



Programmation Orientée Objet C#

Cas Navire

Objectifs du TP :

- ✓ **Améliorer le TP précédent en le professionnalisant et en introduisant de nouveaux concepts**
- ✓ **Gérer les exceptions**
- ✓ S'initier à la programmation objet en Java
- ✓ Utiliser les bonnes pratiques pour le débogage
- ✓ Lire une documentation et savoir l'appliquer
- ✓ Réviser les concepts algorithmiques
- ✓ Se forcer à se poser des questions et en obtenir des réponses

Ce TP n'a pas vocation à être un TP professionnel. Il est seulement péda

Prérequis :

- ✓ **Avoir fini le TP précédent**
- ✓ Bases algorithmiques
- ✓ Bases du langage Java
- ✓ Bases théoriques de la POO
- ✓ Principales fonctionnalités de l'IDE Eclipse

Dans ce TP, vous serez tour à tour :

- ✓ Concepteur de classe :



- ✓ Utilisateur de classe :



Partie 1 : MISE EN PLACE DU PROJET



Il vous est demandé de créer un nouveau projet C# pour éviter les effets de bord dans la mise au point.

Ce projet s'appellera TP2Navire

Chaque classe sera implémentée dans un espace de noms.

Je vous propose cette structure :

Navire.Classesmetier : pour les classes métier : Port et Navire

Navire.Application : pour la classe Programme

Partie 2 : AMELIORATION DU CODE DE LA CLASSE NAVIRE



Revoir le cours sur les tableaux et les collections

Chaque Navire a un identifiant naturel, le numéro imo.

On utilisera donc cet attribut de la classe Navire pour en faire une clé de recherche.



Travail à faire

Proposez une autre classe que la classe List pour stocker les navires présents dans le port.

Répondez aux questions ci-dessous avant de commencer à coder.

1. Quelles méthodes de la classe Port allez vous conserver, éventuellement en les modifiant ?

Méthode	Garder	Supprimer
Port(String nom)		
enregistrerArrivee(Navire navire)		
enregistrerDepart(String imo)		
recupPosition(String imo)		
recupPosition(Navire navire)		
isPresent(Navire navire)		
testerRecupPosition()		
testerRecupPositionV2()		

2. Comment peut-on s'assurer qu'un même navire ne peut être enregistré plusieurs fois dans le port ?

Proposez au moins deux solutions. Il ne vous est pas demandé ici de faire un choix.



Travail à faire

Modifiez la classe Port en conséquence et notamment l'attribut navire et le const
Rajouter un accesseur sur l'attribut navires de la classe Port

Vous effectuerez les tests plus tard.

Partie 3 : QUELQUES AMELIORATIONS

On voudrait fiabiliser l'application et effectuer certains contrôles.

Si un contrôle n'est pas vérifié, le programme doit déclencher une exception.

Faire en sorte :

- ✓ Que la quantité maximale de fret soit toujours supérieure ou égale à 0
- ✓ Que le numéro imo respecte le modèle "IMO9999999" : les 3 lettres IMO suivies de 7 chiffres.

1. Quantité maximale de fret supérieure ou égale à 0.

Ce contrôle doit se faire à l'instanciation d'un objet (constructeur) et lors de la modification de la modification de la quantité maximale de fret pour un navire (setter).



L'idée est de ne pas écrire plusieurs fois la même séquence de code afin de faciliter la maintenance au cas où cette contrainte viendrait à être modifiée

Je vous propose donc d'écrire la règle dans le setter de la donnée membre et d'appeler ce setter depuis le constructeur :

```
public int QteFretMaxi
{
    get => qteFretMaxi;
    set
    {
        if (value >= 0)
        {
            this.qteFretMaxi = value;
        }
        else
        {
            throw new Exception("Erreur, quantité de fret non valide");
        }
    }
}
```

Et pour le constructeur :

```
public Navire(string imo, string nom, string libelleFret, int qteFretMaxi)
{
    this.imo = imo;
    this.nom = nom;
    this.libelleFret = libelleFret;
    this.QteFretMaxi = qteFretMaxi;
}
```

Cette exception ne doit pas mettre fin au programme, elle doit être interceptée et gérée :

- ✓ Dans les méthodes faisant appel au constructeur de la classe Navire
- ✓ Ou aux méthodes appelant la méthode setQteFretMaxi(int qteFretMaxi).



Travail à faire

Vous écrirez la méthode Instanciations() dans la classe Programme

Vous ferez appel à cette méthode depuis la méthode main de la classe Programme

Vous devriez obtenir ceci :

```
Erreur, quantité de fret non valide
Fin normale du programme
```

Avec ce code :

```
private static void Instanciations()
{
    try
    {
        Navire navire = new Navire("IM09427639", "Copper Spirit", "Hydrocarbures", 156827);
        navire = new Navire("IM09839272", "MSC Isabella", "Porte-conteneurs", -197500);
        navire = new Navire("IM08715871", "MSC PILAR", "Porte-conteneurs", 67183);
        navire = new Navire("IM09235232", "FORTUNE TRADER", "Cargo", 74750);
        navire = new Navire("IM09574004", "TRITON SEAHAWK", "Hydrocarbures", 51201);
    }
    catch (Exception ex)
    {
        Console.WriteLine(ex.Message);
    }
}

static void Main()
{
    try
    {
        Instanciations();
        Console.WriteLine("Fin normale du programme");
    }
    catch (Exception ex)
    {
        Console.WriteLine(ex.Message);
    }
    finally { Console.ReadKey(); }
}
```



Travail à faire

Mettre un point d'arrêt sur l'appel de la méthode instanciations et faites **une exécution pas à pas détaillé** jusqu'à ce que vous repérez quelle ligne de code déclenche l'exception et l'endroit où elle est traitée. Poursuivez l'exécution pas à pas

2. Le numéro IMO

Le contrôle est ici moins évident à faire. Vous allez découvrir les **expressions régulières**.



