C1- Code & Go

pool_php_d06

Day 06

Abstract Class & Interfaces

CODING ACADEMY
BY EPITECH

Introduction

Today we will continue to go deeper into OOP, you will keep using all the notions from the previous days, and you will also discover a few new things:

- Abstract class
- Interfaces

We will also make you use Exceptions in today's exercises.

By notation interfaces usually start with an "i" in upper case and abstract class usually starts with an "a" in upper case, but it's not mandatory and it could start with the letter in lower case, or simply not start with those letters.
Except specified otherwise, all messages must be followed by a newline.

Except specified otherwise, the getter and setter name will always be following this format "getAttribute" or "setAttribute", if my attribute name is "someWeirdNameAttribute" its getter will be "getSomeWeirdNameAttribute" (for your general knowledge, it's known as "CamelCase").

Remember, that when your class uses another class, it should include itself and the one needed because the moulinette will only include the bare minimum.

Exercise 01

4pt

Turn in: pool_php_d06/ex_01/AWeapon.php
Restrictions: None.

Create an abstract class "AWeapon" with the following attribute :

- name (a string), the name of the weapon.
- apcost (an integer), the action point cost to use the weapon.
- damage (an integer), the number of damage dealt by the weapon.
- melee (a boolean), representing if the weapon is use for close combat or not.

These attributes will have getter (getName, getApcost, getDamage, isMelee) but no setter.

The constructor of this class will take the name, the apcost and the damage in the following order. If onw of the parameter is not of the good type, you will throw an exception with the following message:

```
Error in AWeapon constructor. Bad parameters.
```

Because it is an exception message there should be **no newline** at the end of this message.

You do not have to handle missing parameters.

By default melee will be false.

You will also add an abstract public method named "attack".
Example :

```
<?php

include_once("AWeapon.php");

new AWeapon();
```

```
Terminal
~/pool_php_d06> php index.php
PHP Fatal error:  Cannot instantiate abstract class AWeapon in /home/gecko/rendu/pool_php_d06
on line 5
~/pool_php_d06>
```

This error means that to test, you will need to create a new class =P.

Exercise 02

2pt

Turn in: pool_php_d06/ex_02/AWeapon.php
pool_php_d06/ex_02/PlasmaRifle.php
pool_php_d06/ex_02/PowerFist.php
Restrictions: None.

Here are the features of the different class inheriting from "AWeapon" that you must create:

- PlasmaRifle :
  - Name : "Plasma Rifle"
  - Damage : 21
  - AP cost : 5
  - Output of attack() : "PIOU"
  - Not Melee
- PowerFist :
  - Name : "Power Fist"
  - Damage : 50
  - AP cost : 8
  - Output of attack() : "SBAM"
  - Melee

The call to attack() must display the output of the weapon followed by a newline

Exercise 03

2pt

Turn in: pool_php_d06/ex_03/IUnit.php
Restrictions: None.

Let's create our first interface. It will be called IUnit.

This interface will be used to determine the base methods that we want our units to implement when they are created.

You must add a few methods to this interface as to do that:

- A public method equip, which will take a parameter.
- A public method attack, which will take a parameter.
- A public method receiveDamage, also taking a parameter.
- A public method moveCloseTo, also taking a parameter.
- A public method recoverAP, taking no parameter.

Exercise 04

3pt

Turn in: pool_php_d06/ex_04/IUnit.php
pool_php_d06/ex_04/AMonster.php
pool_php_d06/ex_04/ASpaceMarine.php
Restrictions: None

Copy your interface from the previous exercise. Now we will actually implement the base of our Monsters and our SpaceMarines. Those two classes must be Abstract class.

For both classes you must create three attributes, a "name", "hp", and "ap" the first onw will be the name of the unit and the second will be its health points, finally the third one will be "Action points", the resource that our unit will use to make an action. Those attributes must have a getter without setter (getName, getHp and getAp). The name will be given during the construction of the unit, however the base hp will depend on the type of the unit so we actually don't know it yet, so let's simply initiate it to 0 for now. It will be the same for AP, so let's put it to 0 too for now.

