FACULTÉ DES SCIENCES ET TECHNOLOGIES | Université de Lille

Semestre 4

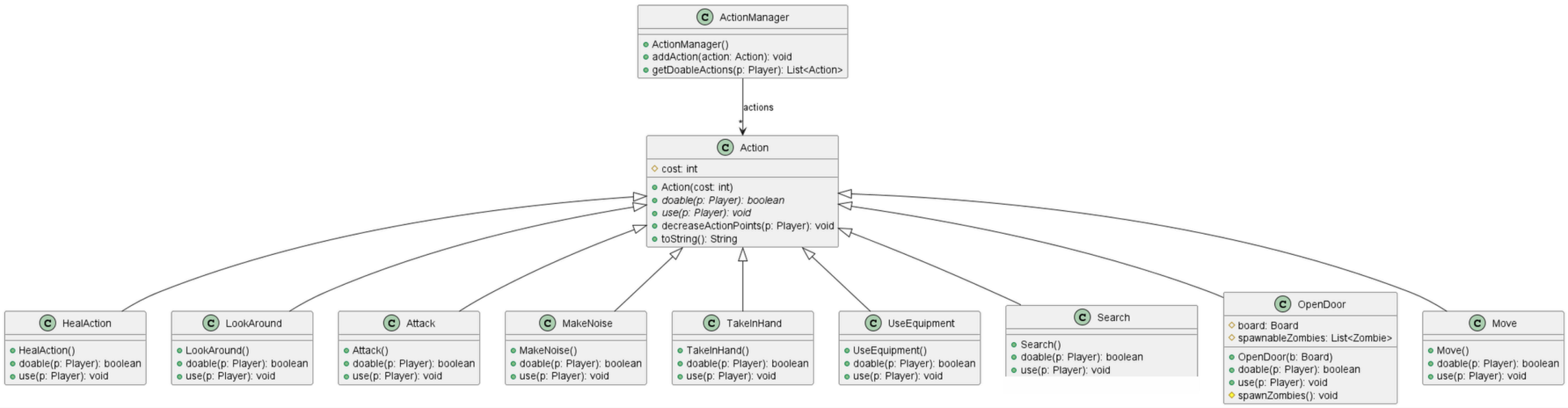
Soutenance projet

Par Elea Krzewinski, Cyril Proy, Guillaume Ponsdesserre, Raphaël Levecque

