PM1/Projekt 3

Aufgabe

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

Ablauf

Ihre Dozierenden halten ein Kick-Off für das Projekt zum stundenplanmässigen Termin für das Modul "Software-Projekt 1" und geben den für Ihre Klasse gültigen Abgabetermin bekannt.

Vorgehen

Wir empfehlen folgendes Vorgehen für dieses Projekt:

Schritt 1: Spielmechanik erlernen, Unterlagen studieren, Arbeitsumgebung einrichten

- Machen Sie sich alle mit dem Spiel vertraut. Am besten spielen Sie dazu gemeinsam oder allein eine (oder auch mehrere) Partien Siedler. Falls niemand im Team ein Siedler Spiel besitzt oder ein physisches Zusammenkommen nicht möglich ist, können Sie auch die online Version des Spiels nutzen. (Link s.u.) Dieses kommt auch mit einem Tutorial, dass anhand einer Partie Siedler in die Spielmechanik einführt.
- 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 (s. Abschnitt Codebasis) im Projekt in Ihrer Entwicklungsumgebung korrekt eingebunden sind.
- Stellen Sie in der Gruppe die grundlegenden Anforderungen zusammen, die sich aus der Aufgabenstellung und der Spielmechanik ergeben. Machen Sie einen Zeitplan, wann Sie welche Teilaufgabe angehen und umgesetzt haben wollen. Entscheiden Sie zudem, ob Sie alle Teilaufgaben lösen wollen – es ist sicher besser Teilaufgaben 1 und eine weiter Teilaufgabe gut zu lösen, als drei Teilaufgaben schlecht zu lösen.

Schritt 2: Lösungsansatz, Klassenmodell und Arbeit an Teilaufgabe 1

- 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.

Schritt 3: Abschluss Teilaufgabe 1 inkl. Tests und Beginn mit Arbeit an weiteren Teilaufgaben

Schritt 4: Abschluss der Arbeiten an den anderen Teilaufgaben (inkl. Tests) und Abgabe

Spielregeln und Anpassungen für Projekt 3

Spielaufbau

Das Regelbuch des Spiels ist online verfügbar:
https://www.catan.de/sites/prod/files/2021-06/CATAN_DasSpiel_Spielregel.pdf
https://www.catan.de/sites/prod/files/2021-06/CATAN_DasSpiel_zuguebersicht.pdf

Eine online Version des Spiels finden Sie hier:

https://catanuniverse.com/game/

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
- Handeln mit anderen Spielern
- Häfen

Je nachdem ob Ihr Projektteam die Teilaufgaben 2 und 3 angeht, wird weitere Funktionalität fehlen:

- Städte (d.h. das Spiel unterstützt nur Siedlungen)
- Spielmechanik bezüglich der längsten Strasse oder des Räubers

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.

Das Spiel endet, wenn jemand N Punkte erreicht hat. Sofern Sie eine Version ohne die Elemente *Städte* oder *längste Strasse* machen, ist N=5. Ansonsten N=7.

