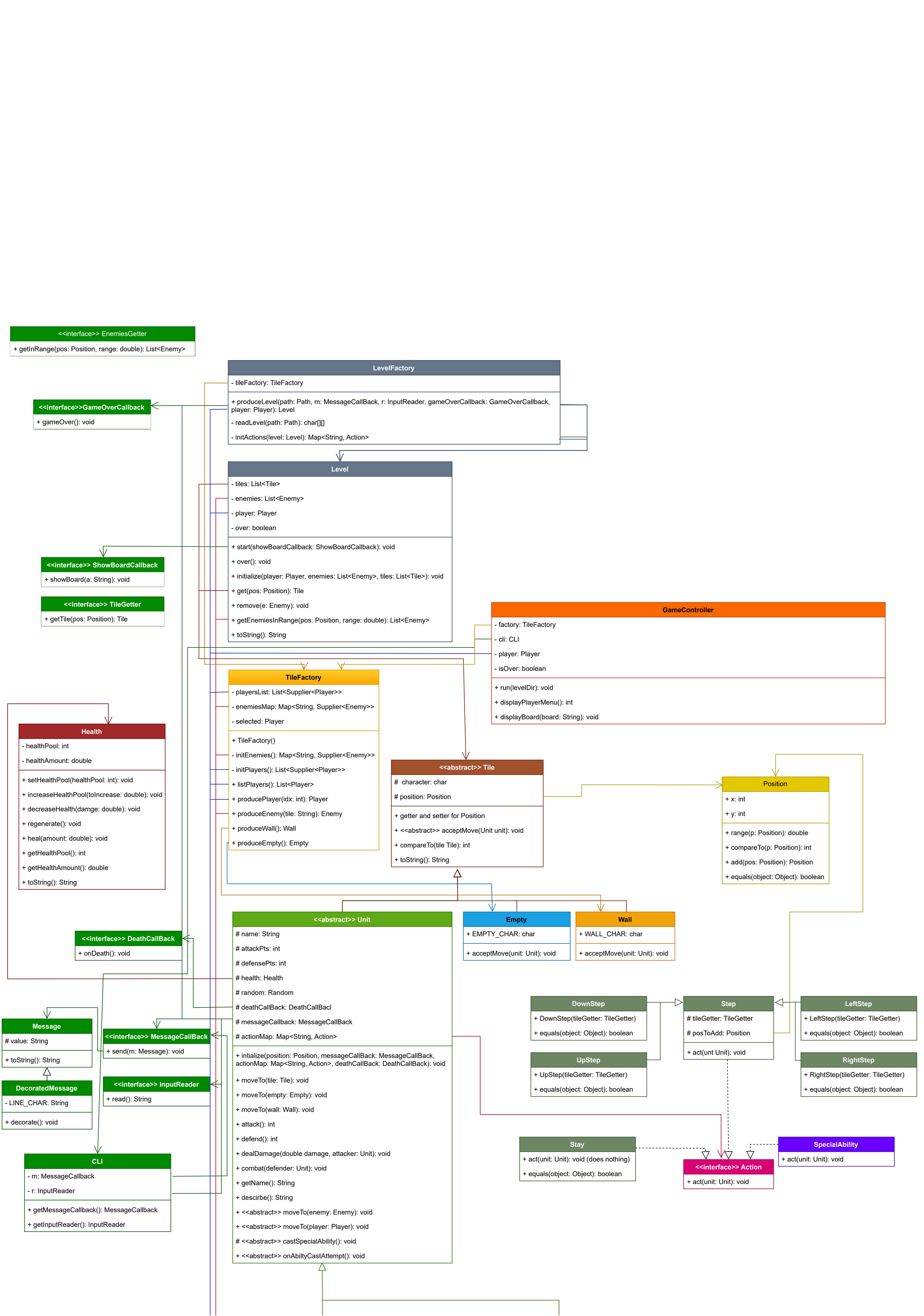
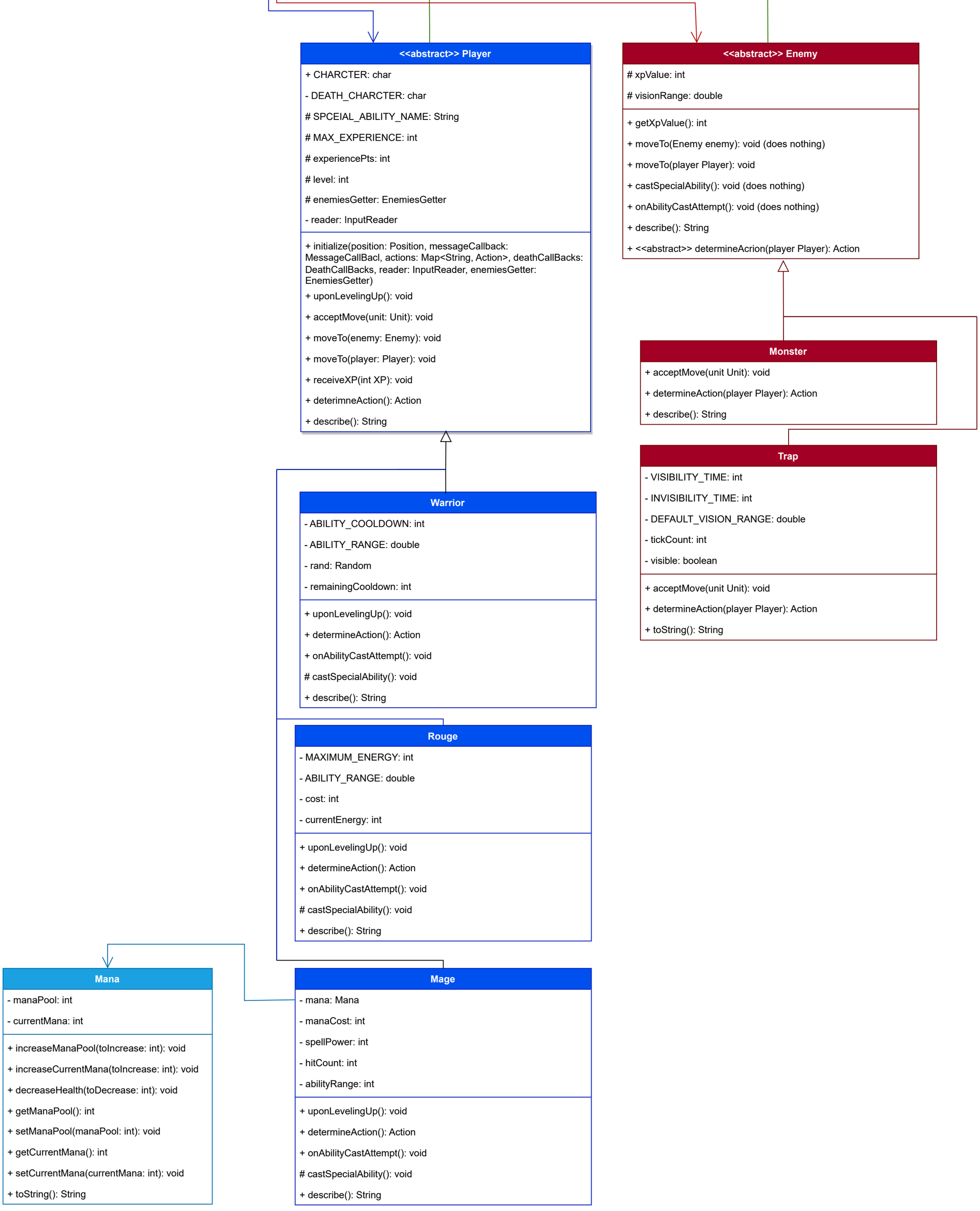

<<interface>> EnemiesGetter
+ getInRange(pos: Position, range: double): List<Enemy>
LevelFactory
- tileFactory: TileFactory
+ produceLevel(path: Path, m: MessageCallBack, r: InputReader, gameOverCallback: GameOverCallback, player: Player): Level
- readLevel(path: Path): char[][]
- initActions(level: Level): Map<String, Action>
<<interface>>GameOverCallback
+ gameOver(): void
Level
- tiles: List<Tile>
- enemies: List<Enemy>
- player: Player
- over: boolean
+ start(showBoardCallback: ShowBoardCallback): void
+ over(): void
+ initialize(player: Player, enemies: List<Enemy>, tiles: List<Tile>): void
+ get(pos: Position): Tile
+ remove(e: Enemy): void
+ getEnemiesInRange(pos: Position, range: double): List<Enemy>
+ toString(): String
<<interface>> ShowBoardCallback
+ showBoard(a: String): void
<<interface>> TileGetter
+ getTile(pos: Position): Tile
GameController
- factory: TileFactory
- cli: CLI
- player: Player
- isOver: boolean
+ run(levelDir): void
+ displayPlayerMenu(): int
+ displayBoard(board: String): void
TileFactory
- playersList: List<Supplier<Player>>
- enemiesMap: Map<String, Supplier<Enemy>>
- selected: Player
+ TileFactory()
- initEnemies(): Map<String, Supplier<Enemy>>
- initPlayers(): List<Supplier<Player>>
+ listPlayers(): List<Player>
+ producePlayer(idx: int): Player
+ produceEnemy(tile: String): Enemy
+ produceWall(): Wall
+ produceEmpty(): Empty
Health
- healthPool: int
- healthAmount: double
+ setHealthPool(healthPool: int): void
+ increaseHealthPool(toIncrease: double): void
+ decreaseHealth(damge: double): void
+ regenerate(): void
+ heal(amount: double): void
+ getHealthPool(): int
+ getHealthAmount(): double
+ toString(): String
<<abstract>> Tile
character: char
position: Position
+ getter and setter for Position
+ <<abstract>> acceptMove(Unit unit): void
+ compareTo(tile Tile): int
+ toString(): String
Position
+ x: int
+ y: int
+ range(p: Position): double
