Lab 07
Assigned: 10/24/2023
Due: 10/30/2023
Theme: Dealing with the Template Method Pattern (TMP)
Points: 40 points

Brushing up our knowledge about the TMP:

++

- A template method defines an algorithm in abstract operations that sub-classes override to provide concrete behavior.
- To implement the invariant parts of an algorithm once and leave it up to subclasses to implement the varied behavior.
- Standard behavior among subclasses should be factored in and localized in a parent class to avoid code duplication.
- Template methods are a fundamental technique for code reuse.

++

Now, let's use the template method pattern for implementing the following scene:

In package lab07, we have (existing code) a class named Player.

- The Player has four properties: id, health, defend, and attack.
  - 'id' is used to identify each Player uniquely.
  - 'health', 'defend', and 'attack' are positive integers.
  - 'attack <= health', meaning for each Player, their attack points should not be more than their health points,
  - 'defend (<= 100)' expresses what percent of the attack a player can defend.
  - If the health point goes below or equal to 0, then a player is declared dead.

Our client wants us to design a 2-player gaming platform so that they may extend the platform to implement similar types of games by tweaking some policies. They don't care if the platform is a class or interface. They requested us to implement a template method that takes care of the combat part (assuming the combatting rules will be the same for every type of game they deploy) and leave some relevant methods unimplemented to tweak the policies of the game. They also expect to initialize the platform with a list of players. Below is the algorithm they want us to implement:

++

Combatting Algorithm:

Input: 2 players
Output: winning Player

Step:

The first Player attacks the second Player (say, P attacks Q).
After the hit, if the other Player is alive, they attack the first Player (Q attacks P).

This process continues until one of them dies.

Return the winner's information.

++

The client wants us to implement two concrete games:

[10 points] Pokemon Battle (Version 01):

- Each Pokemon is a Player. Two Pokemons take part in every battle.
- Pokemon in this version cannot use their defend points to reduce the impact of attacks.
- Hit policies: during every hit, one Pokemon attacks the other with 50% of its attack point. In each iteration, a Player loses its health by the amount of the attacked point (50% of the attack point of the attacker).

[10 points] Mortal Combat:

- Each Character (like Sub-Zero, Mileena, etc.) is a Player. Two characters take part in every battle.
- Hit policies: during every hit, the attacker hits the opponent with all of their attack points. From every hit of the opponent, the defender loses their health by the following rule:

$$defender's\ remaining\ health = current\ health\ point - (attack\ point\ of\ the\ attacker - 10\%\ of\ defend\ points\ of\ the\ defender)$$

To keep our game simple, we will stop here!!!

- Implement the system for the client.
  [10 points each for the 2 classes and
  10 points for the superclass that defines the gaming algorithm]

- Create a Demo class where you test at least one instance of each of the games (Mortal Combat and Pokemon). **[5 * 2 = 10 points]**

Submit all of your source code. Please -

- Write each class in a separate file.
- The Player class has Javadoc comments included in it.
- Use a similar strategy to add JavaDoc comments for each of the classes.