UML Class Diagram

<<interface>> DungeonGraphicalController
+ play():void
+setView(DungeonView):void
+ setGameConfig(row, column, interconnectivity, wrapped, treasurePercentage, numberOfmonsters, Randomizer)
+handleCellClick(Move)

<<interface>> DungeonController
+ play(Dungeon):void

<<interface>> ReadOnlyDungeon
+ getPlayerDescription():Player
+ getPlayerLocationDesc():
+ getAvailableMoves(): Set<Move>
+ getSmell(): SmellLevel
+ isPlayerDead():boolean
+ isGameOver(): boolean
+ playerVisitedEnd():boolean

<<interface>> Location
+ getTreasure(params): Map<Treasure, Integer>
+ getNextMoves(): Set<Move>
+ hasTreasure(): boolean
+ getRow(): int
+ getColumn(): int
+ getArrows(): int
+ hasArrows(): boolean
+ hasMonster(): boolean
+isCave(): boolean

<<interface>> Player
+ getTreasure(): Map<Treasure, Integer>
+ hasTreasure(): boolean
+ getArrows(): int
+ hasArrows():boolean

Treasure
DIAMONDS
RUBIES
SAPPHIRES

<<interface>> DungeonView
+ addListener(DungeonGraphicalController):void
+refresh():void
+ showMessage(String):void
+ makeVisible()
+ setSource(ReadOnlyDungeon)
+ showHelp

DungeonConsoleController
- readable: Readable
- appendable: Appendable

+ play(Dungeon):void

<<interface>> Dungeon
+ movePlayer(Move):void
+ playerPickTreasure():
+ playerPickArrows():void
+ shootArrow(Move, int): ArrowHitOutcome

<<interface>> PlayerPrivate
+ addToTreasure(Treasure,value)
+ fireArrow()

Move
NORTH
SOUTH
EAST
WEST

DungeonSwingController
- view:: DungeonVIew
- dungeon: Dungeon
- modelCopy: Dungeon

Implements all methods of parent

<<interface>> LocationPrivate
+ setNextMove(move):
+ setTreasure(Treasure,quantity):
+pickTreasure(Treasure,quantity):
+ getMonster(): Monster
+ pickArrows(): void
+ hitMonster():void
+ setMonster():void
+ setArrows():void

SmellLevel
NONE
LESS
MORE

DungeonGraphicalView
- listener: DungeonGraphicalController
- dungeon: ReadOnlyDungeon

Implements all methods of parent

<<interface>> Monster
+ getHealth(): int
+ arrowHit(): void
+ isInjured(): boolean
+ isDead(): boolean

DungeonModel
- rows: int
- columns: int
- player: Player
- maze: List<List<LocationPrivate>>
- connectedEdges: List
- randomizer: Randomizer

+ Dungeon(row, column, interconnectivity, wrapped, treasurePercentage, numberOfmonsters, Randomizer) Implements all methods of parents

LocationNode
- treasure: Map<Treasure,Integer>
- isEnd: bool
- nextMoves:List<Move>
- monster: Monster
- arrows: int

Implements all methods of parents

PlayerModel
- collectedTreasure: Map<Treasure,Integer>
- arrows: int

+ getPlayerTreasure(): Map<Treasure,Integer>
+ addToTreasure(Treasure,value)
+ useArrow()

ArrowHitOutcome
MISS,
INJURED,
KILLED

Otyugh
- health: int

Implements all methods of parents

<<Randomizer>>
+ getRandomValue():int

GameRandomizer
- random: Random
- randomSequence: int[]

+ getRandomValue():int

Edge
-x1: int
-y1: int
-x2: int
-y2 : int
-move:Move

+ getMove(): Move

Used in generating maze in Dungeon

Commands
MOVE
PICKUP
SHOOT QUIT
PICKARROWS
PICKTREASURE

package-private

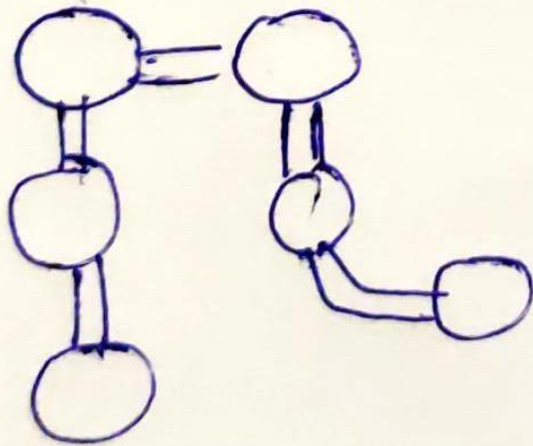has a

1..1

1..1..*

0.1

Menu

Location info

Player info

Text info / Errors

Dungeon View

Testing Plan

1. Every cave has a path, cave cannot be isolated. testAllLocationsAccessible
2. The given percentage of caves have treasure. testTreasureFilledMaze
3. Path length between start and end is always atleast 5.
4. Edge caves in wrapped dungeon are connected
5. Edge caves in not wrapped dungeon are not connected
6. Player picks up treasure correctly. testTreasurePicking(
7. Cave is emptied after player picks treasure. testTreasurePicking(
8. Player moves correctly. testMove
9. Player cannot move across caves without tunnel testInvalidMove
10. Player treasure description is correct
11. Player location is correct testMove
12. Test 5 away getMaze
13. Test tunnels don't have treasure testTreasureFilled
14. Test increasing the interconnectivity increases connecting paths getMaze
15. Test dungeon maze is created correctly. getMaze
16. Test that there is a path from every cave in the dungeon to every other cave in the dungeon.
17. Test that invalid rows value to dungeon throws error.
18. Test that invalid column values to dungeon throws error.
19. Test that invalid interconnectivity value throws error.
20. Test that invalid move value to move player throws error
21. Test that invalid value to add treasure is not accepted
22. Test game ends after player reaches end.
23. Test arrows are distributed correctly
24. Test controller move
25. Test controller shoot
26. Test controller pickup
27. Test invalid move doesn't crash controller
28. Test invalid shoot doesn't crash controller
29. Test invalid pick doesn't crash controller
30. Test game win end.
31. Test game lose end.
32. Test move with mock controller
33. Test shoot with mock controller
34. Test pickup with mock controller
35. Test same actions for GUI based controller.
36. Test mouseclick using fake click event
37. Test key input using fake key event
38. Test menu options
39. Test gui controller makes appropriate changes
40. Test shoot for gui controller

41. Test pickup for gui controller

42. Test move for gui controller

43. Test invalid inputs in gui controller