TP de PHP orienté objet - Fondation.

Un premier exercice de POO

1. Faites une classe `Animal` avec :
   — un attribut privé `nom` ;
   — un attribut `PV` (points de vie), initialisé à 100 par défaut ;
   — des méthodes `manger`, `sePresenter`, `parler`, ... qui affichent l'action faite par votre animal. Manger redonne 5 points de vie (et peut faire un affichage en plus). sePresenter() permet à l'animal de se présenter en donnant son nom et son nombre de points de vie.
2. **(Héritage)** Faites ensuite une classe `Oiseau` et une classe `Reptile` qui héritent de la classe `Animal` et qui redéfinissent les méthodes `sePresenter()` et `parler` (`parler()` affichera « cuicui » pour un oiseau, par exemple). Chacune des classes filles doit avoir un attribut en plus (par exemple pointVenin pour le reptile, hauteur pour l'oiseau qui peut représenter l'altitude moyenne où il évolue).
   Créez un oiseau et un reptile et faites-les se présenter et manger.
3. Inventez une méthode `seBattre()` qui permette aux oiseaux et aux reptiles de se battre.
   Lorsqu'un animal est blessé, ses PV diminuent. Lorsqu'il mange, ses PV augmentent.
   Faites en sorte que les dégâts produits par un reptile soient plus importants que ceux produits par un oiseau. Faites enfin en sorte qu'il y ait une part d'aléatoire dans le combat (utilisez la fonction `rand`).

POO et bataille spatiale

Le mulet a attaqué la fondation et vient de capturer terminus, presque sans combat. Toute sa flotte se retourne maintenant vers les mondes marchands indépendants, dont la flotte est malheureusement considérablement désorganisée. Vous êtes un marchand particulièrement doué en informatique, et c'est pourquoi nous vous avons choisi pour gérer cette flotte et la réorganiser de sorte à ce qu'elle puisse résister en attendant des renforts venus de Trantor, qui ne pourront arriver avant 30 jours. Nous sommes en nette infériorité numérique, nous devons donc compter sur une organisation optimale, autant dire que tous nos espoirs reposent sur vous.

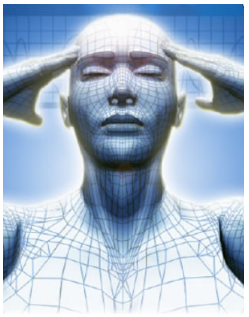
Vous devez gérer à la fois le personnel qui sera affecté à la flotte des vaisseaux et l'ensemble des vaisseaux eux-mêmes. Un membre d'équipage est une **Personne** identifiée de façon unique par son *nom*. Il est également caractérisé par son *age* et son *sexe*. On distingue deux types de personnes :

— les **opérateurs**, qui gèrent la bonne tenue du vaisseau : ils ont un *métier* (pilote, agent d'entretien, etc....), et ils ont une *expérience* dans leur métier, qui est un entier qui représente leur savoir-faire (0 au début, et plus ils travaillent, plus ils gagnent de l'expérience).
— Les **mentalistes** qui sont des membres de la seconde fondation infiltrés chez les marchands indépendants. Ils peuvent influencer les personnes et possèdent une jauge de *mana* (un entier, initialisé à 100 ; ils peuvent influencer les autres personnes, au prix d'une perte de 20 points de mana). Ils obligent ainsi la personne qu'ils influencent à agir.

Quant à la **flotte**, elle a un *nom* dont vous déciderez vous même et est constituée de plusieurs *vaisseaux*. Un **vaisseau** est caractérisé de manière unique par un *nom*, possède un *type* (guerre, transport, etc...) et contient un équipage de 10 personnes au maximum et de deux personnes au minimum. Il possède également une jauge de *vie*, un entier dont la valeur initiale dépend du type du vaisseau, et qui descend quand le vaisseau est touché par un tir ou est abîmé, d'une façon ou d'une autre. Si cet entier arrive à 0, le vaisseau est détruit (il doit être retiré de la flotte).

Vous allez devoir programmer les classes nécessaires à la gestion de la flotte. Avant de programmer une deuxième classe, vérifier que la première fonctionne bien !! Pour cela, vous devez avoir un fichier principal (appelons-le univers.php) qui inclut vos classes, instancie vos objets, et les fait agir.

opératrice (communication)



mentaliste

Première mission - gestion du personnel

Créez une classe **Personne**, dont les opérateurs et les mentalistes hériteront, et munissez-la de méthodes adaptées (un getter qui retourne le nom de la personne, une méthode sePrésenter(), une méthode agir(), qui consiste simplement, pour une personne quelconque, à demander "mais que voulez-vous que je fasse ?"). Une personne a un nom, un prénom, un age, un nombre de points de vie. Testez dans un autre fichier qui va inclure la classe Personne, en créer une instance, et la faire agir.

