ALL SPELLING IS FROM MASTER BRANCH, IF UNSURE CHECK RELEVANT CLASS

ORANGE = TODO

GREEN = NOT STUDENT MADE

BLUE = UML TODO

RELATIONSHIPS STILL NOT ALL DONE SINCE LOTS OF CODE QUITE TIGHTLY COUPLED

NEVER IMPLEMENTED ->

<<interface>>
BattleInterface
+ calculatePlayerDelta()
+ calculateEnemyDelta()

<<interface>>
BattleComponent
+ calculatePlayerDelta(): double
+ calculateEnemyDelta(): double

Battle
- config: Config
- player: Entity
- playerHealth: double
- enemyHealth: double
- enemy: Entity
- enemyType: String
- dungeon: Dungeon
- rounds: ArrayList<BattleComponent> = new ArrayList<BattleComponent>

+ Battle(Entity, Entity, Dungeon)
+ calculatePlayerDelta(): double
+ calculateEnemyDelta(): double
+ addRound(Round): boolean
+ removeRound(Round): boolean
+ getPlayerHealth(): double
+ getEnemyHealth(): double
+ getEnemyType() String
+ getBattleResponse(): BattleResponse

Round
- player: Entity
- playerHealth: double
- playerDelta: double
- enemy: Entity
- enemyHealth: double
- enemyDelta: double
- dungeon: Dungeon
- ItemsUsed: ArrayList<Entity> = new ArrayList<Entity>

+ Round(Entity, Entity, Dungeon)
+ calculatePlayerDelta(): double
+ calculateEnemyDelta(): double
+ getPlayerHealth(): double
+ setPlayerHealth(double): void
+ getPlayerDelta(): double
+ setPlayerDelta(double): void
+ getEnemyDelta(): double
+ setEnemyDelta(double): void
+ getItemsUsed(): ArrayList<Entity>
+ getRoundResponse(): RoundResponse

Many, many fields to write ->

Config
+ all them fields
+ Config(JsonObject)

Entity
- type: string
- id: string
- state: MercenaryState = null
- components: List<Component> = new ArrayList<>
- tags: List<String> = new ArrayList<>

+ Entity(int, int, String)
+ getType(): String
+ getTags(): List<String>
+ addTags(string...): void
+ removeTag(string): void
+ setType(String): void
+ getId(): String
+ getComponent(Class<T>): <T extends Component>
+ removeComponent(Class<T>): <T extends Component>
+ addComponent(Component): void
+ update(): void
+ start(): void
+ getEntityInfoResponse(): EntityResponse
+ isInteractable(): boolean
+ setState(MercenaryState): void
+ getState(): MercenaryState

Dungeon
- Tick: int = 1
- dungeonId: String
- dungeonName: String
- goals: Goals
- entities: List<Entity> = new ArrayList<>
- inventory: Inventory
- config: Config
- player: Entity
- battles: List<Battle> new ArrayList<>

+ Dungeon(String, String)
+ getPlayer(): Entity
+ getConfig(): Config
+ getDungeonId(): String
+ getDungeonName(): String
+ getGoals(): String
+ getGoalsRaw(): Goals
+ getEntities(): List<Entity>
+ getInventory(): Inventory
+ getBattles(): List<Battle>
+ setPlayer(Entity): void
+ addBattle(Battle): void
+ removeEntity(Entity): void
+ addEntity(Entity): void
+ saveGame(): JsonObject
+ findEntity(String): void

DungeonManiaController
- dungeon: Dungeon

+ getSkin(): String
+ getLocalisation(): String
+ dungeons(): List<String>
+ configs(): List<String>
+ newGame(String, String): DungeonResponse
+ getDungeonResponseModel(): DungeonResponse
+ tick(String): DungeonResponse
+ tick(Direction): DungeonResponse
+ build(String): DungeonResponse
+ interact(String): DungeonResponse
+ saveGame(String): DungeonResponse
+ loadGame(String): DungeonResponse
+ allGames(): List<String>

Goals
- currentGoals: ArrayList<String>
- dungeonGoals: ArrayList<String>
- completedGoals: ArrayList<String> = new ArrayList<>

+ Goals(ArrayList<String>)
+ FindGoal(JsonObject, ArrayList<String>): String
+ addGoal(String): void
+ refreshCompleted(): void
+ checkString(String): boolean
+ SingularGoalCase(ArrayList<String>): boolean
+ RemoveCompletedANDCase(ArrayList<String>): void
+ RemoveCompletedORCase(ArrayList<String>, int): boolean
+ removeallbellow(ArrayList<String>, int): void
+ toString(): String
+ update(): void
+ debug(): void

Remove from master branch? ->

Inventory
- inventory: List<Entity>

+ Inventory()
+ getInventory(): List<Entity>
+ addItem(Entity): void
+ removeItem(Entity): void
+ countItem(String): int
+ removeItemCount(String, int): void
+ containsType(String): Entity
+ getInventoryResponse(): ArrayList<ItemResponse>

Component
+ entity: Entity = null

extends

Factory
- config: Config

+ Factory(Config)
+ createEntity(String): Entity
+ createEntity(String, int, int): Entity
+ createEntity(String, int, int, int): Entity
+ createEntity(String, int, int, String): Entity

