Exercice 3.1

Pour cette série d'exercices et les suivants, je vous demanderai de commenter au maximum votre code, en utilisant également les commentaires Docstring.

Exercice 3.2

Faire en sorte que l'affichage des informations d'un navire, d'une coque, et d'une arme se fasse de manière implicite lorsque l'on *print* l'objet correspondant.

Exercice 3.3

Faire en sorte de ne pas pouvoir modifier (ni supprimer) le nom d'un navire depuis une autre classe.

Exercice 3.4

Nous allons distinguer deux types de navires : les navires de guerre et les navires civils.

Les navires de guerre sont des navires normaux auxquels on a ajouté une arme. Seuls les navires de guerre peuvent tirer sur d'autres navires.

Créer une classe *NavireDeGuerre* qui hérite de *Navire*.

Exercice 3.5

Créer une classe *NavireCivil*, qui hérite de *Navire*.

Un navire civil est principalement destiné au tourisme, il possède un nombre de passagers, une capacité de passagers maximale, et une liste de destinations (facultatif).

Un navire civil ne peut bien entendu pas tirer. Il peut en revanche déposer des passagers, et en récupérer des nouveaux, dans la limite de sa capacité maximale.

Exercice 3.6

Créer une classe *Hopital.*

Un hôpital possède un nom, et une note, qui détermine sa capacité à soigner. La note est un nombre décimal qui varie entre 0 et 1.

Le nom d'un hôpital ne peut pas être modifié.

Un hôpital possède une méthode `soigner()`. Elle prend une valeur correspondant à des points de vie en paramètre, et additionne à cette valeur 50 multiplié par la note de l'hôpital et retourne le résultat de ce calcul.

Par exemple, si la note est 0.3, on additionne 50 x 0.3 = 15 à la valeur passée en paramètre.

Exercice 3.7

Un navire médical est à la fois un navire, et à la fois un hôpital.

Créer une classe *NavireMedical*.

Exercice 3.8

Les navires médicaux peuvent soigner des navires, en augmentant leurs points de vie.

Redéfinir la méthode `soigner()`, pour prendre non plus des points de vie en paramètre, mais le navire à soigner lui-même.

Un message doit être affiché, par exemple :

« *N1* a apporté une aide médicale à *N2*.
N2 récupère *xx* points de vie (*xx*/100). »

Dans le *main* de l'application, reprendre le scénario, et ajouter des navires de guerre, des navires civils, des navires médicaux, et faire jouer quelques méthodes de chacune des classes.

Exercice 3.9

Créer des classes d'exception, correspondantes aux différentes exceptions susceptibles d'être levées dans le programme.

Par exemple, lorsque l'on tente de créer un navire avec une puissance supérieure à 100, ou bien encore lorsqu'un navire détruit tente de se déplacer ou tirer.

Exercice 3.10 – Bonus

Regrouper les classes *Navire* (et classes dérivées) dans un fichier, les classes *Coque* et *Arme* dans un autre fichier, de même que pour la classe *Hopital*, et pour les classes d'exceptions. Faire un *main* séparé de autres classes.

Exercice 3.11 – Bonus

Empêcher un navire de guerre de tirer sur lui-même (il ne perd pas de point de puissance de tir dans ce cas-là).

Exercice 3.12 – Bonus

Ajouter une exception qui empêche l'instanciation de la classe *Navire*.

Exercice 3.13 – Bonus

Lorsqu'un navire de guerre tire sur un navire civil ou navire médical, le navire de guerre perd 10 points de dégâts et 10 points de puissance de tir.

Exercice 3.14 – Bonus

Ajouter un paramètre de taille variable nommé dans le constructeur pour prendre en compte les attributs non importants (comme fabriquant, année de construction, puissance moteur, etc.).