

PHP POO day 5

Aujourd'hui, nous allons continuer à approfondir la POO, vous continuerez à utiliser toutes les notions des jours précédents, et vous découvrirez également quelques nouvelles choses :

- Classe abstraite
- Interfaces

Nous vous ferons également utiliser des exceptions dans les exercices d'aujourd'hui.

Par convention, les interfaces commencent généralement par un "i" en majuscule et les classes abstraites par un "a" en majuscule. Les classes abstraites commencent généralement par un "a" en majuscule, mais ce n'est pas obligatoire et on peut commencer par la lettre en minuscule, ou simplement ne pas commencer par ces lettres.

Sauf indication contraire, tous les messages doivent être suivis d'une nouvelle ligne.

Sauf indication contraire, le nom du getter et du setter suivra toujours le format "getAttribute" ou "setAttribute", si le nom de mon attribut est "someWeirdNameAttribute", son getter sera "getSomeWeirdNameAttribute" (pour votre culture générale, c'est connu sous le nom de "CamelCase").

Exercice 1

Fichiers :

ex_01/AWeapon.php

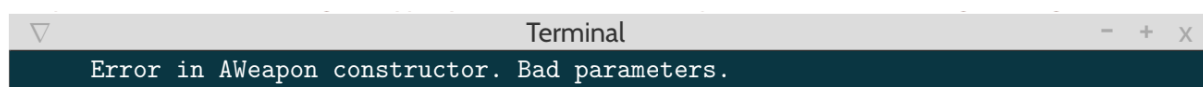
Fichiers exercice : AWeapon.php, exercice.php (to test the code)

Créez une classe abstraite AWeapon ayant les attributs suivants :

- name (une chaîne), le nom de l'arme.
- apcost (un nombre entier), le coût en points d'action pour utiliser l'arme.
- damage (un entier), le nombre de dégâts infligés par l'arme.
- melee (un booléen), représentant si l'arme est utilisée pour le combat rapproché ou non.

Ces attributs auront des getter (getName, getApcost, getDamage, isMelee) mais pas de setter.

Le constructeur de cette classe prendra le nom, l'apcost et les dommages dans l'ordre suivant. Si l'un des paramètres n'est pas du bon type, vous lancerez une exception avec le message suivant :

A screenshot of a terminal window with a dark background. The title bar at the top says "Terminal" and has standard window control buttons (minimize, maximize, close). The main area of the terminal displays the text "Error in AWeapon constructor. Bad parameters." in a light-colored font.

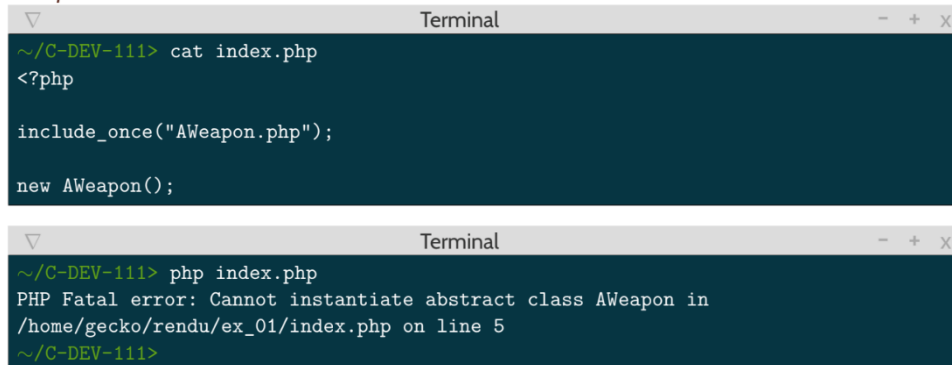
Comme il s'agit d'un message d'exception, il ne doit pas y avoir de nouvelle ligne à la fin de ce message.

Vous n'avez pas à gérer les paramètres manquants.

Par défaut, la mêlée sera fausse.

Vous ajouterez également une méthode publique abstraite appelée "attack".

Example :



```
~/C-DEV-111> cat index.php
<?php

include_once("AWeapon.php");

new AWeapon();

~/C-DEV-111> php index.php
PHP Fatal error: Cannot instantiate abstract class AWeapon in
/home/gecko/rendu/ex_01/index.php on line 5
~/C-DEV-111>
```

Cette erreur signifie que pour tester, vous devrez créer une nouvelle classe =P.

