Model

<<interface>>
Dungeon

+getDungeon:String
+getGameBoardRows(): int
+getGameBoardColumns(): int
+getFinalEdgeList(): List
+getGameBoard(): Cave[][]
+isGameOver(): boolean
+movePlayer(Direction): String
+getWrapping(): boolean
+shootArrow(int distance, Direction direction): String
+pickUpItem(int): String
+quitGame(): String

<<interface>>
Location

+getTreasureList(): List
+getNeighborList(): List
+getArrowListSize(): int
+getMonsterListSize(): int

<<interface>>
Treasure

+getTreasure(): Treasure
+getName(): String

Point2D

-int row
-int column

#getRow
#getColumn

AbstractLocation

-Point2D(int row, column)
-neighborList(List<Integer>)
-treasureList(List<Treasure>)
-arrowList(List<Treasure>)

-setLocation(): void

TreasureImpl

+Enum TreasureType: DIAMOND, SAPHIRE, RUBY
#Ruby
#Diamond
#Sapphire
#TreasureFactory

#TreasureFactory -
#getTreasureFromEnum(TreasureType): Treasure

DungeonImpl

-boolean: wraps
-int: rows
-int: columns
-int: interconnect
-int: treasure
-gameBoard [ ] [ ]
-int: startLocation
-int: endLocation
-List<Edge> potEdgeList
-List<Edge> leftOverEdge
-List<Edge> finalEdgeList
-Player player
-int: difficulty
-RandomNumberGenerator
-int: genSeed
-boolean: quitFlag

-runKruscals(): String
-findCaves(List<Cave>): void
-setUpPlayer(): String
-getPossibleDirection(int): List<Direction>
-getStartPoint(List): String
-findEndPoint(int): String
-findCaveByIndex(int): Cave
-getPotentialEdgeList(): List<Edge>
-getCavesByIndex(): List<Integer>
-addEdge(List<List<Integer>>, int, int): void
-findCaves(List<Cave>):void
-addMonstersToDungeon(List<Cave>): void
-checkForEnd(): boolean
-addArrows(): void
-checkSmell():int
-getOppositeDirection(Direction): Direction

Cave

-int index
-int set
-MonsterList(List<Monster>)

#getRow(): int
#getColumn(): int
#getIndex(): int
#getSet(): int
#adjSet(int): void
#addNeighbor(int): void
#getNeighbors(): List
#addTreasure(Treasure): void
#getTreasureFromCave(): List
#getArrowsFromCave(): List
#addMonster(Monster): void
#getMonsterHealth(): int
#addArrow(CrookedArrow): void
#getMonster(): Monster

<<interface>>
Player

+move(int, List<Direction>, List<Treasure>): void
+getPlayerStatus(): void
+enterDungeon(int, List<Direction>, List<Treasure>): void
+getTreasureList(): List<Treasure>
+shootArrow(Direction, distance): void

Player

-int playerLocation
-List treasureList
-List directions
-List currentTreasure
-List quiver

-updatePlayerLocation(int, List<Direction>, List<Treasure>): void
-updateTreasure(): void

<<interface>>
Monster

+getHealth(): int
+takeDamage(): int

Text

Otyugh

-int health

Edge

-int cave1
-int cave2
-Direction directionToCave1
-Direction directionToCave2

+getDirection(Cave, Cave) :Direction
#compareSets(): boolean
#getLeftSet(): int
#getRightSet(): int
#getLeftIndex(): int
#getRightIndex(): int
#addNeighbors(): void
-getDirectiontoCave1():Direction
-getDriectionToCave2(): Direction
+toString(): String

CrookedArrow

-int distance
-Direction: direction

#shootArrow(int, Direction): void

Driver

Driver

+main

random

RandomNumberGenerator

-int: seed

+getRandomNum(int min, int max): int

Controller

Controller

-readable
-appendable

-playGame(): void
-modelToController(Model): void