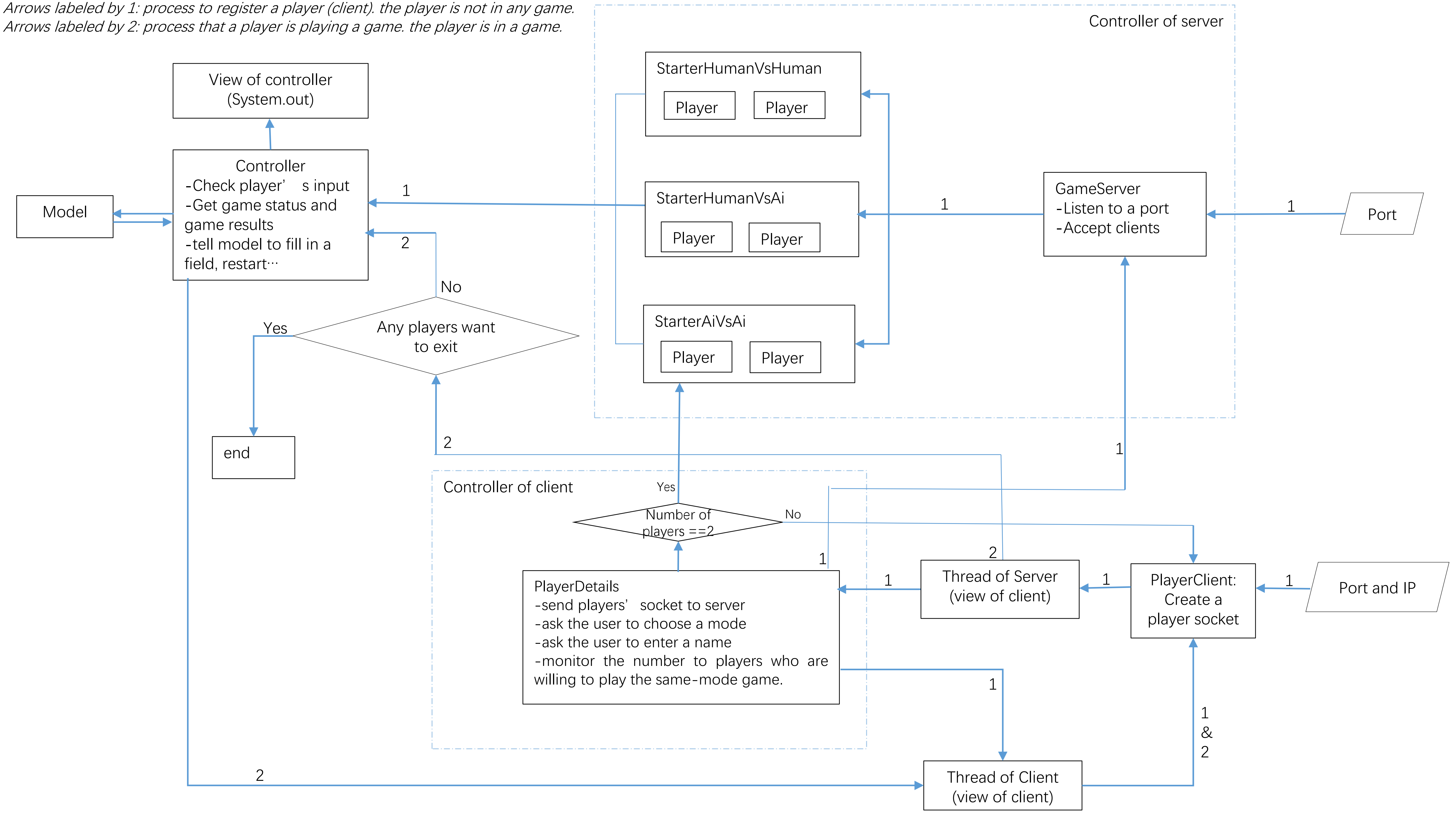
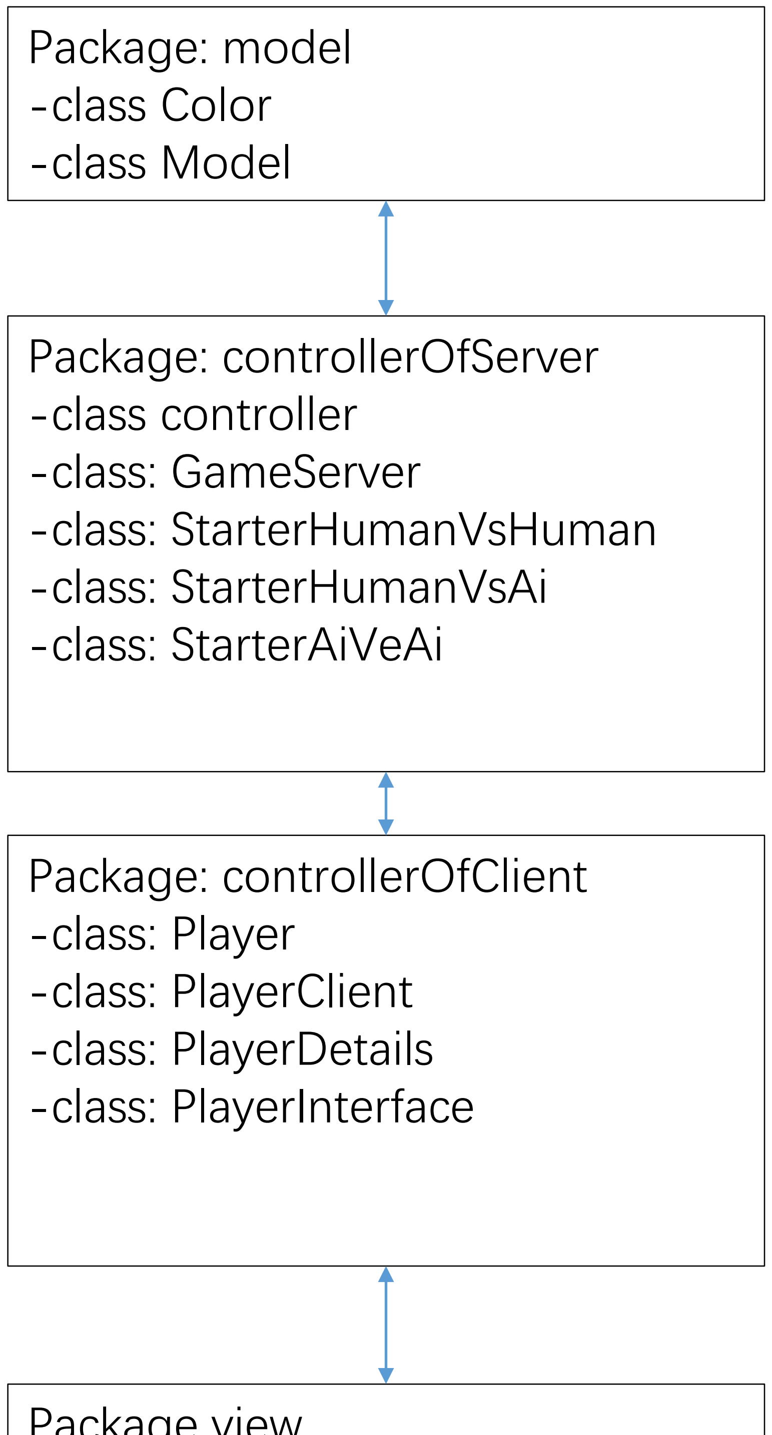
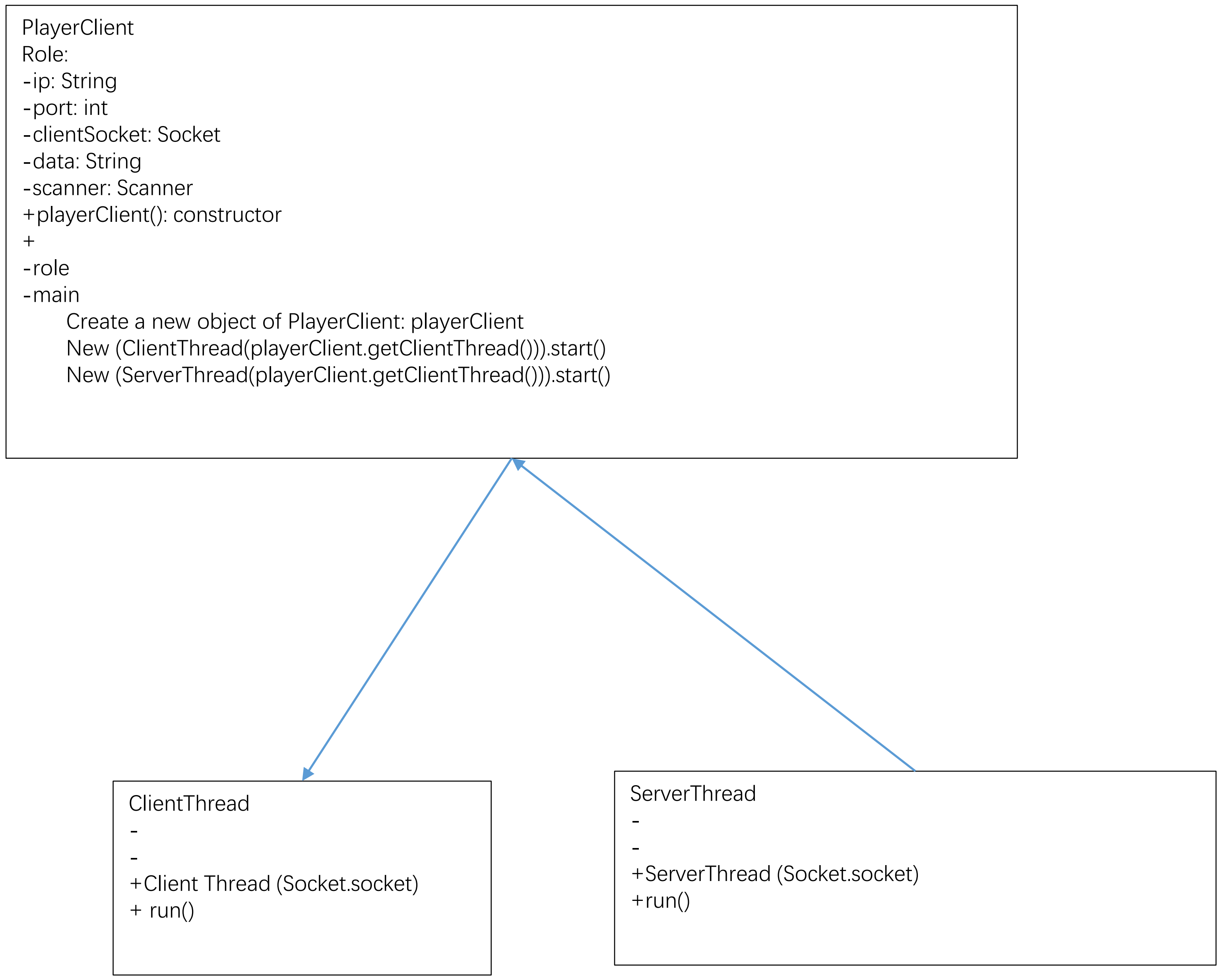
