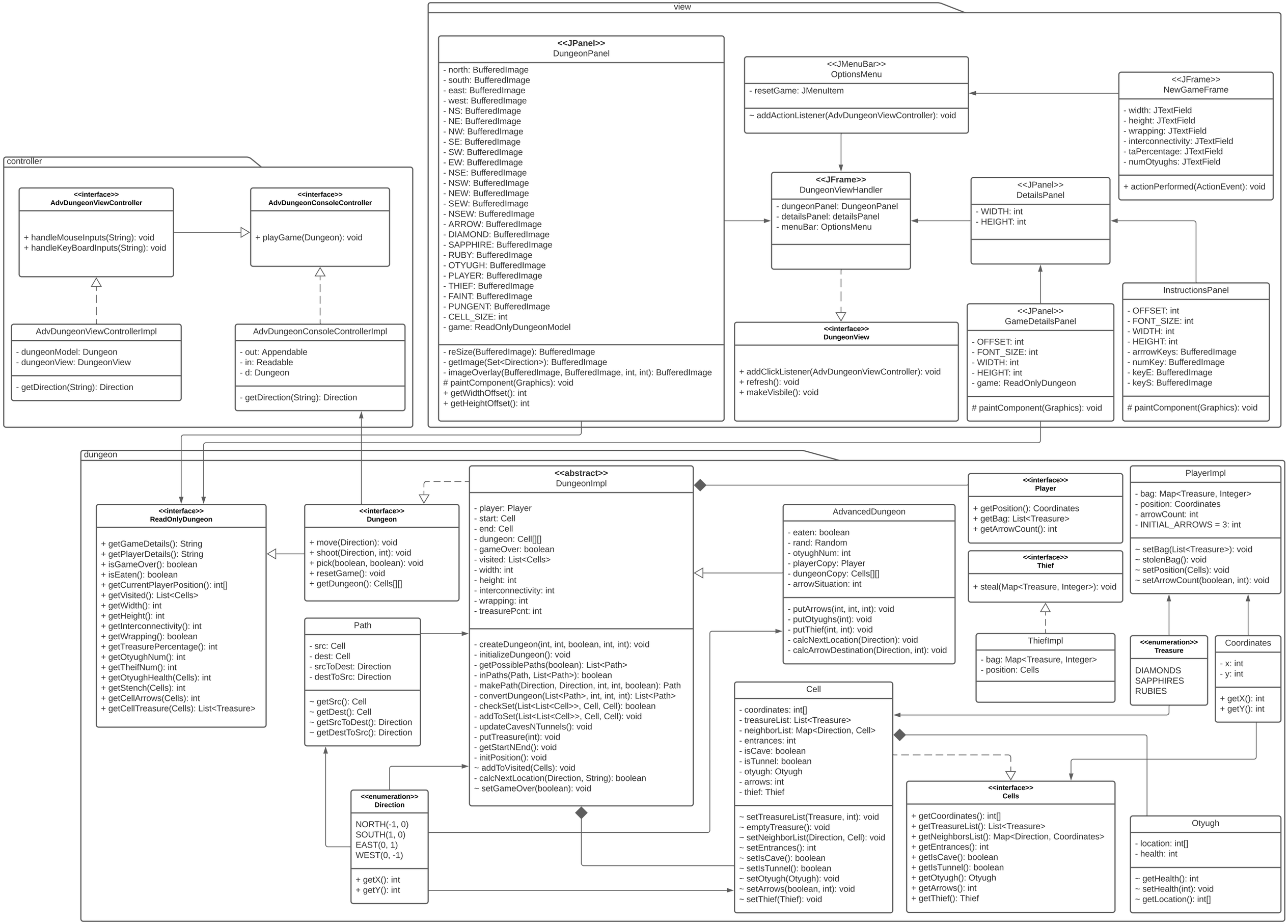
view
<<JPanel>>
DungeonPanel
- north: BufferedImage
- south: BufferedImage
- east: BufferedImage
- west: BufferedImage
- NS: BufferedImage
- NE: BufferedImage
- NW: BufferedImage
- SE: BufferedImage
- SW: BufferedImage
- EW: BufferedImage
- NSE: BufferedImage
- NSW: BufferedImage
- NEW: BufferedImage
- SEW: BufferedImage
- NSEW: BufferedImage
- ARROW: BufferedImage
- DIAMOND: BufferedImage
- SAPPHIRE: BufferedImage
- RUBY: BufferedImage
- OTYUGH: BufferedImage
- PLAYER: BufferedImage
- THIEF: BufferedImage
- FAINT: BufferedImage
- PUNGENT: BufferedImage
- CELL_SIZE: int
- game: ReadOnlyDungeonModel
- reSize(BufferedImage): BufferedImage
- getImage(Set<Direction>): BufferedImage
- imageOverlay(BufferedImage, BufferedImage, int, int): BufferedImage
paintComponent(Graphics): void
+ getWidthOffset(): int
+ getHeightOffset(): int
<<JMenuBar>>
OptionsMenu
- resetGame: JMenuItem
~ addActionListener(AdvDungeonViewController): void
<<JFrame>>
NewGameFrame
- width: JTextField
- height: JTextField
- wrapping: JTextField
- interconnectivity: JTextField
- taPercentage: JTextField
- numOtyughs: JTextField
+ actionPerformed(ActionEvent): void
<<JFrame>>
DungeonViewHandler
- dungeonPanel: DungeonPanel
- detailsPanel: detailsPanel
- menuBar: OptionsMenu
<<JPanel>>
DetailsPanel
- WIDTH: int
- HEIGHT: int
<<interface>>
DungeonView
+ addClickListener(AdvDungeonViewController): void
+ refresh(): void
+ makeVisbile(): void
<<JPanel>>
GameDetailsPanel
- OFFSET: int
- FONT_SIZE: int
- WIDTH: int
- HEIGHT: int
- game: ReadOnlyDungeon
paintComponent(Graphics): void
InstructionsPanel
- OFFSET: int
- FONT_SIZE: int
- WIDTH: int
- HEIGHT: int
- arrrowKeys: BufferedImage
- numKey: BufferedImage
- keyE: BufferedImage
- keyS: BufferedImage
paintComponent(Graphics): void
controller
<<interface>>
AdvDungeonViewController
+ handleMouseInputs(String): void
+ handleKeyBoardInputs(String): void
<<interface>>
AdvDungeonConsoleController
+ playGame(Dungeon): void
AdvDungeonViewControllerImpl
- dungeonModel: Dungeon
- dungeonView: DungeonView
- getDirection(String): Direction
AdvDungeonConsoleControllerImpl
- out: Appendable
- in: Readable
- d: Dungeon
- getDirection(String): Direction
dungeon
<<interface>>
ReadOnlyDungeon
+ getGameDetails(): String
+ getPlayerDetails(): String
+ isGameOver(): boolean
+ isEaten(): boolean
+ getCurrentPlayerPosition(): int[]
+ getVisited(): List<Cells>
+ getWidth(): int
+ getHeight(): int
+ getInterconnectivity(): int
+ getWrapping(): boolean
+ getTreasurePercentage(): int
+ getOtyughNum(): int
+ getTheifNum(): int
+ getOtyughHealth(Cells): int
+ getStench(Cells): int
+ getCellArrows(Cells): int
+ getCellTreasure(Cells): List<Treasure>
<<interface>>
Dungeon
+ move(Direction): void
+ shoot(Direction, int): void