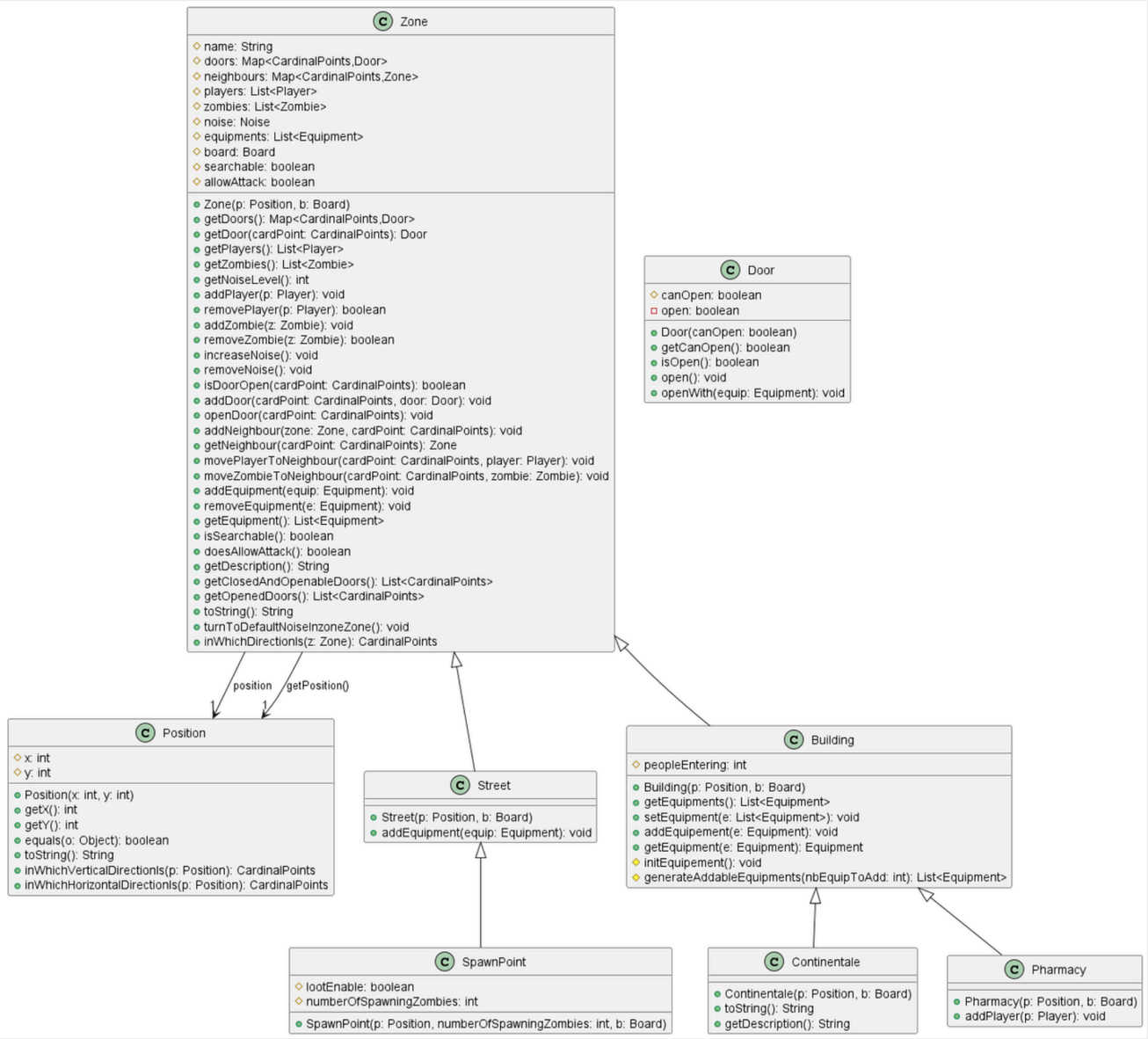
- Avancement du projet
- Choix de modélisation
- Ouverture à l’extension
- Travail en équipe
- Difficultés
- Bilan