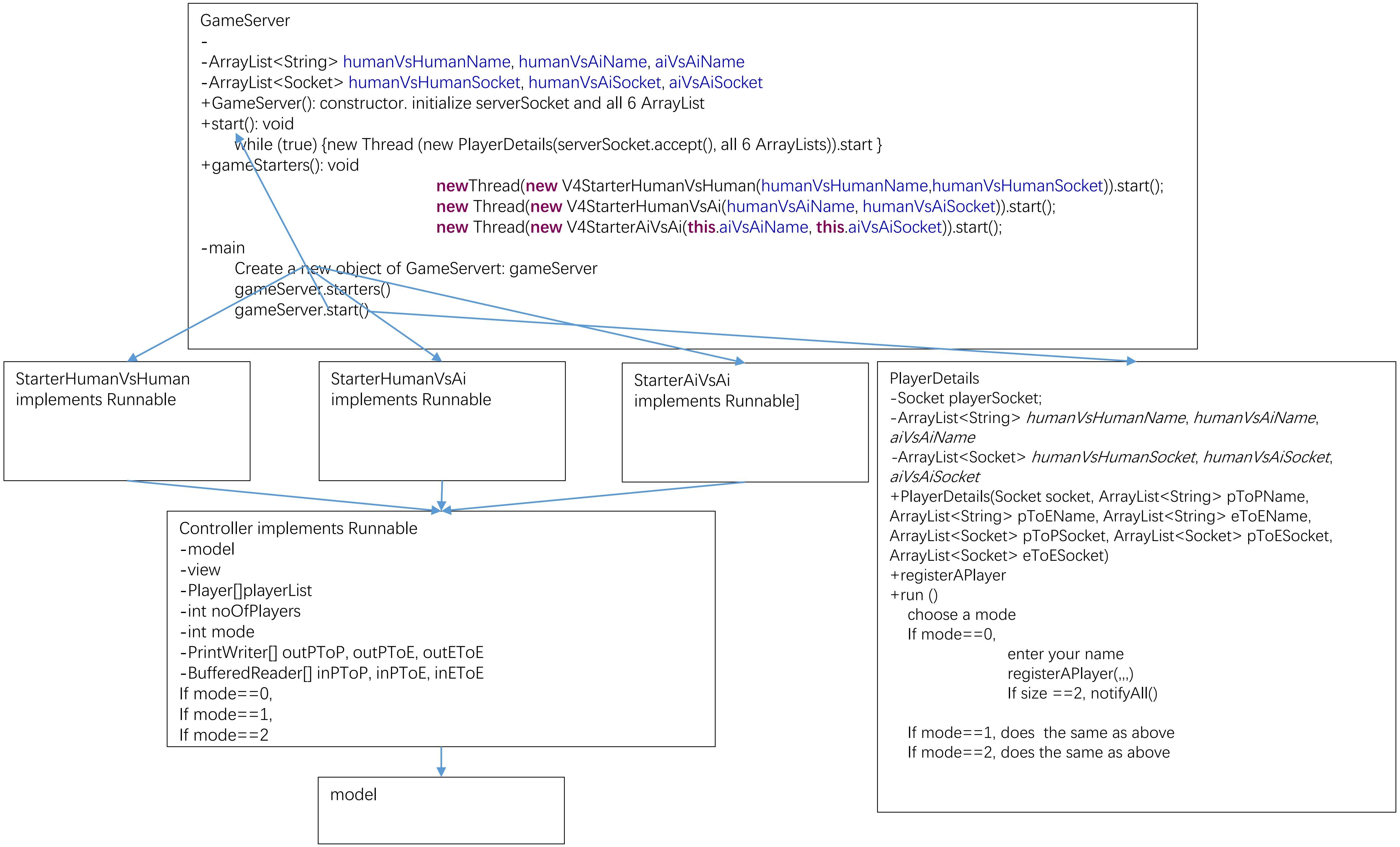
Arrows labeled by 1: process to register a player (client). the player is not in any game.
Arrows labeled by 2: process that a player is playing a game. the player is in a game.



