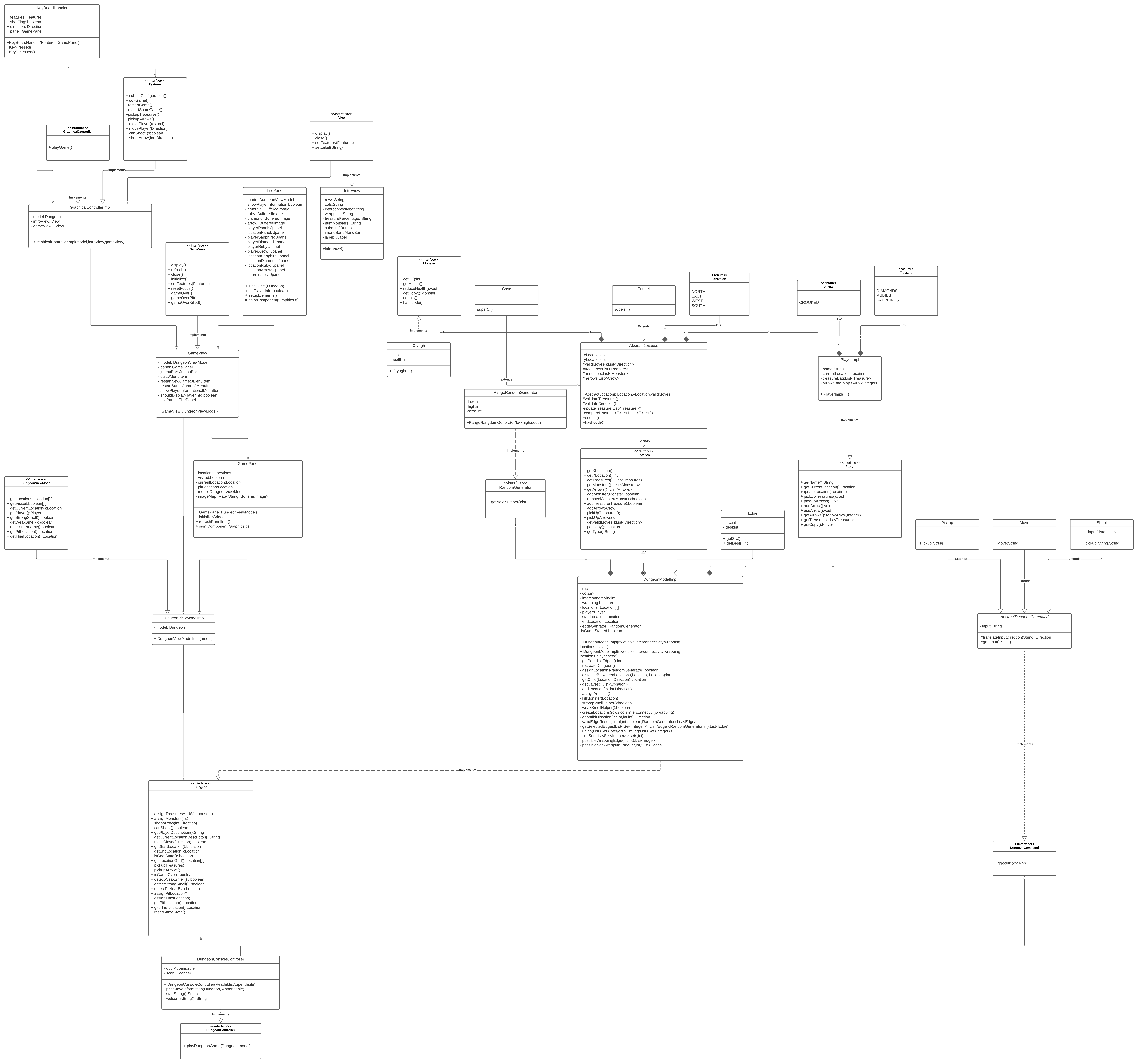
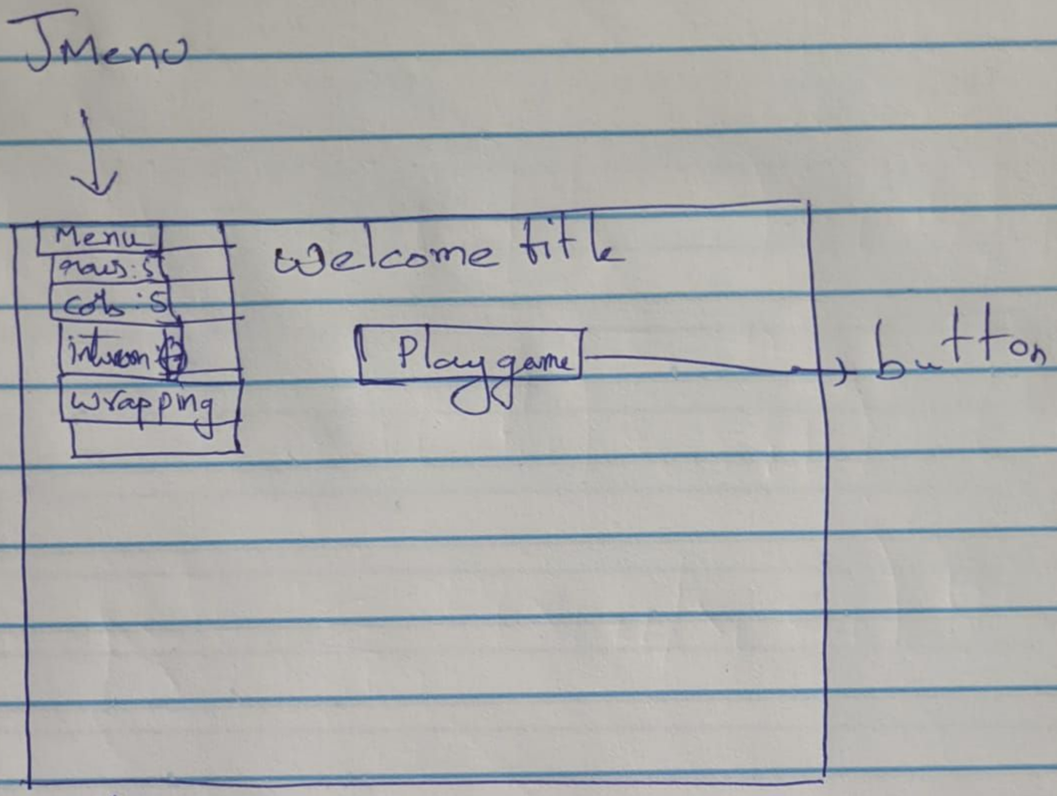
