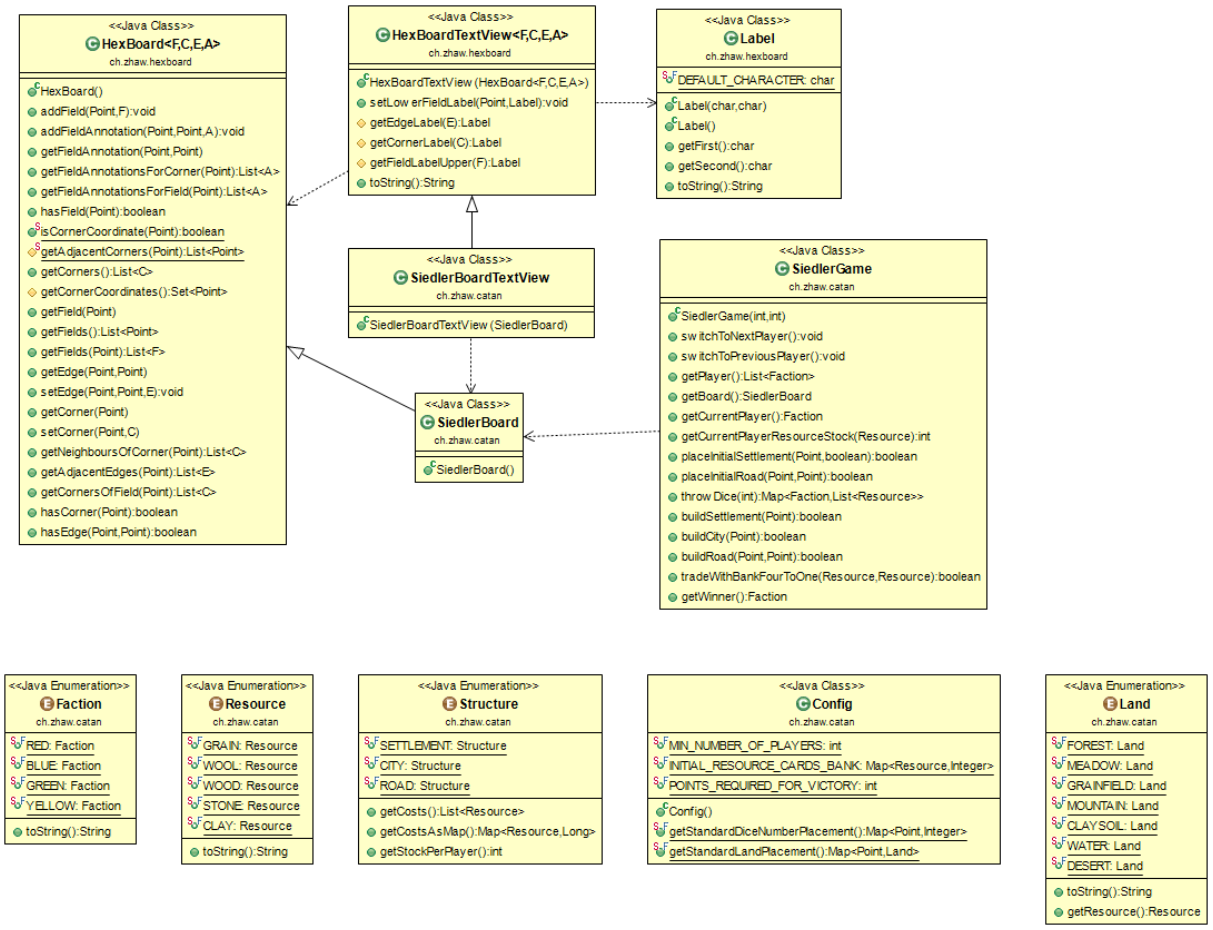

Abbildung 2 Klassendiagram der Ausgangslage

Das Grundgerüst bringt insbesondere folgendes mit:

- **text-io-3.3.0.jar (nicht im Klassendiagramm erfasst):**
  - Eine Bibliothek mit nützlichen Funktionen für die Programmierung von textbasierten Programmen. (Siehe https://text-io.beryx.org/releases/latest/#_textio)
- **ch.zhaw.hexboard:**
  - Package mit Code für die textuelle Darstellung des Siedler Spielbretts

Diese beiden Komponenten **dürfen Sie im Projekt nicht anpassen oder verändern**.