TUI-version Four In A Row




PlayerClient
Role:
-ip: String
-port: int
-clientSocket: Socket
-data: String
-scanner: Scanner
+playerClient(): constructor
+
-role
-main
Create a new object of PlayerClient: playerClient
New (ClientThread(playerClient.getClientThread())).start()
New (ServerThread(playerClient.getClientThread())).start()
ClientThread
-
-
+Client Thread (Socket.socket)
+ run()
ServerThread
-
-
+ServerThread (Socket.socket)
+run()



```
GameServer
-
-ArrayList<String> humanVsHumanName, humanVsAiName, aiVsAiName
-ArrayList<Socket> humanVsHumanSocket, humanVsAiSocket, aiVsAiSocket
+GameServer(): constructor. initialize serverSocket and all 6 ArrayList
+start(): void
    while (true) {new Thread (new PlayerDetails(serverSocket.accept(), all 6 ArrayLists)).start }
+gameStarters(): void
        newThread(new V4StarterHumanVsHuman(humanVsHumanName,humanVsHumanSocket)).start();
        new Thread(new V4StarterHumanVsAi(humanVsAiName, humanVsAiSocket)).start();
        new Thread(new V4StarterAiVsAi(this.aiVsAiName, this.aiVsAiSocket)).start();
-main
    Create a new object of GameServert: gameServer
    gameServer.starters()
    gameServer.start()
```

```
StarterHumanVsHuman
implements Runnable
```

```
StarterHumanVsAi
implements Runnable
```

```
StarterAiVsAi
implements Runnable]
```

```
PlayerDetails
-Socket playerSocket;
-ArrayList<String> humanVsHumanName, humanVsAiName,
aiVsAiName
-ArrayList<Socket> humanVsHumanSocket, humanVsAiSocket,
aiVsAiSocket
+PlayerDetails(Socket socket, ArrayList<String> pToPName,
ArrayList<String> pToEName, ArrayList<String> eToEName,
ArrayList<Socket> pToPSocket, ArrayList<Socket> pToESocket,
ArrayList<Socket> eToESocket)
+registerAPlayer
+run ()
  choose a mode
  If mode==0,
          enter your name
          registerAPlayer(,,,)
          If size ==2, notifyAll()

  If mode==1, does  the same as above
  If mode==2, does the same as above
```

```
Controller implements Runnable
-model
-view
-Player[]playerList
-int noOfPlayers
-int mode
-PrintWriter[] outPToP, outPToE, outEToE
-BufferedReader[] inPToP, inPToE, inEToE
If mode==0,
If mode==1,
If mode==2
```

```
model
```