Helper
+ sharePos(Entity, Entity): boolean

components

components

ItemStat
- attack: double
- defense: double
- durability: int
- potion_duration: int

+ ItemStat(String, Config)
+ getAttack(): double
+ setAttack(double): void
+ getDefense(): double
+ setDefense(double): void
+ getDurability(): int
+ setDurability(int): void
+ getPotion_duration(): int
+ setPotion_duration(int): void

Key
- key: int
- keyString: String

+ Key(int)
+ Key(String)
+ getKey(): int
+ getKeyString(): String

Location
- x: int
- y: int
- z: int = 0
- previousX: int
- previousY: int

+ Location(int, int, int)
+ Location(int, int)
+ getPosition(): Position
+ setPosition(Position): void
+ getX(): int
+ setX(int): void
+ getY(): int
+ setY(int): void
+ getZ(): int
+ setZ(int): void
+ getPreviousX(): int
+ getPreviousY(): int
+ move(Position): void

Power
- active: boolean = false

+ Power()
+ state(): boolean
+ setState(boolean): void

Spider
- direction: int = 1
- arrayLocation: int = -1
- spawn: Position

+ Spider(Position)
+ getSpawn(): Position
+ getDirection(): int
+ setDirection(int): void
+ getArrayLocation(): int
+ setArrayLocation(int): void
+ prediction(): int

Stats
- health: double
- attackDamage: double
- playerState: String
- statusDuration: int
- potionInUse: Entity
- bribe_amount: int
- bribe_radius: int
- recon_radius: int
- isBribed: boolean
- health_increase_rate: double
- health_increase_amount: double
- bribe_fail_rate: double
- statusQueue: ArrayList<Entity> = new ArrayList<Entity>

+ Stats(String, Config)
+ standard getters and setters for each field

systems

systems

Build
- inventory: Inventory
- dungeon: Dungeon

+ Build(Dungeon)
+ update(): List<String>
+ build(String): void

BattleSystem
- dungeon: Dungeon

+ BattleSystem(Dungeon)
+ initiateBattle(): void
+ isEnemy(Entity): boolean
+ updateCurrentPotionEffects(): void

Collision
- entities: List<Entity>
- dungeon: Dungeon
- move: MoveSystem

+ Collision(Dungeon)
+ isColision(Entity, Position): Entity/null
+ isSharePosition(Position, Entity): boolean
+ collisionAction(Entity, Direction, Entity): void
+ unlockDoor(Entity): boolean
+ teleport(Entity): Position

Explosion
- dungeon: Dungeon
- radius: int
- entity: Entity

+ Explosion(Entity, Dungeon)
+ explode(): void
+ recursiveExplode(Position, int): void
+ removeEntity(Position): void

GoalChecker
- dungeon: Dungeon

+ GoalChecker(Entity, Dungeon)
+ isZombies(): boolean
+ treasureGoal(): boolean
+ enemyGoal(): boolean
+ switchGoal(): boolean
+ exitGoal(): boolean
+ checkGoal(): boolean

MoveSystem
- entities: List<Entity>
- dungeon: Dungeon

+ MoveSystem(Dungeon)
+ move(Entity, Position): void
+ movePlayer(Direction): void
+ moveEntity(Entity, Direction): boolean
+ setEntities(List<Entity>): void
+ moveEnemy(Entity): void
+ moveSpider(Entity): void
+ moveZombieOrHydra(Entity): void
+ moveMercenary(Entity): void
+ moveAssassin(Entity): void
+ assassinInRangeOfPlayer(Entity): boolean
+ moveRandom(Entity): void
+ getMaxDistancePosition(Entity): Position
+ entityAdjacentToPlayerPrevLocation(Entity): boolean
+ dijkstra(ArrayList<Position>, Position): Map<Position, Position>
+ getNextPosition(Position, Map<Position, Position>): Position
+ checkIfPositionMerc(Position, Map<Position, Position>): boolean

Response
+ getEntityInforResponse(Entity): EntityResponse
+ getEntityResponses(List<Entity>): List<EntityResponse>
+ getItemInfoResponse(Entity): ItemResponse
+ getItemResponses(List<Entity>): List<ItemResponse>
+ getBattleResponse(Dungeon): List<BattleResponse>
+ getDungeonResponse(Dungeon): DungeonResponse

Tick
- dungeon: Dungeon
- player: Entity = null

+ Tick(Dungeon)
+ Update(): void
+ updateEnemeies(): void
+ preUpdateTiles(): void
+ postUpdateTiles(): void
+ isPowered(Entity): boolean
+ avaliableEdges(Entity): boolean
+ spawn(): void

TileInteraction
- dungeon: Dungeon
- entity: Entity

+ TileInteraction(Dungeon, Entity)
+ Check(): void

UseSystem
- entity: Entity
- dungeon: Dungeon

+ UseSystem(Dungeon, Entity)
+ Use(): void
+ useBomb(): void
+ usePotion(): void

response\models

exceptions

util

response\models

exceptions

util