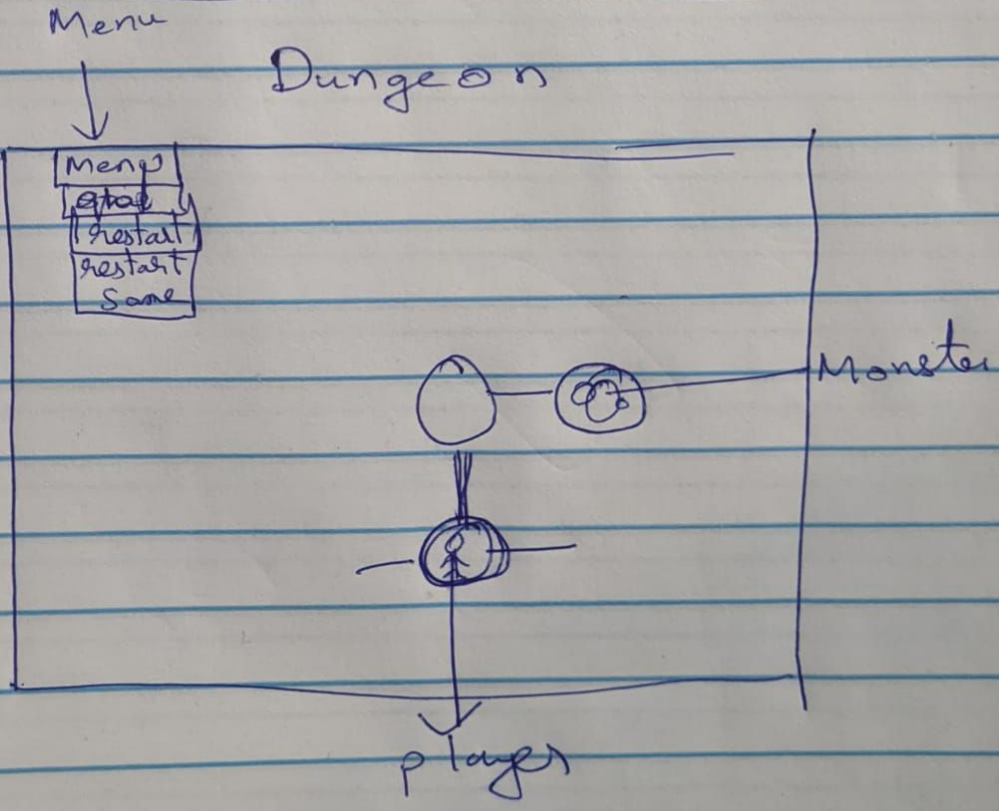

