German University in Cairo
Media Engineering and Technology
Prof. Dr. Slim Abdennadher
Dr. Nourhan Ehab
Dr. Ahmed Abdelfattah

Programming II, *Spring 2022*
The Last of Us: Legacy: **Milestone 2**

Deadline: 11th of May 2023, 11:59 PM

This milestone is a further *exercise* on the concepts of **object oriented programming (OOP)**. By the end of this milestone, you should have a working game engine with all its logic, that can be played on the console if needed. The following sections describe the requirements of the milestone. Refer to the **Game Description Document** for more details about the rules.

- Please note that you are not allowed to change the hierarchy provided in milestone 1 nor the visibility/access modifiers of variables except as explicitly stated in this milestone description. You should also conform to the method signatures provided in this milestone. However, you are free to add more helper methods. You should implement helper methods to include repetitive pieces of code.
- All methods mentioned in the document should be `public`. All class attributes should be `private` with the appropriate access modifiers and naming conventions for the getters and setters as needed.
- You should always adhere to the OOP features when possible. For example, always use the `super` method or constructor in subclasses when possible.
- The model answer for M1 is available on CMS. It is recommended to use this version. A full grade in M1 doesn't guarantee a 100 percent correct code, it just indicates that you completed all the requirements successfully :)
- Some methods in this milestone depend on other methods. If these methods are not implemented or not working properly, this will affect the functionality and the grade of any method that depends on them.
- **You need to carefully read the entire document to get an overview of the game flow as well as the milestone deliverables, before starting the implementation.**
- We define adjacency as the cells within a radius of 1 cell in all directions: Horizontal, Vertical, and Diagonal.
- If any attack results in the death of the target character, that character should immediately be removed from any place that they existed in.
- Before any action is committed, the validity of the action should be checked. For example: Certain actions require action points, other require Collectibles to be available in the hero inventory, etc.
- Some actions will not be possible as well if they violate boundaries or game rules; such as: heroes attacking each other, moving outside map boundaries, moving to occupied cells, etc.
- **All exceptions that could arise from invalid actions should be thrown and handled in this milestone.**

1 Interfaces

1.1 Collectible Interface

Name : `Collectible`

Package : `model.collectibles`

Type : Interface

Description : Interface containing the methods available to objects on the map that can be collected. All `Vaccine`s and `Supplie`s are `Collectible`s.

You should add the following methods to this interface:

1. `void pickUp(Hero h)`: A method that adds the collectible picked by the hero to his corresponding Arraylist .
2. `void use(Hero h)`: A method that removes the used collectible by the hero from corresponding Arraylist .

2 Characters

2.1 `Character` Class

This class should also include the following additional methods:-

- `void attack()`: This method should handle applying the logic of an attack on the character's target. While implementing think of the sub-classes that behave differently, whether they cost an action point or not and what can be extracted generically. Note that characters should only be allowed to attack adjacent cells.
- `void defend(Character c)`: This method is only called whenever a character has been attacked by another character. A defending character causes half his attackDmg to the attacking character.

  Example: *Hero A* (having attack damage 30) attacks *Zombie 1*

  Zombie 1 receives 30 damage from the attack, then **defends**; setting *Hero A* as their target, and dealing 5 damage to *Hero A*.
- `void onCharacterDeath()`: This method should be called whenever a character's currentHp reaches 0. It should handle removing the character from the game, and updating the map.

2.2 `Hero` Class

Note that for any hero moving, attacking, or curing a zombie costs 1 action point. Using the hero's special action on the other hand costs 1 supply instead. Whenever a medic uses their special on a viable target, the target HP is fully restored.

This class should also include the following additional methods:-

- `void move(Direction d)`: A method that is called when the current hero wishes to move in a particular direction. If a movement occurs within a turn, both previously and newly visible cells remain visible until the turn the movement occurred in ends.
- `void useSpecial()` : This method should handle a hero wanting to use his special action, given that the conditions are met. While implementing think of the sub-classes that behave differently, and what can be extracted generically. Refer to the Game description for the different special actions for each hero type.

- `void cure()` : This method should handle using a vaccine to cure a zombie. Heroes can only cure zombies that are in adjacent cells.

3 Engine

3.1 `Game` Class

The below figure demonstrates the orientation of the board This class should also include the

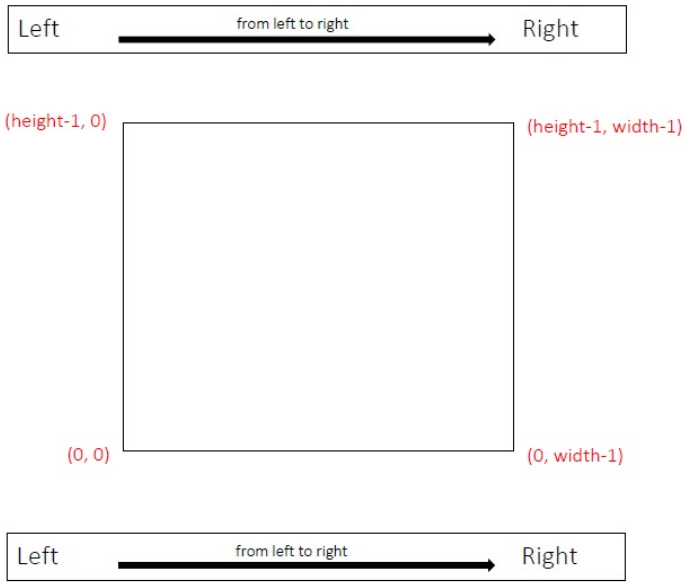

Figure 1: Board Orientation

following additional methods:-

1. `public static void startGame(Hero h)`: This method is called to handle the initial game setup, as such, it should spawn the necessary collectibles (5 Vaccines, 5 Supplies), spawn 5 traps randomly around the map, spawn 10 zombies randomly around the map, add the hero to the controllable heroes pool and removing from the availableHeroes, and finally allocating the hero to the bottom left corner of the map.
2. `public static boolean checkWin()`: This method checks the win conditions for the game. Refer to the Game Description document if needed.
3. `public static boolean checkGameOver()`: This method checks the conditions for the game to end. Refer to the Game Description document if needed.
4. `public static void endTurn()`: This method is called when the player decides to end the turn. The method should allow all zombies to attempt to attack an adjacent hero (if exists), as well as reset each hero's actions, target, and special, update the map visibility in the game such that only cells adjacent to heroes are visible, and finally spawn a zombie randomly on the map.