

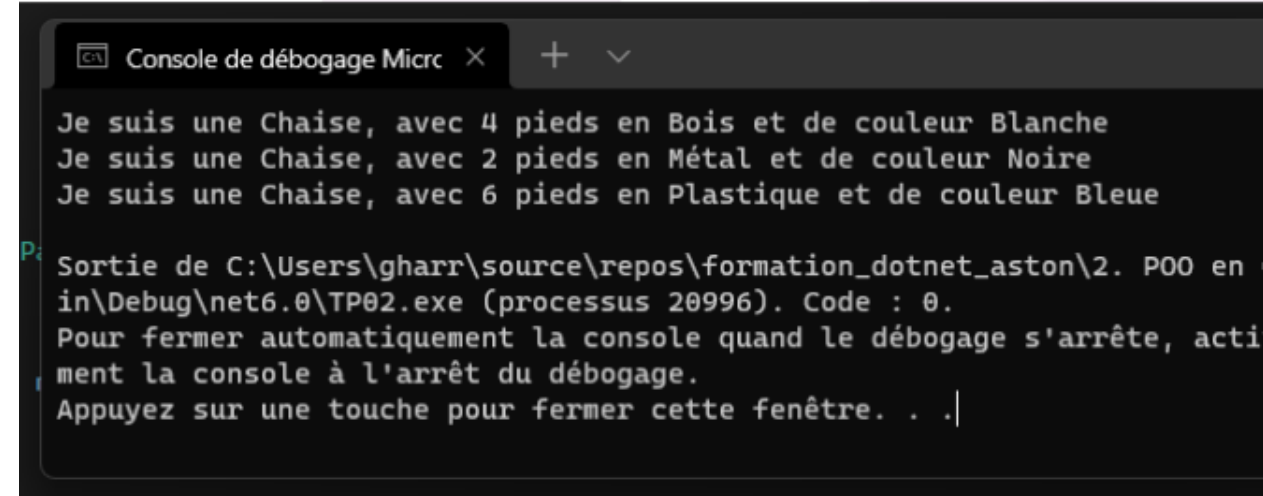
Exercices sur la POO

1) La chaise

Sujet

1. Créer une classe **Chaise** possédant comme variables d'instance le nombre de pieds, le matériaux et la couleur de l'objet
2. Afficher ses informations en surchargeant une méthode de la classe Object
3. La classe Chaise pourra être instanciée avec ou sans paramètres (Constructeur par défaut)
4. Afficher toutes les chaises (Possibilité de simplifier avec une méthode ToString)

Exemple



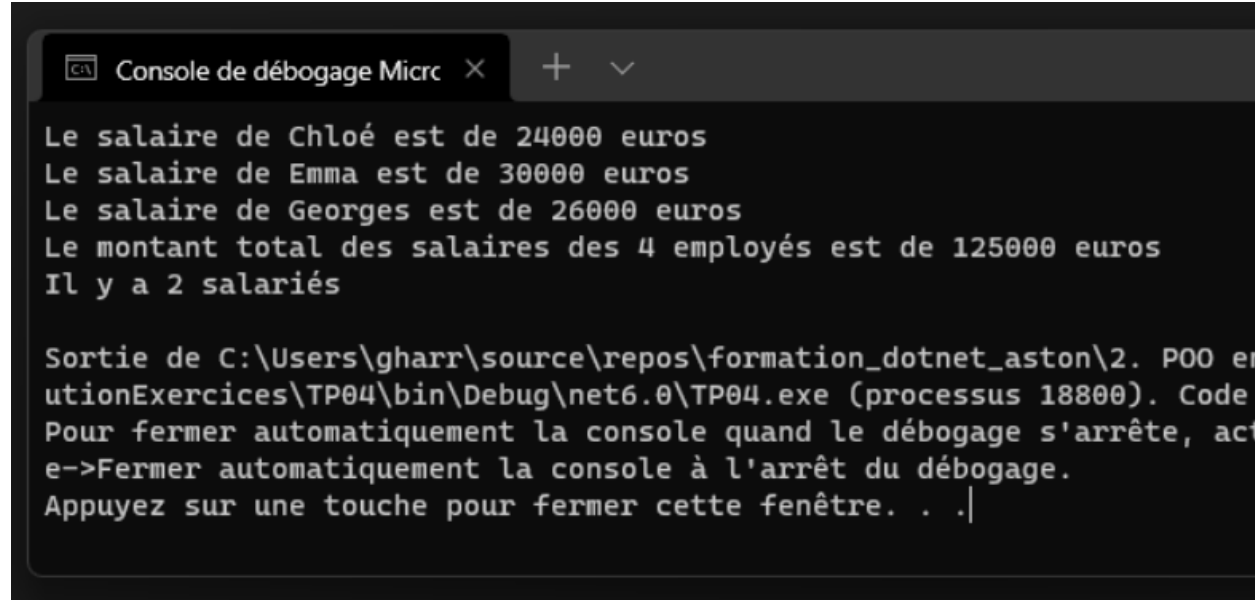
```
Console de débogage Micr...
Je suis une Chaise, avec 4 pieds en Bois et de couleur Blanche
Je suis une Chaise, avec 2 pieds en Métal et de couleur Noire
Je suis une Chaise, avec 6 pieds en Plastique et de couleur Bleue
P: Sortie de C:\Users\gharr\source\repos\formation_dotnet_aston\2. P00 en
in\Debug\net6.0\TP02.exe (processus 20996). Code : 0.
Pour fermer automatiquement la console quand le débogage s'arrête, acti
ment la console à l'arrêt du débogage.
Appuyez sur une touche pour fermer cette fenêtre. . . |
```

