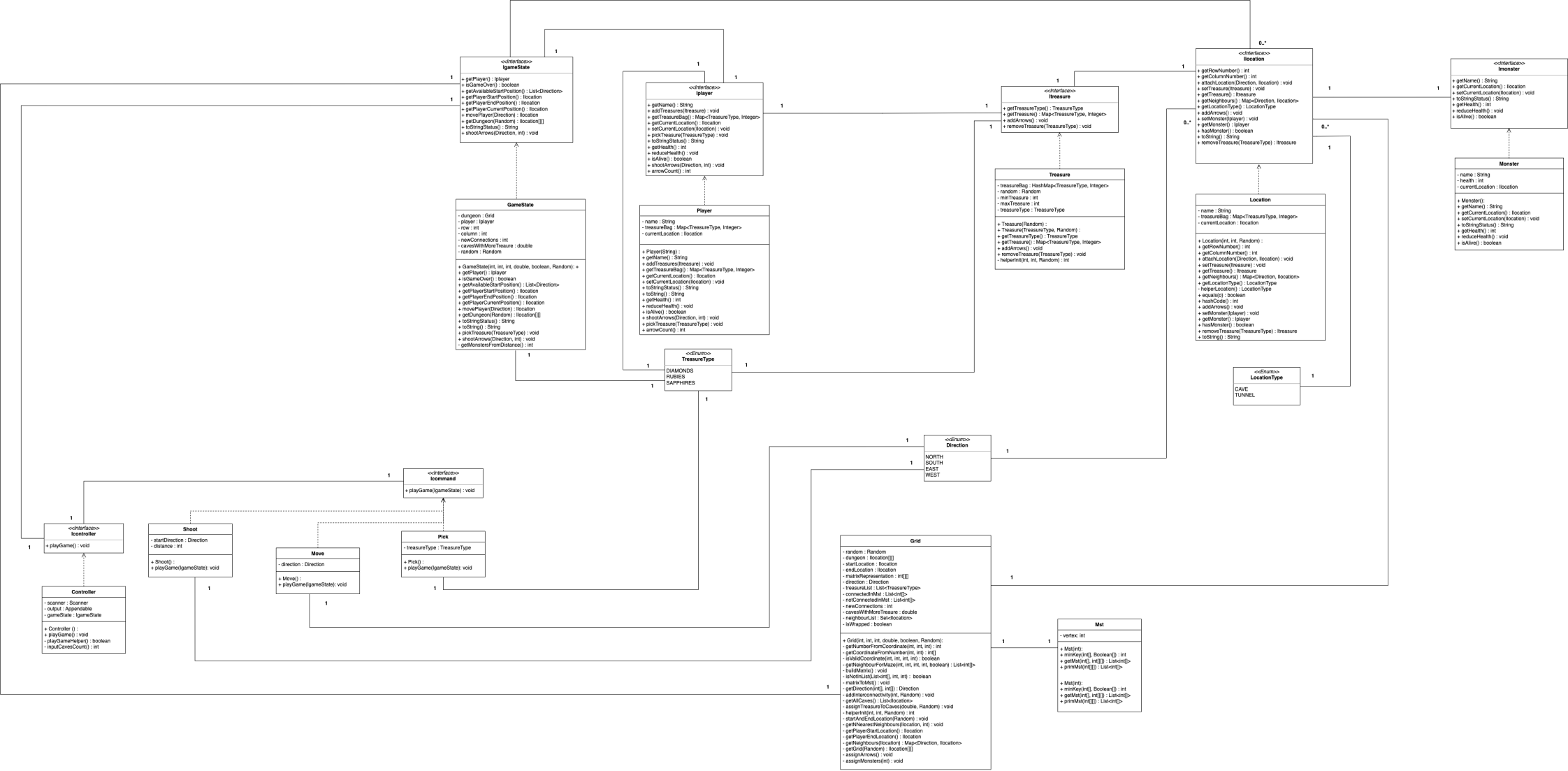
«Interface»
IgameState
+ getPlayer() : Iplayer
+ isGameOver() : boolean
+ getAvailableStartPosition() : List<Direction>
+ getPlayerStartPosition() : Ilocation
+ getPlayerEndPosition() : Ilocation
+ getPlayerCurrentPosition() : Ilocation
+ movePlayer(Direction) : Ilocation
+ getDungeon(Random) : Ilocation[][]
+ toStringStatus() : String
+ shootArrows(Direction, int) : void
«Interface»
Iplayer
+ getName() : String
+ addTreasures(Itreasure) : void
+ getTreasureBag() : Map<TreasureType, Integer>
+ getCurrentLocation() : Ilocation
+ setCurrentLocation(Ilocation) : void
+ pickTreasure(TreasureType) : void
+ toStringStatus() : String
+ getHealth() : int
+ reduceHealth() : void
+ isAlive() : boolean
+ shootArrows(Direction, int) : void
+ arrowCount() : int
«Interface»
Itreasure
+ getTreasureType() : TreasureType
+ getTreasure() : Map<TreasureType, Integer>
+ addArrows() : void
+ removeTreasure(TreasureType) : void
«Interface»
Ilocation
+ getRowNumber() : int
+ getColumnNumber() : int
+ attachLocation(Direction, Ilocation) : void
+ setTreasure(Itreasure) : void
+ getTreasure() : Itreasure
+ getNeighbours() : Map<Direction, Ilocation>
+ getLocationType() : LocationType
+ addArrows() : void
+ setMonster(Iplayer) : void
+ getMonster() : Iplayer
+ hasMonster() : boolean
+ toString() : String
+ removeTreasure(TreasureType) : Itreasure
«Interface»
Imonster
+ getName() : String
+ getCurrentLocation() : Ilocation
+ setCurrentLocation(Ilocation) : void
+ toStringStatus() : String
+ getHealth() : int
+ reduceHealth() : void
+ isAlive() : boolean
GameState
- dungeon : Grid
- player : Iplayer
- row : int
- column : int
- newConnections : int
- cavesWithMoreTreaure : double
- random : Random
+ GameState(int, int, int, double, boolean, Random): +
+ getPlayer() : Iplayer
+ isGameOver() : boolean
+ getAvailableStartPosition() : List<Direction>
+ getPlayerStartPosition() : Ilocation
+ getPlayerEndPosition() : Ilocation
+ getPlayerCurrentPosition() : Ilocation
+ movePlayer(Direction) : Ilocation
+ getDungeon(Random) : Ilocation[][]
+ toStringStatus() : String
+ toString() : String
+ pickTreasure(TreasureType) : void
+ shootArrows(Direction, int) : void
- getMonstersFromDistance() : int
Player
- name : String
- treasureBag : Map<TreasureType, Integer>
- currentLocation : Ilocation
+ Player(String) :
+ getName() : String
+ addTreasures(Itreasure) : void
+ getTreasureBag() : Map<TreasureType, Integer>
+ getCurrentLocation() : Ilocation
+ setCurrentLocation(Ilocation) : void
+ toStringStatus() : String
+ toString() : String
+ getHealth() : int
+ reduceHealth() : void
+ isAlive() : boolean
+ shootArrows(Direction, int) : void
+ pickTreasure(TreasureType) : void
+ arrowCount() : int
Treasure
- treasureBag : HashMap<TreasureType, Integer>
