COMP2500 실습 5

목차

PocuSoft는 '전쟁의 역사'라는 이름의 게임에 회사의 사활을 걸고 있습니다. 이 게임에서 각 플레이어는 선사시대의 야만용사(barbarian)가 되어 데스매치 스타일로 다른 플레이어들과 싸워야 합니다. 이 게임에서 이기려면 다른 플레이어들을 모두 무찌르고 홀로 살아남아야 합니다. 몇 주에 걸친 계획 끝에 팀장님이 여러분에게 매우 중요한 일감을 하나 주었습니다. 야만용사를 나타내는 클래스를 만들라는 거였죠. 스스로 OOP의 전문가라고 자신하고 있는 여러분이라면 이 정도 클래스는 쉽게 뚝딱 만들겠죠? ;)

1. 프로젝트를 준비한다

1. 자, 스트레칭 한 번 하시고... 설계와 프로그래밍 시작하겠습니다. :)

전반적인 규칙

- 여러분이 작성하는 클래스들은 반드시 `academy.pocu.comp2500.lab5` 패키지 안에 속해야 합니다.
- 어떤 클래스 대신 사용할 수 있는 기본 자료형이 존재한다면(예: Boolean/boolean, Integer/int) 그걸 대신 사용하세요. 기본 자료형 대신 클래스 버전이 허용되는 경우는 제네릭(generic) 클래스의 타입(type) 매개변수로 사용할 때 뿐입니다.

2. `Barbarian` 클래스 구현하기

야만용사는 적을 공격하여 피해(damage)를 입힐 수 있습니다. 각 야만용사는 다음과 같은 속성을 가집니다.

- 고유한 이름: 각 플레이어를 식별함
- 공격력(attack)
- 방어력(defense)
- 생명(health points, HP): 공격을 받을 때마다 입은 피해치만큼 HP가 줄어듭니다. HP가 0이 되면 야만용사는 죽습니다.

2.1 생성자를 구현한다

생성자는 다음의 인자들을 받습니다.

- 야만용사의 이름: `String`
- 최대 HP: `int`
- 공격력: `int`
- 방어력: `int`

```
Barbarian barbarian = new Barbarian("Dragonborn Whiterun", 250, 50, 10);
```

2.2 getHp() 메서드를 구현한다

- 이 메서드는 아무 인자도 받지 않습니다.
- 야만용사의 현재 HP를 반환합니다.

```
Barbarian barbarian = new Barbarian("Dragonborn Whiterun", 250, 50, 10);
int hp = barbarian.getHp(); // hp: 250
```

2.3 attack() 메서드를 구현한다

- 이 메서드는 다음의 인자를 받습니다.
    - 공격할 적
- 적을 공격합니다.
- 반환 값은 없습니다.
- 적이 받는 피해치는 다음과 같이 계산합니다.

    ```
    피해치 = (공격자의 공격력 - 방어자의 방어력) / 2
    ```

    계산을 할 때는 `double` 자료형을 사용하고 계산 뒤에는 소수점 이하는 버리세요. (아래 예 참조)

    ```
    예>
    피해치 = (121 - 100) / 2 = 10.5
    피해치 = 10
    ```

- 다른 야만용사에게 공격을 받은 야만용사의 최소 피해치는 1입니다.

```
Barbarian barbarian0 = new Barbarian("Dragonborn Whiterun", 250, 88, 10);
Barbarian barbarian1 = new Barbarian("Ulfric Stormcloak", 300, 70, 15);

barbarian0.attack(barbarian1);

barbarian1.getHp(); // 264
```

2.4 isAlive() 메서드를 구현한다

- 이 메서드는 아무 인자도 받지 않습니다.
- 야만용사가 죽지 않았다면 참을, 죽었다면 거짓을 반환합니다.

```
Barbarian barbarian0 = new Barbarian("Dragonborn Whiterun", 100, 250, 10);
Barbarian barbarian1 = new Barbarian("Ulfric Stormcloak", 100, 70, 50);

barbarian1.isAlive(); // true

barbarian0.attack(barbarian1);

barbarian1.isAlive(); // false
```

3. Gladiator 클래스 구현하기

'전쟁의 역사'를 출시한 지 몇 개월이 지났습니다. 일단 게이머들의 평이 괜찮아 확장팩을 만들기로 결정했습니다. 이름도 그럴듯하게 '전쟁의 역사: 글래디에이터의 시대'라고 합니다. 이 확장팩을 구매한 플레이어들은 야만용사를 글래디에이터로 업그레이드할 수 있습니다. 글래디에이터는 야만용사가 할 수 있는 일을 다 할 수 있을 뿐 아니라 다음과 같은 추가 능력도 얻습니다.

1. 최대 4개의 스킬을 배워 적을 공격할 때 사용할 수 있음

    플레이어는 자유로이 스킬을 배우거나 초기화(배웠던 스킬을 잊어버림)를 할 수 있습니다. 당연히 막강한 스킬일수록 적을 빨리 쓰러뜨릴 수 있죠! 하지만 세상이 이렇게 쉬울 순 없겠죠? 각 공격 스킬은 쓸 때마다 파워 수치(power points)를 하나씩 소모하기에 스킬을 무한히 사용할 수는 없답니다.