2) Le salarié

Sujet

1. Créer une classe **Salarié** ayant pour attributs : le matricule, le service, la catégorie, le nom et le salaire de l'employé
2. Cette classe aura également comme méthode **AfficherSalaire()**
3. Cette classe pourra, via deux champs et une méthode, permettre de savoir le nombre total d'employés, le salaire total et remettre à zéro la valeur du nombre d'employés dans l'entreprise
4. Créer une IHM pour tester le fonctionnement de l'application

Exemple



```
Console de débogage Micr... x + v
Le salaire de Chloé est de 24000 euros
Le salaire de Emma est de 30000 euros
Le salaire de Georges est de 26000 euros
Le montant total des salaires des 4 employés est de 125000 euros
Il y a 2 salariés

Sortie de C:\Users\gharr\source\repos\formation_dotnet_aston\2. P00 er
utionExercices\TP04\bin\Debug\net6.0\TP04.exe (processus 18800). Code
Pour fermer automatiquement la console quand le débogage s'arrête, act
e->Fermer automatiquement la console à l'arrêt du débogage.
Appuyez sur une touche pour fermer cette fenêtre. . .|
```

3) Le pendu

Sujet

1. Réaliser un jeu du pendu en créant une classe **Pendu** qui possédera au minimum comme attributs : le **masque**, le **nombre d'essais** ainsi que le **mot à trouver**. Cette classe aura comme méthodes : **TestChar()**, **TestWin()** et **GenererMasque()**.
2. Le joueur aura **par défaut 10 chances** pour gagner.
3. Utiliser **une autre classe** servant à **générer les mots** pour le jeu, à partir d'un tableau d'entrées potentielles
4. Optionnellement, le joueur pourra **choisir un nombre de coups** pour sa partie
5. Créer une **IHM** pour tester l'application

Exemple

```

C:\Users\gharr\source\repos x + -
=== Paramètres de partie ===
Voulez vous un nombre d'essais spécifique (10 par défaut) ? Y/nY
Combien voulez-vous d'essais ?15
Jeu du pendu généré ! Nombre d'essais : 15
Le mot à trouver : *****
Il vous reste 15 essais
Veuillez saisir une lettre : G
Le mot à trouver : *****
Il vous reste 14 essais
Veuillez saisir une lettre : O
Le mot à trouver : *****O**
Il vous reste 14 essais
Veuillez saisir une lettre : |
  
```

4) Citernes

Le but est de créer une classe qui décrit une citerne d'eau **WaterTank**.

- Elle aura un **poids à vide**, une **capacité totale** et un **niveau de remplissage**.
- La classe proposera également les méthodes suivantes :
 - Une méthode indiquant **le poids total** de la citerne.
 - Une méthode pour **remplir** la citerne avec un **nombre de litre** d'eau.
 - Une méthode pour **vider** la citerne d'eau d'un **nombre de litre** d'eau.
- La classe possédera, également, un attribut pour la **totalité des volumes** des citernes d'eau.
- Créez un programme pour tester votre classe (**IHM**)

Une fois fini ajouter la gestion des cas où la citerne est vide ou déborde :

- la méthode pour remplir renverra **l'excès** d'eau
- la méthode pour vider ne renverra que l'**eau disponible**

```
Poids total de la citerne 1 : 20
Poids total de la citerne 2 : 15
-----
Quantité d'eau dans la citerne 1 : 10
Quantité d'eau dans la citerne 2 : 10
Quantité d'eau dans toutes les citernes : 20
-----
Quantité d'eau dans la citerne 1 après ajout de 11 litres: 20/20
Excès d'eau récupéré : 1
Quantité d'eau dans la citerne 2 après tentative de retrait de 11 litres: 0/10
Quantité d'eau récupérée : 10
-----
Quantité d'eau dans la citerne 1 : 20
Quantité d'eau dans la citerne 2 : 0
Quantité d'eau dans toutes les citernes : 20
```

5) Salariés avec Heritage

Sujet

Créer une classe **Commercial** en dérivant la classe **Salarie**. Cette classe aura 2 propriétés supplémentaires pour calculer un montant de commission : **chiffre d'affaire** et **commission en %**.