C'est un outil qui fait partie de la boîte à outils de tout bon développeur.



Une expression régulière définit un modèle de recherche de chaînes. L'abréviation pour l'expression régulière est regex. Tous les langages modernes intègrent les regex. onnée.



Les expressions régulières peuvent être utilisées pour rechercher, modifier, manipuler ou contrôler la conformité d'une chaîne de caractères à un modèle (pattern).



En C#, on utilise La classe Regex : <https://bre.is/6a84aSrV>



Il n'est pas toujours facile de mettre au point une expression régulière... essayez pour un mail et vous comprendrez vite! Donc soit vous utilisez ce qui existe sur internet, comme la regex de contrôle d'un mail ou d'un numéro de téléphone, ou vous pouvez mettre au point votre regex en utilisant un site en ligne.

Exemples de sites :

<https://regex101.com/>

<https://www.debuggex.com/>

<https://www.freeformatter.com/java-regex-tester.html#ad-output>



Travail à faire

Modifiez la classe Navire afin de faire en sorte que la donnée membre imo soit valide. Vous pourrez tester votre programme en modifiant la méthode instanciations()

```
private static void Instanciations()
{
    try
    {
        Navire navire = new Navire("IMO9427639", "Copper Spirit", "Hydrocarbures", 156827);
        navire = new Navire("IMO9839272", "MSC Isabella", "Porte-conteneurs", 197500);
        navire = new Navire("IMO8715871", "MSC PILAR", "Porte-conteneurs", 67183);
        navire = new Navire("XMT9235232", "FORTUNE TRADER", "Cargo", 74750);
        navire = new Navire("IMO9574004", "TRITON SEAHAWK", "Hydrocarbures", 51201);
    }
    catch (Exception ex)
    {
        Console.WriteLine(ex.Message);
    }
}
```

Vous devriez obtenir ceci à l'exécution :

```
Erreur : IMO non valide
Fin normale du programme
```

Partie 4 : SURCHARGE DU CONSTRUCTEUR

Il est possible maintenant d'instancier un objet de la classe Navire avec seulement son numéro IMO et son nom.

Dans ce cas :

- ✓ Le type de fret est initialisé à "Indéfini"
- ✓ La quantité maximale de fret est initialisée à 0



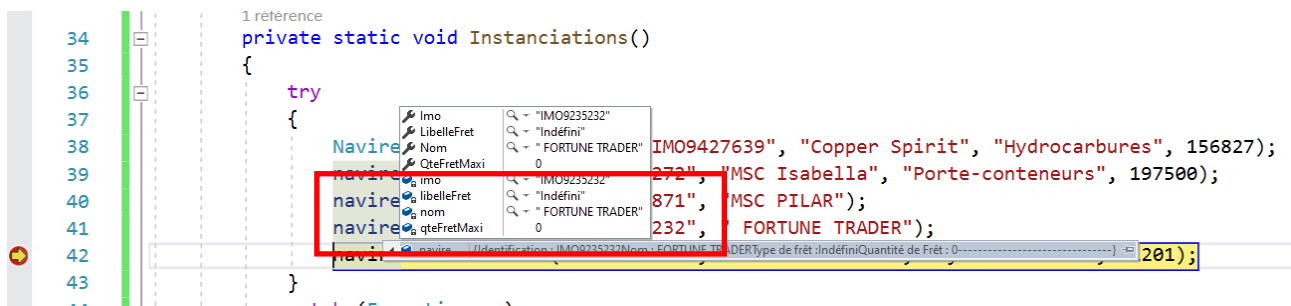
Travail à faire

Ecrivez le nouveau constructeur surchargé de la classe Navire, vous n'avez pas le droit de dupliquer les lignes de code. Vous pourrez utiliser l'opérateur this

Vous pourrez modifier la méthode instanciations() ci-dessous pour mettre en œuvre la surcharge

```
private static void Instanciations()
{
    try
    {
        Navire navire = new Navire("IM09427639", "Copper Spirit", "Hydrocarbures", 156827);
        navire = new Navire("IM09839272", "MSC Isabella", "Porte-conteneurs", 197500);
        navire = new Navire("IM08715871", "MSC PILAR");
        navire = new Navire("IM09235232", "FORTUNE TRADER");
        navire = new Navire("IM09574004", "TRITON SEAHAWK", "Hydrocarbures", 51201);
    }
    catch (Exception ex)
    {
        Console.WriteLine(ex.Message);
    }
}
```

Vous mettrez un point d'arrêt sur la dernière instanciation et vous vérifierez que l'instanciation a bien eu lieu et que le constructeur a bien initialisé toutes les données membres de la classe.



Partie 5 : GESTION DES EXCEPTIONS

Avant d'aller plus loin dans l'écriture des fonctionnalités de l'application.

