Thomas Bekman

Foo Game

Below we included the following classes found in our design and described in detail their methods and purposes. The description for Assignment3.java contains the instructions to play the game.

ABoard.java

This class contains a constructor and methods drawBoard() and updateBoard(). drawBoard() takes in no arguments and has no return type. Its purpose is to print the board to the window based on the values found in the ArrayLists of rows. updateBoard() takes in as an argument ArrayList<ArrayList<Point2D>> gameState. Its purpose is to take in the gameState and update the board's row content based on the player Point2D moves for players one and two found in gameState.

AI.java

This class implements the Player interface and contains the method move(). This method takes in as its argument ArrayList<Object> input and returns an ArrayList<Point2D> object. Its purpose is to generate an AI move. It does this by examining a list of possible moves on the board after it has been updated to contain only empty coordinates (where to there hasn't been a move). It "randomly" selects a move from this list to return.

Assignment3.java

This class is used to run the program and play the game. It creates two players, either Human or AI. Instructions will appear in a window, depending on how you run the file. The first instruction will prompt Player 1 for an input, which should be a "place" play and should be followed by the coordinate (be sure to include spaces like in the following example) you desire to move your piece to. An example is "place 3 3" for a move to (x=3,y=3). A note: the game is currently set to play two AI players against one another. If it's a draw, the program will error and the board will have been filled with pieces.

FooHistory.java

This class implements the History interface and contains the methods addMove(), getCurrentState(), and getTurnCount(). addMove() takes as argument a ArrayList<Point2D> inputMove object. Its purpose is to input a move from a player to a list of their moves. getCurrentState() returns an ArrayList<ArrayList<Point2D>> object. Its purpose is to return an ArrayList of the two ArrayLists belonging to the players that contain their moves up until that point. getTurnCount() returns an int. Its purpose is to return the total current turn number of the game.

FooRules.java
This class implements the Rules interface and contains the methods adjacent(), adjList(), checkWin(), and legalMove(). adjacent() takes as input Point2D point1, Point2D point2 and returns a boolean. Its purpose is to determine if a point is adjacent to a piece or not. adjList() takes as input Point2D point and ArrayList<Point2D> inputList and returns an ArrayList<Point2D> object. Its purpose is to construct a list of adjacent pieces on the board. checkWin() takes as an argument ArrayList<Object> Gamestate and returns a boolean. Its purpose is to determine from the current game state (arrangement of pieces on the board) if a player has won. legalMove() takes as its arguments a ArrayList<Point2D> inputMove and a ArrayList<Object> Gamestate and returns a boolean. Its purpose is to determine:

FooUserInterface.java
This class implements the UserInterface interface and contains the method requestMove(). This move takes as input ArrayList<Object> player and FooHistory GameState and returns an ArrayList<Point2D> object. Its purpose is to output console commands to the user and get user input. In the case of AI, the purpose of this is to send the AI the current game state.

GameMaster.java
This class contains the methods addPlayer() and playGame(). addPlayer() takes as input Player playerToAdd. Its purpose is to add a new player to the game. Players can be human or AI and are limited to two. playGame() takes as input input FooRules Rules, UserInterface FooUserInterface, FooHistory Gamestate, and ABoard Board. Its purpose is to call the start of the game functions.

History.java
This interface contains three methods, addMove(), getCurrentState(), and getTurnCount(). addMove() takes as argument ArrayList<Point2D> inputMove. getCurrentState() returns an ArrayList<ArrayList<Point2D>> and getTurnCount() returns an int. Its purpose is to represent an extendable game history interface.

Human.java
This class implements the Player interface and has a single method move(). It takes as input an ArrayList<Object> input and returns an ArrayList<Point2D>. Its purpose is to accept an array of Strings from UserInterface and returns the appropriate Point2D objects.

Rules.java

This interface contains two methods, checkWin() and legalMove(). checkWin() takes as argument a ArrayList<Object> Gamestate and returns a boolean. legalMove() takes as arguments a ArrayList<Point2D> inputMove and a ArrayList<Object> Gamestate and returns a boolean. Its purpose is to provide an extendable game rules interface.

UserInterface.java

This interface contains only one method, requestMove(). This method takes as input a ArrayList<Object> player and FooHistory GameState and promises to return an ArrayList<Point2D> object. Its purpose is to represent a general, extendable user interface.