**Main**
- main(args: String[]): void

**Helper**
- isInt(str: String): boolean
- isIn(strs: String[], s: String): boolean

**DungeonController**
- playGame(): void
- startGame(wrap: String, row: int, col: int, interconnectivity: int, percentage: int, numMonster: int): void
- restart(): void
- move(direction: String): void
- pick(): void
- shoot(direction: String, distance: int): void
- isWin(): boolean
- isLoose(): boolean
- canShoot(): boolean
- handleCellClick(diresction: String): void

**DungeonControllerImpl**
- prevLocations: Location[][]
- DungeonControllerImpl(model: DungeonModel, view: DungeonView)
- playGame(): void
- startGame(wrap: String, row: int, col: int, interconnectivity: int, percentage: int, numMonster: int): void
- restart(): void
- move(direction: String): void
- pick(): void
- shoot(direction: String, distance: int): void
- isWin(): boolean
- isLoose(): boolean
- canShoot(): boolean
- handleCellClick(diresction: String): void

**Treasure**
- Diamond
- Ruby
- Sapphire

**MyRandom**
- random: Random
- randOrNot: boolean
- MyRandom(randOrNot: boolean)
- nextInt(max: int): int

**DungeonView**
- refresh(): void
- addClickListener(listener: DungeonController): void
- show(playerLocation: String): void

**JFrame**

**ActionListener**

**JPanel**

**DungeonGuiView**
- serialVersionUID: long
- menuBar: JMenuBar
- fileMenu: JMenu
- close: JMenuItem
- buttonPanel: JPanel
- exit: JButton
- direction: JTextField
- move: JButton
- shootDirection: JTextField
- shoot: JButton
- pick: JButton
- dungeonConfig: JTextField
- generate: JButton
- north: JButton
- east: JButton
- south: JButton
- west: JButton
- textArea: JTextArea
- DungeonGuiView(model: DungeonModel)
- refresh(): void
- addClickListener(listener: DungeonController): void
- actionPerformed(e: ActionEvent): void
- setPanel(row: int, col: int): void
- show(information: String): void

**DungeonPanel**
- serialVersionUID: long
- board: Location[][]
- DungeonPanel(row: int, col: int)
- setBoard(board: Location[][]): void
- paintComponent(g: Graphics): void

**DungeonModel**
- create(wrap: String, row: int, col: int, interconnectivity: int): String
- setStartEnd(): void
- setStartEnd(start: int, end: int): void
- addTreasure(percentage: int): void
- addTreasure(percentage: int, noRandom: boolean): void
- enter(): void
- playerDescription(): String
- playerLocation(): String
- move(location: Location): void
- pick(): void
- getRow(): int
- getColumn(): int
- getInterconnectivity(): int
- getNextLocation(location: Location, direction: String): Location
- addMonster(num: int): void
- detectMonster(): int
- shoot(direction: String, distance: int): int
- getLocations(): HashMap<Integer,Location>
- getCaves(): HashMap<Integer,Location>
- getBoard(): Location[][]
- setBoard(board: Location[][]): void

**DungeonModelImpl**
- locations: HashMap<Integer,Location>
- caves: HashMap<Integer,Location>
- edges: HashMap<Integer,HashSet<Integer>>
- row: int
- col: int
- interconnectivity: int
- create(wrap: String, row: int, col: int, interconnectivity: int): String
- setStartEnd(): void
- setStartEnd(start: int, end: int): void
- addTreasure(percentage: int): void
- addTreasure(percentage: int, noRandom: boolean): void
- enter(): void
- playerDescription(): String
- playerLocation(): String
- move(location: Location): void
- pick(): void
- generatePotentialEdges(wrap: String, row: int, col: int): HashMap<Integer,HashSet<Integer>>
- findunionFind: HashMap<Integer,Integer>, x: int): int
- union(unionFind: HashMap<Integer,Integer>, x: int, y: int): void
- generateLocations(): void
- kruskal(potentialEdges: HashMap<Integer,HashSet<Integer>>, restEdges: HashMap<Integer,HashSet<Integer>>, unionFind: HashMap<Integer,Integer>, interconnectivity: int): void
- getTooClose(tooClose: HashSet<Integer>, edges: HashMap<Integer,HashSet<Integer>>): void
- getRow(): int
- getColumn(): int
- getInterconnectivity(): int
- getNextLocation(currLocation: Location, direction: String): Location
- addMonster(num: int): void
- detectMonster(): int
- shoot(direction: String, distance: int): int
- getLocations(): HashMap<Integer,Location>
- getCaves(): HashMap<Integer,Location>
- getBoard(): Location[][]
- setBoard(board: Location[][]): void

**Player**
- move(location: Location): void
- getTreasures(): HashMap<Treasure,Integer>
- addTreasure(treasure: Treasure, num: int): void
- pick(): void
- setArrow(num: int): void
- getArrow(): int
- isDead(): boolean
- setDead(): void
- recover(): void

**PlayerImpl**
- treasures: HashMap<Treasure,Integer>
- arrow: int
- alive: boolean
- PlayerImpl()
- move(location: Location): void
- getTreasures(): HashMap<Treasure,Integer>
- addTreasure(treasure: Treasure, num: int): void
- pick(): void
- setArrow(num: int): void
- getArrow(): int
- isDead(): boolean
- setDead(): void
- recover(): void

**Location**
- getNorth(): boolean
- getWest(): boolean
- getSouth(): boolean
- getEast(): boolean
- getIndex(): int
- addTreasure(): void
- getTreasures(): HashMap<Treasure,Integer>
- resetTreasure(): void
- getRow(): int
- getColumn(): int
- getNextLocation(direction: String): int
- setArrow(): void
- getArrow(): boolean
- resetArrow(): void
- addMonster(): void
- getMonsterHealth(): List<Integer>
- setMonsterHealth(health: int): void
- hasMonster(): boolean
- setVisited(): void
- isVisited(): boolean
- isHere(): boolean
- setPlayerHere(here: boolean): void

random  view  controller  model  gamePanel  prevStart  prevEnd  getPlayer()  getPlayer()  player  getLocation()  location  start  end  getStart()  getEnd()  getLocation()  getStart()  getEnd()  getLocation()  getEnd()  getLocation()  1

| Method | Input | Expected |
|---|---|---|
| addArrow | Percentage of location with arrow | Correct percentage |
| shoot | Specify direction and distance | Hurt monster if there exists |
| move | String of direction | Controller moves model successfully |
| printInformation | Nothing | Controller successfully print information |
| pick | Nothing | Controller control model to pick arrow ar |