Contents

Description

Below is the combined descriptions of each phase of the "Fighting game" project. If working through this project for the first time, it is encouraged to work on one phase at a time, ensuring your execution of the project is correct by comparing with our solution before moving on.

Phase 1

Purpose:

This second project will be spread across three phases to give you the experience of implementing a class and feature-rich program utilizing said class. This first phase consists of implementing a starting form of the class you will be working with until the end of the semester. This class will primarily consist of setting up attributes and their respective getters/setters, as well as some unique methods needed to define the core functionality of the class being created.

Skills

For this project, you will need to:

- Exhibit an understanding of IDE features pertaining to object-oriented development,
- Analyze a simple design document,
- Implement a simple class containing multiple attributes and methods, and
- Write a simple test program for your implementation.

Knowledge:

This assignment will familiarize you with the following important skills:

- How to read a UML diagram,
- How to interpret a series of simple instructions, and
- How to manipulate objects.

Tasks

Challenge

In short You will be creating a `Fighter` class, which will contain the core functionality of a character in some turned-based RPG type game.

In more details Consider the following UML diagram for the `Fighter` class:

In order, the attributes will store:

- The name of the fighter,
- The level of the fighter,
- The current experience the fighter has,
- The max experience the fighter needs to level up,
- The amount of hit points (HP) the fighter has,
  - This is a percentage, that is, the value should only be between 0 and 1, and be displayed using the `P` format specifier.
- The attack stat of the fighter, and
- The defense stat of the fighter.

The methods to be implemented are as follows:

- Getters for all attributes,
- Setters for `name`, `attack`, and `defense`,
- A `LevelUp` method
  - Increments `level`, `attack`, `defense`, and `maxExp` by a predetermined amount.
    - `level`: 1
    - `attack`: 3
    - `defense`: 3

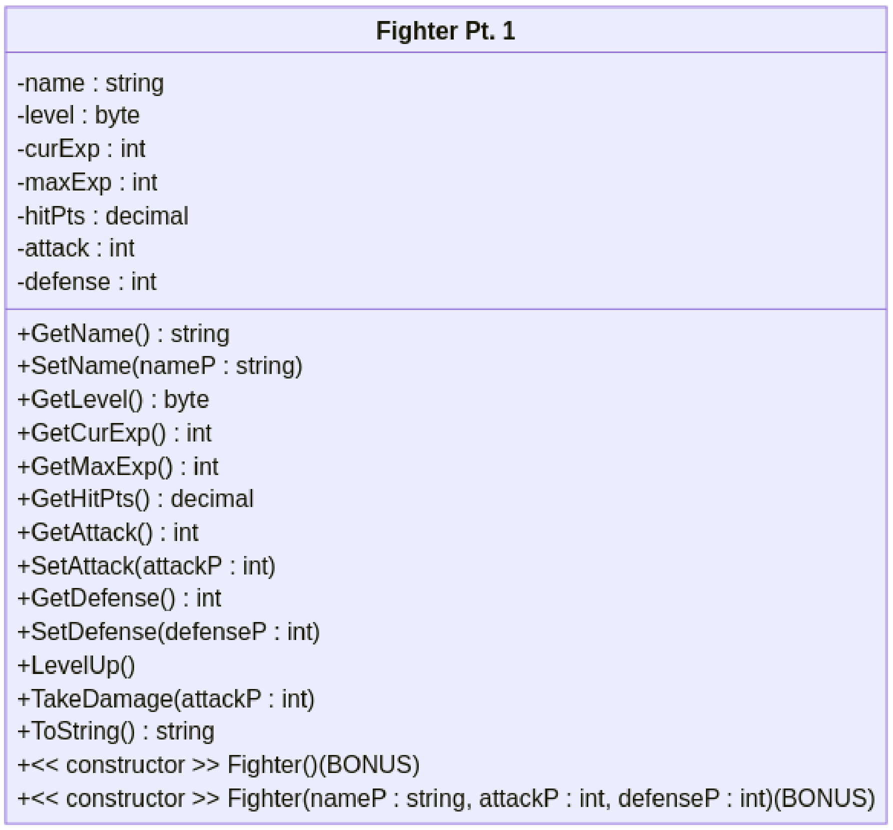


Figure 1: A UML diagram for the Fighter("Fighter Pt. 1") class (text version[2])