KeyBoardHandler
+ features: Features
+ shotFlag: boolean
+ direction: Direction
+ panel: GamePanel
+KeyBoardHandler(Features,GamePanel)
+KeyPressed()
+KeyReleased()
<<interface>> Features
+ submitConfiguration():
+ quitGame()
+restartGame()
+restartSameGame()
+pickupTreasures()
+pickupArrows()
+ movePlayer(row.col)
+ movePlayer(Direction)
+ canShoot():boolean
+ shootArrow(int. Direction)
<<interface>> GraphicalController
+ playGame()
<<interface>> IView
+ display()
+ close()
+ setFeatures(Features)
+ setLabel(String)
Implements
Implements
Implements
GraphicalControllerImpl
- model:Dungeon
- introView:IView
- gameView:GView
+ GraphicalControllerImpl(model,introView,gameView)
TitlePanel
- model:DungeonViewModel
- showPlayerInformation:boolean
- emerald: BufferedImage
- ruby: BufferedImage
- diamond: BufferedImage
- arrow: BufferedImage
- playerPanel: Jpanel
- locationPanel: Jpanel
- playerSapphire: Jpanel
- playerDiamond Jpanel
- playerRuby Jpanel
- playerArrow: Jpanel
- locationSapphire Jpanel
- locationDiamond: Jpanel
- locationRuby: Jpanel
- locationArrow: Jpanel
- coordinates: Jpanel
+ TitlePanel(Dungeon)
+ setPlayerInfo(boolean)
+ setupElements()
# paintComponent(Graphics g)
IntroView
- rows:String
- cols:String
- interconnectivity:String
- wrapping: String
- treasurePercentage: String
- numMonsters: String
- submit: JButton
- jmenuBar:JMenuBar
- label: JLabel
+IntroView()
<<interface>> GameView
+ display()
+ refresh()
+ close()
+ initialize()
+ setFeatures(Features)
+ resetFocus()
+ gameOver()
+ gameOverPit()
+ gameOverKilled()
Implements
GameView
- model: DungeonVIewModel
- panel: GamePanel
- jmenuBar: JmenuBar
- quit:JMenuItem
- restartNewGame:JMenuItem
- restartSameGame;:JMenuItem
- showPlayerInformation:JMenuItem
- shouldDisplayPlayerInfo:boolean
- titlePanel: TitlePanel
+ GameView(DungeonViewModel)
GamePanel
- locations:Locations
- visited:boolean
- currentLocation:Location
- pitLocation:Location
- model:DungeonViewModel
- imageMap: Map<String, BufferedImage>
+ GamePanel(DungeonViewModel)
+ initializeGrid()
+ refreshPanelInfo()
# paintComponent(Graphics g)
<<interface>> DungeonViewModel
+ getLocations:Location[][]
+ getVisited:boolean[][]
+ getCurrentLocation():Location
+ getPlayer():Player
+ getStrongSmell():boolean
+ getWeakSmell():boolean
+ detectPitNearby():boolean
+ getPitLocation():Location
+ getThiefLocation():Location
implements
DungeonViewModelImpl
- model: Dungeon
+ DungeonViewModelImpl(model)
<<interface>> Dungeon
+ assignTreasuresAndWeapons(int)
+ assignMonsters(int)
+ shootArrow(int,Direction)
+ canShoot():boolean
+ getPlayerDescription():String
+ getCurrentLocationDescripton():String
+ makeMove(Direction):boolean
+ getStartLocation():Location
+ getEndLocation():Location
+ isGoalState(): boolean
+ getLocationGrid():Location[][]
+ pickupTreasures()
+ pickupArrows()
+ isGameOver():boolean
+ detectWeakSmell() : boolean
+ detectStrongSmell(): boolean
+ detectPitNearBy():boolean
+ assignPitLocation()
+ assignThiefLocation()
+ getPitLocation():Location
+ getThiefLocation():Location
+ resetGameState()
DungeonConsoleController
- out: Appendable
- scan: Scanner
+ DungeonConsoleController(Readable,Appendable)
- printMoveInformation(Dungeon, Appendable)
- startString():String
- welcomeString(): String