Vous devez ensuite créer les classes **Mentaliste** et **Operateur**. Les attributs de classe de ces deux classes sont définis par l'énoncé plus haut (métier pour les opérateurs, mana pour les mentalistes, expérience pour les opérateurs, etc..). Il faut dans chacune de ces deux classes des méthodes `sePresenter()` qui retournent une chaîne de caractère présentant le mentaliste ou l'opérateur ("je suis un mentaliste homme de 20 ans, il me reste 40 de mana" par exemple). Définissez également une méthode `vieillir()` qui ajoute un an à l'opérateur ou au mentaliste considéré. Définissez la méthode *agir*, qui sera différente pour un mentaliste et pour un opérateur : pour un opérateur, on se contentera pour l'instant de vérifier à quel type d'opérateur nous avons à faire (un pilote, un agent d'entretien, etc...) et d'afficher à l'écran une phrase disant que le vaisseau est en train de décoller, que le vaisseau est nettoyé, etc... Ils gagnent 5 points d'expérience quand c'est réalisé. Évidemment, à terme, il faudra vérifier que le vaisseau est effectivement paré à décoller avant d'agir ! Pour un mentaliste, `agir()` consiste à influencer une personne : on force cette personne à agir elle-même. La méthode `agir` d'un mentaliste prend donc en paramètre la personne à influencer et la fait agir. Cela ne peut se produire que si le mentaliste a assez de mana. Si c'est le cas, il perd 20 points de mana en agissant.

Mission 2 : la gestion des vaisseaux

Créez la classe **Vaisseau** et la classe **Flotte**. Un vaisseau possède un certain nombre de Personnes (10 au plus), l'attribut *equipage* sera donc un tableau. Un vaisseau est également caractérisé par son *nom* unique, par sa *propreté* (propre ou sale, lors de sa création il est propre) et par son *état* (avarié ou en état de marche). Parmi les méthodes que vous pourrez imaginer pour la classe vaisseau voici quelques propositions : une méthode (un "getteur") permettant de donner le nom du vaisseau. Une méthode (un "setteur") permettant d'attribuer un nouveau nom au vaisseau. Une méthode qui dit si le vaisseau est paré à décoller : vérifie qu'il y a entre 2 et 10 membres à bord, que le vaisseau est propre, et en état de marche. Une méthode qui remet le vaisseau en état de décoller (mais il faut vérifier qu'il y a bien à bord un opérateur spécialisé en entretien si le vaisseau est sale, ou un agent de maintenance si le vaisseau est avarié). Cette méthode peut renvoyer 0 en cas d'échec, 1 en cas de succès. Une méthode *décoller()* qui vérifie que le vaisseau est paré à décoller, qui le remet en état de marche si ce n'est pas le cas, et enfin qui vérifie qu'il y a bien un pilote à bord. Si tout cela a pu être fait, on renvoie 1, sinon on renvoie 0 (le décollage n'a pas eu lieu).
Deux autres méthodes permettent également d'affecter ou de retirer un membre de l'équipage d'un vaisseau. Pour affecter une personne, il faut vérifier que son nom ne figure pas déjà dans l'équipage et que l'équipage ne contienne pas déjà 10 personnes.

Quant à la flotte, elle est caractérisée par son nom, et par un ensemble de vaisseaux. Parmi les méthodes à implémenter, il faut pouvoir ajouter et retirer un vaisseau de la flotte.

Mission 3 : mise en place d'une stratégie collective

En se basant sur ce que vous avez codé et appris, individuellement, pendant la première séance, vous devez créer un jeu (tour par tour) qui s'appuie sur les concepts et objets développés. Le jeu ne s'appuiera pas sur des graphismes évolués, il peut être avoir lieu sur une carte fixe (dont la configuration change à chaque tour bien sûr), voire être purement textuel (interactions avec l'utilisateur via des formulaires). Il faudra inventer un système d'authentification utilisant des sessions. Vous pouvez réaliser ce projet tout seul ou à deux, si vous choisissez d'être deux, votre jeu doit être multijoueurs (on doit pouvoir jouer au moins à deux, une base de données est partagée entre les joueurs). Vous serez évalué sur l'originalité du jeu (sur 5), l'ambition/maîtrise technique (sur 10), le rapport (sur 5). Ce projet est à rendre pour le 3 novembre 2024. Bon courage, et n'oubliez pas qu'il en va de l'avenir de l'univers !