- random : Random
- minTreasure : int
- maxTreasure : int
- treasureType : TreasureType
+ Treasure(Random) :
+ Treasure(TreasureType, Random) :
+ getTreasureType() : TreasureType
+ getTreasure() : Map<TreasureType, Integer>
+ addArrows() : void
+ removeTreasure(TreasureType) : void
- helperInit(int, int, Random) : int
Location
- name : String
- treasureBag : Map<TreasureType, Integer>
- currentLocation : Ilocation
+ Location(int, int, Random) :
+ getRowNumber() : int
+ getColumnNumber() : int
+ attachLocation(Direction, Ilocation) : void
+ setTreasure(Itreasure) : void
+ getTreasure() : Itreasure
+ getNeighbours() : Map<Direction, Ilocation>
+ getLocationType() : LocationType
- helperLocation() : LocationType
+ equals(o) : boolean
+ hashCode() : int
+ addArrows() : void
+ setMonster(Iplayer) : void
+ getMonster() : Iplayer
+ hasMonster() : boolean
+ removeTreasure(TreasureType) : Itreasure
+ toString() : String
Monster
- name : String
- health : int
- currentLocation : Ilocation
+ Monster():
+ getName() : String
+ getCurrentLocation() : Ilocation
+ setCurrentLocation(Ilocation) : void
+ toStringStatus() : String
+ getHealth() : int
+ reduceHealth() : void
+ isAlive() : boolean
«Enum»
TreasureType
DIAMONDS
RUBIES
SAPPHIRES
«Enum»
LocationType
CAVE
TUNNEL
«Enum»
Direction
NORTH
SOUTH
EAST
WEST
«Interface»
Icommand
+ playGame(IgameState) : void
«Interface»
Icontroller
+ playGame() : void
Shoot
- startDirection : Direction
- distance : int
+ Shoot() :
+ playGame(IgameState): void
Move
- direction : Direction
+ Move() :
+ playGame(IgameState): void
Pick
- treasureType : TreasureType
+ Pick() :
+ playGame(IgameState): void
Controller
- scanner : Scanner
- output : Appendable
- gameState : IgameState
+ Controller () :
+ playGame() : void
- playGameHelper() : boolean
- inputCavesCount() : int
Grid
- random : Random
- dungeon : Ilocation[][]
- startLocation : Ilocation
- endLocation : Ilocation
- matrixRepresentation : int[][]
- direction : Direction
- treasureList : List<TreasureType>
- connectedInMst : List<int[]>
- notConnectedInMst : List<int[]>
- newConnections : int
- cavesWithMoreTreaure : double
- neighbourList : Set<Ilocation>
- isWrapped : boolean
+ Grid(int, int, int, double, boolean, Random):
- getNumberFromCoordinate(int, int, int) : int
- getCoordinateFromNumber(int, int) : int[]
- isValidCoordinate(int, int, int, int) : boolean
- getNeighbourForMaze(int, int, int, int, boolean) : List<int[]>
- buildMatrix() : void
- isNotInList(List<int[]>, int, int) : boolean
- matrixToMst() : void
- getDirection(int[], int[]) : Direction
- addInterconnectivity(int, Random) : void
- getAllCaves() : List<Ilocation>
- assignTreasureToCaves(double, Random) : void
- helperInit(int, int, Random) : int
- startAndEndLocation(Random) : void