For AMonster, you will add an attribute "damage" and its getter "getDamage", it will be set to 0 for now and will be set later for each monsters, you will also add an attribute "apcost" that will also be set to 0 for now.

equip: A call to this function will display "Monsters are proud and fight with their own bodies."
attack: If the parameter given is not implementing IUnit you will need to throw an exception saying:

```
Error in AMonster. Parameter is not an IUnit.
```

without newline because once again it's an Exception. You don't have to handle the case where no parameter is given.

All monsters will be of melee type, which means that they first need to get in melee range (see moveCloseTo method) before being able to attack their target, so if the monster is not in melee range of the Unit passed as parameter you will display

```
[name]: I'm too far away from [target's name].
```

Where [name] should be replaced by the name of the monster and [target's name] by its target's name.

If our monster is in range, it will then check if it has enough AP to attack, that's when its attributes "ap" and "apcost" matters. To be able to attack it should have more "ap" available than the "apcost" of an attack.

If the attack succeed you must deduce "apcost" from "ap" and call the method "receiveDamage" of the target while passing as parameter the "damage" of the monster, and you should display before calling receiceDamage:

```
[name] attacks [target's name].
```

Where [name] should be replaced by the name of the monster and [target's name] by its target's name.

receiveDamage: This function will always receive an integer representing the damage received by the monster. There is no error to handle. The HP of the monster must be deduced by this amount, however if the HP get to 0 or under this monster will be considered as dead and each of its method should return false from now on (except the getter).

moveCloseTo: If the parameter received is not an IUnit, you will throw the same exception message as in "attack". This function will make the monster move closer to its target, which means that a later call to "attack" will succeed.

We will consider that the monster can only be close to one target at a time.
If the monster is not already close to irs target a call to this function will display:

```
[name] is moving closer to [target's name].
```

Where [name] should be replaced by the name of the monster and [target's name] by its target's name.

recoverAP: a call to this method will increase the "ap" of the monster by 7 at most. It should never go over 50. The ASpaceMarine will be fully created in the following exercise, as of now it should just have what has been specified at the beginning of this exercise, and all the methods of the interface should be empty (it should be possible to instantiate an ASpaceMarine as of now, it juste won't have all functionalities yet) =D

Exercise 05

3pt

Turn in: pool_php_d06/ex_05/IUnit.php
pool_php_d06/ex_05/AMonster.php
pool_php_d06/ex_05/ASpaceMarine.php
pool_php_d06/ex_05/AWeapon.php
pool_php_d06/ex_05/PlasmaRifle.php
pool_php_d06/ex_05/PowerFist.php
Restrictions: None

Copy Your classes from the previous exercise.

Let's continue the creation of our ASpaceMarine class. It should already have a few attributes.
Let's add an attribute "weapon" and its getter "getWeapon"
equip: The parameter received must be an "AWeapon", otherwise you will need to throw an exception with the following message:

```
Error in ASpaceMarine. Parameter is not an AWeapon.
```

without newline because once again it's an Exception. You don't have to handle the case where no parameter is given.
Our SpaceMarine needs to take this new weapon and equip it. If it's done successfully, you will display:

```
[name] has been equipped with a [weapon's name].
```

Where [name] should be replaced by our SpaceMarine's name and [weapon's name] by the weapon's name.
If the weapon has already been taken by another SpaceMarine the function will do nothing. It's up to you to decide how you will handle this.

attack: If the parameter given is not implementing IUnit, you will need to throw an exception saying:

```
Error in ASpaceMarine. Parameter is not an IUnit.
```

without newline because ... It's an Exception. You don't have to handle the case where no parameter is given.

If our SpaceMarine doesn't have any weapon equipped he will say:

```
[name]: Hey, this is crazy. I'm not going to fight this empty handed.
```

Where [name] should be replaced by our SpaceMarine's name and the function will do nothing else.

If the weapon equipped is a melee one and our SpaceMarine is not in range, it will say:

```
[name]: I'm too far away from [target's name].
```

Where [name] should be replaced by our SpaceMarine's name and [target's name] by its target's name.

Like in AMonster, our SpaceMarine will need enough AP to attack. If his available AP are at least equal to its weapon's cost and he is in range (or is using a ranged weapon), you will need to call the "attack" method of the weapon equipped but also the target's receivedDamage method while passing the weapon's damage as parameter.