https://youtu.be/oq75GDVkdKc

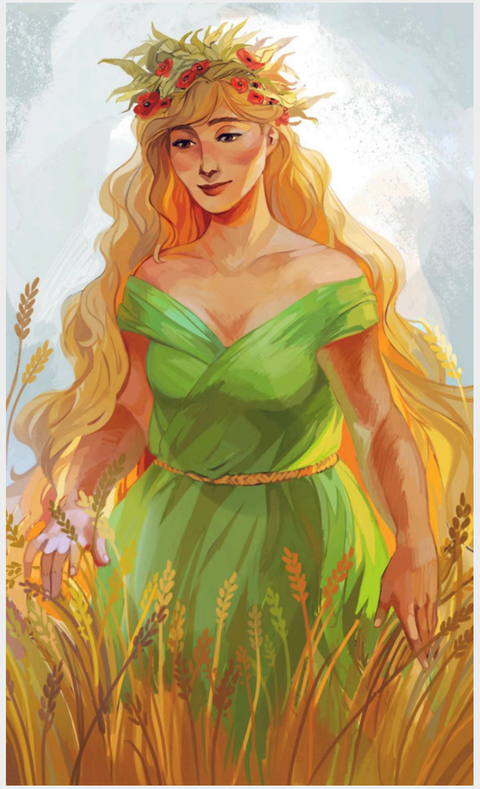
Les actions


ActionManager
ActionManager()
addAction(action: Action): void
getDoableActions(p: Player): List<Action>
actions
*
Action
cost: int
Action(cost: int)
doable(p: Player): boolean
use(p: Player): void
decreaseActionPoints(p: Player): void
toString(): String
HealAction
HealAction()
doable(p: Player): boolean
use(p: Player): void
LookAround
LookAround()
doable(p: Player): boolean
use(p: Player): void
Attack
Attack()
doable(p: Player): boolean
use(p: Player): void
MakeNoise
MakeNoise()
doable(p: Player): boolean
use(p: Player): void
TakeInHand
TakeInHand()
doable(p: Player): boolean
use(p: Player): void
UseEquipment
UseEquipment()
doable(p: Player): boolean
use(p: Player): void
Search
Search()
doable(p: Player): boolean
use(p: Player): void
OpenDoor
board: Board
spawnableZombies: List<Zombie>
OpenDoor(b: Board)
doable(p: Player): boolean
use(p: Player): void
spawnZombies(): void
Move
Move()
doable(p: Player): boolean
use(p: Player): void

Les zones


Zone
name: String
doors: Map<CardinalPoints,Door>
neighbours: Map<CardinalPoints,Zone>
players: List<Player>
zombies: List<Zombie>
noise: Noise
equipments: List<Equipment>
board: Board
searchable: boolean
allowAttack: boolean
Zone(p: Position, b: Board)
getDoors(): Map<CardinalPoints,Door>
getDoor(cardPoint: CardinalPoints): Door
getPlayers(): List<Player>
getZombies(): List<Zombie>
getNoiseLevel(): int
addPlayer(p: Player): void
removePlayer(p: Player): boolean
addZombie(z: Zombie): void
removeZombie(z: Zombie): boolean
increaseNoise(): void
removeNoise(): void
isDoorOpen(cardPoint: CardinalPoints): boolean
addDoor(cardPoint: CardinalPoints, door: Door): void
openDoor(cardPoint: CardinalPoints): void
addNeighbour(zone: Zone, cardPoint: CardinalPoints): void
getNeighbour(cardPoint: CardinalPoints): Zone
movePlayerToNeighbour(cardPoint: CardinalPoints, player: Player): void
moveZombieToNeighbour(cardPoint: CardinalPoints, zombie: Zombie): void
addEquipment(equip: Equipment): void
removeEquipment(e: Equipment): void
getEquipment(): List<Equipment>
isSearchable(): boolean
doesAllowAttack(): boolean
getDescription(): String
getClosedAndOpenableDoors(): List<CardinalPoints>
getOpenedDoors(): List<CardinalPoints>
toString(): String
turnToDefaultNoiseInzoneZone(): void
inWhichDirectionIs(z: Zone): CardinalPoints
Door
canOpen: boolean
open: boolean
Door(canOpen: boolean)
getCanOpen(): boolean
isOpen(): boolean
open(): void
openWith(equip: Equipment): void
position
getPosition()
Position
x: int
y: int
Position(x: int, y: int)
getX(): int
getY(): int
equals(o: Object): boolean
toString(): String
inWhichVerticalDirectionIs(p: Position): CardinalPoints
inWhichHorizontalDirectionIs(p: Position): CardinalPoints
Street
Street(p: Position, b: Board)
addEquipment(equip: Equipment): void
Building
peopleEntering: int
Building(p: Position, b: Board)
getEquipments(): List<Equipment>
setEquipment(e: List<Equipment>): void
addEquipement(e: Equipment): void
getEquipment(e: Equipment): Equipment
initEquipement(): void
generateAddableEquipments(nbEquipToAdd: int): List<Equipment>
SpawnPoint
lootEnable: boolean
numberOfSpawningZombies: int
SpawnPoint(p: Position, numberOfSpawningZombies: int, b: Board)
Continentale
Continentale(p: Position, b: Board)
toString(): String
getDescription(): String
Pharmacy
Pharmacy(p: Position, b: Board)
addPlayer(p: Player): void

Demeter qui hérite de Player:



