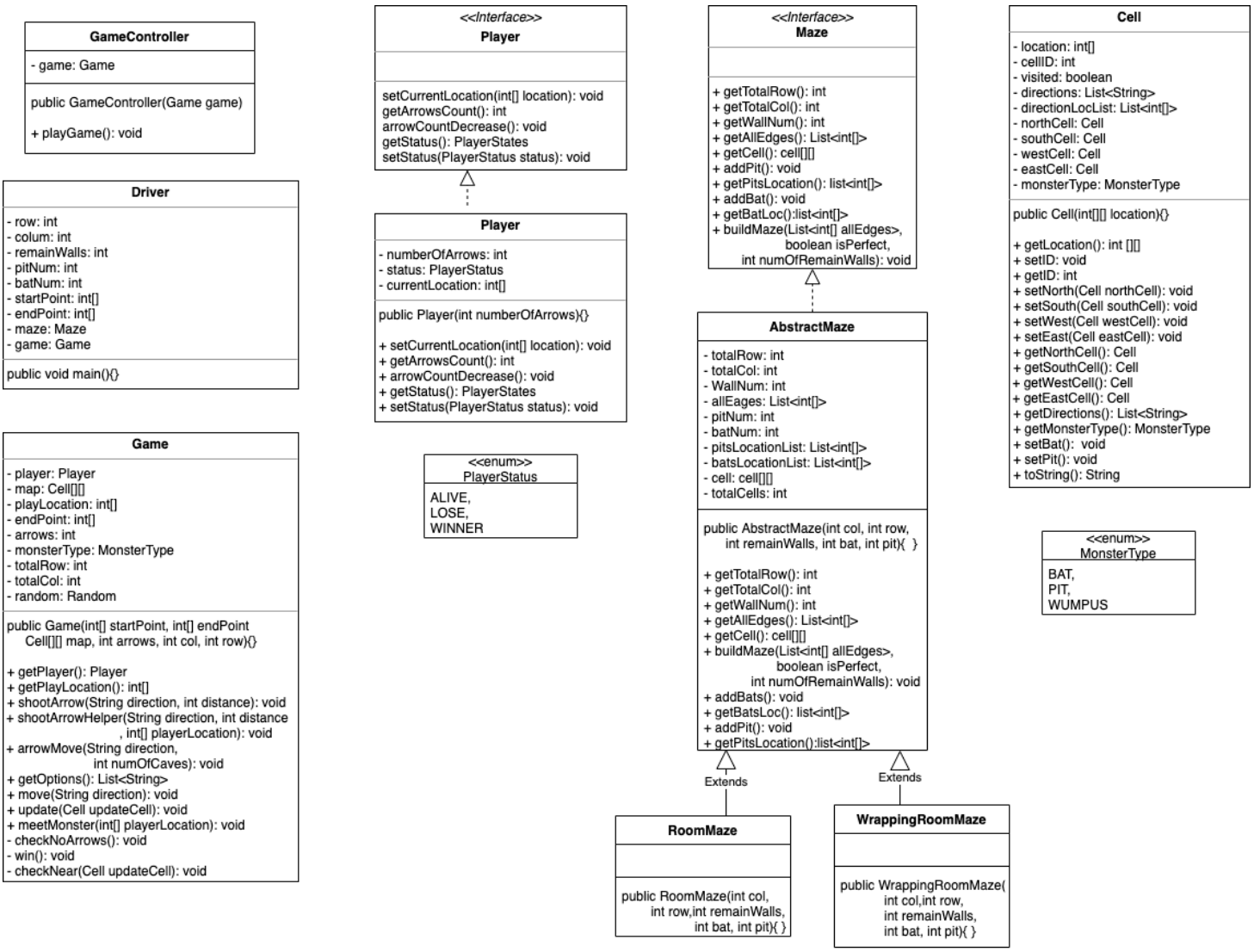
HW5
UML:


GameController
- game: Game
public GameController(Game game)
+ playGame(): void
Driver
- row: int
- colum: int
- remainWalls: int
- pitNum: int
- batNum: int
- startPoint: int[]
- endPoint: int[]
- maze: Maze
- game: Game
public void main(){}
Game
- player: Player
- map: Cell[][]
- playLocation: int[]
- endPoint: int[]
- arrows: int
- monsterType: MonsterType
- totalRow: int
- totalCol: int
- random: Random
public Game(int[] startPoint, int[] endPoint
Cell[][] map, int arrows, int col, int row){}
+ getPlayer(): Player
+ getPlayLocation(): int[]
+ shootArrow(String direction, int distance): void
+ shootArrowHelper(String direction, int distance
, int[] playerLocation): void
+ arrowMove(String direction,
int numOfCaves): void
+ getOptions(): List<String>
+ move(String direction): void
+ update(Cell updateCell): void
+ meetMonster(int[] playerLocation): void
- checkNoArrows(): void
- win(): void
- checkNear(Cell updateCell): void
<<Interface>>
Player
setCurrentLocation(int[] location): void
getArrowsCount(): int
arrowCountDecrease(): void
getStatus(): PlayerStates
setStatus(PlayerStatus status): void
Player
- numberOfArrows: int
- status: PlayerStatus
- currentLocation: int[]
public Player(int numberOfArrows){}
+ setCurrentLocation(int[] location): void
+ getArrowsCount(): int
+ arrowCountDecrease(): void
+ getStatus(): PlayerStates
+ setStatus(PlayerStatus status): void
<<enum>>
PlayerStatus
ALIVE,
LOSE,
WINNER
<<Interface>>
Maze
+ getTotalRow(): int
+ getTotalCol(): int
+ getWallNum(): int
+ getAllEdges(): List<int[]>
+ getCell(): cell[][]