Exercice 2

Fichiers :

ex_02/AWeapon.php

ex_02/PlasmaRifle.php

ex_02/PowerFist.php

Voici les caractéristiques des différentes classes héritant de "AWeapon" que vous devez créer :

- PlasmaRifle :

- Name : "Plasma Rifle" (fusil à plasma)
- Damage: 21
- AP cost: 5
- Sortie de attack() : "PIOU"
- Not Mêlée

- PowerFist :

- Name: "Power Fist" (poing puissant)
- Damage : 50
- AP cost : 8

- Sortie de l'attack() : "SBAM"
- Mêlée

L'appel à attack() doit afficher la sortie de l'arme suivie d'une nouvelle ligne.

Exercice 3

Fichiers :

ex_03/IUnit.php

Créons notre première interface. Elle s'appellera IUnit.

Cette interface sera utilisée pour déterminer les méthodes de base que nous voulons que nos unités mettent en œuvre lorsqu'elles sont créées.

Pour ce faire, vous devez ajouter quelques méthodes à cette interface :

- Une méthode publique equip, qui prendra un paramètre.
- Une méthode publique attack, qui prendra un paramètre.
- Une méthode publique receiveDamage, prenant également un paramètre.
- Une méthode publique moveCloseTo, prenant également un paramètre.
- Une méthode publique recoverAP, sans paramètre.

Exercice 4

Fichiers :

ex_04/IUnit.php

ex_04/AMonster.php

ex_04/ASpaceMarine.php

Copiez votre interface de l'exercice précédent. Maintenant nous allons réellement implémenter la base de nos Monstres et de nos SpaceMarines. Ces deux classes doivent être des classes abstraites.

Pour ces deux classes, vous devez créer trois attributs, un "nom", un "hp" et un "ap". Le premier sera le nom de l'unité et le second sera ses points de vie, enfin le troisième sera les "points d'action", la ressource que notre unité utilisera pour effectuer une action.

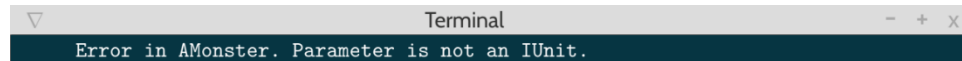
Ces attributs doivent avoir un getter sans setter (getName, getHp et getAp).

Le nom sera donné lors de la construction de l'unité, cependant le hp de base dépendra du type d'unité, donc nous ne le connaissons pas encore, donc initialisons-le simplement à 0 pour le moment. Il en sera de même pour AP, donc mettons-le aussi à 0 pour le moment.

Pour AMonster, vous ajouterez un attribut "damage" et son getter "getDamage", il sera mis à 0 pour le moment et sera défini plus tard pour chaque monstre, vous ajouterez également un attribut "apcost" qui sera également fixé à 0 pour l'instant.

equip : Un appel à cette fonction affichera "Monsters are proud and fight with their own bodies."

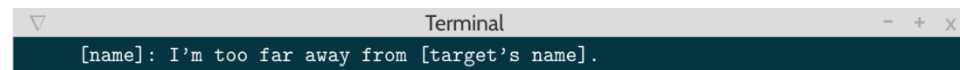
attack : Si le paramètre donné n'implémente pas IUnit, vous devrez lancer une exception disant :



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

Sans nouvelle ligne, car il s'agit encore une fois d'une exception. Vous n'avez pas à gérer le cas où aucun paramètre n'est donné.

Tous les monstres seront de type mêlée, ce qui signifie qu'ils doivent d'abord se mettre à portée de mêlée (voir la méthode CloseTo) avant de pouvoir attaquer leur cible, donc si le monstre n'est pas à portée de mêlée de l'Unité passée en paramètre, vous afficherez

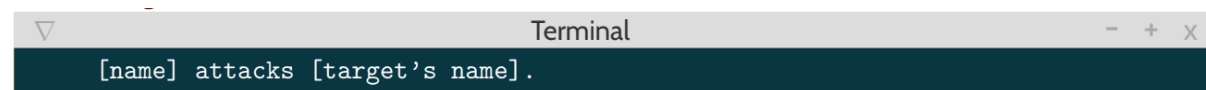


```
Terminal
[name]: I'm too far away from [target's name].
```

Où [name] doit être remplacé par le nom du monstre et [target's name] par le nom de sa cible.