Implements
<<interface>> DungeonController
+ playDungeonGame(Dungeon model)
<<interface>> Monster
+ getID():int
+ getHealth():int
+ reduceHealth():void
+ getCopy():Monster
+ equals()
+ hashcode()
Implements
Otyugh
- id:int
- health:int
+ Otyugh(....)
1
1
Cave
super(...)
extends
Tunnel
super(...)
Extends
<<enum>> Direction
NORTH
EAST
WEST
SOUTH
1**4
1
1..*
<<enum>> Arrow
CROOKED
1
1...*
1
<<enum>> Treasure
DIAMONDS
RUBIES
SAPPHIRES
1..*
1
AbstractLocation
-xLocation:int
-yLocation:int
#validMoves():List<Direction>
#treasures:List<Treasure>
# monsters:List<Monster>
# arrows:List<Arrow>
+AbstractLocation(xLocation,yLocation,validMoves)
#validateTreasures()
#validateDirection()
-updateTreasure(List<Treasure>()
-compareLists(List<T> list1,List<T> list2)
+equals()
+hashcode()
RangeRandomGenerator
-low:int
-high:int
-seed:int
+RangeRangdomGenerator(low,high,seed)
implements
<<interface>> RandomGenerator
+ getNextNumber():int
1
1
Extends
<<interface>> Location
+ getXLocation():int
+ getYLocation():int
+ getTreasures(): List<Treasures>
+ getMonsters(): List<Monsters>
+ getArrows(): List<Arrows>
+ addMonster(Monster):boolean
+ removeMonster(Monster):boolean
+ addTreasure(Treasure):boolean
+ addArrow(Arrow)
+ pickUpTreasures();
+ pickUpArrows();
+ getValidMoves():List<Direction>
+ getCopy():Location
+ getType():String
1..*
1
Edge
- src:int
- dest:int
+ getSrc():int
+ getDest():int
PlayerImpl
- name:String
- currentLocation:Location
- treasureBag:List<Treasure>
- arrowsBag:Map<Arrow,Integer>
+ PlayerImpl(....)
Implements
<<interface>> Player
+ getName():String
+ getCurrentLocation():Location
+updateLocation(Location)
+ pickUpTreasures():void
+ pickUpArrows():void
+ addArrow():void
+ useArrow():void
+ getArrows(): Map<Arrow,Integer>
+ getTreasures:List<Treasure>
+ getCopy():Player
1
1
DungeonModelImpl
- rows:int
- cols:int
- interconnectivity:int
- wrapping:boolean
- locations: Location[][]
- player:Player
- startLocation:Location
- endLocation:Location
- edgeGenrator: RandomGenerator
-isGameStarted:boolean
+ DungeonModelImpl(rows,cols,interconnectivity,wrapping locations,player)
+ DungeonModelImpl(rows,cols,interconnectivity,wrapping locations,player,seed)
- getPossibleEdges():int
- recreateDungeon()
- assignLocations(randomGenerator):boolean
- distanceBetweeenLocations(Location, Location):int
- getChild(Location,Direction):Location
- getCaves():List<Location>
- addLocation(int int Direction)
- assignArtifacts()
- killMonster(Location)
- strongSmellHelper():boolean
- weakSmellHelper():boolean
- createLocations(rows,cols,interconnectivity,wrapping)
- getValidDirection(int,int,int,int):Direction
- validEdgeResult(int,int,int,boolean,RandomGenerator):List<Edge>
- getSelectedEdges(List<Set<Integer>>,List<Edge>,RandomGenerator,int):List<Edge>
- union(List<Set<Integer>> ,int int):List<Set<integer>>
- findSet(List<Set<Integer>> sets,int)
- possibleWrappingEdge(int,int):List<Edge>
- possibleNonWrappingEdge(int,int):List<Edge>
implements
Pickup
+Pickup(String)
Move
+Move(String)
Shoot
-inputDistance:int
+pickup(String,String)
Extends
Extends
Extends
AbstractDungeonCommand
- input:String
#translateInputDirection(String):Direction
#getInput():String
Implements
<<interface>> DungeonCommand
+ apply(Dungeon Model)