        * `maxExp`: 50
    - Resets `hitPts` back to full health (that is, back to 1)
- A `TakeDamage` feature:
    - Takes in `attackP` as a parameter (that is, the damage dealt by the opposing fighter)
    - Deduct the result of the following formula from `hitPts`
        * `((decimal)attackP/10) / (((decimal)defense/10) + 4m)`
    - If `hitPts` is below 0, set `hitPts` to 0.
- A `ToString` method
    - This should neatly display all the attributes for the class.

Example Once creating my fighter, Totally McRealGuy, this is what I display to the console:

```
Name: Totally McRealGuy
Level: 0
HP: 100.00%
Exp: 0/100
Attack: 15
Defense: 5
```

Notice how `hitPts` (abbreviated as HP) is displayed as a percentage and attack and defense was set.

TotallyMcRealGuy leveled up! Let's check his stats now:

```
Name: Totally McRealGuy
Level: 1
HP: 100.00%
Exp: 0/150
Attack: 18
Defense: 8
```

Notice how `maxExp`, `attack`, and `defense` has incremented.

Bonus We'll add two new constructors to the class, one that takes no parameters, and another that takes in parameters for `name`, `attack`, and `defense`. The former should have an empty body, while the latter should assign the parameters to their respective attributes.

Phase 2

Purpose

This is the second phase of your three-phase project for the course. This phase will consist of extending your Fighter class to add some more functionality, then creating a user-interactive program that collectsthe data from the user to make a Fighter object with the class.

Skills

For this project, you will need to:

- Exhibit an understanding of IDE features pertaining to object-oriented development,
- Expand on the class you made last time with new methods,
- Construct a interactive program,
- Validate inputs from the user using boolean logic, and
- Create an object from the class you constructed.

Knowledge

This assignment will familiarize you with the following important skills:

- How to manipulate objects,
- How to use boolean systems (`if`/`switch`, loops), and
- How to validate user input

Tasks

Challenge

In short You will be adding four additional methods into the `Fighter` class you constructed in phase 1, then implementing a program to construct a `Fighter` object based on the *validated* input given by the user.

