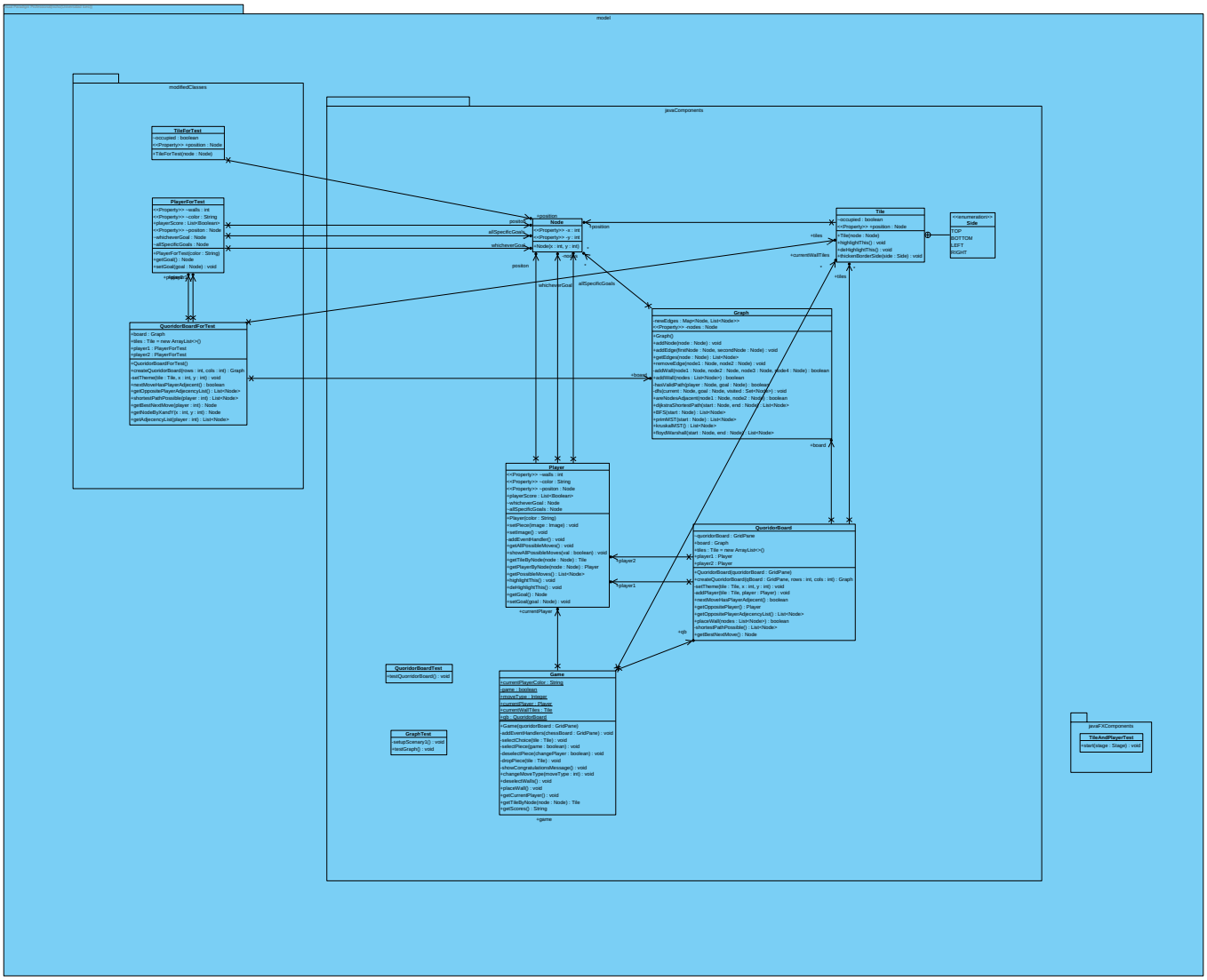
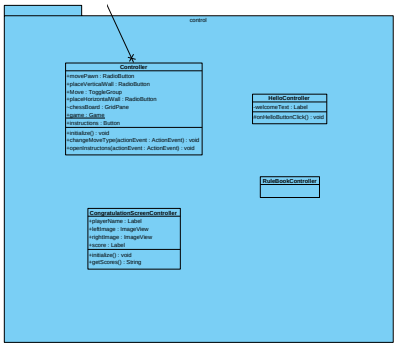

