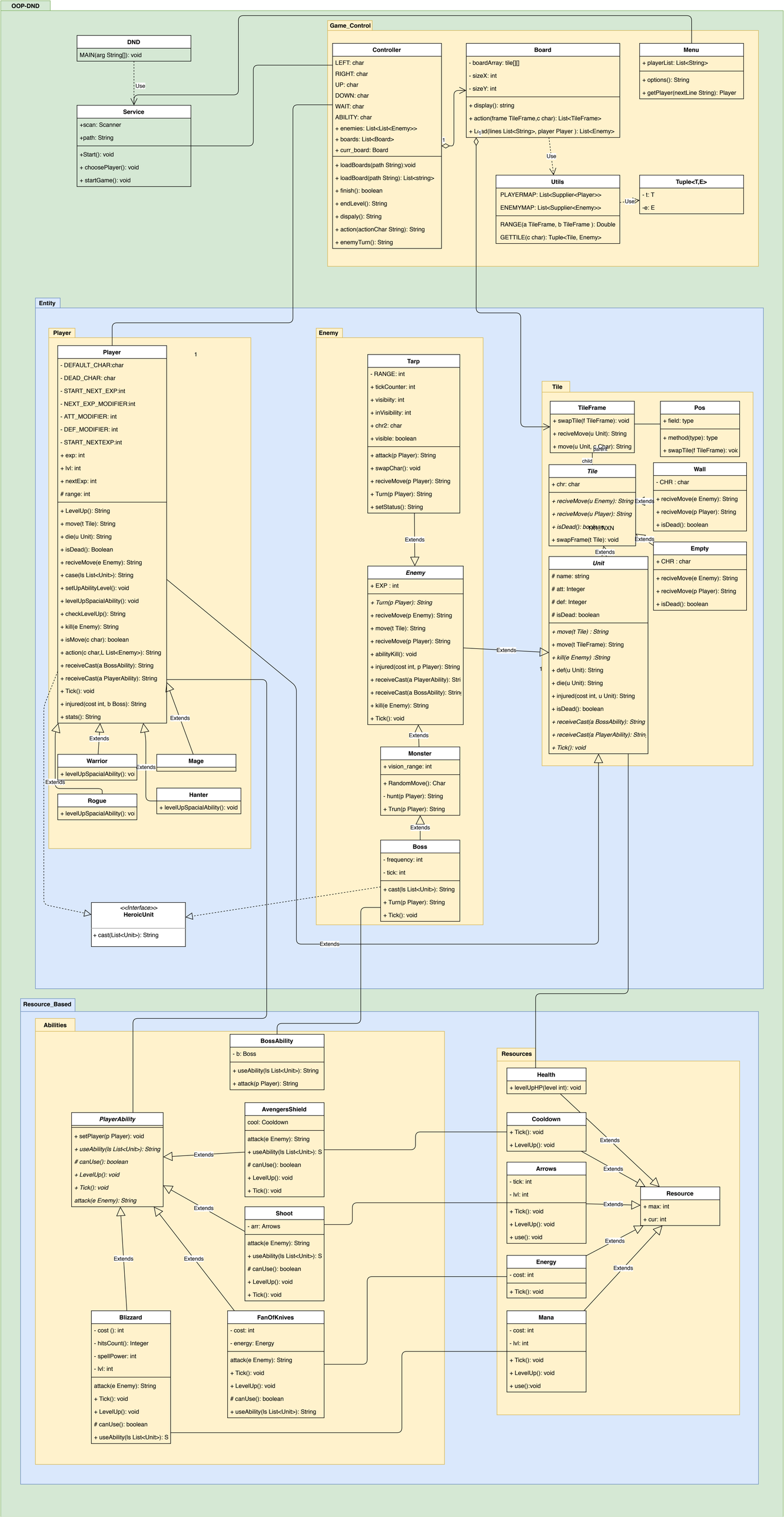
OOP-DND
DND
MAIN(arg String[]): void
Use
Service
+scan: Scanner
+path: String
+Start(): void
+ choosePlayer(): void
+ startGame(): void
Game_Control
Controller
LEFT: char
RIGHT: char
UP: char
DOWN: char
WAIT: char
ABILITY: char
+ enemies: List<List<Enemy>>
+ boards: List<Board>
+ curr_board: Board
+ loadBoards(path String):void
+ loadBoard(path String): List<string>
+ finish(): boolean
+ endLevel(): String
+ dispaly(): String
+ action(actionChar String): String
+ enemyTurn(): String
1
Board
- boardArray: tile[][]
- sizeX: int
- sizeY: int
+ display(): string
+ action(frame TileFrame,c char): List<TileFrame>
+ Load(lines List<String>, player Player): List<Enemy>
Use
Menu
+ playerList: List<String>
+ options(): String
+ getPlayer(nextLine String): Player
Utils
PLAYERMAP: List<Supplier<Player>>
ENEMYMAP: List<Supplier<Enemy>>
RANGE(a TileFrame, b TileFrame): Double
GETTILE(c char): Tuple<Tile, Enemy>
Use
Tuple<T,E>
- t: T
-e: E
Entity
Player
Player
- DEFAULT_CHAR:char
- DEAD_CHAR: char
- START_NEXT_EXP:int
- NEXT_EXP_MODIFIER:int
- ATT_MODIFIER: int
- DEF_MODIFIER: int
- START_NEXTEXP:int
+ exp: int
+ lvl: int
+ nextExp: int
# range: int
+ LevelUp(): String
+ move(t Tile): String
+ die(u Unit): String
+ isDead(): Boolean
+ reciveMove(e Enemy): String
+ case(ls List<Unit>): String
+ setUpAbilityLevel(): void
+ levelUpSpacialAbility(): void
+ checkLevelUp(): String
+ kill(e Enemy): String
+ isMove(c char): boolean
+ action(c char,L List<Enemy>): String
+ receiveCast(a BossAbility): String
+ receiveCast(a PlayerAbility): String
+ Tick(): void
+ injured(cost int, b Boss): String
+ stats(): String
1
Extends
Extends
Extends
Extends
Extends
Warrior
+ levelUpSpacialAbility(): vo
Mage
Rogue
+ levelUpSpacialAbility(): vo
Hanter
+ levelUpSpacialAbility(): void
Enemy
Tarp
- RANGE: int
+ tickCounter: int
+ visibiity: int
+ inVisibility: int
+ chr2: char
+ visible: boolean
+ attack(p Player): String
+ swapChar(): void
+ reciveMove(p Player): String
+ Turn(p Player): String
+ setStatus(): String
Extends
Enemy
+ EXP : int
+ Turn(p Player): String
+ reciveMove(p Enemy): String
+ move(t Tile): String
+ reciveMove(p Player): String
+ abilityKill(): void
+ injured(cost int, p Player): String
+ receiveCast(a PlayerAbility): Stri
+ receiveCast(a BossAbility): String
+ kill(e Enemy): String
+ Tick(): void
Extends
Monster
