Implementation of Parcheesi

Project Overview:

Our Parcheesi is organized as a server-client architecture. Each Player is organized with MVC architecture, such that independent AI players can be modularly plugged in.

Controller asks players to take their turns, facilitates communication between Board and Players. Board keeps track of pieces and spots, determines legal moves. Players take their turns, asking Controller about legal moves.

What follows is a high-level overview of our planned implementation.

Main script:
Instantiate a controller, register players and board with controller.
Launch successive game rounds until a player has won.
Display that player, and terminate.

Model consists of Controller (the server/supervisor), AIPlayer, and HPlayer.
Interface Definitions (Typescript):

Controller:

```
interface IController { … }
interface ITurn { … }

namespace Controller {
        private class Controller implements IController { … }
        private controller Controller()

        export controller
}
// facilitates interaction between board and players, keeps them siloed
// a singleton (namespace singleton pattern)

private class SPlayer
// represents a player internally for the Controller

private class Die
// represents a die

private class Dice
```

// wraps two individual dice

private class Turn implements ITurn
// represents a single turn of the game

Players:
public interface IPlayer

public abstract class Player implements IPlayer
// the common pieces of Player

public class MPlayer extends Player
// represents a machine player

public class HPlayer extends Player
// represents a human player

View:

- The actual implementation of the view consists of two layers:
    - BoardView - visual representation of board, allows all players to see current state at all times (observe movements/placements of all human and machine player pieces)
    - HView - for human players - overlaid on board during turn, lets user choose move (no graphical view specifically for machine players)

public interface IBoardView
// the board view interface

public class BoardView
// the board gui - shows current player's turn, dice roll results, layout of pieces.

public interface IHView
// the human view interface

public class HView implements IHView
// the view for human players - information and selectors for your turn options

Stories

Use cases:

<u>User starts a game.</u>

Controller creates a board and players for each color. It then plays successive rounds of the game until a player has won, displays that player, and terminates.

Playing a turn.
Controller asks the player to roll dice. Based on that result, it informs the player of all possible moves, for each of their pieces. It then allows the player to make legal moves until none are available (multiple turns are triggered only in the case of rolling doubles). Finally, it moves to the next player.

Ending the game.
Controller checks after each player's turn if they have won. Game ends once anyone has won.

Strategy for Automated Player

Roll, if dice contain 5 (or multiple 5s) and player has pieces at start, free as many as possible. If moves remain, play legal moves with forward most piece possible, moving to progressively less forward pieces as required to make all possible moves. Once out of legal moves, if doubles roll again and repeat procedure, otherwise done for the turn.