Vous allez créer une classe GestionPortException dédiée à l'application. Elle héritera de la classe Exception. Elle ne fera pas grand-chose de plus que la classe Exception, mais vous écrirez votre première classe héritée.

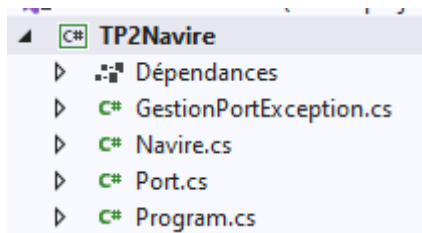
Cette classe sera dans l'espace de noms GestionNavire.Exceptions



Travail à faire

Créez cette classe graphiquement dans le bon espace de noms
Complétez le code comme indiqué ci-dessous

Votre classe devrait apparaître dans l'explorateur de projet :



```
namespace GestionNavire.Exceptions
{
    2 références
    class GestionPortException : Exception
    {
        1 référence
        public GestionPortException(string message)
            : base("Erreur de : " + System.Environment.UserName + " le " + DateTime.Now.ToLocalTime() +
                "\n" + message) {}
    }
}
```



Travail à faire

Expliquez ce que va faire la ligne encadrée.

Vous pouvez tester avec la méthode Instanciations et un mauvais numéro IMO.



Désormais votre application ne déclenchera plus que des exceptions de ce type avec l'instruction throw.

Partie 6 : AJOUT D'UN NAVIRE DANS LE PORT



Il faut s'assurer, avant d'enregistrer un navire, qu'il n'est pas présent dans le port (test sur numéro IMO).

Dans la documentation de la classe Dictionary, vous avez dû lire qu'elle n'acceptait pas les clés dupliquées.



Travail à faire

Dans ce qui suit, vous allez créer :

- ✓ la donnée membre port dans la classe Programme
- ✓ la méthode enregistrerArrivee(Navire navire) dans la classe Port
- ✓ la méthode testerEnregistrerArrivee () dans la classe Programme

Vous allez créer la donnée membre port dans la classe Programme et vous l'instancierez dans la méthode main()

```
class Program
{
    private static Port port;
    0 références
    static void Main()
    {
        try
        {
            port = new Port("Toulon");
            //Instanciations();
            Console.WriteLine("Fin normale du programme");
        }
        catch (Exception ex)
        {
            Console.WriteLine(ex.Message);
        }
        finally {Console.ReadKey(); }
    }
}
```

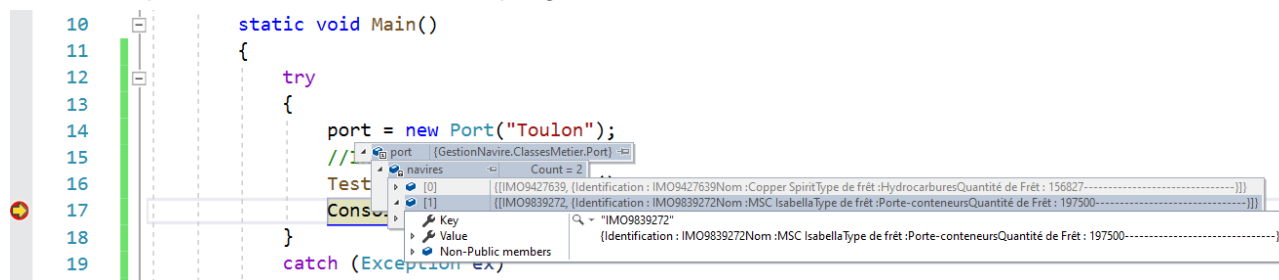
Puis vous allez créer la méthode **enregistrerArrivee(Navire navire)** dans la classe Port. Son rôle est uniquement d'ajouter le navire passé en paramètre quand les différents contrôles sont vérifiés.

```
public void EnregistrerArrivee(Navire navire)
{
    if (this.navires.Count < this.nbNaviresMax)
    {
        this.navires.Add(navire.Imo, navire);
    }
    else
    {
        throw new GestionPortException("Ajout impossible, le port est complet");
    }
}
```

Puis la méthode `TesterEnregistrerArrivee()` de la classe Programme.


```
private static void TesterEnregistrerArrivee()
{
    try
    {
        Navire navire = new Navire("IMO9427639", "Copper Spirit", "Hydrocarbures", 156827);
        port.EnregistrerArrivee(navire);
        navire = new Navire("IMO9839272", "MSC Isabella", "Porte-conteneurs", 197500);
        port.EnregistrerArrivee(navire);
    }
    catch (GestionPortException ex)
    {
        Console.WriteLine(ex.Message);
    }
}
```

Mettez un point d'arrêt avant la fin du programme et observez l'attribut navires de la classe Port :



Vous pouvez aussi rajouter un espion dans la fenêtre d'espions pour observer le contenu de cet attribut :

Espion 1		
Rechercher (Ctrl+E) Profondeur de recherche : 3		
Nom	Valeur	Type
port	{GestionNavire.ClassesMetier.Port}	GestionNavi
navires	Count = 2	System.Coll
[0]	{[IMO9427639, {Identification : IMO9427639Nom :Copper SpiritType de frêt :HydrocarburesQuantité de Frêt : 156827-----}]}	System.Coll
Key	"IMO9427639"	string
Value	{Identification : IMO9427639Nom :Copper SpiritType de frêt :HydrocarburesQuantit...	GestionNavi
Non-Public members		
[1]	{[IMO9839272, {Identification : IMO9839272Nom :MSC IsabellaType de frêt :Porte-co...}]}	System.Coll
Key	"IMO9839272"	string
Value	{Identification : IMO9839272Nom :MSC IsabellaType de frêt :Porte-conteneursQuant...	GestionNavi
Non-Public members		
Raw View		
nbNaviresMax	5	int
nom	"Toulon"	string
Ajouter un élément à espionner		



