PlayerImpl

-List<Treasure> treasure
-LocationNode currentLocation

+PlayerImpl()

<<interface>>
Player

+lookForTreasure():
+pickUpTreasure(LocationNode location):
+getLocation(): LocationNode
+setLocation(): void
+getDescription(): String
+getLocationDescription(): String
+movePlayer(Direction)

<<enumeration>>
Treasure

DIAMOND
RUBY
SAPPHIRE

Wall

+Wall()
+isWall(): boolean

Dungeon

-LocationGraphADT locationGraph
-Player player
-Location start
-Location end

+Dungeon(int m, int n, boolean enableWrap, int interconnectivity)
+Dungeon(int m, int n, boolean enableWrap, int interconnectivity, int ...random)
-decideStartAndEnd()
-generateLocationGraph()
-addNodeToGraph(LocationNode node)
-addEdgeToGraph(LocationNode from, LocationNode to, Direction direction)
-generateNodesAndEdges()
-KruskalOnNumbers(): List<MatrixPosition[2]>
-addTreasureToCave(int posX, int posY, Treasure treasure)

<<interface>>
Dungeon

+addPlayer(): void
+addTreasure(int percentage): void
+getPlayerDescription(): String
+movePlayer(Direction direction): void
+getPlayerLocationDescription(): String
+pickTreasure(): String

<<interface>>
LocationGraphADT

+addLocationNode(int posX, int posY)
+addConnection(LocationNode from, LocationNode to, Direction)

<<interface>>
LocationNode

+ default isTunnel(): boolean
+ default isCave(): boolean
+ default isWall(): boolean
+ default hasLocationAt(Direction direction): boolean
+ default getLocationAt(Direction direction): Location
+ getDescription(): String
+ hasTreasure(): boolean
+ addTreasure(Treasure treasure): void
+ getPos(): MatrixPosition;

AbstractLocation

#NeighbouringLocations: HashMap<Direction, LocationNode> {read only}
#MatrixPosition pos {read only}

#AbstractLocation(MatrixPosition pos)
#setNeighbour(Direction direction, LocationNode location): void

---

Tunnel

+Tunnel(MatrixPosition pos)
+isTunnel(): boolean

LocationGraph

-List<LocationNode> locations

Cave

List<Treasure> treasure
Randomizer randomizer

+Cave(MatrixPosition pos)
+isCave(): boolean

MatrixPosition

-int i
-int j

+MatrixPosition(int i, int j)

<<interface>>
Randomizer

+getRandomIntBetween(int a, int b): int

<<enumeration>>
Directions

NORTH
EAST
SOUTH
WEST

+getOpposite(): Direction

Java.util.Random

ActualRandomizer

-Random random

PseudoRandomizer

-List<Integer> random

-Sai Dheeraj Reddy Gadwala

Testing Plan for project 3: Dungeons:

1. Test if the dungeon constructor generates the locationsGraph as expected using pseudo random inputs.
2. Test if add player adds player to dungeon as expected.
3. Test if add treasure(percentage), adds treasure to the given percent of caves.
4. Test if percentage argument of addTreasure is valid.
5. Test if direction of move Player is valid.
6. Test if getPlayerDescription throws error when player has not been added yet.
7. Test if getPlayerDescription gives the correct description if player has already been added.
8. Test if movePlayer does not move player through walls.
9. Test if movePlayer throws error when player has not been initialized.
10. Test if movePlayer moves player as expected.
11. Test if getPlayerLocation gives the correct player location.
12. Test if getPlayerLocation throws error when player has not been initialized.
13. Test if player picks treasure as expected and a proper message is returned about picking the treasure or not finding treasure at that location.
14. Test if tunnels have 2 neighbouring locations and 2 neighbouring walls.
15. Test if caves have 1, 3, or 4 neighbouring locations.
16. Test if interconnectivity is in a valid range.
17. Test that caves can have more than 1 treasure
18. Test if tunnels cannot have treasure
19. Test if treasure is removed from cave after a player picks it up
20. Test if treasure picked by player is exactly what he picked.
21. Test if game ends after player reaches end
22. Test if every cave in the graph is reachable
23. Test if wrapped Dungeons have wraps.
24. Test if unwrapped Dungeons do not wrap.
25. Test if start and end have at least a distance of 5 between them.
26. Test if all locations are at unique positions.
27. Test if treasure is not null in methods that add treasure to caves and players.
28. Test if Actual Randomizer works as expected.
29. Test if Pseudo Randomizer works as expected.

#Note: These are the tests I currently have in mind. More tests will be added in the implementation phase.