humanVsHumanName

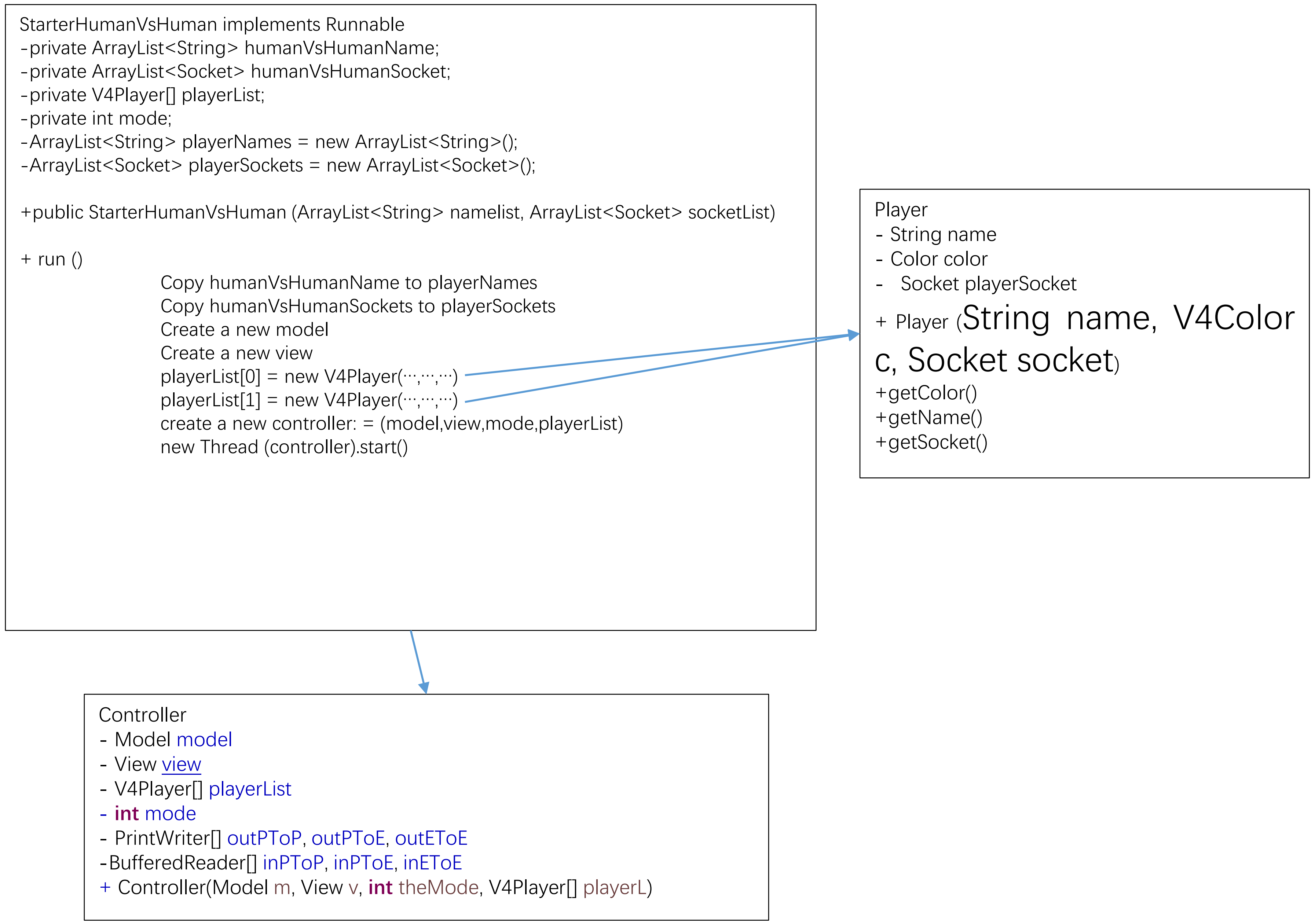
humanVsHumanSocket

humanVsAiName

humanVsAiSocket

AiVsAiName

AiVsAiNameSocket

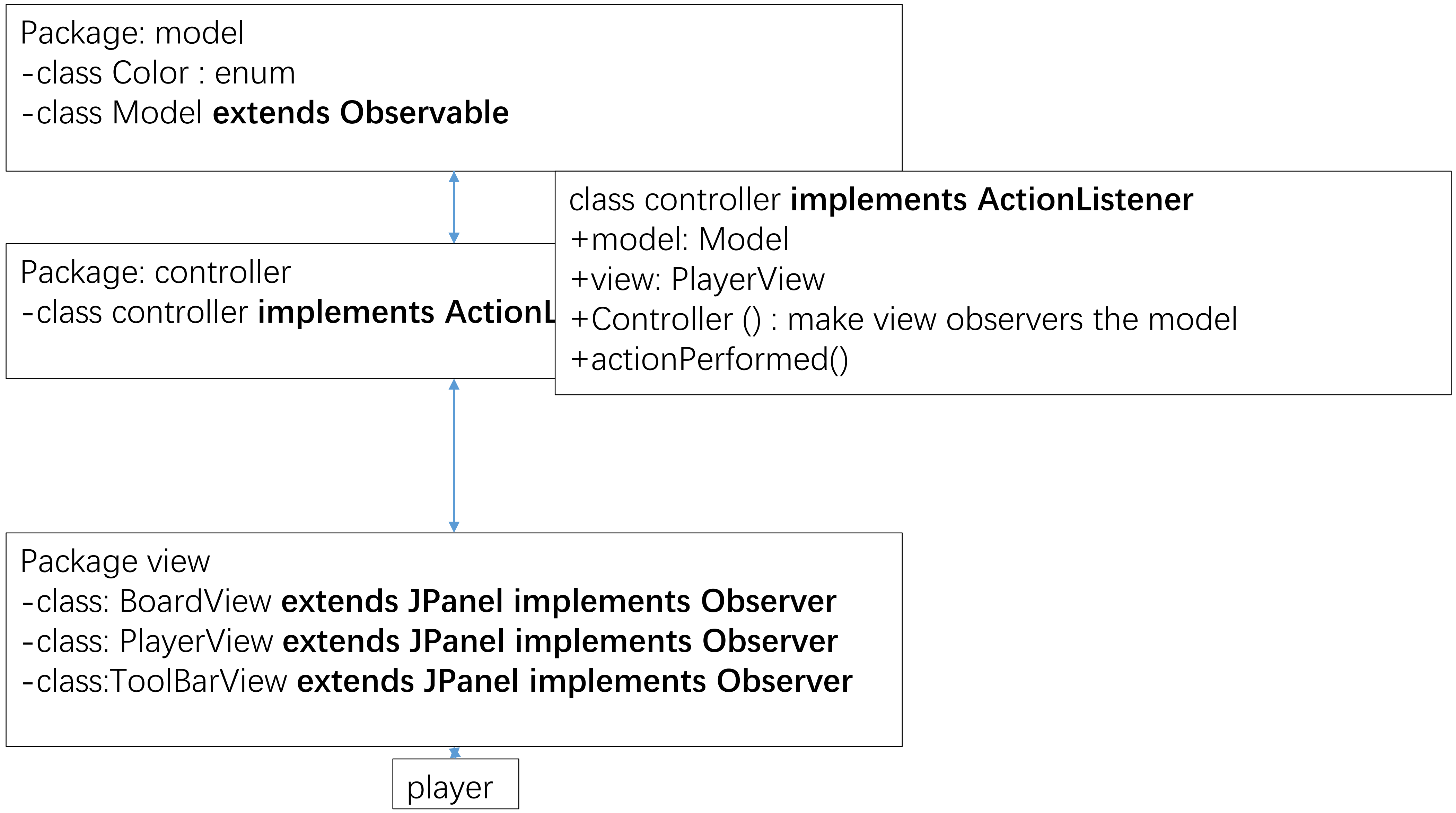


```
StarterHumanVsHuman implements Runnable
-private ArrayList<String> humanVsHumanName;
-private ArrayList<Socket> humanVsHumanSocket;
-private V4Player[] playerList;
-private int mode;
-ArrayList<String> playerNames = new ArrayList<String>();
-ArrayList<Socket> playerSockets = new ArrayList<Socket>();

+public StarterHumanVsHuman (ArrayList<String> namelist, ArrayList<Socket> socketList)

+ run ()
        Copy humanVsHumanName to playerNames
        Copy humanVsHumanSockets to playerSockets
        Create a new model
        Create a new view
        playerList[0] = new V4Player(…,…,…)
        playerList[1] = new V4Player(…,…,…)
        create a new controller: = (model,view,mode,playerList)
        new Thread (controller).start()
```

```
Player
- String name
- Color color
-   Socket playerSocket
+ Player (String  name,  V4Color c, Socket socket)
+getColor()
+getName()
+getSocket()
```

```
Controller
- Model model
- View view
- V4Player[] playerList
- int mode
- PrintWriter[] outPToP, outPToE, outEToE
-BufferedReader[] inPToP, inPToE, inEToE
+ Controller(Model m, View v, int theMode, V4Player[] playerL)
```