Si notre monstre est à portée, il va alors vérifier s'il a assez de AP pour attaquer, c'est là que ses attributs "ap" et "apcost" ont de l'importance. Pour être en mesure d'attaquer, il doit avoir plus ou autant de AP disponibles que le "coût d'ap" d'une attaque.

Si l'attaque réussit vous devez déduire "apcost" de "ap" et appeler la méthode "receiveDamage" de la cible en passant en paramètre le "damage" du monstre, et vous devez afficher avant d'appeler receiveDamage :



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

Où [name] doit être remplacé par le nom du monstre et [target's name] par le nom de sa cible.

recoverAP : un appel à cette méthode augmentera l'ap du monstre de 7 au maximum. Il ne doit jamais dépasser 50.

L'ASpaceMarine sera entièrement créé dans l'exercice suivant, pour l'instant il doit juste avoir ce qui a été spécifié au début de cet exercice, et toutes les méthodes de l'interface doivent être vides (il doit être possible d'instancier un ASpaceMarine dès maintenant, il n'aura juste pas encore toutes les fonctionnalités) =D

Exercice 5

Fichiers :

ex_05/IUnit.php

ex_05/AMonster.php

ex_05/ASpaceMarine.php

ex_05/AWeapon.php

ex_05/PlasmaRifle.php

ex_05/PowerFist.php

Copiez vos classes des exercices précédents.

Continuons la création de notre classe ASpaceMarine. Elle devrait déjà avoir quelques attributs.

Ajoutons un attribut "weapon" et son getter "getWeapon".

equip : Le paramètre reçu doit être un "AWeapon", sinon vous devrez lever une exception avec le message suivant :

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

Sans nouvelle ligne, car il s'agit encore une fois d'une exception. Vous n'avez pas à gérer le cas où aucun paramètre n'est donné.

Notre SpaceMarine doit prendre cette nouvelle arme et l'équiper. Si cela est fait avec succès, vous afficherez :

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

Où [name] doit être remplacé par le nom de notre SpaceMarine et [weapon's name] par le nom de l'arme. Si l'arme a déjà été prise par un autre SpaceMarine, la fonction ne fera rien. C'est à vous de décider comment vous allez gérer cela.

attack: Si le paramètre donné n'implémente pas IUnit, vous devrez lancer une exception disant :

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

Sans nouvelle ligne parce que... C'est une exception. Vous n'avez pas à traiter le cas où aucun paramètre n'est donné.

Si notre SpaceMarine n'a pas d'arme équipée, il dira :

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

Où [name] doit être remplacé par le nom de notre SpaceMarine et la fonction ne fera rien d'autre.

Si l'arme équipée est une arme de mêlée et que notre SpaceMarine n'est pas à portée, il dira :

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

Où [noma] doit être remplacé par le nom de notre SpaceMarine et [target's name] par le nom de sa cible.

Comme dans AMonster, notre SpaceMarine aura besoin de suffisamment de PA pour attaquer. Si ses PA disponibles sont au moins égaux au coût de son arme et qu'il est à portée (ou qu'il utilise une arme à distance), il faudra appeler la méthode "attack" de l'arme équipée mais aussi la méthode receivedDamage de la cible en passant les dégâts de l'arme en paramètre.

De plus, si l'attaque est réussie, vous devez déduire le coût de l'arme aux AP du SpaceMarine et vous devriez l'afficher avant l'appel à receivedDamage :

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

Où [name] doit être remplacé par le nom de notre SpaceMarine, [target's name] par le nom de sa cible et [weapon's name] par le nom de l'arme équipée.

receivedDamage : Cette fonction fonctionnera exactement comme celle d'AMonster.

moveCloseTo : Cette fonction fonctionnera exactement comme celle d'AMonster.

recoverAP : Cette fonction fonctionnera exactement comme celle d'AMonster, sauf qu'elle récupérera 9 PA au lieu de 7.

Exercice 6

Fichiers :

ex_06/IUnit.php

ex_06/AMonster.php

ex_06/ASpaceMarine.php

ex_06/AWeapon.php

ex_06/PlasmaRifle.php

ex_06/PowerFist.php

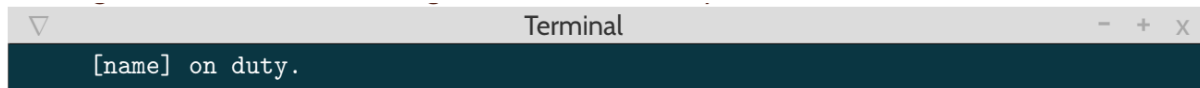
ex_06/TacticalMarine.php

ex_06/AssaultTerminator.php

Copiez vos classes de l'exercice précédent.