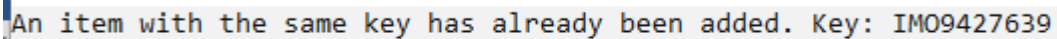
Travail à faire

Dans ce qui suit, vous allez créer :

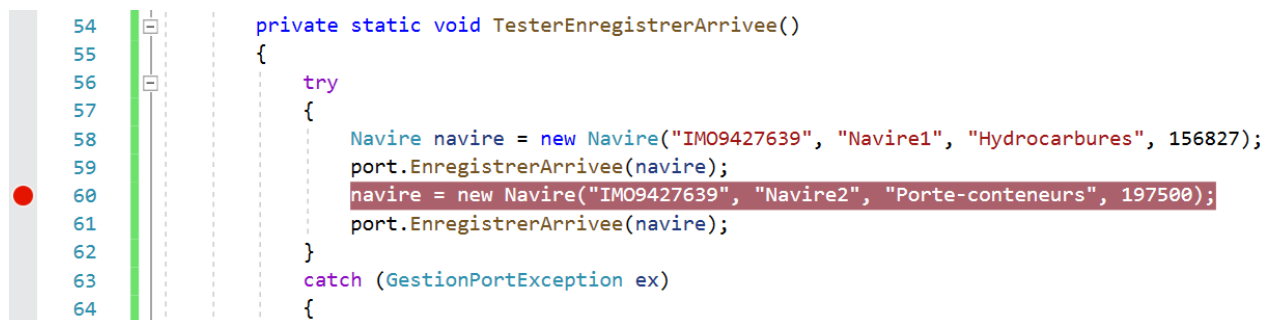
- ✓ Vous allez rajouter 2 navires ayant le même numéro IMO et vous constaterez ce qu'il s'est passé.

```
private static void TesterEnregistrerArrivee()
{
    try
    {
        Navire navire = new Navire("IM09427639", "Navire1", "Hydrocarbures", 156827);
        port.EnregistrerArrivee(navire);
        navire = new Navire("IM09427639", "Navire2", "Porte-conteneurs", 197500);
        port.EnregistrerArrivee(navire);
    }
    catch (GestionPortException ex)
    {
    }
}
```

Vous devriez obtenir ce message :



Mettez un point d'arrêt ici :



```
54 private static void TesterEnregistrerArrivee()
55 {
56     try
57     {
58         Navire navire = new Navire("IM09427639", "Navire1", "Hydrocarbures", 156827);
59         port.EnregistrerArrivee(navire);
60         navire = new Navire("IM09427639", "Navire2", "Porte-conteneurs", 197500);
61         port.EnregistrerArrivee(navire);
62     }
63     catch (GestionPortException ex)
64     {
65     }
66 }
```



Faites une exécution pas à pas du programme vous remarquerez que l'exception a été interceptée au niveau de la méthode Main().



Pourquoi l'exception n'a-t-elle pas été interceptée par la méthode TesterEnregistrerArrivee() ?

Grâce à la documentation, on peut voir quel type d'exception peut être déclenché par la méthode Add: <https://bre.is/kJvfTzk4>

Exceptions

ArgumentNullException

key a la valeur null.

ArgumentException

Un élément possédant la même clé existe déjà dans [Dictionary<TKey,TValue>](#).



En fait cette exception interne a été déclenchée dans la méthode *EnregistrerArrivee* de la classe *Port*. C'est là qu'il aurait fallu l'intercepter et la gérer en propageant une exception de type *GestionPortException* qui sera interceptée dans la méthode *TesterEnregistrerArrivee*.

```
public void EnregistrerArrivee(Navire navire)
{
    try
    {
        if (this.navires.Count < this.nbNaviresMax)
        {
            this.navires.Add(navire.Imo, navire);
        }
        else
        {
            throw new GestionPortException("Ajout impossible, le port est complet");
        }
    }
    catch (ArgumentException)
    {
        throw new GestionPortException("Le navire " + navire.Imo + " est déjà enregistré");
    }
}

private static void TesterEnregistrerArrivee()
{
    Navire navire = null;
    try
    {
        navire = new Navire("IM09427639", "Copper Spirit", "Hydrocarbures", 156827, 12000);
        port.EnregistrerArrivee(navire);
        navire = new Navire("IM09427639", "Copper Spirit", "Hydrocarbures", 156827, 12000);
        port.EnregistrerArrivee(navire);
    }
    catch (GestionPortException ex)
    {
        Console.WriteLine(ex.Message);
    }
}
```



Travail à faire

Expliquez pourquoi a-t-on a modifié la déclaration de la variable *navire* de type *Navire* ?

A l'exécution, vous devriez obtenir ceci :

```
Erreur de : Benoît le 25/01/2020 00:14:05
Le navire IM09427639 est déjà enregistré
```



Travail à faire



Dans ce qui suit, vous allez créer la méthode `testerEnregistrerArriveeV2()` dans la classe `Programme` et qui va essayer de rajouter des navires à la classe `Port` au-delà de la capacité du port.