model
modifiedClasses
TileForTest
+occupied : boolean
<<Property>> +position : Node
+TileForTest(node : Node)
PlayerForTest
<<Property>> -walls : int
<<Property>> -color : String
+playerScore : List<Boolean>
<<Property>> +position : Node
-whichever Goal : Node
-allSpecificGoals : Node
+PlayerForTest(color : String)
+getGoal() : Node
+setGoal(goal : Node) : void
+player1
+player2
QuoridorBoardForTest
+board : Graph
+tiles : Tile = new ArrayList<>()
+player1 : PlayerForTest
+player2 : PlayerForTest
+QuoridorBoardForTest()
+createQuoridorBoard(rows : int, cols : int) : Graph
+setTheme(tile : Tile, x : int, y : int) : void
+nextMoveHasPlayerAdjacent() : boolean
+getOppositePlayerAdjacentList() : List<Node>
+shortestPathPossible(player : int) : List<Node>
+getBestNextMove(player : int) : Node
+getNodeByXandY(x : int, y : int) : Node
+getAdjacencyList(player : int) : List<Node>
javaComponents
Node
<<Property>> -x : int
<<Property>> -y : int
+Node(x : int, y : int)
+position
position
allSpecificGoals
whichever Goal
position
whichever Goal
allSpecificGoals
Tile
-occupied : boolean
<<Property>> +position : Node
+Tile(node : Node)
+highlightThis() : void
+deHighlightThis() : void
+thickenBorderSide(side : Side) : void
+tiles
+currentWallTiles
+tiles
<<enumeration>>
Side
TOP
BOTTOM
LEFT
RIGHT
Graph
-newEdges : Map<Node, List<Node>>
<<Property>> -nodes : Node
+Graph()
+addNode(node : Node) : void
+addEdge(firstNode : Node, secondNode : Node) : void
+getEdges(node : Node) : List<Node>
+removeEdge(node1 : Node, node2 : Node) : void
-addWall(node1 : Node, node2 : Node, node3 : Node, node4 : Node) : boolean
+addWall(nodes : List<Node>) : boolean
-hasValidPath(player : Node, goal : Node) : boolean
-dfs(current : Node, goal : Node, visited : Set<Node>) : void
+areNodesAdjacent(node1 : Node, node2 : Node) : boolean
+dijkstraShortestPath(start : Node, end : Node) : List<Node>
+BFS(start : Node) : List<Node>
+primMST(start : Node) : List<Node>
+KruskalMST() : List<Node>
+floydWarshall(start : Node, end : Node) : List<Node>
+board
+board
Player
<<Property>> -walls : int
<<Property>> -color : String
<<Property>> -position : Node
+playerScore : List<Boolean>
-whichever Goal : Node
-allSpecificGoals : Node
+Player(color : String)
+setPiece(image : Image) : void
+setImage() : void
-addEventHandler() : void
+getAllPossibleMoves() : void
+showAllPossibleMoves(val : boolean) : void
+getTileByNode(node : Node) : Tile
+getPlayerByNode(node : Node) : Player
+getPossibleMoves() : List<Node>
+highlightThis() : void
+deHighlightThis() : void
+getGoal() : Node
+setGoal(goal : Node) : void
+currentPlayer
+player2
+player1
QuoridorBoard
-quoridorBoard : GridPane
+board : Graph
+tiles : Tile = new ArrayList<>()
+player1 : Player
+player2 : Player
+QuoridorBoard(quoridorBoard : GridPane)
+createQuoridorBoard(pBoard : GridPane, rows : int, cols : int) : Graph
+setTheme(tile : Tile, x : int, y : int) : void
-addPlayer(tile : Tile, player : Player) : void
+nextMoveHasPlayerAdjacent() : boolean
+getOppositePlayer() : Player
+getOppositePlayerAdjacentList() : List<Node>
+placeWall(nodes : List<Node>) : boolean
-shortestPathPossible() : List<Node>
+getBestNextMove() : Node
+qb
QuoridorBoardTest
+testQuoridorBoard() : void
GraphTest
+setupScenary() : void
+testGraph() : void
Game
+currentPlayerColor : String
-game : boolean
+moveType : Integer
+currentPlayer : Player
+currentWallTiles : Tile
+qb : QuoridorBoard
+Game(quoridorBoard : GridPane)
-addEventHandlers(chessBoard : GridPane) : void
-selectChoice(tile : Tile) : void
-selectPiece(game : boolean) : void
-deselectPiece(changePlayer : boolean) : void
-dropPiece(tile : Tile) : void
-showCongratulationsMessage() : void
+changeMoveType(moveType : int) : void
+deselectWalls() : void
+placeWall() : void
+getCurrentPlayer() : void
+getTileByNode(node : Node) : Tile
+getScores() : String
+game
javaFXComponents
TileAndPlayerTest
+start(stage : Stage) : void


control
Controller
+movePawn : RadioButton
+placeVerticalWall : RadioButton
+Move : ToggleGroup
+placeHorizontalWall : RadioButton
+chessBoard : GridPane
-game : Game
+instructions : Button
+initialize() : void
+changeMoveType(actionEvent : ActionEvent) : void
+openInstructions(actionEvent : ActionEvent) : void
HelloController
-welcomeText : Label
#onHelloButtonClick() : void
RuleBookController
CongratulationScreenController
+playerName : Label
+leftImage : ImageView
+rightImage : ImageView
+score : Label
+initialize() : void
+getScores() : String

HelloApplication
+start(stage : Stage) : void
+main(args : String[]) : void
+openWin() : void
+openInstructions() : void