The Board With The Figures

Grundidee

Die Architektur des Spiels basiert auf dem Model-View-Controller-Pattern. Die Idee dahinter ist, den Programmcode in die drei funktional voneinander unabhängigen Pakete Ein- und Ausgabe, sowie das Datenmodell zu untergliedern und voneinander abzukapseln. Über die Eingabe steuert der Nutzer das Programm, im Datenmodell wird die Eingabe verarbeitet und der Zustand des Programms aktualisiert und über die Ausgabe dem Nutzer zurückgegeben. Der Programmstart wird im *Main*-Paket ausgelöst. Durch die Trennung erreichen wir einen übersichtlichen, klar strukturierten Code, der einfach zu warten und im späteren Entwicklungsprozess flexibel erweiterbar ist.

Das Datenmodell (*Model*)

Hier arbeitet das Backend unseres Schachspiels. Neben den Spielregeln wird im Model der Programmablauf festgelegt und der aktuelle Zustand des Spiels gespeichert.

Da alle Schachfiguren bis auf ihre Bewegungsmöglichkeiten ähnliche Eigenschaften aufweisen, haben wir uns für die Erstellung einer abstrakten Figuren-Superklasse entschieden, die ihre Attribute und Methoden an die einzelnen Figurenklassen vererbt. Dies trägt zur Übersichtlichkeit des Codes bei, verringert Redundanz und beugt potenzielle Fehlerquellen vor. Außerdem lassen sich Effizienz und Performance des Programms steigern.

Die einzelnen Figurenklassen *Knight, Pawn, King, Bishop, Queen* und *Rook* sind dann im Einzelnen für die Instanziierung der jeweiligen Figuren verantwortlich. Ihre Regeln der Bewegungsmöglichkeiten sind nach Vorlage des *Strategy*-Patters in eigene Klassen ausgelagert, da z.B. das Bewegungsmuster der Dame lediglich eine Kombination der Muster des Turms und des Bischofs ist.

Über die Klasse *Position* werden alle denkbaren Koordinaten verarbeitet und z.B. die Figuren mit ihren Koordinaten auf dem Feld verknüpft oder Zielkoordinaten verarbeitet.

Jede Figur wird dann in der Klasse *Tile* mit einem einzelnen Schachfeld verknüpft, mit seinem Kürzel versehen und in der Klasse *Board* initialisiert. Das Schachbrett besteht also aus einer Menge von Feldern, auf denen sich die Figuren befinden. Desweiteren werden in *Board*, nachdem das Spiel gestartet wurde, die Bewegungsmöglichkeiten der einzelnen Figuren berechnet und der aktuelle Zustand des Spiels gespeichert.

Parallel zum Board existiert zudem die Klasse *Cemetery*. Auf dem Friedhof werden die geschlagenen Figuren in einer Liste abgelegt und können fortan vom Nutzer abgerufen werden.

Die Spielregeln werden durch das *RulesInterface* und die Klasse *Normal* realisiert. Sollen weitere Spielmodi hinzugefügt werden bzw. nach anderen Regeln gespielt werden, kann man neben *Normal* weitere Regel-Klassen hinzufügen.

Für die Erstellung der Spieler verwenden wir die abstrakte Klasse *Player*. Ein Spieler kann sowohl ein Mensch sein, der seine Eingaben über die Computerperipherie macht als auch eine KI. Beide werden durch einzelne Klassen repräsentiert. Um gegen unterschiedlich starke Computergegner antreten zu können, wird in der Klasse *Computer* entsprechend der Auswahl des Nutzers eine KI erstellt. Als Vorlage dient die abstrakte Klasse *Intelligence*. Als Unterklasse gibt es bisher nur eine einfache KI implementiert. In einer späteren Iteration kommt noch eine weitere Unterklasse für eine intelligente KI hinzu.

Alle bisher aufgezählten Komponenten des *Model*-Pakets laufen in der Klasse *Game* zusammen. Hier werden das Board, der Friedhof und die Spieler initialisiert, unterschiedliche Spielmodi verwaltet und das Spiel gestartet und gestoppt. Außerdem bildet das *Game* die Schnittstelle aus dem *Model*-Paket zur Ein- und zur Ausgabe.

Eingabe (*Controller*)

Hier werden die Eingaben, die der Nutzer über die Konsole macht, entgegengenommen und auf Semantik und Syntax überprüft. Bei der Nutzung des 2D-GUI werden Mausklicks entsprechend der Zeigerposition klassifiziert.

Gültige Eingaben werden in der entsprechenden *Controller*-Klasse validiert und lösen Steuerbefehle aus, die Veränderungen im *Model* und in der *View* verursachen. Es können z.B. Figuren auf dem Schachbrett bewegt oder die Sprache geändert werden. Ungültige Eingaben werden abgefangen und mit entsprechenden Meldungen beantwortet.

Ausgabe (*View*)

Im Paket *View* wird das Ausgabeverhalten des Programms gesteuert und die Konsole oder das GUI als Reaktion auf die Nutzereingaben, die Veränderungen am Model auslösen, aktualisiert. Die Klasse *Console* regelt alle Konsolenausgaben des Programms, *Playground* steuert das 2D-GUI. Beide Klassen erhalten ihre gemeinsame Funktionalität aus dem *ViewInterface*. Für die Darstellung des aktuellen Zustands des Spiels stehen *Playground* und *Console* direktem Kontakt mit der Klasse *Game* aus dem *Model*-Paket.