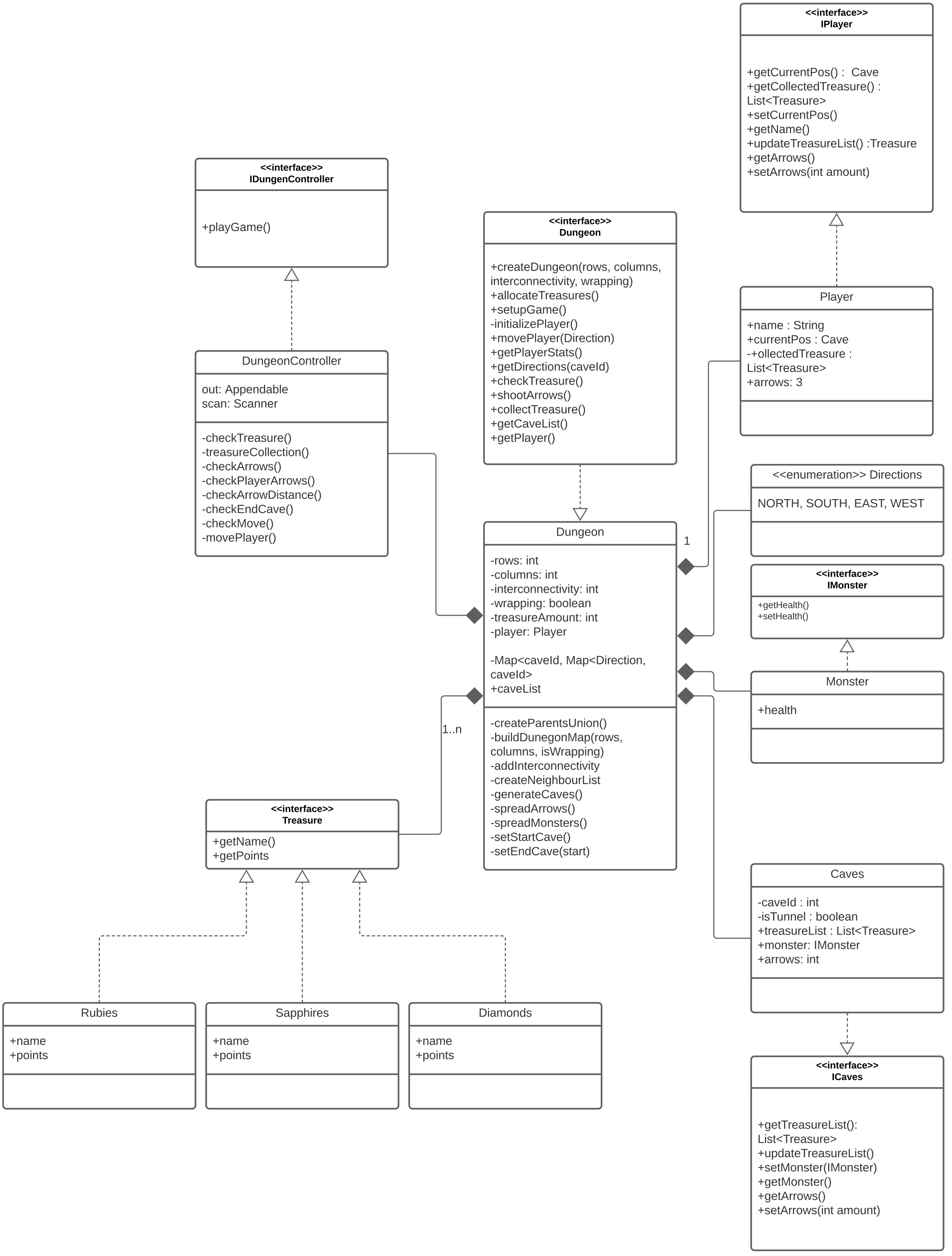
<<interface>>
IDungenController
+playGame()
DungeonController
out: Appendable
scan: Scanner
-checkTreasure()
-treasureCollection()
-checkArrows()
-checkPlayerArrows()
-checkArrowDistance()
-checkEndCave()
-checkMove()
-movePlayer()
<<interface>>
Dungeon
+createDungeon(rows, columns, interconnectivity, wrapping)
+allocateTreasures()
+setupGame()
-initializePlayer()
+movePlayer(Direction)
+getPlayerStats()
+getDirections(caveId)
+checkTreasure()
+shootArrows()
+collectTreasure()
+getCaveList()
+getPlayer()
Dungeon
-rows: int
-columns: int
-interconnectivity: int
-wrapping: boolean
-treasureAmount: int
-player: Player
-Map<caveId, Map<Direction, caveId>
+caveList
-createParentsUnion()
-buildDunegonMap(rows, columns, isWrapping)
-addInterconnectivity
-createNeighbourList
-generateCaves()
-spreadArrows()
-spreadMonsters()
-setStartCave()
-setEndCave(start)
1
1..n
<<interface>>
IPlayer
+getCurrentPos() : Cave
+getCollectedTreasure() : List<Treasure>
+setCurrentPos()
+getName()
+updateTreasureList() :Treasure
+getArrows()
+setArrows(int amount)
Player
+name : String
+currentPos : Cave
-+ollectedTreasure : List<Treasure>
+arrows: 3
<<enumeration>> Directions
NORTH, SOUTH, EAST, WEST
<<interface>>
IMonster
+getHealth()
+setHealth()
Monster
+health
<<interface>>
Treasure
+getName()
+getPoints
Caves
-caveId : int
-isTunnel : boolean
+treasureList : List<Treasure>
+monster: IMonster
+arrows: int
Rubies
+name
+points
Sapphires
+name
+points
Diamonds
+name
+points
<<interface>>
ICaves
+getTreasureList(): List<Treasure>
+updateTreasureList()
+setMonster(IMonster)
+getMonster()
+getArrows()
+setArrows(int amount)

