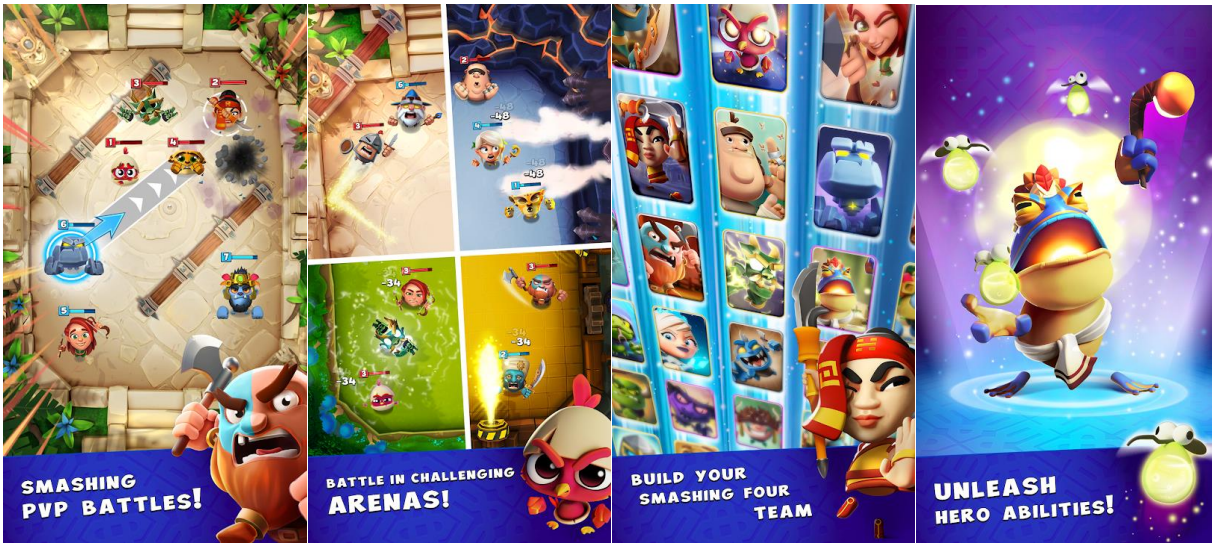
CMP 1002 – Object-Oriented Programming – Spring 2020

Smashing Four Simulator Project

(DEADLINE: 15th of May 23:59)

"Prove your skill in 1v1 PvP! Execute your strategy in different arenas! Smash your way to the victory!"

Outline: In this project, you are expected to write a program that simulates the fundamental aspects of the combat of the game Smashing Four by Geewa.



Smashing Four is an online, card collectable, PvP multiplayer game where you build a squad of four smashing heroes and battle against other players in spectacular arenas. In a combat, up to 4 units fight with another up to 4 units in a turn-based fashion. The combat is divided into discrete rounds, and in each round, one unit of each team has a single turn where it uses one skill and arena shows its effect with a random chance.

Turn order:

At the start of the game, for each unit of each team, the game generates a random number between 1-4. This order stands until the game finishes. Then, the game chooses the team that will start to the battle.

Character Types:

- *Warrior*: is a pure warrior. It has a chance to attack twice on the same opponent.
  - *HP*: 100 – 250
  - *Attack*: 30 – 50
  - *Defence*: 20 – 30
  - *Chance*: 12% – 16%

- *Defender*: is like a castle. Its health points are more than others. It has a chance to strike back after defending.
  - *HP*: 300 – 600
  - *Attack*: 15 – 35
  - *Defence*: 50 – 70
  - *Chance*: 10% – 13%

- *Wizard*: brings the magic on the field. It has a chance to attack multiple opponents at the same time.
    - *HP*: 90 – 200
    - *Attack*: 10 – 30
    - *Defence*: 15 – 25
    - *Chance*: 12% – 16%

- *Healer*: is a sine qua non in the battle. It has a chance to heal its teammate 20% of teammate's HP after attacking.
    - *HP*: 100 – 250
    - *Attack*: 10 – 30
    - *Defence*: 15 – 30
    - *Chance*: 15% – 23%

Arena Types:

- *Casual*: does not affect the battle.
- *FireSpin*: has a chance(10%) to attack (20 pure damage) each unit on the field after each round.
- *Valhalla*: has a chance(5%) to brings the death unit back to the field with 10% of HP.
- *Aurora:* has a chance(10%) to heal (10 healing point) each unit on the field after each round.