+ addPit(): void
+ getPitsLocation(): list<int[]>
+ addBat(): void
+ getBatLoc():list<int[]>
+ buildMaze(List<int[] allEdges>,
boolean isPerfect,
int numOfRemainWalls): void
AbstractMaze
- totalRow: int
- totalCol: int
- WallNum: int
- allEages: List<int[]>
- pitNum: int
- batNum: int
- pitsLocationList: List<int[]>
- batsLocationList: List<int[]>
- cell: cell[][]
- totalCells: int
public AbstractMaze(int col, int row,
int remainWalls, int bat, int pit){ }
+ getTotalRow(): int
+ getTotalCol(): int
+ getWallNum(): int
+ getAllEdges(): List<int[]>
+ getCell(): cell[][]
+ buildMaze(List<int[] allEdges>,
boolean isPerfect,
int numOfRemainWalls): void
+ addBats(): void
+ getBatsLoc(): list<int[]>
+ addPit(): void
+ getPitsLocation():list<int[]>
Extends
Extends
RoomMaze
public RoomMaze(int col,
int row,int remainWalls,
int bat, int pit){ }
WrappingRoomMaze
public WrappingRoomMaze(
int col,int row,
int remainWalls,
int bat, int pit){ }
Cell
- location: int[]
- cellID: int
- visited: boolean
- directions: List<String>
- directionLocList: List<int[]>
- northCell: Cell
- southCell: Cell
- westCell: Cell
- eastCell: Cell
- monsterType: MonsterType
public Cell(int[][] location){}
+ getLocation(): int [][]
+ setID: void
+ getID: int
+ setNorth(Cell northCell): void
+ setSouth(Cell southCell): void
+ setWest(Cell westCell): void
+ setEast(Cell eastCell): void
+ getNorthCell(): Cell
+ getSouthCell(): Cell
+ getWestCell(): Cell
+ getEastCell(): Cell
+ getDirections(): List<String>
+ getMonsterType(): MonsterType
+ setBat(): void
+ setPit(): void
+ toString(): String
<<enum>>
MonsterType
BAT,
PIT,
WUMPUS