Testing Plan – Project 4

Testing plan for dungeon:

1. Test if each cave in the dungeon has a path to other cave.
2. If the interconnectivity is 0, no cycles should be present in the dungeon grid.
3. Max interconnectivity should not be greater than the number of potential cycles that could exists in the dungeon grid.
4. Error should be thrown for negative rows and columns passed.
5. If interconnectivity is greater than 0, then the number of cycles in the dungeon grid should be equal to the interconnectivity degree.
6. Test that wrapping exists correctly is user wants wrapping dungeon.
7. The start and end caves of the dungeon should at least have the distance of 5.
8. The dungeon should at least contain 6 caves to maintain the minimum distance of 5 between start and end caves.
9. The treasure allocation to the caves should be handled properly (eg: if user input is 20% and number of caves is 100, then 20 caves should receive treasure.)
10. Player should be instantiated correctly with start as the players initial position.
11. The player should be able to move only in 4 directions (north, south, east, west).
12. If for any move is not possible then "invalid move" should be displayed.
13. The game should end once the player reaches the end cave.
14. Test player description.

Testing plan for caves:

1. If the connection for a particular cave is 2 then it should be named as a tunnel.
2. If the connection is 1(dead end) or greater than 2 then it should not be termed as a tunnel.
3. Treasure list should be updated properly once the random assignment of the treasure is done.

Testing treasure:

1. There should be at least 3 types of treasure - diamond, rubies, sapphires.

Testing plan for Player:

1. The current position of the player should be the start of the dungeon.
2. The current location of the player should be updated correctly after every move.
3. Test if the collected list of treasure should be updated after every move in which the player moves to the location having treasure.

Testing plan for arrows:

1. Amount same as treasure
2. Can be found in tunnels
3. Pass freely through tunnel
    - Shoot an arrow from a given position
    - Can manually set the player position to a particular cave and then determine the expected end cave.
        - Single way through cave (eg: east-west)

        - Check if given a start position and end position
        - The arrow visits the correct locations in order
    - Player starts with 3 **crooked arrows**
    - Gives direction and distance (caves and not tunnels) to shoot
    - Distance must be exact to hit the Otyugh
    - 2 hits to kill the Otyugh

Testing plan for Otyugh:

1. Should be present in the end cave.
2. Should not be in the start.
3. Should not be present in tunnels.

Testing Plan for Controller:

1. Test for null model.
2. Test for FailingAppendable
3. Test for Invalid values
4. Test if model works correctly for treasure and arrows pickup.
5. Test if model works correctly for inputs in which player is killed.
6. Test if model works correctly for inputs in which player wins.