- 10000000 points de vie
- 10 points d'action
- peut soigner

```java
package zombicide.actors.players;

/** Class of Demeter */
public class Demeter extends Player{

    /**
     * builds a new Demeter
     */
    public Demeter() {
        super();
        this.lifePoints = 10000000;
        this.actionPoints = 10;
        this.demeter = true;
    }

    /**
     * heal
     */
    public void heal() {
        Player chosenPlayer = this.choosePlayer(this.getPlayersOfZone());
        chosenPlayer.addLifePoints(2);
    }
}
```

Nouveau zombie

Boss qui hérite de Zombie:

- 1000 points de vie
- 1 point d'action
- 1 point de dégât

```java
package zombicide.actors.zombie;

/** Class of Boss */
public class Boss extends Zombie {
    /**
     * Builds a Boss
     */
    public Boss() {
        super(1000,1,1);
    }

    /**
     * decrease the life points of damage if damage > 1
     * and analyze if the zombie is still alive, if not removes it
     * @param damage the damage the zombie took
     */
    public boolean takeDamage(int damage){
        if (damage > 1) {
            return super.takeDamage(damage);
        }
        else {
            return false;
        }
    }
}
```

```java
protected List<Zombie> generateSpawnableZombies(int nbOfZombiesToSpawn) {
    List<Zombie> spawnableZombies = new ArrayList<>();
    int n=(int)(Math.random() * 100);
    if (n<=1) {
        spawnableZombies.add(new Boss());
    }
    spawnableZombies.add(new Abomination());
    spawnableZombies.add(new Huge());
    for(int i = 0; i < nbOfZombiesToSpawn ; i++) {
        spawnableZombies.add(new Runner());
        spawnableZombies.add(new Walker());
    }
    return spawnableZombies;
}
```

Sacrifice qui hérite de Action:

- 3 points d'actions

```java
/** Constructor of the action sacrifice */
public Sacrifice() {
    super(3);
}
/**
 * Check if the action is doable
 * @param p the player
 * @return true if the action is doable, false otherwise
 */
public boolean doable(Player p) {
    return p.canSacrifice();
}
/**
 * Sacrifice the player
 * @param p the player
 */
public void use(Player p) {
    p.sacrifice();
    this.decreaseActionPoints(p);
}
```

Nouvelle arme

Bow qui hérite de Weapon:

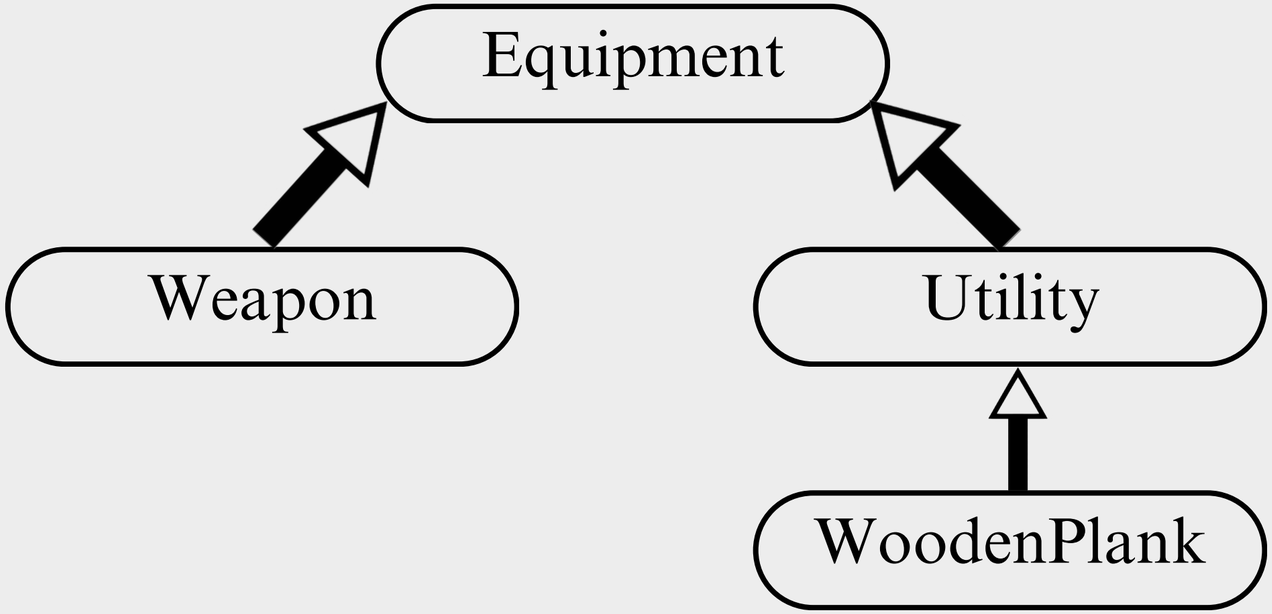
- 1 lancer de dé
- seuil 3
- dégât 2
- portée 0 à 2
- n'ouvre pas les portes
- est silencieux

```java
package zombicide.equipments;

/**Class of Bow**/
public class Bow extends Weapon {
    /**
     * Builds a Bow
     */
    public Bow() {
        super(false,1,3,2,2,true);
    }
}
```

```java
/**
 * generate the addable equipment
 * @return the list of addable equipment
 */
protected List<Equipment> generateAddableEquipments(int nbEquipToAdd) {
    List<Equipment> addableEquipments = new ArrayList<Equipment>();
    for(int i = 0; i < nbEquipToAdd; i++) {
        addableEquipments.add(new Axe());
        addableEquipments.add(new Rifle());
        addableEquipments.add(new Pistol());
        addableEquipments.add(new InfraredGlasses());
        addableEquipments.add(new Map(this.board));
        addableEquipments.add(new Crowbar());
        addableEquipments.add(new AidKit());
        addableEquipments.add(new MasterKey());
        addableEquipments.add(new Chainsaw());
        addableEquipments.add(new Bow());
    }
    return addableEquipments;
}
```