- getNNearestNeighbours(Ilocation, int) : void
- getPlayerStartLocation() : Ilocation
- getPlayerEndLocation() : Ilocation
- getNeighbours(Ilocation) : Map<Direction, Ilocation>
- getGrid(Random) : Ilocation[][]
- assignArrows() : void
- assignMonsters(int) : void
Mst
- vertex: int
+ Mst(int):
+ minKey(int[], Boolean[]) : int
+ getMst(int[], int[][]) : List<int[]>
+ primMst(int[][]) : List<int[]>
+ Mst(int):
+ minKey(int[], Boolean[]) : int
+ getMst(int[], int[][]) : List<int[]>
+ primMst(int[][]) : List<int[]>
1
0..*

Class			
GameState			
	Method	**Input**	**Expected Value**
	testGetPlayer	getPlayer() Player.getName()	"Player"
	testIsGameOver	- movePlayer(Direction.WEST) movePlayer(Direction.SOUTH) movePlayer(Direction.EAST) movePlayer(Direction.EAST) movePlayer(Direction.EAST)	FALSE TRUE
	testGetAvailableDirectionFromPlayerPosition	getAvailableDirectionFromPlayerPosition()	"WEST"
	testGetPlayerStartLocation	getPlayer() Player.getCurrentLocation()	getPlayerStartLocation()
	testGetPlayerEndLocation	getPlayer() Player.getPlayerEndLocation()	"(3, 4)"
	testMovePlayer	Player.getPlayerStartLocation() movePlayer(Direction.WEST) movePlayer(Direction.SOUTH)	"(2, 2)" "(2, 1)" "(3, 1)"
	testPlayerCurrentLocation	getPlayerCurrentLocation()	"(2, 2)"
	testInterconnectivity	getDungeon()	getNeighbours().size()
	testTreasureInRightLocation	getDungeon() Dungeon.isCave() Dungeon.getTreaure	TRUE
	testTreasurePercentage	getDungeon() Dungeon.isCave() Dungeon.getTreaure	
	testPlayerTravelStatus	movePlayer(Direction.WEST) movePlayer(Direction.SOUTH)	"(2, 2), (2, 1), (3, 1)"
	testPlayerPickingTreasure	movePlayer(Direction.WEST) movePlayer(Direction.SOUTH) movePlayer(Direction.EAST) movePlayer(Direction.EAST) movePlayer(Direction.EAST)	"(2, 2) (2, 1) (3, 1), (3, 2) (3, 3) (3, 4)"
	testTreasureNullAfterPicking	getDungeon() locationInfo() movePlayer(Direction.WEST) movePlayer(Direction.SOUTH) movePlayer(Direction.WEST) movePlayer(Direction.NORTH) Player.travelStatus()	"(2, 2) (2, 1, (3, 1) (3, 0) (2, 0)"
	testIllegalMove	movePlayer(Direction.NORTH)	IllegalArgumentException
	testDungeonCreation	getDungeon() getNeighbours().size()	
	testPathLength	getPlayerStartLocation() getPlayerEndLocation()	distance = getPlayerStartLocation() - getPlayerEndLocation() distance >= 5
	testMonsterLocation	getDungeon() isCave() getMonster()	TRUE

	testNumberOfMonsters	getDungeon() isCave() getMonster()	 7
	testPickArrows	getPlayerStartLocation().getName() getPlayerEndLocation().getName() pickTreasure(TreasureType) getPlayer().getTreasureBag().get(TreasureType)	(5, 3) (3, 0) 5
	testShootArrowTwice	 getPlayerStartLocation().getName() getPlayerEndLocation().getName() pickTreasure(TreasureType) getPlayer().getTreasureBag().get(TreasureType) movePlayer() shootArrows(Direction, 1) shootArrows(Direction, 1) getDungeon()[i][j].getMonster().getHealth()	(5, 3) (3, 0) 4 0