Il est enfin temps de créer notre véritable Space Marine. Comme vous l'aurez deviné, tous les SpaceMarines devront hériter de la classe ASpaceMarine.

Commençons par TacticalMarine. Lors de sa création, il devrait dire :

A terminal window with a title bar containing a dropdown arrow, the word "Terminal", and window control buttons (minimize, maximize, close). The terminal area has a dark background and displays the text "[name] on duty." in a light-colored monospace font.

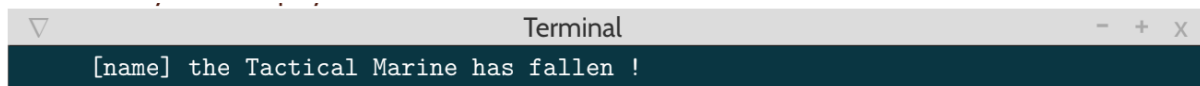
Où [name] doit être remplacé par le nom de notre SpaceMarine.

Son nom sera toujours reçu lors de sa création.

Lors de la création, vous équiperez votre TacticalMarine d'un PlasmaRifle.

Un TacticalMarine aura 100Hp et 20AP par défaut.

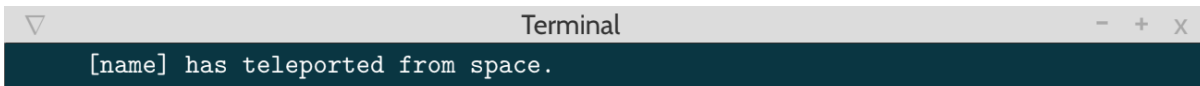
A sa mort, vous afficherez :

A terminal window with a title bar containing a dropdown arrow, the word "Terminal", and window control buttons (minimize, maximize, close). The terminal area has a dark background and displays the text "[name] the Tactical Marine has fallen !" in a light-colored monospace font.

Où [name] doit être remplacé par le nom de notre SpaceMarine.

Un Marine tactique étant... Tactique régénère ses PA plus rapidement que les autres Marines. Il récupérera en fait 12 PA au lieu de 9 lorsque le recoverAP sera appelé.

Parlons maintenant de AssaultTerminator. Lors de sa création, il recevra également son nom et affichera :

A terminal window with a title bar containing a dropdown arrow, the word "Terminal", and window control buttons (minimize, maximize, close). The terminal area has a dark background and displays the text "[name] has teleported from space." in a light-colored monospace font.

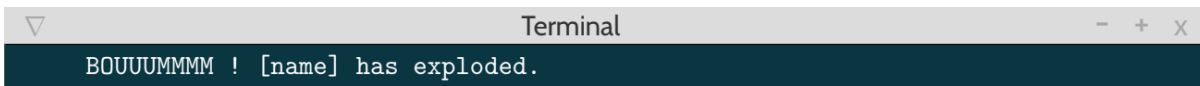
Où [name] doit être remplacé par le nom de notre SpaceMarine.

Par défaut, il possède 150HP et 30AP.

Lors de sa création, vous devez l'équiper d'un PowerFist.

De plus, AssaultTerminator étant un peu plus résistant que les autres Marines, lorsque sa méthode "receiveDamage" sera appelée, elle réduira les dégâts de 3. Cependant, les dégâts reçus ne peuvent pas être réduits en dessous de 1.

Pendant sa destruction, vous afficherez :

A terminal window with a title bar containing a dropdown arrow, the word "Terminal", and window control buttons (minimize, maximize, close). The terminal area has a dark background and displays the text "BOUUUMMMM ! [name] has exploded." in a light-colored monospace font.

Où [name] doit être remplacé par le nom de notre SpaceMarine.

Exercice 7

Fichiers :

ex_07/IUnit.php

ex_07/AMonster.php

ex_07/ASpaceMarine.php

ex_07/AWeapon.php

ex_07/PlasmaRifle.php

ex_07/PowerFist.php

ex_07/TacticalMarine.php

ex_07/AssaultTerminator.php

ex_07/RadScorpion.php

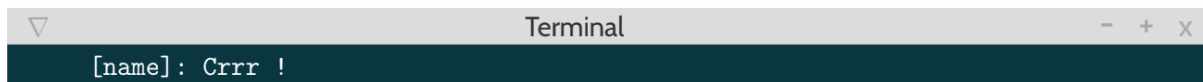
ex_07/SuperMutant.php

Copiez vos classes de l'exercice précédent.