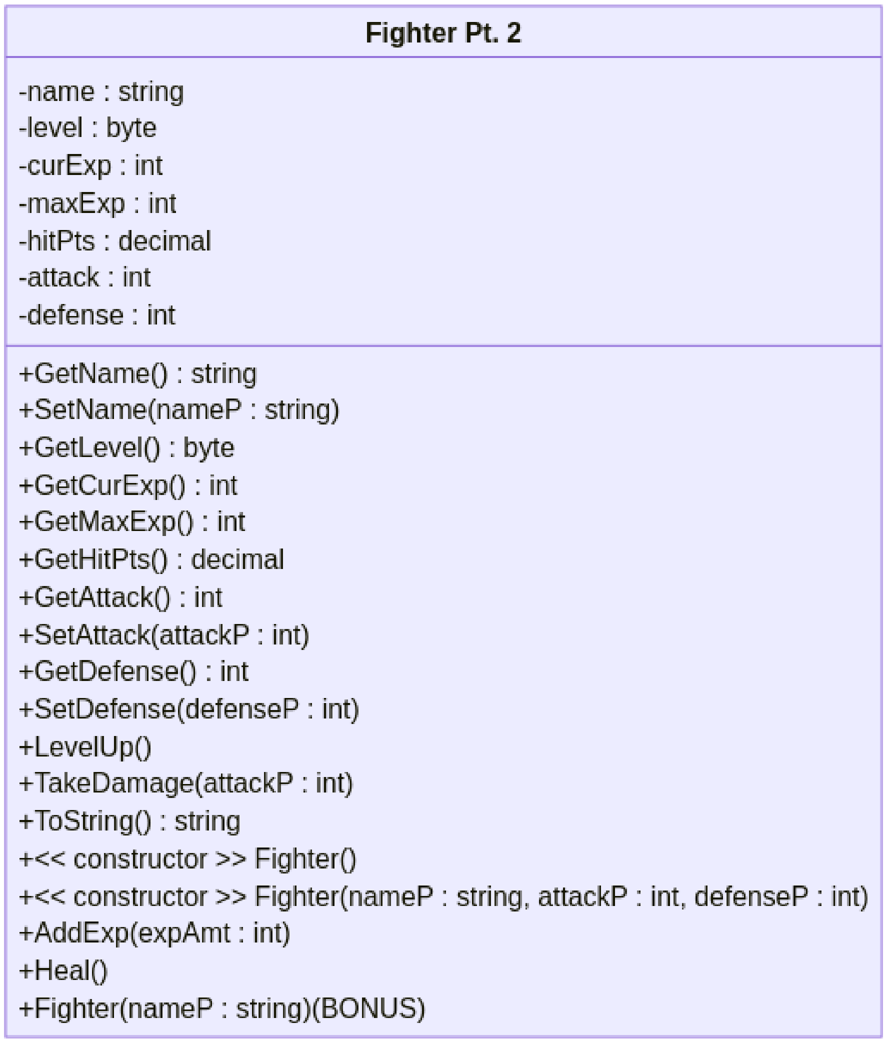
In more details First, download the Project 2 Phase 2 Template: (LINK HERE)

(Note: You must use the template to complete this project unless you got a perfect score on Phase 1)

Consider the updated UML Diagram for the `Fighter` class:

In `Fighter.cs`...

- Add a no-args constructor that sets the attribute values to:
  - name: "Default"
  - level: 1
  - curExp: 0
  - maxExp: 100
  - hitPts: 1m
  - attack: 10



```
Fighter Pt. 2
-name : string
-level : byte
-curExp : int
-maxExp : int
-hitPts : decimal
-attack : int
-defense : int
+GetName() : string
+SetName(nameP : string)
+GetLevel() : byte
+GetCurExp() : int
+GetMaxExp() : int
+GetHitPts() : decimal
+GetAttack() : int
+SetAttack(attackP : int)
+GetDefense() : int
+SetDefense(defenseP : int)
+LevelUp()
+TakeDamage(attackP : int)
+ToString() : string
+<< constructor >> Fighter()
+<< constructor >> Fighter(nameP : string, attackP : int, defenseP : int)
+AddExp(expAmt : int)
+Heal()
+Fighter(nameP : string)(BONUS)
```

Figure 2: A UML diagram for the Fighter("Fighter Pt. 2") class (text version[4])

    - defense: 10
- Add a populated constructor, which will use the same default values above except for the attributes that have parameters
- Add an `AddExp` method
    - Takes in `expAmt` and adds this to `curExp`.
    - If `curExp` exceeds `maxExp`, deduct `maxExp` from `curExp` and call `LevelUp`
- Add a `Heal` method
    - Sets `hitPts` to 1

In `Program.cs`...

- Create a program that prompts the user to input the values to create your fighter.
    - Ask the user for the name of their fighter (no validation required on this part)
    - Ask the user for their attack and defense stats
        * Attack and defense must be non-negative integer values whose combined total does not exceed 20
    - Once done, should display the newly created character to the console.