	testArrowCount	getDungeon() getTreasure().containsKey(TreasureType)	 5
	testIsGameOver	getPlayerStartLocation().getName() getPlayerEndLocation().getName() isGameOver() pickTreasure(TreasureType) movePlayer() shootArrows(Direction) shootArrows(Direction) movePlayer() shootArrows(Direction) shootArrows(Direction) isGameOver()	(5, 3) (3, 0) False True
	testMonsterInEndLocation	getPlayerEndLocation().getName() getMonster().getName() getMonster().getHealth()	(3, 0) "Otyugh" 100
	testMonsterInStartLocation	getPlayerStartLocation().getName() getMonster	(5, 3) Null
	testArrowGoingThroughTunnel	getPlayerStartLocation().getName() getPlayerEndLocation().getName() shootArrows(Direction) getDungeon[i][j].getMonster().getHealth()	(5, 3) (3, 0) 50
	testArrowMissesMonster	getPlayerStartLocation().getName() getPlayerEndLocation().getName() shootArrows(Direction) getDungeon[i][j].getMonster().getHealth()	(5, 3) (3, 0) 100
	testTreasureCountDecrease	 getPlayerStartLocation().getName() getPlayerEndLocation().getName() movePlayer(Direction) movePlayer(Direction) pickTreasure(TreasureType) getPlayer().getCurrentLocation().getTreasure().get(TreasureType)	(5, 3) (3, 0) 2

GameTestWrapping			
	testGameStateWrapping	GameState(-10, -10, -1, "wrapping", 20, null)	IllegalArgumentException
	testGetPlayer	GameState.getPlayer().getName()	"Player"
	testIsGameOver	GameState.isGameOver() GameState.movePlayer(Direction.NORTH); GameState.movePlayer(Direction.NORTH); GameState.movePlayer(Direction.NORTH); GameState.movePlayer(Direction.NORTH); GameState.movePlayer(Direction.EAST); GameState.movePlayer(Direction.EAST); GameState.movePlayer(Direction.EAST);	FALSE TRUE
	testGetAvailableDirectionsFromPlayerPosition	GameState.getAvailableDirectionsFromPlayerPosition().toString() GameState.getAvailableDirectionsFromPlayerPosition().size()	"[SOUTH, EAST, WEST]" 3
	testGetPlayerStartLocation	GameState.getPlayerStartLocation().toString()	"(2, 4)"
	testGetPlayerEndLocation	GameState.getPlayerEndLocation().toString()	"(4, 7)"
	testMovePlayer	GameState.getDungeon() GameState.getPlayer().getCurrentLocation().toString() GameState.movePlayer(Direction.EAST) GameState.getPlayer().getCurrentLocation().toString() GameState.movePlayer(Direction.NORTH) GameState.movePlayer(Direction.NORTH) GameState.movePlayer(Direction.EAST) GameState.movePlayer(Direction.NORTH) GameState.getPlayer().getCurrentLocation().toString()	"(2, 4)" "(2, 5)" "(9, 6)"
	TestTreasureInRightLocation	GameState.getDungeon() isCave() getTreasure()	TRUE
	testToStringTravelStatus	GameState.getDungeon() isCave() getTreasure() toStringPlayerTravelStatus()	"Player has traveled to the following locations: [(2, 4)].\n" + "Treasures: {}"