2. 휴식(rest)을 통해 HP를 회복할 수 있음

    HP가 10만큼 회복됩니다. 파워 수치도 더불어 1만큼 회복되는데 글래디에이터가 현재 알고 있는 모든 공격 스킬에 대해 1만큼 회복되는 겁니다.

그럼 이 일은 누가 해야 할까요? 당연히 `Barbarian` 클래스를 제작했었던 여러분이겠죠? `Gladiator`와 `Move` 클래스를 만들어주세요.

3.1 `Move` 클래스를 구현한다

- `Move` 클래스는 공격 스킬(attack move)을 나타냅니다.
- 생성자는 다음의 인자를 받아야 합니다.
  - 공격 스킬의 이름: `String`
  - 공격 스킬의 파워: `int`
  - 최대 파워 수치: `int`. 본 스킬을 사용할 수 있는 최대 횟수를 나타냅니다.
- 이 외에 필요한 메서드가 있다면 자유로이 만드세요. public 메서드도 괜찮습니다.

```
Move move = new Move("Hadoken", 100, 20);
```

3.2 `addMove()` 메서드를 구현한다

- 이 메서드는 1개의 인자를 받습니다.
  - 공격 스킬 개체
- 공격 스킬 개체를 스킬 목록에 추가합니다.
- 이미 알고 있는 스킬은 추가할 수 없습니다.
- 스킬을 성공적으로 추가했다면 참을, 아니라면 거짓을 반환합니다.

```
Gladiator gladiator = new Gladiator("Dragonborn Whiterun", 100, 250, 10);
Move move = new Move("Hadoken", 100, 20);

gladiator.addMove(move); // true
```

3.3 `removeMove()` 메서드를 구현한다

- 이 메서드는 1개의 인자를 받습니다.
  - 공격 스킬의 이름: `String`
- 지정된 이름과 같은 공격 스킬을 제거합니다.
- 공격 스킬을 찾아서 지웠다면 참을, 아니라면 거짓을 반환합니다.

```
Gladiator gladiator = new Gladiator("Dragonborn Whiterun", 100, 250, 10);
Move move = new Move("Hadoken", 100, 20);

gladiator.addMove(move); // true
gladiator.removeMove("Hadoken"); // true
```

3.4 또 다른 `attack()` 메서드를 구현한다

- 이 메서드는 다음의 인자들을 받습니다.
  - 적을 공격할 때 사용할 스킬의 이름: `String`
  - 공격할 적
- 지정된 공격 스킬을 사용하여 적을 공격합니다.
- 이 메서드는 아무것도 반환하지 않습니다.
- 그 스킬을 모를 경우 공격에 실패합니다.
- 남아있는 파워 수치가 충분치 않다면 공격에 실패합니다.
- 적이 받는 피해치는 다음과 같이 계산합니다.

  ```
  피해치 = (공격자의 공격력 / 방어자의 방어력 * 스킬의 파워) / 2
  ```

  계산을 할 때는 `double` 자료형을 사용하고 계산 뒤에는 소수점 이하는 버리세요.
- 공격에 성공하면 최소 1의 피해를 입힙니다.

```
Gladiator gladiator0 = new Gladiator("Dragonborn Whiterun", 100, 250, 10);
Gladiator gladiator1 = new Gladiator("Ulfric Stormcloak", 1000, 300, 77);
Move move = new Move("Hadoken", 120, 20);

gladiator0.addMove(move); // true

gladiator0.attack("Hadoken", gladiator1);

gladiator1.getHp(); // 806
```

3.5 `rest()` 메서드를 구현한다

- 이 메서드는 아무 인자도 받지 않습니다.
- HP는 10만큼, 파워 수치는 알고 있는 스킬마다 1씩 회복합니다.

```
Gladiator gladiator0 = new Gladiator("Dragonborn Whiterun", 100, 250, 10);
Gladiator gladiator1 = new Gladiator("Ulfric Stormcloak", 1000, 300, 77);
Move move = new Move("Hadoken", 120, 20);

gladiator0.addMove(move);

gladiator0.attack("Hadoken", gladiator1);

gladiator1.getHp(); // 806

gladiator1.rest();

gladiator1.getHp(); // 816
```

4. `Knight` 클래스 구현하기

게임을 서비스한 지 5년이 지났습니다. 유지보수 모드로 전환하기 전에 마지막 확장팩을 출시하겠답니다. 마지막 확장팩은 '전쟁의 역사: 원탁의 기사'라는 이름을 붙었네요. 플레이어는 글래디에이터를 기사(knight)로 진화해 추가적인 능력을 얻을 수 있습니다.

1. 각 기사는 펫(pet) 한 마리를 동반할 수 있습니다.
2. 기사와 펫은 콤보 공격을 할 수 있습니다.