Nouvel équipement



- La classe Utility représente entre autres les équipements : carte, trousse de soins, clé (destinés à être utilisés lors de l’action “utiliser un équipement”)
- Elle définit une méthode abstraire use() qui servira dans les classes qui en hérite à définir et déclencher l’utilisation de l’équipement
- La classe WoodenPlank redéfinit simplement la méthode use() présente dans Utility

Travail en équipe

1. **Faire en sorte que lorsqu'un player tue un zombie, il gagne un niveau**
   → Et donc, vérifier s'il franchit un palier (enum ?), si oui : augmenter ses points d'action (takeDamageFrom() renvoie true si Zombie tué ?)

2. **Créer un attribut currentActionPoints, en plus de actionPoints dans Player**
   → Pour permettre de réinitialiser les point d'actions courant après un tour (donc changer decreaseActionPoints() pour diminuer currentActionPoints et pas actionPoints, faire une méthode reinitActionPoint())

3. **Améliorer la méthode move() des zombies**
   › En créant une méthode inWhichDirectionIs(Position) dans Position, puis une autre similaire dans Zone qui utilise celle de Position. Elle renvoie un CardinalPoint qui indique dans quelle direction se situe l'autre Zone. Puis mettre noisiestZone en paramètre.

4. **Continuer le main du livrable 3**
   → Faire attaquer et bouger tous les zombies. Pour simplifier : faire une méthode getAllZombies() dans Board ?

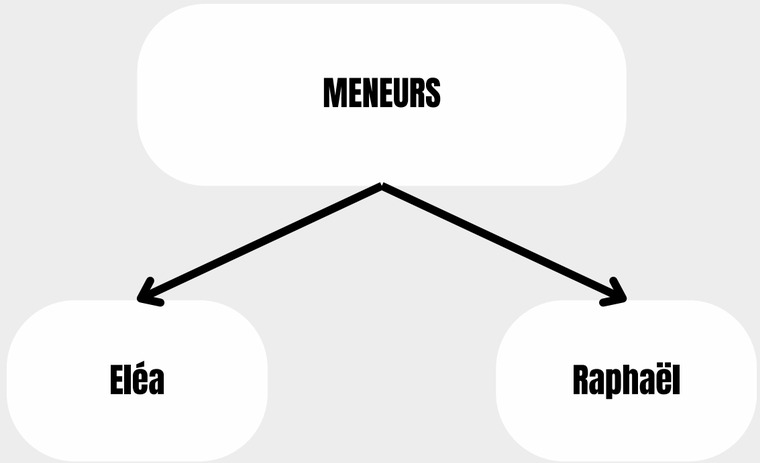
5. **Éventuellement voir avec la prof la pertinence de l'attribut board chez les acteurs et zone**