Intro Screen



Dungeon



MONSTER DUNGEON TESTING PLAN

V:1.0

1. Testing Player

1.1 Testing Constructor and getMethods

Testing construction	**Input**	**Expected Value**
Name	new PlayerImpl("Bob",location)	"Bob"
Name Empty	new PlayerImpl("",location)	IllegalArgumentException
Name null	new PlayerImpl(null,location)	IllegalArgumentException
Location	new PlayerImpl("Bob",location)	Correct X coordinate and Y coordinate for the location treasures and valid moves must be same
Location null	new PlayerImpl(null,location)	IllegalArgumentException
Treasure	new PlayerImpl("Bob",location)	Check getTreasures list empty()
Arrows	new PlayerImpl("Bob",location)	Check getArrows list empty()

1.2 Testing Methods in player

Testing	**Input**	**Expected Value**
pickUpTreasures	Treasures at the location	Treasures must be added to current players list
pickUpTreasures	Empty Treasure	No treasures must be added to players treasure.
pickUpArrows	Arrows at the location	Arrows must be added to current players list
pickUpArrows	Empty Arrows	No Arrows must be added to players treasure.
getCopy()	Defensinsive copy check	Check all the attributes of the player.

2. Testing Location

2.1 Testing Constructor and getMethods

Testing construction	**Input**	**Expected Value**
Location	new Cave(2,3,directionList)	X Coordinate:2 Y Coordinate:3 getTreasures empty Valid DIrections has all the directions added
Location	new Tunnel(2,3,directionList)	X Coordinate:2 Y Coordinate:3 getTreasures empty Valid DIrections has all the directions added
Location direction in direction list is null	new Cave(2,3,directionList)	IllegalArgumentException
Location direction in direction list is null	new Tunnel(2,3,directionList)	IllegalArgumentException
Location X null	new Cave(null,3,directionList)	IllegalArgumentException
Location Y null	new Cave(2,null,directionList)	IllegalArgumentException
Location Valid Directions null or empty	new Cave(2,3,null)	IllegalArgumentException
Location X null	newTunnel(null,3,direction List)	IllegalArgumentException
Location Y null	new Tunnel(2,null,directionList)	IllegalArgumentException
Location Valid Directions null or empty	new Tunnel(2,3,null)	IllegalArgumentException
Location with more than 2 valid directions	new Tunnel(2,3,list)	IllegalArgumentException

2.2 Testing Methods in Location

Testing	**Input**	**Expected Value**
addTreasure	Cave add treasure	Treasure list must be populated and the result is true.
addTreasure	Tunnel add treasure	Treasure list must be empty and the result must be false.
addArrow	Cave add arrow	Arrow list must be populated and the result is true.
addArrow	Tunnel add Arrow	Arrow list must be populated and the result is true.
addMonster	Cave add Monster	Add monster to the cave. Return true.
addMonster	Tunnel add Monster	.return false.
removeTreasure	Cave and tunnel remove treasure	Remove treasure if it exists.
removeArrow	Cave and Tunnel remove arrow	Remove the arrow if it exists.
removeMonster	Cave and tunnel remove monsters.	Remove Monster if it exists.
getTreasures	Tunnel and cave	Get all treasures.
getArrows	Cave and Tunnel	Get all arrows.
getMonsters	Cave and Tunnel	Get all Monsters.
updateLocation	Location	Check if the players current location has been updated.
updateLocation null	Location	IllegalArgumentException

3. Testing Monster

3.1 Testing Constructor and getMethods

Testing construction	**Input**	**Expected Value**
id	new Otyugh(0).getId()	Monster id
Id negative	new Otyugh(-1).getId()	IllegalArgumentException
health	getHealth()	2
health	reduceHealth()	1
equals	Check 2 equal	Expected output
compareTo	Check compareTo	Compare two inputs
toString	Compare Two Strings	Compare two strings

3.2 Testing Methods in player

Testing	**Input**	**Expected Value**
health	reduceHealth()	reduceHealth

4. Testing Dungeon Controller

4.1 Testing Constructor and getMethods

Testing construction	Input	Expected Value
DungeonController Check Output	new dungeon controller with input and output	Failing Appendable to test if the controller really throws Illegal State exception
DungeonController Input null	new dungeon controller With input and output.	IllegalArgumentException
DungeonController Output null	new dungeon controller With input and output.	IllegalArgumentException
DungeonController Check input.	new dungeon Mock Model with a log to check if the model is receiving the correct input.	Model should receive the correct input.