Eine weitere Komponente ist das Package **ch.zhaw.catan**. Dieses soll den Code für das eigentliche Spiel enthalten. Einige wenige Klassen (z.T. nur deren Gerüst), liefern wir bereits mit:

- **Config.java:**
  - Diese Klasse enthält einiges an nützlichen Konstanten, Enums und Methoden. So z.B. wie viele Strassenelemente jedem Spieler zur Verfügung stehen oder einen Enum für

die Landtypen und Ressourcentypen die es gibt u.v.m. **Diese Klasse darf nicht verändert werden.**

- **SiedlerGame:**
  - Die **öffentliche Schnittstelle** dieser Klasse ist gegeben und **darf nicht verändert werden**. Die Implementation der Schnittstelle ist jedoch Ihre Aufgabe.
- **SiedlerBoard:**
  - Diese Klasse erbt von der Klasse HexBoard und ist aktuell noch leer. Hier können Sie die für ein Siedler Spielbrett nützliche/spezifische Ergänzungen reintun, z.B. die Speicherung der Zuordnung der Zahlenchips zu den Landfeldern. Im Klassenkopf können Sie zudem festlegen, welche Datentypen die Ecken (Corner), Kanten (Edge) und Felder (fields) auf dem HexBoard speichern können.
- **SiedlerBoardTextView:**
  - Diese Klasse erbt von der Klasse HexBoardView und ist aktuell noch fast leer. Hier können Sie die für die Darstellung eines Siedler Spielbretts nützliche/spezifische Ergänzungen reintun, z.B. Code der dafür sorgt, dass die Werte der Zahlenchips in der Textrepräsentation des Spielbretts enthalten sind.

Schliesslich gibt es noch die Klasse **Dummy**. Dummy.java enthält eine kleine Demo-Anwendung, die einige nützliche Funktionen resp. Anwendungen des mitgelieferten Codes demonstriert.

Ausgehend von dieser Ausgangslage müssen sie folglich «nur» das Grundgerüst vervollständigen. Wir erzeugen so eine Situation, die etwas praxisnäher ist: Sie müssen auch Code verstehen und nutzen, der von Dritten geschrieben wurde.

Anmerkungen zur Umsetzung

- Es reicht, wenn Sie, falls eine Aktion eines Spielers nicht ausführbar ist (z.B. zu wenig Ressourcen oder die Platzierung einer Siedlung auf einer bereits existierenden Siedlung), ausgeben, dass die Aktion nicht ausführbar war. Woran es lag, muss nicht ersichtlich sein.
- Sie können alle in der Vorlesung "Programmieren 1" bis und mit Semesterwoche 12 durchgenommenen Konstrukte benutzen.
- Sie dürfen weiter alle Funktionalität der mitgelieferten Bibliothek TextIO nutzen sowie alles in java.util. Es lohnt sich, die vielen Möglichkeiten von Listen, Maps etc. genauer anzuschauen. Ein gutes Beispiel ist die merge Methode von Maps: `mymap.merge(key, myInteger, Integer::sum);` addiert bei vorhandenem Integer Wert für den gegebenen Schlüssel den Wert myInteger hinzu. Bei nicht vorhandenem Wert fügt es den Wert myInteger in die Map ein.
- Nutzen Sie für die Entgegennahme von Befehlen ein mehrstufiges Vorgehen. z.B. zuerst den Befehl (z.B. SETTLEMENT). Danach fragen Sie nach der x-Position und dann nach der y-Position.
- Nutzen Sie die Coaching-Möglichkeit – Ihr Betreuer/Coach berät Sie gerne bei der Umsetzung.