Example Here is an example of execution, where the user input is u̲n̲d̲e̲r̲l̲i̲n̲e̲d̲, and hitting "enter" is represented by "↵": s

```
MAKE YOUR FIGHTER!!!!
Name: Totally McRealguy↵
Set your attack and defense (combined total must not
 ↪  exceed 20).
Attack: no↵
Invalid entry, try again.
Attack: 21↵
Invalid entry, try again.
Attack: 15↵
Defense: 15↵
Combined total exceeds 20, try again.
Set your attack and defense (combined total must not
 ↪  exceed 20).
Attack: 15↵
Defense: 5↵
YOUR FIGHTER HAS BEEN CREATED!!!
Name: Totally McRealguy
Level: 1
HP: 100.00%
Exp: 0/100
Attack: 15
Defense: 5
```

Bonus Create a third constructor that takes in only a name parameter and creates a fighter using *randomly generated* attack and defense values. These values must also not allow the combined total of these to be greater than 20. (For information on Random[5])

Phase 3

Purpose

In this final phase of your second project, you will be utilizing the full extent of your Fighter class to make a game loop focused on the player fighter fighting randomly generated opponent fighters.

Skills

For this project, you will need:

- Exhibit an understanding of IDE features pertaining to object-oriented development,
- Expand on the class you made last time with new methods,
- Construct a interactive program,
- Validate inputs from the user using boolean logic, and
- Create an object from the class you constructed.

Knowledge

This assignment will familiarize you with the following important skills:

- How to manipulate objects,
- How to generate random numbers,
- How to use boolean systems (`if`/`switch`, loops), and
- How to validate user input

Tasks

Challenge

In short You will be adding one more method to your `Fighter` class before extending your implementation in `Program.cs` to make a game loop where the player fighter fights rounds of randomly generated opponent fighters.

[5]https:/princomp.github.io/lectures/misc/random

In more details First, download the Project 2 Phase 3 Template: (LINK HERE)

(Note: You must use the template to complete this project unless you got a perfect score on Phase 2)

Consider the updated UML Diagram for the `Fighter` class:

In `Fighter.cs`… - Add one more constructor that takes in a string parameter for `name` and a byte parameter for the player's level (named `playerLvl` in the UML) - Randomly generates `atk` and `def` values (combined total must not exceed 20) - Set attributes to the following values: - name: name parameter - level: `playerLvl` - curExp: `0` - maxExp: `50 * level` - hitPts: `1` - attack: `atk + (3 * (level - 1))` - defense: `def + (3 * (level - 1))`

In `Program.cs`… - (Optional step for neatness) After character creation, prompt user to press any key to continue, then clear the screen once any key is pressed - Create the game loop. That is, make player fighter fight against randomly generated opponent fighters until player fighter is defeated - Loop should proceed as follows: - Creates a new random opponent fighter - Prints player and opponent fighter stats - Ask player to pick move (either attack or defend) and validate input to ensure that player enters a valid move - Randomly decide opponent's move - Do the following based on both fighters' moves… - If player attacks… - and opponent attacks, then both take damage with the opposing fighter's attack stat - and opponent defends, then opponent takes damage with the player's attack stat / 2 - If player defends… - and opponent attacks, then player takes damage with the opponent's attack stat / 2 - and opponent defends, then nothing happens - If opponent is defeated, then… - Heal player - Add `25 * opponent's level` exp to player's curExp - Continue to next round - If player is defeated, then exit loop and display how many rounds the player beat.

Example See the video demo on D2L

Bonus For this final phase, there are two bonuses: 1. Modify the ToString method in `Fighter.cs` to display like this:

```
Name: Totally McRealguy
Level: 1
HP: [█████████████████████████] 100.00%
Exp: 0/100
Attack: 10
Defense: 10
```

- Note: The number of blocks displayed should be based on how much health the fighter has, and should center so that if blocks are