4.2 Testing Methods in DungeonController

Testing	Input	Expected Value
playDungeonGame	Valid input Move	Move work correctly for given input.
playDungeonGame	Invalid Input Move	Invalid command and invalid direction strings should be returned.
playDungeonGame	Valid input Pickup treasures	Treasures should be picked up
playDungeonGame	Invalid input Pickup treasures.	Treasures should not be picked up.
playDungeonGame	Valid Input Pickup arrows.	Arrows should be picked up.
playDungeonGame	Invalid Input pickup arrows	Arrows should not be picked up.

playDungeonGame	Valid Input Shoot arrow	Check the arrow shot to the monster.
playDungeonGame	Valid input Shoot arrow	Check arrow shot misses monster for a longer distance.
playDungeonGame	Invalid Input Shoot arrow	Print relevant message.

5. Testing Dungeon

5.1 Testing Constructor and getMethods

Testing construction	Input	Expected Value
DungeonModel	DungeonModel(3,3,5,true, player)	Check in toString() if the dungeon attributes are correct. And check player with equals
DungeonModel rows 0	DungeonModel(0,3,5,true, player)	IllegalArgumentException
DungeonModel cols 0	DungeonModel(3,0,5,true, player)	IllegalArgumentException
DungeonModel player null	DungeonModel(3,3,0,true, player)	IllegalArgumentException
DungeonModel player null	DungeonModel(3,3,0,true, player)	Player check current location should match start location and should have picked up treasures
DungeonModel start location	Check start location	getStartLocation() Start location must not be null
DungeonModel end location	Check end location	getEndLocation() End location must not be null.
getLocation()	DungeonModel(3,3,5,true, player) use cave generator	Check all the locations have formed correctly with the right interconnectivity, wrapping and also the edges
getLocation()	DungeonModel(3,3,5,false, player) use cave generator	Check all the locations have formed correctly with the right interconnectivity, wrapping and also the edges
getLocation()	DungeonModel(3,3,5,true, player) use cave generator	Check all the locations have formed correctly with the right interconnectivity wrapping and the edges
getLocation()	DungeonModel(3,3,5,true,	Check if all the edges with

	player) use cave generator	two valid directions are a tunnel or not. Move the player and check the description.

5.2 Testing Methods in Dungeon

Testing	Input	Expected Value
assignTreasures(20)	DungeonModel(3,3,5,true, player) use treasure generator	Check all locations. More than 20 percent should have treasures
assignTreasures(30)	DungeonModel(3,3,5,true, player) use cave generator	Check all locations. More than 30 percent should have treasures
getPlayerDescription()	1)Initialize Dungeon Move player	Check player treasures as well as the current location
getPlayerDescription()	1)Initialize Dungeon Move player to end node.	Check player treasures as well as the current location
getCurrentLocatiionDescription()	1)Initialize Dungeon Use cave generator and treasure generator Move player to a location	Validate treasures and valid moves in the location
getCurrentLocatiionDescription()	1)Initialize Dungeon Use cave generator and treasure generator Move player to a start location	Validate treasures and valid moves in the location
getCurrentLocatiionDescription()	1)Initialize Dungeon Use cave generator and treasure generator Move player to a end location	Validate treasures and valid moves in the location
makeMove(Direction)	Valid direction	Check if the playerDescription() current location of player changed

		and player picked up treasures.
makeMove(Direction)	Valid direction	Check if the Current Location Description if the output is as expected.
makeMove(Direction)	Invalid direction	IllegalArgumentException
makeMove(Direction)	Reached the goal state an d trying to move.	IllegalStateException
isGoalState()	Current location=End state	Return true
isGoalState()	Current location ! = End state	Return false
Start state and end state	Check if the distance between start and end is at least 5	Use DFS or BFS
assignThiefLocation	Check if the thief location is not null	Shouldnt be null
Assign PitLocation	Check the pit location	Shouldnt be null
assignThiefLocation	Check if the player and thief location are same player is robbed	Player robbed.
Assign PitLocation	Check if game over after player falls into pit	Player should not fall into the pit.

6. Testing Graphical Controller

Testing Constructor and getMethods

Testing construction	**Input**	**Expected Value**
GraphicalController	Create new Mock Model Mock view and introview	Check for the correct output and methods called
GraphicalController	checkMove	Move the player
GraphicalController	checkShoot.	Check shooting in graphical controller
GraphicalController	Check Pickup	Pickup and shoot.