GUI-Version Four In a Row



Package: model
-class Color : enum
-class Model **extends Observable**

class controller **implements ActionListener**
+model: Model
+view: PlayerView
+Controller () : make view observers the model
+actionPerformed()

Package: controller
-class controller **implements ActionL**

Package view
-class: BoardView **extends JPanel implements Observer**
-class: PlayerView **extends JPanel implements Observer**
-class:ToolBarView **extends JPanel implements Observer**

player

class: BoardView **extends JPanel implements Observer**
-tiles[][]: Jbuttons
-model: Model
-foreColoer: color
-backColor: Color
-playerView:Playerview
+BoarView():
    create an array of buttons
    set the color of buttons before clicked and after clicked.
+registerControllers ():
    add ActionListener() to each Jbuttons, each Jbutton will invokes a movement in the model
+update ()
    shows the color on the button.
    If this field is occupied by red player, the button shows "red" . Otherwise, the buttons shows "blue" .
    if the game is over, shows which line has 4 pieces together.

class: TooBarView **extends JPanel implements Observer**
-newGameButtons: Jbutton
-chooseMode: Jbutton
-getHintButton: Jbutton
-guitGameButton: Jbutton
-label: Jlabel
-showHint: Jlabel
+ToolBarView():

create all the buttons, set their texts, fonts, sizes, status

create all the labels, set their texts, fonts.

+registerControllers ():

add ActionListener() to each Jbuttons, each Jbutton will invokes a reaction in the model, such as start a new game, or change mode, or get hint.

+update ()

defines how the text on the mode button should be switched, after a click.

shows message to player what to do next.

shows whether anybody wins or it is a draw

class: PlayerView **extends JPanel implements Observer**
-player1: JRadiobutton
-player2: JRadiobutton
-group: ButtonGroup
-JTextField, Jlabel, and int for setting AI thinking time
+PlayerView():
set the location and size of the PlayerView
create two buttons. Set their size and status.
the button group makes two buttons contradict to each other
+registerControllers ():
add ActionListener() to each radio button. Set player’ s turn based on which JRadioButton is clicked.
+update ()
change the player’ s name according to the mode
if player wants to start a new game, clear the player turn
if game starts, update the player turn
set “AI thinking” text when AI is thinking

Class: model
playerTurn =0: red player’ s turn
playerTurn =1: blue player’ s turn
The turn is switched by +1 or -1

Class: Main
playerTurn =0: red player’ s turn
playerTurn =1: blue player’ s turn

The turn is switched by +1 or -1