+ pick(boolean, boolean): void
+ resetGame(): void
+ getDungeon(): Cells[][]
<<abstract>>
DungeonImpl
- player: Player
- start: Cell
- end: Cell
- dungeon: Cell[][]
- gameOver: boolean
- visited: List<Cells>
- width: int
- height: int
- interconnectivity: int
- wrapping: int
- treasurePcnt: int
- createDungeon(int, int, boolean, int, int): void
- initializeDungeon(): void
- getPossiblePaths(boolean): List<Path>
- inPaths(Path, List<Path>): boolean
- makePath(Direction, Direction, int, int, boolean): Path
- convertDungeon(List<Path>, int, int, int): List<Path>
- checkSet(List<List<Cell>>, Cell, Cell): boolean
- addToSet(List<List<Cell>>, Cell, Cell): void
- updateCavesNTunnels(): void
- putTreasure(int): void
- getStartNEnd(): void
- initPosition(): void
~ addToVisited(Cells): void
- calcNextLocation(Direction, String): boolean
~ setGameOver(boolean): void
AdvancedDungeon
- eaten: boolean
- rand: Random
- otyughNum: int
- playerCopy: Player
- dungeonCopy: Cells[][]
- arrowSituation: int
- putArrows(int, int, int): void
- putOtyughs(int): void
- putThief(int, int): void
- calcNextLocation(Direction): void
- calcArrowDestination(Direction, int): void
<<interface>>
Player
+ getPosition(): Coordinates
+ getBag: List<Treasure>
+ getArrowCount(): int
PlayerImpl
- bag: Map<Treasure, Integer>
- position: Coordinates
- arrowCount: int
- INITIAL_ARROWS = 3: int
~ setBag(List<Treasure>): void
~ stolenBag(): void
~ setPosition(Cells): void
~ setArrowCount(boolean, int): void
<<interface>>
Thief
+ steal(Map<Treasure, Integer>): void
ThiefImpl
- bag: Map<Treasure, Integer>
- position: Cells
<<enumeration>>
Treasure
DIAMONDS
SAPPHIRES
RUBIES
Coordinates
- x: int
- y: int
+ getX(): int
+ getY(): int
Path
- src: Cell
- dest: Cell
- srcToDest: Direction
- destToSrc: Direction
~ getSrc(): Cell
~ getDest(): Cell
~ getSrcToDest(): Direction
~ getDestToSrc(): Direction
<<enumeration>>
Direction
NORTH(-1, 0)
SOUTH(1, 0)
EAST(0, 1)
WEST(0, -1)
+ getX(): int
+ getY(): int
Cell
- coordinates: int[]
- treasureList: List<Treasure>
- neighborList: Map<Direction, Cell>
- entrances: int
- isCave: boolean
- isTunnel: boolean
- otyugh: Otyugh
- arrows: int
- thief: Thief
~ setTreasureList(Treasure, int): void
~ emptyTreasure(): void
~ setNeighborList(Direction, Cell): void
~ setEntrances(): int
~ setIsCave(): boolean
~ setIsTunnel(): boolean
~ setOtyugh(Otyugh): void
~ setArrows(boolean, int): void
~ setThief(Thief): void
<<interface>>
Cells
+ getCoordinates(): int[]
+ getTreasureList(): List<Treasure>
+ getNeighborsList(): Map<Direction, Coordinates>
+ getEntrances(): int
+ getIsCave(): boolean
+ getIsTunnel(): boolean
+ getOtyugh(): Otyugh
+ getArrows(): int
+ getThief(): Thief
Otyugh
- location: int[]
- health: int
~ getHealth(): int
~ setHealth(int): void
~ getLocation(): int[]