	testPlayerPickingTreasure	GameState.movePlayer(Direction.EAST); GameState.movePlayer(Direction.EAST); GameState.movePlayer(Direction.SOUTH); GameState.movePlayer(Direction.SOUTH); GameState.movePlayer(Direction.WEST); GameState.movePlayer(Direction.WEST); GameState.movePlayer(Direction.SOUTH); GameState.movePlayer(Direction.EAST); GameState.movePlayer(Direction.EAST); toStringPlayerTravelStatus()	"Player has traveled to the following locations: [(2, 4) (2, 5) (2, 6) " + "(3, 6) (4, 6) (4, 5) (4, 4) (5, 4) (5, 5) (5, 6)].\n" + "Treasures: {DIAMONDS=14, RUBIES=9, SAPPHIRES=15}"
	testTreasureNullAfterPick	GameState.movePlayer(Direction.EAST) GameState.movePlayer(Direction.EAST) GameState.movePlayer(Direction.SOUTH) printPlayerTravelStatus() movePlayer(Direction.WEST) movePlayer(Direction.EAST) printPlayerTravelStatus()	"Player has traveled to the following locations: [(2, 4) (2, 5) " + "(2, 6) (3, 6)].\n" + Treasures: {DIAMONDS=7, RUBIES=2, SAPPHIRES=9}" Player has traveled to the following locations: [(2, 4) (2, 5) " + "(2, 6) (3, 6) (3, 5) (3, 6)].\n" + "Treasures: {DIAMONDS=7, RUBIES=2, SAPPHIRES=9}"
	testIllegalMove	movePlayer(Direction.NORTH)	IllegalArgumentException
	testPathLength	getPlayerStartLocation() getPlayerEndLocation()	distance = getPlayerStartLocation() - getPlayerEndLocation() distance >= 5

	testPlayerMovementsAllDirections	GameState(6, 10, 1000, "wrapping", 20, random) getAvailableDirectionsFromPlayerPosition().toString() movePlayer(Direction.NORTH) getPlayerCurrentLocation().toString() getAvailableDirectionsFromPlayerPosition().toString() movePlayer(Direction.SOUTH) getPlayerCurrentLocation().toString() getAvailableDirectionsFromPlayerPosition().toString() movePlayer(Direction.EAST) getPlayerCurrentLocation().toString() getAvailableDirectionsFromPlayerPosition().toString() movePlayer(Direction.WEST) getPlayerCurrentLocation().toString()	"[SOUTH, WEST, EAST, NORTH]" "(3, 8)" "[EAST, WEST, SOUTH, NORTH]" "(4, 8)" "[WEST, EAST, SOUTH, NORTH]" "(4, 9)" "[WEST, EAST, SOUTH, NORTH]" "(4, 8)"

Location			
	testIncorrectLocation	new Location()	IllegalArgumentException
	testGetName	getName()	"(0, 0)"
	testGetRowCoordinate	getRowCoordinate()	0
	testGetColCoordinate	getColCoordinate()	0
	testJoinLocationToNorthDirection	Location.joinLocationToNorthDirection(Location()) Location.getNeighbours().get(Direction.NORTH)	Location(0, 2, rand)
	testJoinLocationToSouthDirection	Location.joinLocationToSouthDirection(Location()) Location.getNeighbours().get(Direction.SOUTH)	Location(1, 2, rand)
	testJoinLocationToEastDirection	Location.joinLocationToEastDirection(Location()) Location.getNeighbours().get(Direction.EAST)	Location(1, 2, rand)
	testJoinLocationToWestDirection	Location.joinLocationToWestDirection(Location()) Location.getNeighbours().get(Direction.WEST)	Location(1, 2, rand)