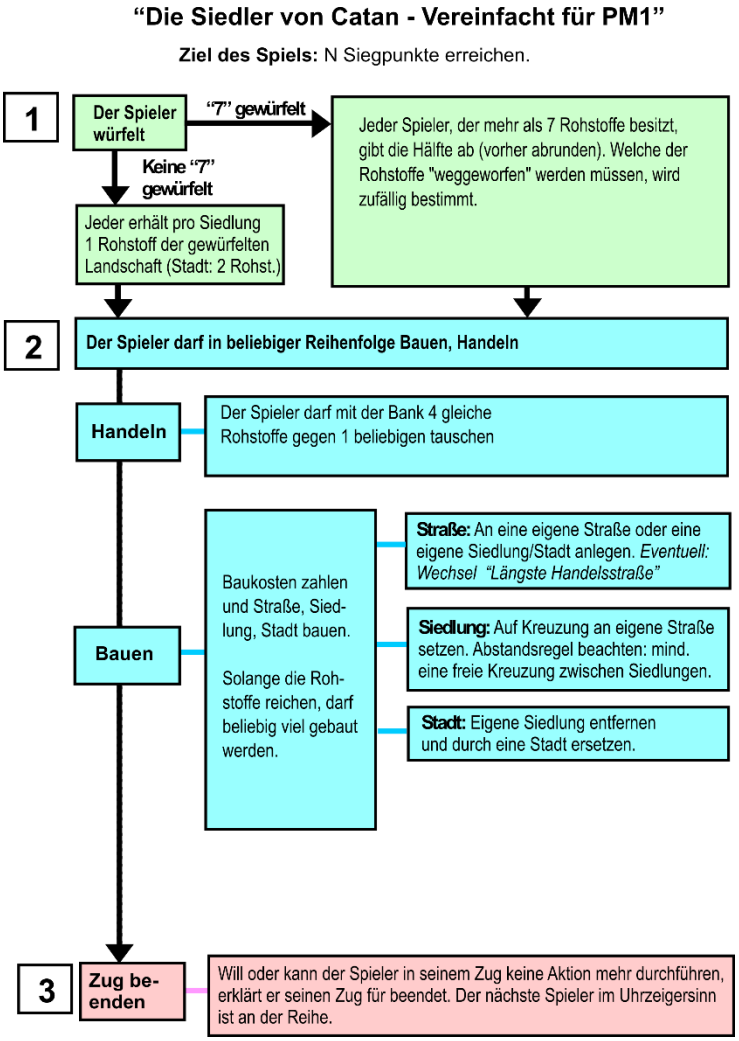

Abbildung 1 Spielablauf

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-3.4.0 (nicht im Klassendiagramm erfasst):**
  - Eine Bibliothek mit nützlichen Funktionen für die Programmierung von textbasierten Programmen. Siehe auch https://text-io.beryx.org/releases/latest/#_textio
- **slf4j-api-2.0.0-alpha1:**
  - Eine Bibliothek, die von der TextIO Bibliothek benötigt wird.
- **ch.zhaw.hexboard:**
  - Package mit Code für die textuelle Darstellung des Siedler Spielbretts

Diese Komponenten **dürfen Sie im Projekt nicht anpassen oder verändern**. Die Bibliothek TextIO müssen Sie in Ihr Projekt in der IDE einbinden. Wie das genau gemacht werden muss, hängt von Ihrer IDE ab. Abbildung 3 Einbindung von Bibliotheken zeigt, wo Sie das für die Eclipse IDE machen.