Vorgehen

Gehen Sie stufenweise vor:

- Laden Sie die ZIP-Datei mit dem Grundgerüst des Projekts herunter und checken Sie den Inhalt in ein für das Projekt 3 analog zu den vorderen Projekten neu erstellen Repository ein. Stellen Sie sicher, dass die beiden Bibliotheken im Unterverzeichnis /lib im Projekt in Ihrer Entwicklungsumgebung korrekt eingebunden sind.
- Stellen Sie in der Gruppe die grundlegenden Anforderungen zusammen, die sich aus der Aufgabenstellung und der Spielanleitung ergeben. Entscheiden Sie, ob und wann Sie die optionalen Anforderungen zu erweiterten Softwarefunktionalität (Städte, längste Strasse) umsetzen.
- Einigen Sie sich in der Gruppe auf einen Lösungsansatz und ein Klassenmodell.
- Verteilen Sie Verantwortlichkeiten. Eine einfache Variante wäre, dass jedes Gruppenmitglied für eine bis zwei Klassen verantwortlich ist, andere Aufteilungen sind aber denkbar.
- Erarbeiten Sie die Klassendefinition und setzen Sie diese um.

Ablauf

Das Projekt dauert von SW10 bis SW12. Ihre Dozierenden halten ein Kick-Off für das Projekt zum normalen Termin für das Modul "Software-Projekt 1" in SW10 und geben den für Ihre Klasse gültigen Abgabetermin bekannt.

Abgabe

- Der von Ihnen erarbeitete Code muss zur Abgabe bis zur Deadline auf GitHub hochgeladen werden. (Laden Sie bitte keine Binärdateien hoch.)
- Informieren Sie Ihren Betreuer per E-Mail über die URL des verwendeten Git-Repos. Dies hat keine Auswirkungen auf die Abgabefrist, Sie können bis zuletzt Updates einspielen.

Bewertung

Für diese Aufgabe erhalten Sie max. 20 Punkte. In die Bewertung fliessen die folgenden Kriterien ein:

Allgemeine Anforderung (all-or-nothing)

Voraussetzung für Punkterteilung: Das Programm ist lauffähig. Kriterium: Vorführung oder Test durch Betreuer. Ein nicht lauffähiges Programm erhält 0 Punkte.

Software: Grundfunktionalität (10 P.)

- Das Programm besitzt die geforderte Funktionalität. Kriterium: Vorführung oder Test durch Betreuer.
- Sie halten die Vorgaben hinsichtlich einsetzbarer Konstrukte und Clean Code ein und Ihr Code ist sauber dokumentiert. Kriterium: Codeanalyse durch Betreuer, ggf. Erläuterung.
- Sie haben eine sinnvolle Aufteilung in Klassen und Klassendefinitionen gefunden. Dokumentation der Klassenhierarchie ist mit hochzuladen. Kriterium: Analyse durch Betreuer, ggf. Erläuterung.
- Sie haben sinnvolle Test Cases für die Klasse SiedlerGame.java definiert und diese erfolgreich ausgeführt. Kriterium: Dokumentation über Test Cases und erfolgreiche Tests, diese sind mit hochzuladen.

Software: Erweiterung «Städte» (2 P.)

- Mittels Vererbung realisierte Funktionalität für Städte

Software: Erweiterung «Längste Strasse» (4 P.)

- Funktionalität zum Bestimmen der längsten Strasse auf dem Spielfeld

Projekt (4 P.)

- Ihr Vorgehen war planmässig, Fortschritte und Änderungen wurden strukturiert beschlossen und dokumentiert. Kriterium: Nachvollziehbare Dokumentation vorhanden zu Problemen, Beschlüssen und Änderungen. Dokumente, wie Pläne und Meeting Minutes sind auf GitHub mit hochzuladen.
- Alle Gruppenmitglieder haben gleichermassen Code beigetragen und auf GitHub eingecheckt. Eine einfache Variante wäre, dass jedes Gruppenmitglied für bestimmte Klassen verantwortlich ist, andere Aufteilungen sind aber denkbar. Kriterium: Check durch GitHub Log, Dokumentation im Code.

Anhang: Spielfeld mit überlagertem Koordinatennetz