Backgammon - Architektur

Model-View-Controller

- Grundlegendes Architekturmuster: MVC
    - Model: Abstrakte Repräsentation der Daten
    - View: UI-Elemente und Ansicht (greift auf Model zu)
    - Controller: Stellt die Verbindung zwischen View und Model, führt Abläufe wie z.B. Spielzüge aus

Model-View-Controller

- backgammon.game
  - Board
    Repräsentation des Spielbretts
  - Dice
    Würfel. Wichtig: activate() und roll()
  - Field
    Ein Feld auf dem Spielbrett
  - Move
    Ein (möglicher oder auszuführender) Spielzug
  - Piece
    Eine einzelne Spielfigur

Model-View-Controller

- backgammon.players
  - Player
    Eine Repräsentation eines Spielers
  - ArtificialPlayer
    erbt von Player, grundlegender KI-Spieler

Model-View-Controller

- backgammon.ui
  - BoardView
    Spielbrettdarstellung
  - Constants
  - DicePanel
    lässt den Spieler würfeln
  - DiceView
    zeigt das Würfergebnis an
  - FieldRegion
  - GameWindow
    organisiert die Views, zeichnet neu
  - PieceView
    zeichnet eine Spielfigur

Model-View-Controller

- backgammon.ui
  - ArtificialPlayerView
    organisiert das Zeichnen der KI-Visualisierung, verwaltet ArtificialPlayer und ThinkView
  - ThinkView
    zeichnet die KI-Visualisierung

Model-View-Controller

- backgammon.game
  - Game
    Verwaltet Spielzüge, Würfel und Spielbrett

Observer

- Grundlegendes Architekturmuster: Observer

  - Observer: Klasse die eine andere Klasse beobachtet und auf Änderungen reagiert

    in diesem Fall: Game

  - Observable: Klasse die beobachtet wird und Observer über Änderungen informiert

    in diesem Fall: Dice

Artificial Player

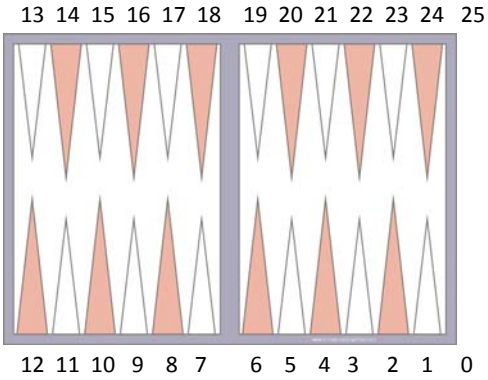
```
public class AIPlayer extends ArtificialPlayer {
    public AIPlayer() {
        AIPlayerView view = new AIPlayerView();
        this.setThinkView(view);
    }

    public void calculateMove(Dice dice, Board board) {
        // calculate the player's move here
    }

    public Object clone() {
        return new AIPlayer();
    }

    public String getDescription() {
        return "AI Player";
    }
}
```

Artificial Player View

```
public class AIPlayerView extends ThinkView {
    public void onPaint(Graphics g) {
        // draw your visualization here
    }
}
```

Spielmodell

- Board enthält ein Array von 26 Fields.
- Außerdem: darkCenterField und whiteCenterField für geworfene Steine.
- Jedes dieser Fields enthält einen Vector von Pieces.



Hilfreiche Methoden

- Move enthält die Methode

```
public static Vector<Vector<Move>> calculatePossibleMoves(Dice dice,
    Board board, boolean whiteTurn)
```

- Diese gibt einen Vector von Vector von Moves aus:
- Jeder Move ist ein elementarer Spielzug, d.h. die Bewegung eines Spielsteins von einem Feld zu einem anderen
- Der Vector von Moves ist ein möglicher Spielzug eines Spielers, der sich aus zwei oder vier (Pasch) elementaren Zügen zusammensetzt
- Der Vector von Vector von Moves schließlich ist eine Liste von möglichen Zügen

Hilfreiche Methoden

- BoardView enthält die Methode

```
public static int[] convertFieldToBoardCoordinates(int fieldId, int j)
```

- Diese Methode konvertiert abstrakte Felder auf dem Spielbrett in Bildschirmkoordinaten, z.B. zur Visualisierung