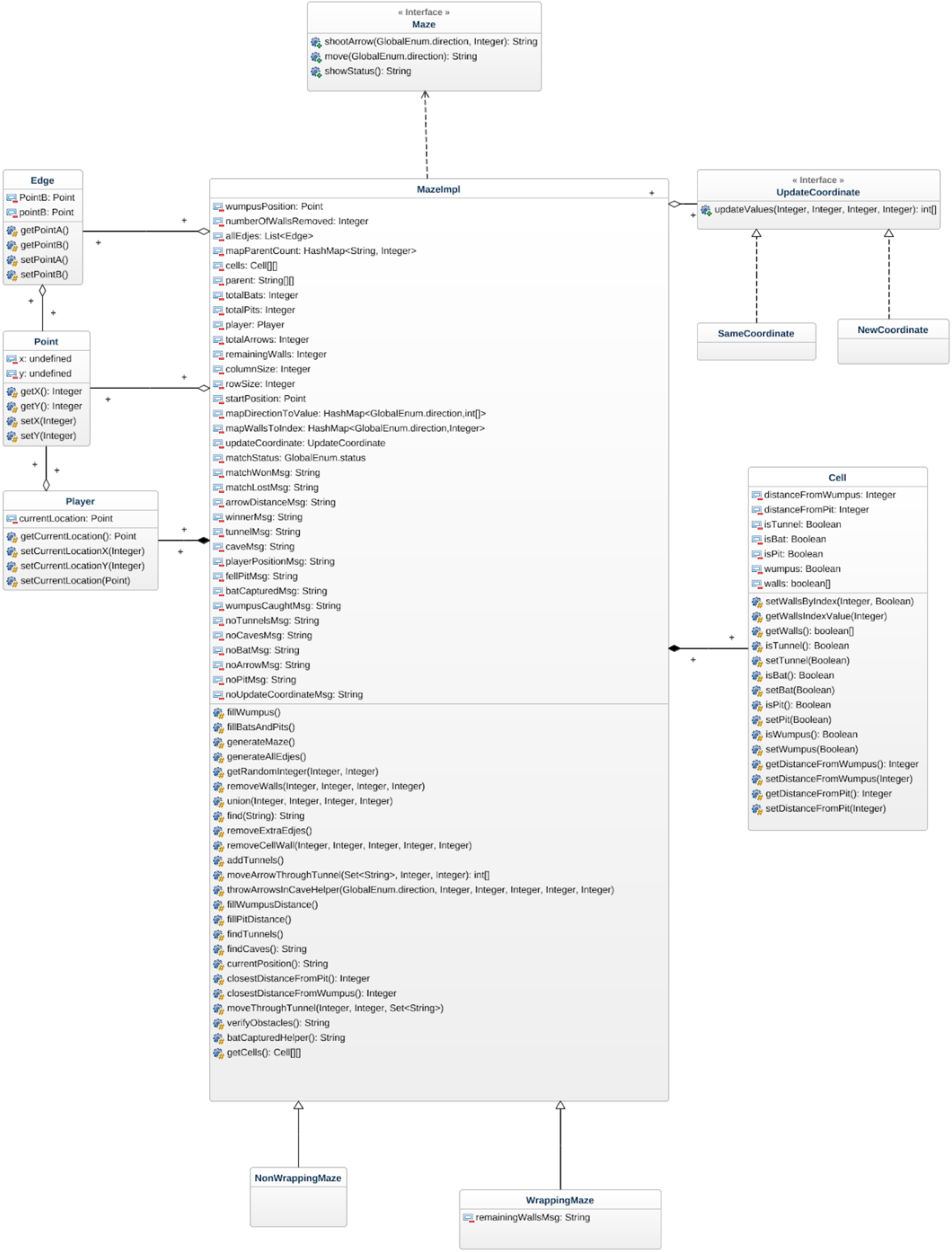
HTW Controller

UML


« Interface »
Maze
shootArrow(GlobalEnum.direction, Integer): String
move(GlobalEnum.direction): String
showStatus(): String
Edge
PointB: Point
pointB: Point
getPointA()
getPointB()
setPointA()
setPointB()
Point
x: undefined
y: undefined
getX(): Integer
getY(): Integer
setX(Integer)
setY(Integer)
Player
currentLocation: Point
getCurrentLocation(): Point
setCurrentLocationX(Integer)
setCurrentLocationY(Integer)
setCurrentLocation(Point)
MazeImpl
wumpusPosition: Point
numberOfWallsRemoved: Integer
allEdjes: List<Edge>
mapParentCount: HashMap<String, Integer>
cells: Cell[][]
parent: String[][]
totalBats: Integer
totalPits: Integer
player: Player
totalArrows: Integer
remainingWalls: Integer
columnSize: Integer
rowSize: Integer
startPosition: Point
mapDirectionToValue: HashMap<GlobalEnum.direction,int[]>
mapWallsToIndex: HashMap<GlobalEnum.direction,Integer>
updateCoordinate: UpdateCoordinate
matchStatus: GlobalEnum.status
matchWonMsg: String
matchLostMsg: String
arrowDistanceMsg: String
winnerMsg: String
tunnelMsg: String
caveMsg: String
playerPositionMsg: String
fellPitMsg: String
batCapturedMsg: String
wumpusCaughtMsg: String
noTunnelsMsg: String
noCavesMsg: String
noBatMsg: String
noArrowMsg: String
noPitMsg: String
noUpdateCoordinateMsg: String
fillWumpus()
fillBatsAndPits()
generateMaze()
generateAllEdjes()
getRandomInteger(Integer, Integer)
removeWalls(Integer, Integer, Integer, Integer)
union(Integer, Integer, Integer, Integer)
find(String): String
removeExtraEdjes()
removeCellWall(Integer, Integer, Integer, Integer, Integer)
addTunnels()
moveArrowThroughTunnel(Set<String>, Integer, Integer): int[]
throwArrowsInCaveHelper(GlobalEnum.direction, Integer, Integer, Integer, Integer, Integer)
fillWumpusDistance()
fillPitDistance()
findTunnels()
findCaves(): String
currentPosition(): String
closestDistanceFromPit(): Integer
closestDistanceFromWumpus(): Integer
moveThroughTunnel(Integer, Integer, Set<String>)
verifyObstacles(): String
batCapturedHelper(): String
getCells(): Cell[][]
« Interface »
UpdateCoordinate