	testIsCave	Location.joinLocationToNorthDirection(Location()) Location.joinLocationToSouthDirection(Location()) Location.joinLocationToEastDirection(Location()) Location.joinLocationToWestDirection(Location()) Location.isCave()	TRUE
	testSetTreasure	Location.joinLocationToNorthDirection(Location()) Location.joinLocationToSouthDirection(Location()) Location.joinLocationToEastDirection(Location()) Location.joinLocationToWestDirection(Location()) Location.setTreasure()	null
	testSetTreasureInTunnel	Location.joinLocationToNorthDirection(Location()) Location.joinLocationToSouthDirection(Location()) Location.setTreasure()	null
	testGetTreasure	Location.joinLocationToNorthDirection(Location()) Location.joinLocationToSouthDirection(Location()) Location.joinLocationToEastDirection(Location()) Location.joinLocationToWestDirection(Location()) Location.setTreasure() Location.getTreasure()	"{SAPPHIRES=4, RUBIES=8, DIAMONDS=9}"

	testGetNeighbours	Location.joinLocationToNorthDirection(Location()) Location.joinLocationToSouthDirection(Location()) Location.joinLocationToEastDirection(Location()) Location.joinLocationToWestDirection(Location()) Location.getNeighbours().size() Location.getNeighbours().get(Direction.NORTH) Location.getNeighbours().get(Direction.SOUTH)	 4 Location(0, 2, rand) Location(2, 2, rand)
	testLocationInfo	Location.joinLocationToNorthDirection(Location()) Location.setTreasure() Location.LocationInfo()	"(0, 0), treasure: {SAPPHIRES=4, RUBIES=8, DIAMONDS=9}, " + "neighbours: {NORTH=(0, 2)}"
Player			
	testPlayerNull	Player(null)	IllegalArgumentException
	testGetName	Player.getName()	"Player"
	testaddTreasure	Treasure.getTreasure()	
	testGetTreasure	Player.getTreasures()	
	testAddTwoTreasures	Player.addTreasure(Treasure) Player.addTreasure(Treasure)	
	testGetCurrentLocation	Player.setCurrentLocation(Location(0, 0, random))	Location(0, 0, random))
	testGetCurrentLocationEmpty	Player.getCurrentLocation()	null
	testSetCurrentLocation	Player.getCurrentLocation()	Player.setCurrentLocation(Location(0, 0, random))
	testMovePlayer	Player.getCurrentLocation() Player.getCurrentLocation()	Player.setCurrentLocation(Location(0, 0, random)) Player.setCurrentLocation(Location(0, 1, random))
	testTravelStatus	Player.setCurrentLocation(Location(0, 0, random)) Player.setCurrentLocation(Location(1, 0, random)) Player.setCurrentLocation(Location(2, 0, random)) Player.toStringTravelStatus()	 "Player has traveled to the following locations: [(0, 0) (1, 0) (2, 0)].\n" + "Treasures: {}"
	testTravelStatus	getDungeon() locationInfo() movePlayer(Direction.WEST) movePlayer(Direction.SOUTH) movePlayer(Direction.WEST) movePlayer(Direction.NORTH) Player.toStringTravelStatus()	 "Player has traveled to the following locations: [].\n" + "Treasures: {}"