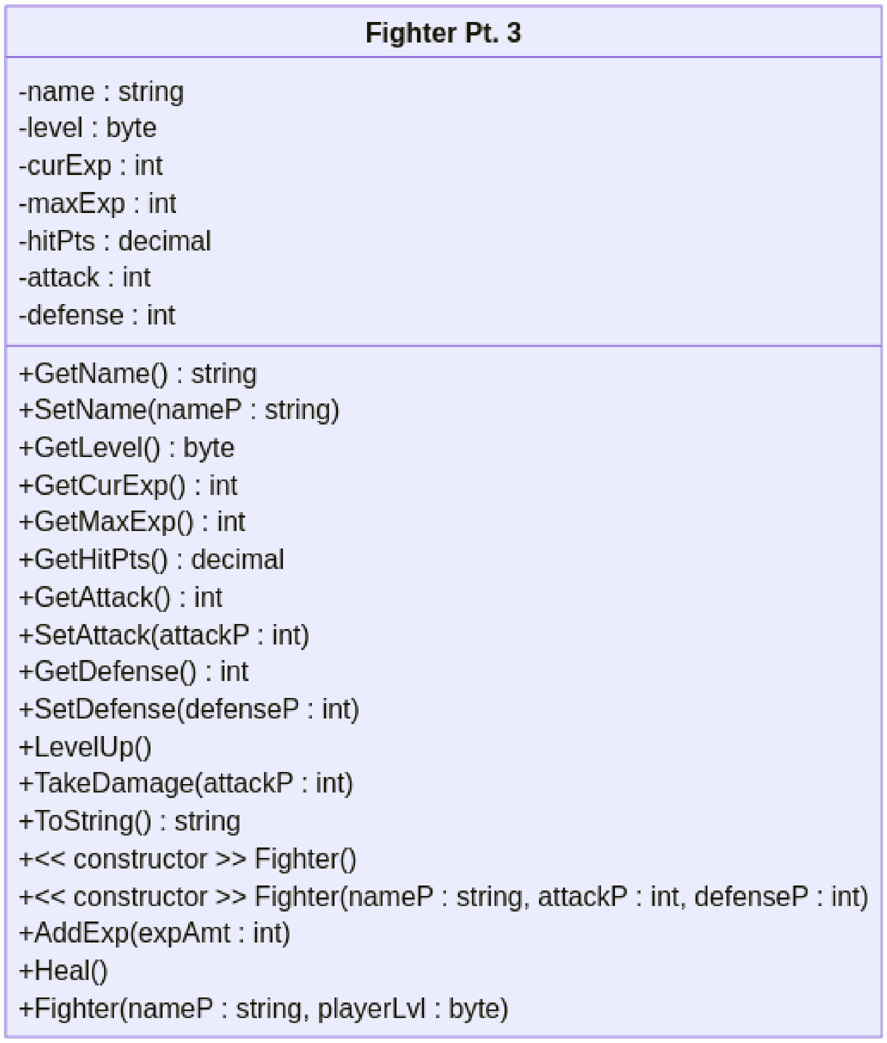


Figure 3: A UML diagram for the Fighter("Fighter Pt. 3") class (text version[7])

missing (e.g., `HP: [████████████            ] 50.00%`) The escape character for the block is `\u2588`.
- Hint: See the official documentation for string interpolation[8] and a relevant StackOverflow post about repeating characters[9].

2. Add a new move that either fighter can use, Heavy Attack, which will add the additional conditions for each move:

- If player attacks…
  - and opponent heavy attacks, then both take damage with the opposing fighter's attack stat * 2
- If player defends…
  - and opponent heavy attacks, then player takes damage with the opponent's attack stat
- If player heavy attacks…
  - and opponent attacks, then both take damage with the opposing fighter's attack stat * 2
  - and opponent defends then opponent takes damage with the player's attack stat
  - and opponent heavy attacks, then both take damage with the opposing fighter's attack stat * 4

Solution

Project 2 Phase 3 Solution (LINK HERE)

[8]https://learn.microsoft.com/en-us/dotnet/csharp/language-reference/tokens/interpolated
[9]https://stackoverflow.com/a/411762