콤보 공격은 그다지 복잡하지 않습니다. 기사가 적을 공격할 때, 펫도 그 적을 같이 공격할 뿐입니다.

이번에도 팀장님이 여러분에게 `Pet`과 `Knight` 클래스를 만들라고 지시했습니다. 이 클래스들만 만들면 신작 개발팀으로 옮겨준다고 하시니 열심히 하도록 합시다.

4.1 `Pet` 클래스를 구현한다

- `Pet` 클래스의 생성자는 다음의 인자를 받아야 합니다.
  - 펫의 이름: `String`
  - 펫의 공격력: `int`
- 이 외에 필요한 메서드가 있다면 자유로이 만드세요. public 메서드도 괜찮습니다.

```
Pet pet = new Pet("Yoshi", 50);
```

4.2 `setPet()` 메서드를 구현한다

- 이 메서드는 1개의 인자를 받습니다.
  - 펫 개체
- 인자로 들어온 펫을 동반하는 펫으로 설정(set)합니다.
- 이 메서드는 아무것도 반환하지 않습니다.
- 인자로 `null`이 들어오면 더 이상 동반하는 펫이 없어야 합니다.

```
Knight knight = new Knight("Dragonborn Whiterun", 100, 250, 10);
Pet pet = new Pet("Yoshi", 50);

knight.setPet(pet);
```

4.3 `attackTogether()` 메서드를 구현한다

- 이 메서드는 1개의 인자를 받습니다.
  - 콤보 공격을 받을 적
- 공격자와 펫이 적을 공격합니다.

- 이 메서드는 아무것도 반환하지 않습니다.
- 이 공격을 사용하려면 펫을 가지고 있어야 합니다.
- 적이 받는 피해치는 다음과 같이 계산합니다.

  ```
  피해치 = (공격자의 공격력 + 펫의 공격력 – 방어자의 방어력) / 2
  ```

  계산을 할 때는 `double` 자료형을 사용하고 계산 뒤에는 소수점 이하는 버리세요.
- 공격에 성공하면 최소 1의 피해를 입힙니다.

```
Knight knight0 = new Knight("Dragonborn Whiterun", 100, 250, 10);
Knight knight1 = new Knight("Ulfric Stormcloak", 1000, 300, 77);
Pet pet = new Pet("Yoshi", 100);

knight0.setPet(pet);

knight0.attackTogether(knight1);

knight1.getHp(); // 864
```

5. 본인 컴퓨터에서 테스트하는 법

- `Program.java`를 아래처럼 바꾼 뒤 실행하세요.

```
package academy.pocu.comp2500.lab5.app;

import academy.pocu.comp2500.lab5.Barbarian;
import academy.pocu.comp2500.lab5.Gladiator;
import academy.pocu.comp2500.lab5.Knight;
import academy.pocu.comp2500.lab5.Move;
import academy.pocu.comp2500.lab5.Pet;

public class Program {

    public static void main(String[] args) {
        Barbarian barbarian0 = new Barbarian("Dragonborn Whiterun", 250, 210, 60);
        Barbarian barbarian1 = new Barbarian("Ulfric Stormcloak", 200, 70, 10);

        barbarian0.attack(barbarian1);

        assert barbarian1.getHp() == 100;
        assert barbarian1.isAlive();

        barbarian0.attack(barbarian1);

        assert barbarian1.getHp() == 0;
        assert !barbarian1.isAlive();

        Gladiator gladiator0 = new Gladiator("Parthunax", 250, 210, 10);
        Gladiator gladiator1 = new Gladiator("Zoro", 100, 150, 65);
        Move move0 = new Move("Gomu Gomu no pistol", 50, 10);
        Move move1 = new Move("Thunderbolt", 90, 15);
        Move move2 = new Move("Ice Beam", 90, 10);
        Move move3 = new Move("Surf", 90, 15);

        assert gladiator0.addMove(move0);
        assert gladiator0.addMove(move1);
        assert gladiator0.addMove(move2);
        assert gladiator0.addMove(move3);

        assert gladiator0.removeMove("Surf");

        gladiator0.attack("Gomu Gomu no pistol", barbarian0);

        assert barbarian0.getHp() == 163;
        assert barbarian0.isAlive();

        gladiator0.attack("Gomu Gomu no pistol", gladiator1);

        assert gladiator1.getHp() == 20;

        gladiator1.rest();

        assert gladiator1.getHp() == 30;

        Knight knight0 = new Knight("Naruto", 252, 310, 99);
        Knight knight1 = new Knight("Sasuke", 250, 290, 111);
        Pet pet0 = new Pet("Giant Toad", 180);

        knight0.setPet(pet0);

        knight0.attackTogether(gladiator0);

        assert gladiator0.getHp() == 10;

        knight0.attackTogether(knight1);

        assert knight1.getHp() == 61;
    }
}
```

6. 커밋, 푸시 그리고 빌드 요청

이건 어떻게 하는지 이제 다 아시죠? :)