Vous allez aussi créer des try/catch imbriqués dans le `Main` afin de ne pas interrompre l'exécution du programme en cas d'exception

```
private static void TesterEnregistrerArriveeV2()
{
    Navire navire = null;
    try
    {
        port.EnregistrerArrivee(new Navire("IMO9839272", "MSC Isabella", "Porte-conteneurs", 197500));
        port.EnregistrerArrivee(new Navire("IMO8715871", "MSC PILAR"));
        port.EnregistrerArrivee(new Navire("IMO9235232", "FORTUNE TRADER", "Cargo", 74750));
        port.EnregistrerArrivee(new Navire("IMO9405423", "SERENEA", "Tanker", 158583));
        port.EnregistrerArrivee(new Navire("IMO9574004", "TRITON SEAHAWK", "Hydrocarbures", 51201));
        port.EnregistrerArrivee(new Navire("IMO9748681", "NORDIC SPACE", "Tanker", 157587));
    }
    catch (GestionPortException ex)
    {
        Console.WriteLine(ex.Message);
    }
    catch (ArgumentException)
    {
        throw new GestionPortException("Le navire " + navire.Imo + " est déjà enregistré");
    }
}
```

Méthode `Main` :

```
static void Main()
{
    try
    {
        port = new Port("Toulon");
        //Instanciations();
        try {TesterEnregistrerArrivee(); }
        catch(GestionPortException ex)
        { Console.WriteLine(ex.Message); }

        try { TesterEnregistrerArriveeV2(); }
        catch (GestionPortException ex)
        { Console.WriteLine(ex.Message); }
        Console.WriteLine("Fin normale du programme");
    }
    catch (Exception ex)
    {
        Console.WriteLine(ex.Message);
    }
    finally { Console.ReadKey(); }
}
```



Exécutez, vous devriez obtenir ceci :

```
Erreur de : Benoît le 25/01/2020 07:43:04
Le navire IM09427639 est déjà enregistré
Erreur de : Benoît le 25/01/2020 07:43:04
Ajout impossible, le port est complet
Fin normale du programme
```

Vous remarquez que l'exception levée dans la procédure testerEnregistrerArrivee() n'a pas mis fin à l'exécution du programme.



Travail à faire

Faites une exécution pas à pas afin de voir comment le programme s'est déroulé.

Mettez un point d'arrêt sur la dernière instruction du programme et constatez les navires dans le port, il ne devrait y en avoir 5 et pas de doublons.

```
Console.WriteLine("Fin normale du programme");
```

Vous devriez obtenir ceci :

Espion 1		
Rechercher (Ctrl+E)		
Profondeur de recherche: 3		
Nom	Valeur	Type
port	{GestionNavire.ClassesMetier.Port}	GestionNavire.ClassesMet...
navires	Count = 5	System.Collections.Generi...
[0]	{[IMO9427639, {Identification : IMO9427639Nom :Copper SpiritType de frêt :Hydroc...	System.Collections.Generi...
[1]	{[IMO9839272, {Identification : IMO9839272Nom :MSC IsabellaType de frêt :Porte-co...	System.Collections.Generi...
[2]	{[IMO8715871, {Identification : IMO8715871Nom :MSC PILARType de frêt :IndéfiniQ...	System.Collections.Generi...
[3]	{[IMO9235232, {Identification : IMO9235232Nom :FORTUNE TRADERType de frêt :Car...	System.Collections.Generi...
[4]	{[IMO9405423, {Identification : IMO9405423Nom :SERENEAType de frêt :TankerQuan...	System.Collections.Generi...
Raw View		
nbNaviresMax	5	int
nom	"Toulon"	string

Partie 7 : DEPART D'UN NAVIRE DU PORT

Méthode **EnregistrerDepart(String imo)** de la classe Port

Peu de changements ici, on retire le navire dont le numéro IMO est égal au numéro IMO passé en paramètre.

Elle doit au préalable vérifier si le navire est bien présent dans la collection navires de la classe Port et déclencher une exception de type GestionPortException si on tente d'enregistrer le départ d'un navire qui n'est pas dans la collection navires

Cette méthode n'est plus surchargée

Vous testerez l'enregistrement du départ d'un navire avec la méthode testerEnregistrerDepart() de la classe Programme :

**Travail à faire**

Ecrire la méthode enregistrerDepart(String imo) de la classe Port.

Vous aurez à chercher la méthode de la classe Dictionary qui va supprimer le navire.

Vous écrirez ensuite la méthode testerEnregistrerDepart() de la classe Programme

```
public void EnregistrerDepart(String imo)
{
    this.navires.Remove(imo);
}

public static void TesterEnregistrerDepart()
{
    try
    {
        port.EnregistrerDepart("IM09427639");
        port.EnregistrerDepart("IM09405423");
        port.EnregistrerDepart("IM01111111");
    }
    catch (GestionPortException ex)
    {
        Console.WriteLine(ex.Message);
    }
}
```

Extrait de la méthode main :

```
try { TesterEnregistrerArriveeV2(); }
catch (GestionPortException ex)
{ Console.WriteLine(ex.Message); }

try { TesterEnregistrerDepart(); }
catch (GestionPortException ex)
{ Console.WriteLine(ex.Message); }
```

Vous devriez obtenir ce résultat à l'exécution :

```
Erreur de : Benoît le 25/01/2020 07:58:54
Le navire IM09427639 est déjà enregistré
Erreur de : Benoît le 25/01/2020 07:58:54
Ajout impossible, le port est complet
Fin normale du programme
```

Et si vous exécutez le programme en mode debug avec le point d'arrêt ici :



28

```
Console.WriteLine("Fin normale du programme");
```

Vous constatez que les 2 navires ont été supprimés.