Nous allons enfin créer nos monstres ! Comme vous l'avez deviné, toutes les classes de monstres devront hériter de la classe "AMonster".

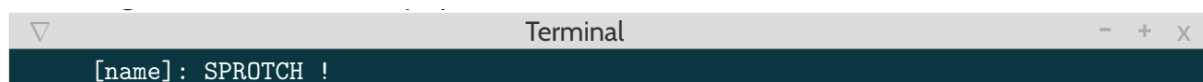
Nos monstres auront un nom générique. Ils seront tous appelés "RadScorpion" ou "SuperMutant" suivi d'un identifiant. Par exemple, le premier RadScorpion existant sera appelé "RadScorpion #1", le second sera "RadScorpion #2". De ce fait, nos Monstres ne prendront aucun paramètre lors de leur création.

Créons d'abord celui de RadScorpion, pendant sa création il s'affichera :

A terminal window titled "Terminal" with a dark background. The text "[name]: Crrr !" is displayed in white.

Où [name] doit être remplacé par le nom du monstre.

Et pendant sa destruction, il affichera :

A terminal window titled "Terminal" with a dark background. The text "[name]: SPROTCH !" is displayed in white.

Où [name] doit être remplacé par le nom du monstre.

RadScorpion est un monstre assez commun, il n'aura que 80HP cependant, il commencera avec le maximum de

AP, 50, chacune de ses attaques lui infligera 25 dégâts et lui coûtera 8 AP.

Cependant, ils peuvent être assez effrayants, c'est pourquoi nous considérerons que s'ils attaquent un marine qui n'est pas un "AssaultTerminator", ils infligeront des dégâts doubles (parce que le Marine sera trop effrayé pour s'éloigner).

Maintenant, nous allons créer notre SuperMutant.


```
Terminal
[name]: Roaarr !
```

Où [name] doit être remplacé par le nom du monstre.

Et pendant sa destruction, il affichera :

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

Où [name] doit être remplacé par le nom du monstre.

Ils commenceront avec 170HP, 20AP. Chacune de ses attaques infligera 60 dégâts mais coûtera 20AP.

Lorsque le SuperMutant récupère de la PA, il récupère également des HP, avec un maximum de 10HP récupérés par appel (170HP étant leur pleine santé).

Exercice 8

Fichiers :

ex_08/IUnit.php

ex_08/AMonster.php

ex_08/ASpaceMarine.php

ex_08/AWeapon.php

ex_08/PlasmaRifle.php

ex_08/PowerFist.php

ex_08/TacticalMarine.php

ex_08/AssaultTerminator.php

ex_08/RadScorpion.php

ex_08/SuperMutant.php

ex_08/SpaceArena.php

Copiez vos classes de l'exercice précédent.

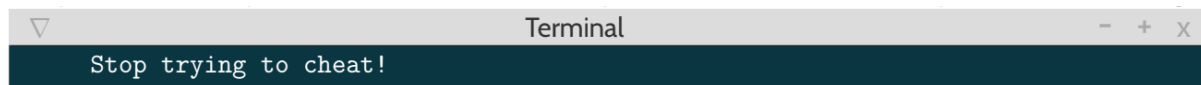
Créez une classe SpaceArena.

Nous utiliserons cette classe pour simuler des combats entre des équipes de Monstres et des équipes de SpaceMarines.

Cette classe aura 3 méthodes.

La première sera "enlistMonsters", elle prendra un tableau de monstres comme paramètre.

Si un élément du paramètre n'est pas un "AMonster", vous devrez lever une exception avec le message :

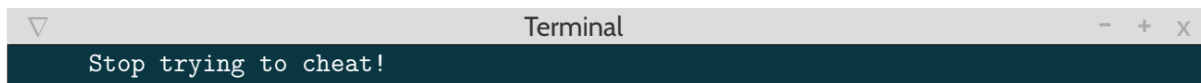
A terminal window with a title bar that says "Terminal" and standard window controls (minimize, maximize, close). The terminal content shows the text "Stop trying to cheat!" in a monospaced font.