Moreover, if the attack has been successful, you should deduce the weapon's cost to the SpaceMarine's AP and you should display before the call to receivedDamage:

```
[name] attacks [target's name] with a [weapon's name].
```

Where [name] should be replaced by our SpaceMarine's name, [target's name] by its target's name and [weapon's name] bu the equipped weapon's name.

receivedDamage: This function will work exactly like the AMonster one.

moveCloseTo: This function will work exactly like the AMonster one. recoverAP: This function will work exactly like the AMonster one, except that it will rcover 9 AP instead of 7.

Exercise 06

2pt

Turn in: pool_php_d06/ex_06/IUnit.php
pool_php_d06/ex_06/AMonster.php
pool_php_d06/ex_06/ASpaceMarine.php
pool_php_d06/ex_06/AWeapon.php
pool_php_d06/ex_06/PlasmaRifle.php
pool_php_d06/ex_06/PowerFist.php
pool_php_d06/ex_06/TacticalMarine.php
pool_php_d06/ex_06/AssaultTerminator.php
Restrictions: None

Copy Your classes from the previous exercise.

It is finally time to create our actual Space Marine. As you should have guessed, all SpaceMarines will need to inherit from the ASpaceMarine class.

Let's begin with TacticalMarine. During its creation ot should say:

```
        [name] on duty.
```

Where [name] should be replaced by our SpaceMarine's name.

Its name will always be received during its creation.

When being created, you will equip your TacticalMarine with a PlasmaRifle.

A TacticalMarine will have 100Hp and 20AP by default.

On its death you will display:

```
        [name] the Tactical Marine has fallen !
```

Where [name] should be replaced by our SpaceMarine's name.

A Tactical Marine being ... Tactical will regenerate its AP faster than other Marines. It will actually take back 12 AP instead of 9 when recoverAP will be called.

Let's talk about AssaultTerminator now. On its creation it will also receive its name and will display :

```
        [name] has teleported from space.
```

Where [name] should be replaced by our SpaceMarine's name.

By default it will possess 150HP and 30AP.
During its creation you should equip it with a PowerFist.

Moreover, AssaultTerminator being a bit more resistant than other Marines when his method "receivedDamage" will be called it will reduce the damage by 3. However, the damage received can't be reduced under 1.

During its destruction you will display:

```
        BOUUUMMMM ! [name] has exploded.
```

Where [name] should be replaced by our SpaceMarine's name.

Exercise 07

2pt

Turn in: pool_php_d06/ex_07/IUnit.php
pool_php_d06/ex_07/AMonster.php
pool_php_d06/ex_07/ASpaceMarine.php
pool_php_d06/ex_07/AWeapon.php
pool_php_d06/ex_07/PlasmaRifle.php
pool_php_d06/ex_07/PowerFist.php
pool_php_d06/ex_07/TacticalMarine.php
pool_php_d06/ex_07/AssaultTerminator.php
pool_php_d06/ex_07/RadScorpion.php
pool_php_d06/ex_07/SuperMutant.php
Restrictions: None

Copy Your classes from the previous exercise.

We will now finally create our Monsters! As you should have guessed, all monsters class will have to inherit from the "AMonster" class.

Our monsters will have a generic name. They will all be called "RadScorpion" or "SuperMutant" followed by an id. For example, the first RadScorpion existing will be called "RadScorpion #1", the second one will be "RadScorpion #2". Because of this, our Monsters will not take any parameter during their creation.

Let's create the RadScorpion one first, during its creation it will display:

```
[name]: Crrr!
```

Where [name] should be replaced by the monster's name.

And during its destruction it will display:

```
[name]: SPROTCH!
```

Where [name] should be replaced by the monster's name.

RadScorpion is a fairly common monster, it will only have 80HP however, it will start with the maximum AP, 50, each one of its attacks will deal 25 damage and will cost him 8 AP.
However, they can be pretty scary, that's why we will consider that if they are attacking a marine that is not an "AssaultTerminator" they will deal double damage (because the Marine will be too scared to move away).

Now let's create our SuperMutant.

```
[name]: Roaarrr!
```

Where [name] should be replaced by the monster's name.

And during its destruction it will display:

```
[name]: Urgh!
```

Where [name] should be replaced by the monster's name.

They will start with 170HP, 20AP. Each one of its attack will deal 60 damages but will come with the price of 20AP.