Espion 1		
Rechercher (Ctrl+E) ← → Profondeur de recherche : 3 A		
Nom	Valeur	
port	{GestionNavire.ClassesMetier.Port}	T
navires	Count = 3	S
[0]	{{IMO9839272, {Identification : IMO9839272Nom :MSC IsabellaType de frêt :Porte-co...	S
[1]	{{IMO8715871, {Identification : IMO8715871Nom :MSC PILARType de frêt :IndéfiniQ...	S
[2]	{{IMO9235232, {Identification : IMO9235232Nom :FORTUNE TRADERType de frêt :Car...	S
Raw View		
nbNaviresMax	5	ir



Vous constatez aussi que la tentative de suppression d'un navire qui n'est pas dans le port n'a pas déclenché d'exception.



Travail à faire

Ré-écrire la méthode enregistrerDepart(String imo) de la classe Port.

Cette méthode doit déclencher une exception GestionPortException si on tente de supprimer un Navire non présent dans le port.

Vous utiliserez les méthodes de la classe Dictionary pour effectuer ce contrôle.

Vous devriez maintenant obtenir le résultat suivant :

```
Erreur de : Benoît le 25/01/2020 08:08:24
Le navire IMO9427639 est déjà enregistré
Erreur de : Benoît le 25/01/2020 08:08:24
Ajout impossible, le port est complet
Erreur de : Benoît le 25/01/2020 08:08:24
impossible d'enregistrer le départ du navire IMO1111111, il n'est pas dans le port
Fin normale du programme
```

Partie 8 : AJOUT DE FONCTIONNALITE : DECHARGEMENT DES NAVIRES

Chaque port est équipé de zones de stockage pour décharger les navires ayant pour libelleFret "Porte-conteneurs"

Une zone de stockage est susceptible de stocker tous types de conteneurs d'un navire porte-conteneurs.

Elle est caractérisée par

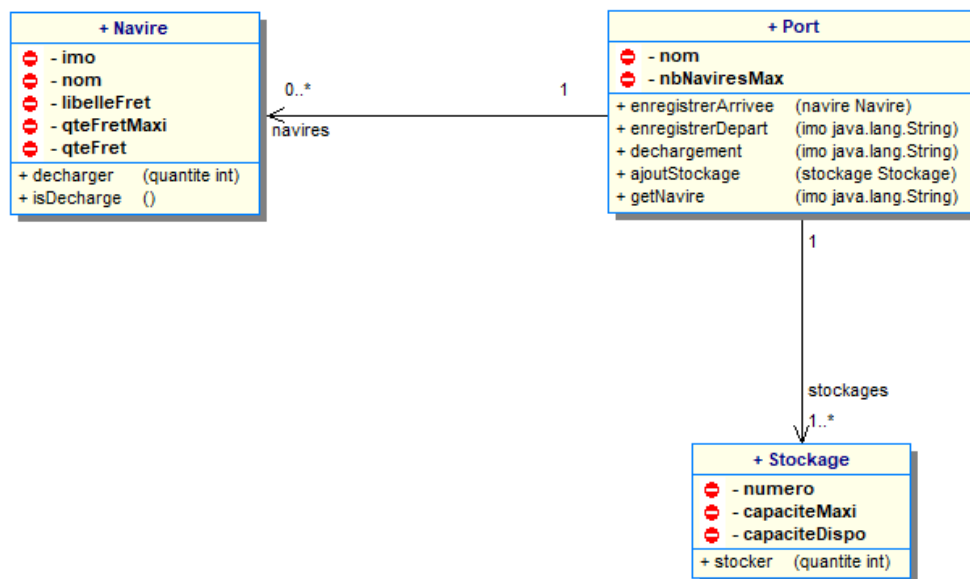
- ✓ une capacité de stockage maximum exprimée en tonnes
- ✓ une capacité de stockage disponible exprimée aussi en tonnes.

Lorsque sa capacité de stockage est égale à zéro, cela signifie que la zone de stockage est remplie et ne peut donc plus accueillir de marchandise.

Lors d'une opération de déchargement, le navire porte-conteneurs parcourt les zones de stockage séquentiellement sur le quai. Le navire décharge autant de fret que la zone peut en recevoir puis il continue jusqu'à ce qu'il soit entièrement déchargé.