updateValues(Integer, Integer, Integer, Integer): int[]
SameCoordinate
NewCoordinate
Cell
distanceFromWumpus: Integer
distanceFromPit: Integer
isTunnel: Boolean
isBat: Boolean
isPit: Boolean
wumpus: Boolean
walls: boolean[]
setWallsByIndex(Integer, Boolean)
getWallsIndexValue(Integer)
getWalls(): boolean[]
isTunnel(): Boolean
setTunnel(Boolean)
isBat(): Boolean
setBat(Boolean)
isPit(): Boolean
setPit(Boolean)
isWumpus(): Boolean
setWumpus(Boolean)
getDistanceFromWumpus(): Integer
setDistanceFromWumpus(Integer)
getDistanceFromPit(): Integer
setDistanceFromPit(Integer)
NonWrappingMaze
WrappingMaze
remainingWallsMsg: String

Explanation

In this design, we use a Maze interface that will be used by the client for iterating with the maze. Through this interface, we can know about the status of the maze, shoot an arrow in the maze, move in the maze. The maze is also inherited by NonWrappingMaze and WrappingMaze depending upon we want to overlap coordinates or not. The implementation of the operation provided by the maze interface is present in MazeImpl. Moreover, the maze is represented in the form on a 2D array of Cell’s object, and each point is represented by the Point class object. Thus Cell and Point class are aggregated to MazeImpl. Also, Edge class is aggregated as the matrix has a lot of edges between points. In addition to that, Player class is aggregate to the maze as a player is present in the maze and they follow as 'HAS-A' relationship. We have an interface UpdatedCoordinates which is applied depending on whether the maze is wrapping or non-wrapping. This interface application is based on the strategy design pattern.

Test Plan

Following is the test plan that we are going to follow for testing the maze.

1) We will perform move in different situation such as moving towards humpus/bats/pits/wall
2) We will perform shooting of arrow such that arrow hits and misses
3) We will verify smell of wumpus and wrath of pit
4) We will verify if arrow can be shot through tunnel
5) We will verify if player can move through tunnel
6) We will verify if bat and pits can stay on the came cave