	testPickTreasure	movePlayer(Direction.SOUTH) setTreasure(Itreasure) getTreasure.toString() setCurrentLocation() pickTreasure(TreasureType.RUBIES) pickTreasure(TreasureType.SAPPHIRES) pickTreasure(TreasureType.RUBIES)	 "Sapphires 2, Rubies 8, Diamonds 9" "Sapphires 3, Rubies 10, Arrows 2, Diamonds 9"
	testPickArrow	addArrow() addArrow() addArrow() Location.getTreasure().toString() Player.setTreasure(location) Player.pickTreasure() Player.getTreasure().toString() Location.getTreasure().toString()	 Arrows 3 Arrows 4 Arrows 1
	testGetHealth	Player.getHealth()	100
	testReduceHealth	reduceHealth()	0

	testIsAlive	isAlive() reduceHealth() isAlive()	True False
	testAfterShooting	movePlayer(Direction.SOUTH) setCurrentLocation(location) Player.shootArrows(Direction.SOUTH, 1) Player.shootArrows(Direction.SOUTH, 1) Player.shootArrows(Direction.SOUTH, 1) Player.shootArrows(Direction.SOUTH, 1) Player.shootArrows(Direction.NORTH, 1)	 Arrows 2 Arrows 1 Arrows 0 IllegalArgumentException "No Arrows remaining" Illegal Argument Exception "Direction illegal"
	testHalfSurvivalDies	Location.attachLocation(Direction.SOUTH) Location.setTreasure(Itreasure) Location.setMonster() Player.getHealth() Player.isAlive() Location.getMonster().getHealth() Location.getMonster().reduceHealth() Location.getMonster().getHealth() Player.setCurrentLocation(Ilocation) Player.getHealth() Player.isAlive()	 100 TRUE 100 50 0 FALSE
	testHalfSurvivalSurvives	Location.attachLocation(Direction.SOUTH) Location.setTreasure(Itreasure) Location.setMonster() Player.getHealth() Player.isAlive() Location.getMonster().getHealth() Location.getMonster().reduceHealth() Location.getMonster().getHealth() Player.setCurrentLocation(Ilocation) Player.getHealth() Player.isAlive()	 100 TRUE 100 50 100 TRUE

Treasure			
	testIncorrectTreasure	Treasure(null)	IllegalArgumentException

	testGetTreasure	Treasure.getTreasure()	expected.put(TreasureType.RUBIES, 4);
			expected.put(TreasureType.SAPPHIRES, 4);
			expected.put(TreasureType.DIAMONDS, 1);

	testRemoveTreasure	movePlayer(Direction.EAST) setTreasure(Treasure) getTreasure().toString() removeTreasure(TreasureType.RUBIES) removeTreasure(TreasureType.RUBIES)	 "Sapphires 4, Rubies 3, Diamonds 1" "Sapphires 4, Rubies 2, Diamonds 1" "Sapphires 4, Rubies 1, Diamonds 1"
	testAddArrows	addArrows() addArrows() addArrows() toString()	 "Sapphires 2, Rubies 8, Arrows 3, Diamonds 9"

	testRemoveArrows	addArrows() addArrows() addArrows() toString() removeTreasure() toString()	"Sapphires 2, Rubies 8, Arrows 3, Diamonds 9" "Sapphires 2, Rubies 8, Arrows 2, Diamonds 9"
Monster			
	testGetName	Monster.getName()	"Otyugh"
	testGetCurrentLocation	Monster.setCurrentLocation(Location(0, 0, random))	Location(0, 0, random))
	testSetCurrentLocation	Monster.getCurrentLocation()	Monster.setCurrentLocation(Location(0, 0, random))