Il s'arrête de parcourir les zones de stockage quand il est entièrement déchargé.

our prendre en compte cette nouvelle fonctionnalité, on a modifié le diagramme de classes :



Classe Navire :

- ✓ Ajout de l'attribut **qteFret** : quantité de Fret à bord du navire exprimé en tonnes, accessible en lecture seule. Il doit être renseigné au moment de l'instanciation du Navire. Cette quantité ne peut être inférieure à 0 et supérieure à la quantité de fret maximum du navire. Message de l'exception dans ces cas là : *"Valeur incohérente pour la quantité de fret stockée dans le navire"*
- ✓ Ajout de la méthode **Decharger(int quantite)** : va retirer la quantité passée en paramètre de la quantité de fret du Navire.
La quantité passée en paramètre doit être positive. Si ce n'est pas le cas, déclenchement d'une exception de la classe GestionPortException avec le message "la quantité à décharger ne peut être négative ou nulle".
Si la quantité passée en paramètre est supérieure à la quantité de fret du navire, on déclenchera une exception GestionPortException avec le message " Impossible de décharger plus que la quantité de fret dans le navire"
- ✓ Ajout de la méthode **EstDecharge()** qui renvoie vrai si le navire est entièrement déchargé (qteFret=0), false sinon.



Travail à faire

- ✓ Modifiez la classe Navire pour prendre en charge cette évolution.
- ✓ Les constructeurs doivent être modifiés, ainsi que toutes les instanciations actuelles qui vous ont permis de faire les tests.
- ✓ Effectuer les tests d'instanciation.

Classe Stockage (Espace de nom : GestionNavire.ClassesMetier) :

- ✓ **numero** :entier : en lecture seule. Il s'agit du numéro du stockage dans le port.

- ✓ **capacitéMaxi (entier)** : capacité de stockage maximum en tonnes. Modifiable et accessible depuis l'extérieur de la classe en lecture et écriture
- ✓ **capaciteDispo (entier)** : capacité de stockage disponible exprimé en tonnes. Non modifiable par une autre méthode que la méthode stocker. Accessible en lecture depuis l'extérieur de la classe
- ✓ **Constructeurs** : le premier admettra les paramètres numero, capaciteMaxi et capacité Dispo, le deuxième les paramètres numero, capaciteMaxi. La capaciteDispo sera alors initialisée avec la capaciteDispo. Bien entendu la capacité maximum doit être supérieure à 0 sinon déclenchement de l'exception GestionPortException avec comme message "Impossible de créer un stockage avec une capacité négative". La capacitéDispo, ne doit pas être supérieure à la capaciteMaxi.
- ✓ **Stocker(int quantite)** : va stocker la quantité passée en paramètre dans le stockage. Vous aurez 2 possibilités pour effectuer la mise à jour :
 - soit diminuer la capacité disponible de la valeur passée en paramètre, après les contrôles d'usage
 - soit réaffecter la capacité disponible avec la quantité disponible diminuée de la quantité à stocker.La quantité passée en paramètre doit être positive. Si ce n'est pas le cas, déclenchement d'une exception de la classe GestionPortException avec le message "la quantité à stocker dans un stockage ne peut être négative ou nulle". Si la quantité passée en paramètre est supérieure à la quantité disponible dans le stockage, on déclenchera une exception GestionPortException avec le message "Impossible de stocker plus que la capacité disponible dans le stockage"



Travail à faire

Créez la classe Stockage

Testez avec les méthodes ci-dessous de la classe Programm.



Observez la Property *CapaciteDispo* qui est public en get et privée en set !

```
public int CapaciteDispo { get => capaciteDispo;

    private set {
        if (value <= 0)
        {
            throw new GestionPortException("la quantité à stocker dans un stockage ne peut être négative ou nulle");
        }
        else if (this.capaciteDispo >= value)
        {
            // On peut stocker
            this.capaciteDispo -= value;
        }
        else
        {
            throw new GestionPortException("Impossible de stocker plus que la capacité disponible dans le stockage");
        }
    }
}
```



Il y a plusieurs bugs dans cette méthode.... A vous de les trouver et de les corriger

Testez avec la méthode `TesterInstanciationsStockage()`, vous devriez obtenir ceci :

```
Erreur de : Benoît le 26/01/2020 09:53:14
la quantité à stocker dans un stockage ne peut être négative ou nulle
Erreur de : Benoît le 26/01/2020 09:53:15
Impossible de stocker plus que la capacité disponible dans le stockage
Fin normale du programme
```

```
public static void TesterInstanciationsStockage()
{
    try
    { new Stockage(1, 15000); }
    catch (GestionPortException ex)
    { Console.WriteLine(ex.Message); }
    try
    { new Stockage(2, 12000, 10000); }
    catch (GestionPortException ex)
    { Console.WriteLine(ex.Message); }
    try
    { new Stockage(3, -25000, -10000); }
    catch (GestionPortException ex)
    { Console.WriteLine(ex.Message); }
    try
    { new Stockage(4, 15000, 20000); }
    catch (GestionPortException ex)
    { Console.WriteLine(ex.Message); }
}
```

Classe Port :

- ✓ Ajout de l'attribut **stockages** : objet de la classe `List<Stockage>` qui stockera des objets de la classe `Stockage`.



On considère que les zones de stockage sont situées sur le quai dans l'ordre où elles sont créées dans la collection `stockage` de la classe `Port`

- ✓ Ajout de la méthode **Dechargement(String imo)** : après s'être assuré que le navire est dans le port, et comme indiqué ci-dessus, le navire va parcourir les zones de stockages jusqu'à ce qu'il soit entièrement déchargé. Si le Navire n'a pas pu être entièrement déchargé, la méthode va déclencher une exception qui sera traitée par l'application. Le message d'erreur : "Le navire IMOxxxxxxx n'a pas pu être entièrement déchargé, il reste : xx tonnes".
Si le Navire passé en paramètre n'est pas du type "Porte-conteneurs", déclenchement d'une exception `GestionPortException`. (Message : "Ce type de Navire ne peut être



déchargé dans des zones de stockage"). Cette Exception sera traitée dans la partie applicative.