When SuperMutant recoverAP, they will also recover HP, with a maximum of 10HP recovered by call (170HP being their full health).

Exercise 08

2pt

Turn in: pool_php_d06/ex_08/IUnit.php
pool_php_d06/ex_08/AMonster.php
pool_php_d06/ex_08/ASpaceMarine.php
pool_php_d06/ex_08/AWeapon.php
pool_php_d06/ex_08/PlasmaRifle.php
pool_php_d06/ex_08/PowerFist.php
pool_php_d06/ex_08/TacticalMarine.php
pool_php_d06/ex_08/AssaultTerminator.php
pool_php_d06/ex_08/RadScorpion.php
pool_php_d06/ex_08/SuperMutant.php
pool_php_d06/ex_08/SpaeArena.php
Restrictions: None

Copy Your classes from the previous exercise.

Create a class SpaceArena.
We will use this class to simulate fights between teanms of Monsters and teams of SpaceMarines.

This class will have 3 methods.
The first one will be "enlistMonsters", it will take an array of Monsters as parameter.
If any element of the parameter is not an "AMonster" you will need to throw an exception with the message:

```
Stop trying to cheat!
```

Without newline, because it's an Exception message.
It will add the monsters given as parameter to the one already registered to fight.

You will then create an "enlistSpaceMarines" method, that will take an array of SpaceMarines as parameter.
If any element of the parameter is not an "ASpaceMarine" you will need to throw an exception with the message:

```
Stop trying to cheat!
```

Without newline, because it's an Exception message.
It will add the SpaceMarines given as parameter to the one already registered to fight.
It should not be possible to add a warrior which is already enlisted for a fight.
If an exception has been thrown, the warriors placed before the cheater in the array should still be enlisted. (For example, if enlistMonsters receives an array like this [RadScorpion, RadScorpion, somethingElse, SuperMutant], both RadScorpion should be enlisted but the SuperMutant should not be.)

Now you will create the main method, "fight". It will take no parameter.

This function will make the registered teams of monsters and spaceMarines fight between them.
If no monsters are registered it will say:

```
No monster available to fight.
```

If no spaceMarines are registered it will say:

```
Those cowards ran away.
```

If at least one of the two team is missing the function will stop there by returning false.

Here is how a fight should process:

- When a new round between a monster and spaceMarine begin, the spaceMarine always go first.
- The one playing will try to attack, if it's a success its turn is over. If it failed because he wasn't in range, he will go closer. If it failed because he didn't have enough AP, he will call is method "recoverAP" once.
- The turn then goes to the opponent. This process repeats until one of the opponent has fallen. The winner call his function "recoverAP" once before starting its next fight.
- If the spaceMarine has won, the second monster comes in the arena, and the whole process starts again until one of the two teams has been defeated (If the monster has won, then second spaceMarine enters the fray).

Every time a monster or a spaceMarine enters the fight, you will display:

```
[name] has entered the arena.
```

The SpaceMarine shall always be presented first if both fighters enter at the same time.

At the end of the fight you will display:

```
The [team's name] are victorious.
```

Where [team's name] will be replaced by "monsters" or "spaceMarines".

Every victor will stay in the arena waiting for the next fight.

Example :

```
<?php

include_once("SpaceArena.php");
include_once("AssaultTerminator.php");
include_once("TacticalMarine.php");
include_once("RadScorpion.php");
include_once("SuperMutant.php");

$arena = new SpaceArena();

$arena->enlistMonsters([new RadScorpion(), new SuperMutant(), new RadScorpion()]);
$arena->enlistSpaceMarines([new TacticalMarine("Joe"), new AssaultTerminator("Abaddon"), new
    TacticalMarine("Rose")]);

$arena->fight();

$arena->enlistMonsters([new SuperMutant(), new SuperMutant()]);
$arena->fight();
```