+ compareTo(p: Position): int
+ add(pos: Position): Position
+ equals(object: Object): boolean
<<interface>> DeathCallBack
+ onDeath(): void
<<abstract>> Unit
name: String
attackPts: int
defensePts: int
health: Health
random: Random
deathCallBack: DeathCallBacl
messageCallback: MessageCallBack
actionMap: Map<String, Action>
+ intialize(position: Position, messageCallBack: MessageCallBack, actionMap: Map<String, Action>, deathCallBack: DeathCallBack): void
+ moveTo(tile: Tile): void
+ moveTo(empty: Empty): void
+ moveTo(wall: Wall): void
+ attack(): int
+ defend(): int
+ dealDamage(double damage, attacker: Unit): void
+ combat(defender: Unit): void
+ getName(): String
+ describe(): String
+ <<abstract>> moveTo(enemy: Enemy): void
+ <<abstract>> moveTo(player: Player): void
# <<abstract>> castSpecialAbility(): void
+ <<abstract>> onAbiltyCastAttempt(): void
Empty
+ EMPTY_CHAR: char
+ acceptMove(unit: Unit): void
Wall
+ WALL_CHAR: char
+ acceptMove(unit: Unit): void
Message
value: String
+ toString(): String
DecoratedMessage
- LINE_CHAR: String
+ decorate(): void
<<interface>> MessageCallBack
+ send(m: Message): void
<<interface>> InputReader
+ read(): String
CLI
- m: MessageCallback
- r: InputReader
+ getMessageCallback(): MessageCallback
+ getInputReader(): InputReader
DownStep
+ DownStep(tileGetter: TileGetter)
+ equals(object: Object): boolean
UpStep
+ UpStep(tileGetter: TileGetter)
+ equals(object: Object): boolean
Step
tileGetter: TileGetter
posToAdd: Position
+ act(unt Unit): void
LeftStep
+ LeftStep(tileGetter: TileGetter)
+ equals(object: Object): boolean
RightStep
+ RightStep(tileGetter: TileGetter)
+ equals(object: Object): boolean
Stay
+ act(unit: Unit): void (does nothing)
+ equals(object: Object): boolean
<<interface>> Action
+ act(unit: Unit): void
SpecialAbility
+ act(unit: Unit): void


<<abstract>> Player
+ CHARCTER: char
- DEATH_CHARCTER: char
SPCEIAL_ABILITY_NAME: String
MAX_EXPERIENCE: int
experiencePts: int
level: int
enemiesGetter: EnemiesGetter