- ✓ Ajout de la méthode **AjoutStockage()** : ajoute un objet de la classe stockage dans la collection stockages.
- ✓ Ajout de la méthode **GetNavire(String imo)** : retourne le navire ayant comme numéro IMO le numéro IMO passé en paramètre. S'il ce navire n'existe pas dans la collection, la méthode renvoie *null*.



À vous de trouver la bonne méthode !!!



Travail à faire

Modifiez la classe Port pour prendre en compte les modifications demandées.
Testez avec les méthodes suivantes de la classe Programm.

Ajout de stockages dans le port, méthode de la classe Program :

```
public static void AjouterStockages()
{
    port.AjoutStockage(new Stockage(1, 160000));
    port.AjoutStockage(new Stockage(2, 12000));
    port.AjoutStockage(new Stockage(3, 25000));
    port.AjoutStockage(new Stockage(4, 15000));
    port.AjoutStockage(new Stockage(5, 15000));
    port.AjoutStockage(new Stockage(6, 15000));
    port.AjoutStockage(new Stockage(7, 15000));
    port.AjoutStockage(new Stockage(8, 15000));
    port.AjoutStockage(new Stockage(9, 35000));
    port.AjoutStockage(new Stockage(10, 19000));
}
```

Et vous devriez arriver à ce résultat :

```
Fin de chargement du Port
Erreur de : Benoît le 27/01/2020 11:40:46
Le navire IM09427639 est déjà enregistré
Erreur de : Benoît le 27/01/2020 11:40:46
Ajout impossible, le port est complet
----- Début des déchargements -----
Navire IM09839272 déchargé
Erreur de : Benoît le 27/01/2020 11:40:46
Impossible de décharger le navire IM01111111 il n'est pas ans le port ou n'est pas un porte conteneurs
Erreur de : Benoît le 27/01/2020 11:40:46
Impossible de décharger le navire IM09574004 il n'est pas ans le port ou n'est pas un porte conteneurs
Navire IM09786841 déchargé
Erreur de : Benoît le 27/01/2020 11:40:46
le navire IM09776432 n'a pas pu être entièrement déchargé, il reste 184000 tonnes
----- fin des déchargements -----
Erreur de : Benoît le 27/01/2020 11:40:46
impossible d'enregistrer le départ du navire IM01111111, il n'est pas dans le port
Fin normale du programme
```



Méthode TesterDechargerNavires()

```
public static void TesterDechargerNavires()
{
    try
    {
        String imo = "IMO9839272";
        port.Dechargement(imo);
        Console.WriteLine("Navire " + imo + " déchargé");
        port.EnregistrerDepart(imo);
    }
    catch (GestionPortException ex) { Console.WriteLine(ex.Message); }
    try
    {
        string imo = "IMO1111111";
        port.Dechargement(imo);
        Console.WriteLine("Navire " + imo + " déchargé");
    }
    catch (GestionPortException ex) { Console.WriteLine(ex.Message); }
    try
    {
        string imo = "IMO9574004";
        port.Dechargement(imo);
        Console.WriteLine("Navire " + imo + " déchargé");
    }
    catch (GestionPortException ex) { Console.WriteLine(ex.Message); }
    try
    {
        port.EnregistrerArrivee(new Navire("IMO9786841", "EVER GLOBE", "Porte-conteneurs", 198937, 190000));
        string imo = "IMO9786841";
        port.Dechargement(imo);
        Console.WriteLine("Navire " + imo + " déchargé");
        port.EnregistrerDepart(imo);
    }
    catch (GestionPortException ex) { Console.WriteLine(ex.Message); }
    try
    {
        port.EnregistrerArrivee(new Navire("IMO9776432", "CMACGM LOUIS BLERIOT", "Porte-conteneurs", 202684, 190000));
        string imo = "IMO9776432";
        port.Dechargement(imo);
        Console.WriteLine("Navire " + imo + " déchargé");
    }
    catch (GestionPortException ex) { Console.WriteLine(ex.Message); }
}
}
```

Méthode Main() :

```
static void Main()
{
    try
    {
        port = new Port("Toulon");
        InitPort();
        //Instanciations();
        try { TesterEnregistrerArrivee(); }
        catch (GestionPortException ex)
        { Console.WriteLine(ex.Message); }
        try { TesterEnregistrerArriveeV2(); }
        catch (GestionPortException ex)
        { Console.WriteLine(ex.Message); }
        Console.WriteLine("-----");
        Console.WriteLine("----- Début des déchargements -----");
        Console.WriteLine("-----");
        AjouterStockages();
        TesterDechargerNavires();
        Console.WriteLine("-----");
        Console.WriteLine("----- fin des déchargements -----");
        Console.WriteLine("-----");
        try { TesterEnregistrerDepart(); }
        catch (GestionPortException ex) { Console.WriteLine(ex.Message); }
        Console.WriteLine("Fin normale du programme");
    }
    catch (Exception ex)
    {
        Console.WriteLine(ex.Message);
    }
    finally { Console.ReadKey(); }
}
```

Fin de la partie 2

*Si vous avez tout compris de ce TD, vous êtes sur la bonne voie.
Sinon, posez vous des questions, interrogez votre professeur et persistez*