Terminal

```
~/pool_php_d06> php index.php
RadScorpion #1:  Crrr!
SuperMutant #1:  Roaarrr !
RadScorpion #2:  Crrr!
Joe on duty.
Joe has been equipped with a Plasma Rifle.
Abaddon has teleported from space.
Abaddon has been equipped with a Power Fist.
Rose on duty.
Rose has been equipped with a Plasma Rifle.
Joe has entered the arena.
RadScorpion #1 has entered the arena.
Joe attacks RadScorpion #1 with a Plasma Rifle.
PIOU
RadScorpion #1:  I'm too far away from Joe.
RadScorpion #1 is moving closer to Joe.
Joe attacks RadScorpion #1 with a Plasma Rifle.
PIOU
RadScorpion #1 attacks Joe.
Joe attacks RadScorpion #1 with a Plasma Rifle.
PIOU
RadScorpion #1 attacks Joe.
Abaddon has entered the arena.
Abaddon:  I'm too far away from RadScorpion #1.
Abaddon is moving close to RadScorpion #1.
RadScorpion #1:  I'm too far away from Abaddon.
RadScorpion #1 is moving closer to Abaddon.
Abaddon attacks RadScorpion #1 with a Power Fist.
SBAM
SuperMutant #1 has entered the arena.
Abaddon:  I'm too far away from SuperMutant #1.
RadScorpion #1:  SPROTCH !
Abaddon is moving close to SuperMutant #1.
SuperMutant #1:  I'm too far away from Abaddon.
SuperMutant #1 is moving closer to Abaddon.
Abaddon attacks SuperMutant #1 with a Power Fist.
SBAM
SuperMutant #1 attacks Abaddon.
Abaddon attacks SuperMutant #1 with a Power Fist.
SBAM
Abaddon attacks SuperMutant #1 with a Power Fist.
SBAM
Abaddon attacks SuperMutant #1 with a Power Fist.
SBAM
RadScorpion #2 has entered the arena.
Abaddon:  I'm too far away from RadScorpion #2.
SuperMutant #1:  Urgh !
Abaddon is moving closer to RadScorpion #2.
RadScorpion #2:  I'm too far away from Abaddon.
RadScorpion #2 is moving closer to Abaddon.
Abaddon attacks RadScorpion #2 with a Power Fist.
SBAM
```

Terminal

```
RadScorpion #2 attacks Abaddon.
Abaddon attacks RadScorpion #2 with a Power Fist.
SBAM
The spaceMarines are victorious.
SuperMutant #2:  Roaarrr !
SuperMutant #3:  Roaarrr !
Abaddon has entered the arena.
SuperMutant #2 has entered the arena.
Abaddon:  I'm too far away from SuperMutant #2.
RadScorpion #2:  SPROTCH!
Abaddon is moving closer to SuperMutant #2.
SuperMutant #2:  I'm too far away from Abaddon.
SuperMutant #2 is moving closer to Abaddon.
Abaddon attacks SuperMutant #2 with a Power Fist.
SBAM
SuperMutant #2 attacks Abaddon.
Abaddon attacks SuperMutant #2 with a Power Fist.
SBAM
Abaddon attacks SuperMutant #2 with a Power Fist.
SBAM
SuperMutant #2 attacks Abaddon.
Rose has entered the arena.
Rose attacks SuperMutant #2 with a Plasma Rifle.
PIOU
SuperMutant #2:  I'm too far away from Rose.
Rose attacks SuperMutant #2 with a Plasma Rifle.
PIOU
SuperMutant #2:  I'm too far away from Rose.
Rose attacks SuperMutant #2 with a Plasma Rifle.
PIOU
SuperMutant #2:  I'm too far away from Rose.
SuperMutant #2 is moving closer to Rose.
Rose attacks SuperMutant #2 with a Plasma Rifle.
PIOU
SuperMutant #3 entered the arena.
Rose attacks SuperMutant #3 with a Plasma Rifle.
PIOU
SuperMutant #3:  I'm too far away from Rose.
SuperMutant #3 is moving closer to Rose.
Rose attacks SuperMutant #3 with a Plasma Rifle.
PIOU
SuperMutant #3 attacks Rose.
Rose attacks SuperMutant #3 with a Plasma Rifle.
PIOU
Rose attacks SuperMutant #3 with a Plasma Rifle.
PIOU
SuperMutant #3 attacks Rose.
The monsters are victorious.
SuperMutant #3:  Urgh!
SuperMutant #2:  Urgh!
Joe the Tactical Marine has fallen !
BOUUUMMMM ! Abaddon has exploded.
Rose the Tactical Marine has fallen!
```