- reader: InputReader
+ initialize(position: Position, messageCallback: MessageCallBacl, actions: Map<String, Action>, deathCallBacks: DeathCallBacks, reader: InputReader, enemiesGetter: EnemiesGetter)
+ uponLevelingUp(): void
+ acceptMove(unit: Unit): void
+ moveTo(enemy: Enemy): void
+ moveTo(player: Player): void
+ receiveXP(int XP): void
+ deterimneAction(): Action
+ describe(): String
<<abstract>> Enemy
xpValue: int
visionRange: double
+ getXpValue(): int
+ moveTo(Enemy enemy): void (does nothing)
+ moveTo(player Player): void
+ castSpecialAbility(): void (does nothing)
+ onAbilityCastAttempt(): void (does nothing)
+ describe(): String
+ <<abstract>> determineAcrion(player Player): Action
Monster
+ acceptMove(unit Unit): void
+ determineAction(player Player): Action
+ describe(): String
Trap
- VISIBILITY_TIME: int
- INVISIBILITY_TIME: int
- DEFAULT_VISION_RANGE: double
- tickCount: int
- visible: boolean
+ acceptMove(unit Unit): void
+ determineAction(player Player): Action
+ toString(): String
Warrior
- ABILITY_COOLDOWN: int
- ABILITY_RANGE: double
- rand: Random
- remainingCooldown: int
+ uponLevelingUp(): void
+ determineAction(): Action
+ onAbilityCastAttempt(): void
# castSpecialAbility(): void
+ describe(): String
Rouge
- MAXIMUM_ENERGY: int
- ABILITY_RANGE: double
- cost: int
- currentEnergy: int
+ uponLevelingUp(): void
+ determineAction(): Action
+ onAbilityCastAttempt(): void
# castSpecialAbility(): void
+ describe(): String
Mana
- manaPool: int
- currentMana: int
+ increaseManaPool(toIncrease: int): void
+ increaseCurrentMana(toIncrease: int): void
+ decreaseHealth(toDecrease: int): void
+ getManaPool(): int
+ setManaPool(manaPool: int): void
+ getCurrentMana(): int
+ setCurrentMana(currentMana: int): void
+ toString(): String
Mage
- mana: Mana
- manaCost: int
- spellPower: int
- hitCount: int
- abilityRange: int
+ uponLevelingUp(): void
+ determineAction(): Action
+ onAbilityCastAttempt(): void
# castSpecialAbility(): void
+ describe(): String