+ vision_range: int
+ RandomMove(): Char
- hunt(p Player): String
+ Trun(p Player): String
Extends
Boss
- frequency: int
- tick: int
+ cast(ls List<Unit>): String
+ Turn(p Player): String
+ Tick(): void
Extends
1
Tile
TileFrame
+ swapTile(f TileFrame): void
+ reciveMove(u Unit): String
+ move(u Unit, c Char): String
Pos
+ field: type
+ method(type): type
+ swapTile(f TileFrame): void
Parent
child
Tile
+ chr: char
+ reciveMove(u Enemy): String
+ reciveMove(u Player): String
+ isDead(): boolean
+ swapFrame(t Tile): void
Extends
Extends
Extends
Wall
- CHR : char
+ reciveMove(e Enemy): String
+ reciveMove(p Player): String
+ isDead(): boolean
Empty
+ CHR : char
+ reciveMove(e Enemy): String
+ reciveMove(p Player): String
+ isDead(): boolean
Unit
name: string
att: Integer
def: Integer
isDead: boolean
+ move(t Tile) : String
+ move(t TileFrame): String
+ kill(e Enemy) :String
+ def(u Unit): String
+ die(u Unit): String
+ injured(cost int, u Unit): String
+ isDead(): boolean
+ receiveCast(a BossAbility): String
+ receiveCast(a PlayerAbility): Strin
+ Tick(): void
<<Interface>>
HeroicUnit
+ cast(List<Unit>): String
Extends
Resource_Based
Abilities
BossAbility
- b: Boss
+ useAbility(ls List<Unit>): String
+ attack(p Player): String
PlayerAbility
+ setPlayer(p Player): void
+ useAbility(ls List<Unit>): String
# canUse(): boolean
+ LevelUp(): void
+ Tick(): void
attack(e Enemy): String
AvengersShield
cool: Cooldown
attack(e Enemy): String
+ useAbility(ls List<Unit>): S
# canUse(): boolean
+ LevelUp(): void
+ Tick(): void
Extends
Shoot
- arr: Arrows
attack(e Enemy): String
+ useAbility(ls List<Unit>): S
# canUse(): boolean
+ LevelUp(): void
+ Tick(): void
Extends
Extends
Extends
Blizzard
- cost (): int
- hitsCount(): Integer
- spellPower: int
- lvl: int
attack(e Enemy): String
+ Tick(): void
+ LevelUp(): void
# canUse(): boolean
+ useAbility(ls List<Unit>): S
FanOfKnives
- cost: int
- energy: Energy
attack(e Enemy): String
+ Tick(): void
+ LevelUp(): void
# canUse(): boolean
+ useAbility(ls List<Unit>): String
Resources
Health
+ levelUpHP(level int): void
Cooldown
+ Tick(): void
+ LevelUp(): void
Extends
Extends
Arrows
- tick: int
- lvl: int
+ Tick(): void
+ LevelUp(): void
+ use(): void
Extends
Resource
+ max: int
+ cur: int
Extends
Energy
- cost: int
+ Tick(): void
Extends
Mana
- cost: int
- lvl: int
+ Tick(): void
+ LevelUp(): void
+ use():void