Attack and Dealing Damage:

When a unit uses a combat skill over another unit, then the game makes a couple of checks if it is how much damage the attacker dealt with the defender.

- Actual damage points are calculated with attacker's *attack* points and defender's *defence* points via the following formula:

$$actualDamage = \lceil Attack * (100/\lceil 100 + Defence \rceil) \rceil$$

- If the unit takes damage, its HP is subtracted by *actualDamage* value to a minimum of 0.
    - A unit that has 0 HP instantly dies.

Turn Steps:

1. According to the order, the unit chooses one opponent to attack. (Get input from the user to conduct which opponent)
2. The unit performs its special ability if it is lucky.
3. The next unit of the opponent performs Step 1 – 2.
4. Arena shows its effect within its chance rate.
5. Step 1 – 4 are repeated until the game finishes.

Game Steps:

Create at least eight characters.

You need to build the teams (4 vs 4). This can be random or selectable by the user (It is up to you)

Create an arena to play. This can be random or selectable by the user (It is up to you)

The teams are randomly sorted for attack order. The game selects a team to start combat.

At the beginning of each round, print current HP of units.

Process the turn steps explained above.

After each attack, print out which unit attacked to which unit, how much, used a special ability

Print if arena performs its effect.

At the end of each round, print out "End of Round 1, 2, …"

At the end of the combat, which team has won, and which units are alive.

Object-Oriented Programming Requirements & Hints:

1. You are expected to have the following classes based on the mechanics of the game:
   - class Unit
     - class Warrior: public Unit
     - class Defender: public Unit
     - class Wizard: public Unit
     - class Healer: public Unit
   - class Arena
2. You are expected to write appropriate fields and functions to each of the classes above *Example: class Unit has int maxHP, int hp, int chance, …*
3. Local fields of **ALL** classes **MUST BE PROTECTED**.
4. **DO NOT USE POINTERS, INSTEAD USE SMART POINTERS**
5. While designing the classes, if two classes having the same parent class having similar functionality, try to write it as a virtual function
6. You may add new classes, global functions, etc.

NOTE: Official Smashing Four combat mechanism are more complicated as are described in this document. You are required to implement what is written in the document, NOT what is in the game. However, if you want to experiment you can add additional components of the game

NOTE: All calculations will be done as real (i.e., double) operations but the result will be **ROUNDED UP** to int.

NOTE: In random number calculations, you can use the rand() function as well as the srand() function to set up the randomization seed value. If you wish, you can use more complicated randomization tools.

Character Creation:

You may create the characters randomly or precisely at the beginning of the game. The stats of each character must be within the range specified above for their type. You can see some examples:

- Warrior w1(122, 30, 28, 12);
- Warrior w2(206, 39, 29, 13);
- Warrior w3(106, 40, 21, 16);
- Warrior w4(184, 32, 20, 14);
- Warrior w5(203, 36, 21, 13);
- Warrior w6(140, 30, 29, 12);
- Warrior w7(170, 40, 30, 14);

- Defender d1(536, 19, 66, 19);
- Defender d2(309, 34, 51, 15);
- Defender d3(374, 25, 50, 11);
- Defender d4(455, 19, 60, 27);
- Defender d5(438, 25, 68, 11);
- Defender d6(508, 29, 59, 16);
- Defender d7(347, 31, 60, 17);

- Wizard wz1(116, 22, 16, 14);
- Wizard wz2(98, 22, 15, 16);
- Wizard wz3(154, 11, 20, 14);
- Wizard wz4(90, 30, 17, 12);
- Wizard wz5(136, 24, 15, 14);
- Wizard wz6(94, 13, 19, 15);
- Wizard wz7(145, 23, 22, 13);

- Healer h1(139, 24, 19, 18);
- Healer h2(218, 30, 23, 17);
- Healer h3(224, 29, 29, 20);
- Healer h4(199, 13, 26, 16);
- Healer h5(179, 21, 30, 17);
- Healer h6(211, 19, 26, 18);
- Healer h7(146, 22, 25, 23);

THE STATS OF CHARACTERS ARE CONSTANT DURING THE GAME. ONLY YOU NEED TO DEFINE ANOTHER VARIABLE TO USE CURRENT HEALTH POINT OF CHARACTER. THE CURRENT HEALTH POINT CANNOT BE EITHER MORE THAN HEALTH POINT STAT OR LESS THAN ZERO.