UML Class Diagram

Driver
```
+ main(String[] args)
```

<<interface>> GameViewInterface
```
+ void refresh();
+ void removeStartPanel();
+ void makeVisible();
+ void dungeonView();
+ void refreshDungeon();
+ void setListeners(ActionListener clicks, KeyListener keys,
GuiControllerInterface controller);
+ List<String> getGameInitialValues();
+ int endGamePopUp(String message);
+ void quitGame();
+ void setStartPanel();
+ void removeDungeonView();
+ void disposeFrame();
+ int getNewInputValues(String value);
+ void playerHearSound(String soundMessage);
+ void updateModel(ReadonlyModel model);
+ void popUpDialog(String configMessage);
```

<<interface>> GuiControllerInterface
```
+ start()
+handleMouseClick(int,int)
```

<<interface>> ConsoleControllerInterface
```
+ start()
```

<<interface>> GameModel
```
+ printMaze(): String
+ checkWinning(): boolean
+ availablePaths(): String
+ checkValidMove(String move) : boolean
+ moveToPosition(String move)
+ availableTreasure() : String
+ mazeDetails() : Integer[]
+ playerLocationDetails() : int
+ playerTreasureDetails() : HashMap<Treasure, Integer>
+ checkTreasureInCave() : boolean;
+ pickupTreasure(treasure)
+ pickupArrow()
+ ceckArrow() : boolean
+ shoot() : String
+ isPlayerAlive() : boolean
+ smell() : String
+ checkPlayerArrow() : boolean
+ playerArrowDetails() : int
```

<<interface>> ReadOnlyModel
```
+ String[] playerLocationDetails();
+ String smell();
+ boolean isPlayerAlive();
+ int playerArrowDetails();
+ HashMap<CaveJewels, Integer>
playerTreasureDetails();
+ String getPlayerName();
+ Set<Integer> getVisitedCave();
+ boolean checkWinning();
```

RandomNumber

<<interface>> RandomInterface
```
+ getRandomNumber(int , int ) : int
```

RandomFroTesting

GuiController
```
- GameModel model;
- Game View  view;
- RandomInterface random;
- List<Interger> inputs;
- List<String> gameInputs;

+ GuiController(model, view, random)
```

ControllerConsole
```
- final GameModel model;
- final Appendable out;
- final Scanner scan;

+ ControllerConsole(model, appendable, readable)
```

GameFrame
```
- StartPanel startPanel;
- GameMenu menu;
- JSpinner rowSpinner;
- JSpinner colSpinner;
- JSpinner interconnectivitySpinner;
- JSpinner treasurePercentageSpinner;
- JSpinner monsterCountSpinner;
- JTextField playerName;
- JButton playButton;
- JComboBox wrapList;
- ReadonlyModel model;
- DungeonPanel dungeonPanel;
- JSplitPane splitPane;
- ActionListener clicks;
- PopUpWindow pop;
- GuiControllerInterface controller;
- GameInformationPanel infoPanel;
- List<String> inputs;

+  GameFrame(String dungeonName, ReadonlyModel model)
```

<<interface>> CommandInterface
```
+ command(GameModel model, Appendable out, Scanner scan)
```

Move

Pickup

Shoot

<<interface>> PlayerInterface
```
+ setTreasureCount(HashMap<Treasure, Integer>
newTreasureCount) ;
+ getTreasureCount(Treasure treasure) : int
+ getPlayerLocation() : int
+ setPlayerLocation(int nextCaveId)
```

Game
```
-dungeon : DungeonInterface
-player : PlayerInterface
-currentCave : CaveInterface
- random : RandomInterface

+ Game(int , int , int , boolean ,int , String , RandomInterface )
```

<<interface>> CaveInterface
```
+ getCaveId() : int
+ getRowNo() : int
+ getColumnNo() : int
+ setTunnel(boolean tunnel) :
+ isTunnel() : boolean
+ setDirection(Direction direction, Integer
caveId) :
+ getDirection() : HashMap<Direction,
Integer>
+ setCaveTreasure(Treasure treasure, int
value) :
+ getCaveTreasure() : HashMap<Treasure,
Integer>
+ isArrow()
+ setArrow()
+ getMonster()
+ setMonster()
```

Player
```
-playerName:String
-playerLocation:int
-treasureCount: HashMap<Treasure, Integer>

+Player(Strng playerName,location)
```

<<interface>> DungeonInterface
```
+ getCaveList() : List<List<CaveInterface>>
+ getStartCave(): CaveInterface
+ getEndCave(): CaveInterface
+ printDungeon(CaveInterface currentCave): String
+ CaveInterface getCave(int nextCaveId): CaveInterface
+ updateTreasure(CaveInterface cave)
```

Cave
```
caveId : int
rowNo : int
columnNo : int
caveTreasure : HashMap<Jewels,Integer>
direction : HashMap<Path,Integer>
isTunnel : boolean

+ Cave(int caveId, int rowNo, int columnNo)
+ Cave( CaveInterface )
```

<<enumeration>> PATH
```
NORTH
SOUTH
EAST
WEST
```

<<enumeration>> Jewels
```
DIAMONDS
SAPHIRE
RUBY
```

Dungeon
```
- int xcoordinate
- int ycoordinate
- int interconnectivity
- int treasurePercentage
- boolean isWrapping
- boolean flagStartEndAssigned
- CaveInterface startCave
- CaveInterface endCave
- List<List<CaveInterface>> caveList
- List<CaveInterface[]> possiblePath
- List<CaveInterface[]> leftOverList
- List<List<Integer>> startEndPath
- RandomInterface random

+ Dungeon(int , int , int , int , boolean , RandomInterface )
- initializeDungeon()
- initializeCaveTreasure()
- updateTunnel()
- createLinks(List<HashSet<Integer>> hashSetList)
- connectLeftover()
- getSets(List<HashSet<Integer>> hashSetList, Integer
caveId): HashSet
- updateCaveDirection(CaveInterface[] possiblePath1)
- createSets() : List<HashSet<Integer>>
- createPossiblePathList()
- initializeCave()
- selectStartEndCave()
- getCaveObject() : CaveInterface
- checkDistance(CaveInterface tempStartCave,
CaveInterface tempEndCave)
- List<Integer>[] createAdjacencyList() : CaveInterface
- printAllPaths(int startCaveId, int endCaveId,
List<Integer>[] adjList)
- printAllPathsUtil(int u, int d, boolean) isVisited,
                    List<Integer> localPathList,
List<Integer>[] adjList)
```

<<interface>> MonsterInterface
```
+ getHealth()
+ decreaseHealth()
+ getMonsterId()
```

Otyugh
```
-- monsterId : int
-- health : int

+Otyugh(monsterId)
```

Extends

Game | − | ☐ | ✕ |

| Settings | Options |

Welcome To Dungeons:

Player Name :   [_____]

No. of Rows :   [_____]

No of Columns :   [_____]

Interconnectivity : [_____]

Treasure Percentage: [_____]

Monster Count :   [_____]

Wrapping Status :   [_____ ⊗]

Play Game

Game   — □ X

Settings | Options.

△

Player

◁   ▷ ▽
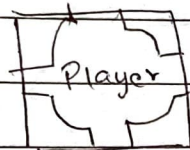
Welcome to Dungeons

Player Position :

Treasure : Diamonds:
           Ruby:
           Sapphire.

Sound :
Smell :