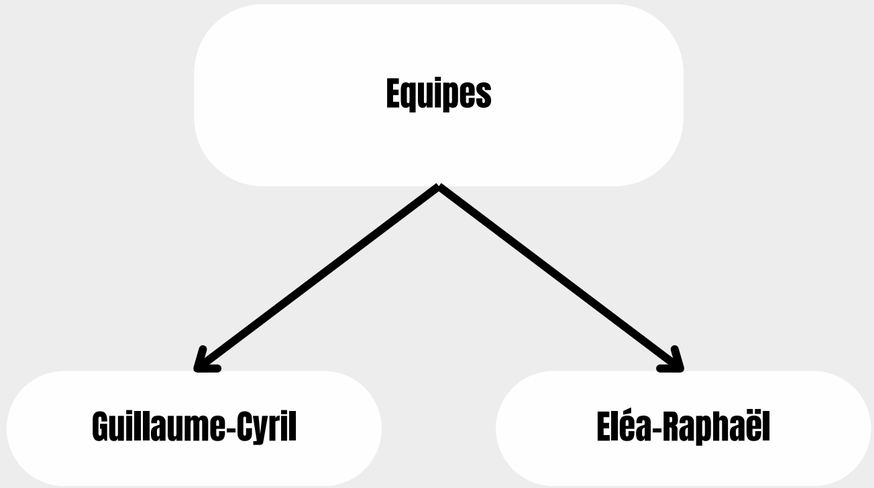
Selon moi :

- Concernant l'attribut board chez les acteurs :
  - Player :
    Pas besoin d'attribut board, il suffit de déplacer la méthode getReachableZombies() et CheckAvailable() dans Weapon et utiliser le système de voisin pour chercher les zombies atteignables **(voir le code que j'avais envoyé sur discord) ceci simplifierait également la méthode attack() de player en se passant de downcast Equipment › Weapon**
  - Zombie :
    Pas besoin si on améliore la méthode move() comme décrit plus haut

- Concernant l'attribut board chez les Zone :
  Si on passe board en attribut des zones, alors le système de voisins que nous avons crée précédemment devient obsolète puisqu'on peut utiliser board pour obtenir les zone adjacentes

Si j'ai bien compris, on a board dans Zone pour initialiser l'équipement Map dans chaque zone. Cependant, on pourrait déplacer la méthode qui initialise les équipements des zones dans Board, ou dans Game, ainsi nul besoin d'un attribut board pour les Zone et le système de voisins reste pertinent.

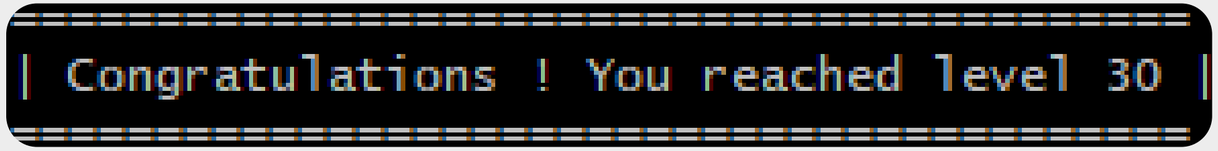




JUnit

UNIFIED MODELING LANGUAGE™

Bilan



- Passer d’une description textuelle à une conceptualisation UML puis à un programme java

- Progresser dans le travail en équipe, organiser et répartir le travail, exprimer ses idées à ses camarades

- Approfondir l’expérience de la programmation orientée objet, l’utilisation de GIT, et la création d’UML