Sans nouvelle ligne, car il s'agit d'un message d'exception.

Elle ajoutera les monstres donnés en paramètre à ceux déjà enregistrés pour combattre.

Vous créerez ensuite une méthode "enlistSpaceMarines", qui prendra un tableau de SpaceMarines comme paramètre.

Si un élément du paramètre n'est pas un "ASpaceMarine", vous devrez lever une exception avec le message suivant :

A terminal window with a title bar that says "Terminal" and standard window controls (minimize, maximize, close). The terminal content shows the text "Stop trying to cheat!" in a monospaced font.

Sans nouvelle ligne, car il s'agit d'un message d'exception.

Il ajoutera les SpaceMarines donnés en paramètre à celui qui est déjà inscrit pour combattre.

Il ne devrait pas être possible d'ajouter un guerrier qui est déjà inscrit pour un combat.

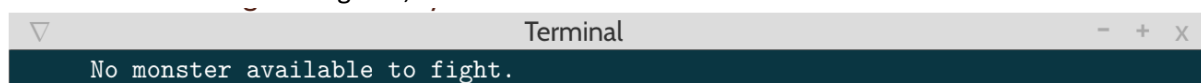
Si une exception a été levée, les guerriers placés avant le tricheur dans le tableau devraient quand même être enrôlés.

(Par exemple, si enlistMonsters reçoit un tableau comme celui-ci [RadScorpion, RadScorpion, somethingElse, SuperMutant], les deux RadScorpion devraient être enrôlés mais pas le SuperMutant).

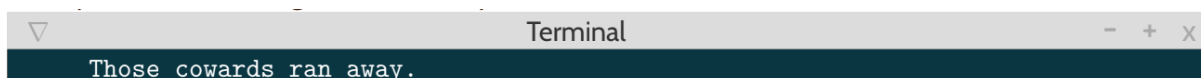
Vous allez maintenant créer la méthode principale, "fight". Elle ne prendra aucun paramètre.

Cette fonction fera combattre les équipes de monstres et de Marines de l'espace enregistrées entre elles.

Si aucun monstre n'est enregistré, il dira :

A terminal window with a title bar that says "Terminal" and standard window controls (minimize, maximize, close). The terminal content shows the text "No monster available to fight." in a monospaced font.

Si aucune SpaceMarine n'est enregistrée, il dira :

A terminal window with a title bar that says "Terminal" and standard window controls (minimize, maximize, close). The terminal content shows the text "Those cowards ran away." in a monospaced font.

Si au moins une des deux équipes est manquante, la fonction s'arrête là en retournant false.

Voici comment doit se dérouler un combat :

- Lorsqu'un nouveau tour entre un monstre et un SpaceMarine commence, le SpaceMarine passe toujours en premier.

- Celui qui joue essaiera d'attaquer, si c'est un succès son tour est terminé. Si ça échoue parce qu'ils n'étaient pas à portée, il se rapprochera. S'il a échoué parce qu'il n'avait pas assez de PA, il utilisera sa méthode "recoverAP" (récupérer PA) une fois.

- Le tour passe ensuite à l'adversaire. Ce processus se répète jusqu'à ce que l'un des adversaires soit tombé. Le vainqueur appelle une fois sa fonction "recoverAP" avant de commencer son prochain combat.

- Si le SpaceMarine a gagné, le deuxième monstre entre dans l'arène, et tout le processus recommence jusqu'à ce que l'une des deux équipes soit vaincue (si le monstre a gagné, le second SpaceMarine entre dans l'arène).

Chaque fois qu'un monstre ou un Marin de l'espace entre dans le combat, vous affichez :

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

Le SpaceMarine sera toujours présenté en premier si les deux combattants entrent en même temps.

A la fin du combat, vous afficherez :

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

Où [team's name] sera remplacé par "monstres" ou "spaceMarines".

Chaque vainqueur restera dans l'arène en attendant le prochain combat.

Example :

```
Terminal
~/C-DEV-111> cat index.php
<?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
~/C-DEV-111> php index.php
RadScorpion #1: Crrr!
SuperMutant #1: Roaarr !
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
```

```
Terminal
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
RadScorpion #2 attacks Abaddon.
Abaddon attacks RadScorpion #2 with a Power Fist.
SBAM
The spaceMarines are victorious.
SuperMutant #2: Roaarr !
SuperMutant #3: Roaarr !
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
```

```
Terminal
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!
```