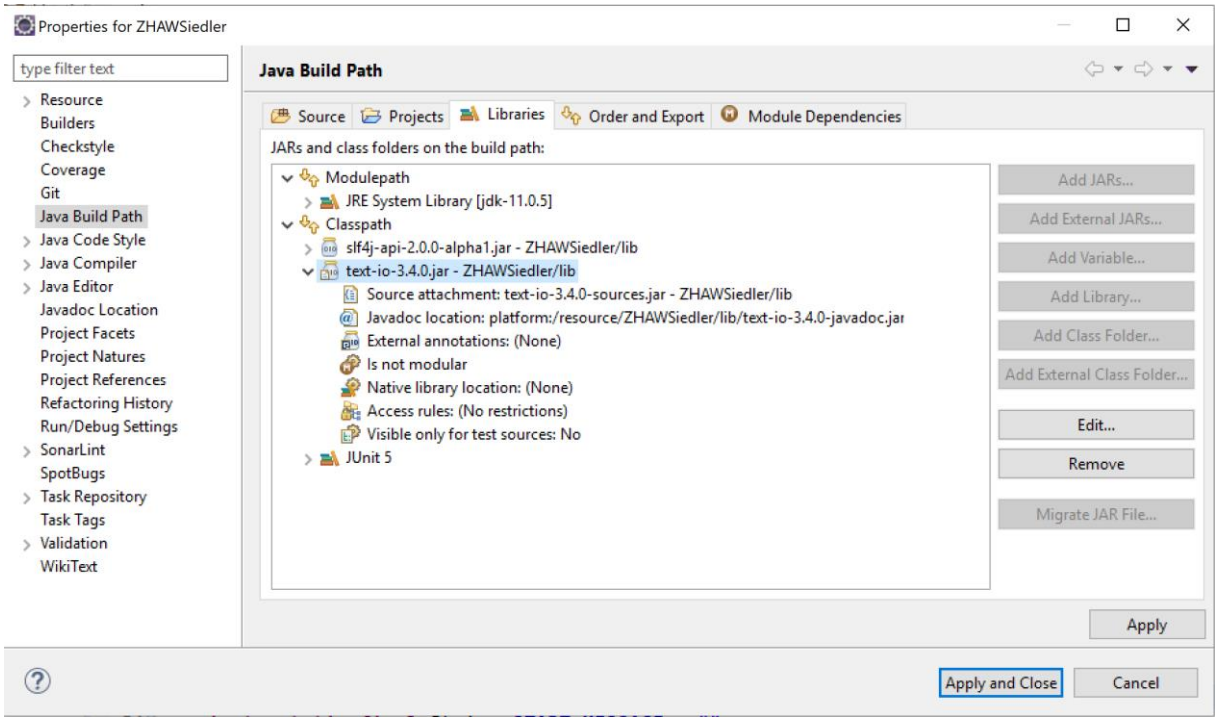

Abbildung 3 Einbindung von Bibliotheken

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 nützliche Konstanten, Enums und Methoden. U.a., wie viele Strassenelemente jedem Spieler zur Verfügung stehen oder einen Enum für die Land- und Ressourcentypen die es gibt. **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.
  - Insbesondere muss noch folgendes sichergestellt sein:
    - Wird ein SiedlerGame Objekts erzeugt, muss dabei auch das zum Spielen notwendige SiedlerBoard erzeug und gemäss Phase I initialisiert werden.
    - Beim Aufruf der Methoden für Spielaktionen, wie z.B. buildRoad(), müssen alle zugehörigen Anpassungen am Spielzustand vorgenommen, resp. ausgelöst werden (z.B. Anpassen der Ressourcen der Spielenden und der Bank).
- **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.

Neben dem Code im src Ordner, finden Sie im test Ordner die Klasse SiedlerGameTestBasic. Diese enthält bereits einige Unit Tests, die mit Ausnahme des Tests requirementBuildCity alle erfolgreich sein müssen um die allgemeinen Anforderungen (all-or-nothing) zu erfüllen. Ihre eigenen Tests können Sie in der Klasse SiedlerGameTest schreiben. Schauen Sie sich auch die Klasse **ThreePlayerStandard** im Package games im test Ordner an. Diese könnte hilfreich sein, da diese einige nützliche Spielsituationen/Spielzustände generieren kann, die Sie als Ausgangslage für Tests nutzen können.

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 zum Abgabetermin 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 berät Sie gerne bei der Umsetzung.

Abgabe

Der von Ihnen erarbeitete Code muss zur Abgabe bis zur Deadline auf GitHub hochgeladen werden. (Laden Sie bitte keine Binärdateien hoch.)

Bewertung

Für diese Aufgabe erhalten Sie max. 30 Punkte. In die Bewertung fliessen die folgenden Kriterien ein:

Allgemeine Anforderung (all-or-nothing)

Voraussetzung für Punkterteilung:

- Das Programm ist lauffähig. Ein nicht lauffähiges Programm erhält 0 Punkte.
- Die mitgelieferten Tests in der Klasse SiedlerGameTestBasic laufen durch und müssen mit Ausnahme des Tests requirementBuildCity erfolgreich sein.
- Alle Gruppenmitglieder haben substanziell zum Code beigetragen und auf GitHub eingecheckt. Eine einfache Variante wäre, dass jedes Gruppenmitglied für bestimmte Klassen verantwortlich ist, andere Aufteilungen sind aber denkbar.

Teilaufgabe 1: Grundfunktionalität (18 P.)

- Das Programm besitzt die geforderte Funktionalität.
- Sie halten die Vorgaben hinsichtlich einsetzbarer Konstrukte und Clean Code ein und Ihr Code ist sauber dokumentiert.
- Sie haben eine sinnvolle Aufteilung in Klassen und Klassendefinitionen gefunden. Das Klassendiagramm ist mit hochzuladen. Für das Klassendiagramm muss die in PROG1 vermittelte Notation verwendet werden. Abweichungen davon sind als Legende mitzuliefern. https://moodle.zhaw.ch/pluginfile.php/397809/mod_folder/content/0/klassendiagramm.pdf
- Sie haben sinnvolle Test Cases für die Klasse **SiedlerGame** definiert und diese erfolgreich ausgeführt. Die Dokumentation eines Test Cases umfasst mindestens folgenden Elementen: Kurzbeschrieb, Äquivalenzklasse, ob positiver/negativer Test, Ausgangszustand, Input resp. ausgeführte Aktionen und erwarteter Output resp. Zustand.

Teilaufgabe 2: Erweiterung «Städte» (5 P.)

- Mittels Vererbung realisierte Funktionalität für Städte

Teilaufgabe 3: «Längste Strasse» oder «Räuber» (7 P.)

Um die 7 Punkte zu erhalten, müssen Sie EINE der beiden Erweiterungen geeignet umsetzen:

- Implementation der Spiellogik betreffend die längste Strasse (s. Regelbuch und Almanach)
- Implementation der Spiellogik betreffend den Räuber. Im Unterschied zu den originalen Regeln, werden Ressourcenkarte und Mitspieler, bei dem der Räuber eine Ressourcenkarte stiehlt, nicht vom aktuellen Spieler ausgewählt. Beides wird zufällig ausgewählt. Für das Stehlen kommen wie beim Original, nur Mitspieler infrage, die eine Siedlung am Feld haben, wo der Räuber platziert wird.

Anhang: Spielfeld mit überlagertem Koordinatennetz