	testGetHealth	Monster.getHealth()	100
	testReduceHealth	reduceHealth()	0
	testIsAlive	isAlive() reduceHealth() isAlive()	True False
Controller			
	testMovePlayer	Controller(StringReader, Appendable, IgameState) Controller.playGame() Appendable.toString.contains("Moving NORTH")	TRUE
	testPickTreasureArrow	Controller(StringReader, Appendable, IgameState) Controller.playGame() Appendable.toString.contains("Picking up Arrows") Appendable.toString.contains("You are in a cave and you have 4 Arrows.")	TRUE TRUE
	testShootArrow	Controller(StringReader, Appendable, IgameState) Controller.playGame() Appendable.toString.contains("Shooting in NORTH at distance of 1")	TRUE
	testShootInvalidDirection	Controller(StringReader, Appendable, IgameState) Controller.playGame() Appendable.toString.contains("Direction is not valid")	TRUE
	testShootInvalidDistance	Controller(StringReader, Appendable, IgameState) Controller.playGame() Appendable.toString.contains("Distance is not valid")	TRUE
	testShootToKillAndEnterCave	Controller(StringReader, Appendable, IgameState) Controller.playGame() Appendable.toString.contains("You see a dead monster here.")	TRUE
	testQuitGame	Controller(StringReader, Appendable, IgameState) Controller.playGame() Appendable.toString.contains("You quit the game..")	TRUE
	testInvalidCommandInput	Controller(StringReader, Appendable, IgameState) Controller.playGame() Appendable.toString.contains("Not a valid command.")	TRUE
	testInvalidCavesCount	Controller(StringReader, Appendable, IgameState) Controller.playGame() Appendable.toString.contains("Invalid caves count")	TRUE
	testWinGame	Controller(StringReader, Appendable, IgameState) Controller.playGame() Appendable.toString.contains("You win! You made it to the end.")	TRUE
	testLessPungentSmell	Controller(StringReader, Appendable, IgameState) Controller.playGame() Appendable.toString.contains("You slightly smell something nearby")	TRUE

	testMorePungentSmell	Controller(StringReader, Appendable, IgameState) Controller.playGame() Appendable.toString.contains("You smell something terrible nearby")	TRUE
	testMoveNorth	Controller(StringReader, Appendable, IgameState) Controller.playGame() Appendable.toString.contains("Moving NORTH")	TRUE
	testMoveSouth	Controller(StringReader, Appendable, IgameState) Controller.playGame() Appendable.toString.contains("Moving SOUTH")	TRUE
	testMoveEast	Controller(StringReader, Appendable, IgameState) Controller.playGame() Appendable.toString.contains("Moving EAST")	TRUE
	testMoveWest	Controller(StringReader, Appendable, IgameState) Controller.playGame() Appendable.toString.contains("Moving WEST")	TRUE
	testTreasureInfo	Controller(StringReader, Appendable, IgameState) Controller.playGame() Appendable.toString.contains("You find 9 Diamonds, 3 Sapphires, 9 Rubies")	TRUE