- Créer les **deux constructeurs** de la classe Commercial. Ne pas oublier d'appeler les constructeurs équivalents de **la classe de base** (mère).
- Surcharger la méthode **AfficherSalaire()** pour calculer le salaire réel (**fixe + commission**).
- Ajoutez des méthodes **ToString** pour l'affichage des salariés et commerciaux.
- Écrire un programme (**IHM**) qui permet à une entreprise de 20 Employés (Salariés et commerciaux) :
 - D'**ajouter** des employés
 - D'**afficher les salaires** de chaque employé
 - De **rechercher** un employé par le début de son nom et afficher son **salaire**

Exemple

```
D:\Programation\Utopios\Fo...
===== Gestion des employés =====
1-- Ajouter un employé
2-- Afficher le salaire des employés
3-- Rechercher un employé
0-- Quitter

Entrez votre choix : _
```

```
D:\Programation\Utopios\Fo...
=== Ajouter un employé ===
1-- Salarié
2-- Commerciale
0-- Retour

Entrez votre choix :
```

5) Salariés avec Heritage (suite exemples)

```
D:\Programation\Utopios\FormationC#-NET\Codes\Découv...
=== Ajouter un employé ===
1-- Salarié
2-- Commerciale
0-- Retour

Entrez votre choix : 1
Merci de saisir le nom : Apeupré Jean-Michel
Merci de saisir le matricule : M001
Merci de saisir la categorie: C001
Merci de saisir le service : S001
Merci de saisir le salaire: 1800
```

```
D:\Programation\Utopios\FormationC#-NET\Codes\Découv...
=== Ajouter un employé ===
1-- Salarié
2-- Commerciale
0-- Retour

Entrez votre choix : 2
Merci de saisir le nom : Duss Jean-Claude
Merci de saisir le matricule : M002
Merci de saisir la categorie: C002
Merci de saisir le service : S002
Merci de saisir le salaire: 1300
Merci de saisir le chiffre d'affaire du commerciale : 13000
Merci de saisir la commission : 3
```

```
D:\Programation\Utopios\FormationC#-NET\Codes\Découv...
==== Salaire des employés ====
-----
TpClasseSalarieHeritage.Classes.Salarie
Le salaire fixe de Apeupré Jean-Michel est de 1800 euros
-----
TpClasseSalarieHeritage.Classes.Commerciale
Je suis un commerciale
Le salaire fixe de Duss Jean-Claude est de 1300 euros
Le salaire avec commission de Duss Jean-Claude est 1690 euros
-----
==== Gestion des employés ====
1-- Ajouter un employé
2-- Afficher le salaire des employés
3-- Rechercher un employé
0-- Quitter

Entrez votre choix :
```

```
D:\Programation\Utopios\FormationC#-NET\Codes\Découv...
==== Recherche employé par nom ====
Merci de saisir le nom : Duss Jean-Claude
Le salaire fixe de Duss Jean-Claude est de 1300 euros
Le salaire avec commission de Duss Jean-Claude est 1690 euros
==== Gestion des employés ====
1-- Ajouter un employé
2-- Afficher le salaire des employés
3-- Rechercher un employé
0-- Quitter

Entrez votre choix :
```

6) Compte bancaire

Sujet

1. Créer une classe **abstraite** **CompteBancaire**
 Cette classe aura : un **solde**, un **client** et une **liste d'opérations**(dépôt ou retrait).
2. Créer les classes : **ComptePayant**, **CompteEpargne**, **CompteCourant** qui héritent de **CompteBancaire**.
3. Créer une classe **Client** avec les attributs suivants : **nom**, **prénom**, **identifiant**, **liste des comptes** et numéro de téléphone. On créera le client au début de l'application
4. Créer une classe **Opération** avec les attributs suivants : **numéro**, **montant** et **statut** (depot/retrait en **enum**)
5. Créer une **IHM** pour tester l'application. Pour un compte au choix de l'utilisateur on pourra effectuer un **dépôt**, un **retrait** ou **afficher le solde et les opérations**

Exemple

```

C:\Users\gharr\source\repo: x + v
=== Menu Principal ===
1. Lister les comptes bancaires
2. Créer un compte bancaire
3. Effectuer un dépôt
4. Effectuer un retrait
5. Afficher les opérations et le solde
0. Quitter le programme
Votre choix : 2
=== Création de Compte ===
1. Créer un compte courant
2. Créer un compte épargne
3. Créer un compte payant
0. Annuler la création de compte
Votre choix : |
  
```


7) Forum

Sujet

1. Créer une classe **Forum** ayant les attributs suivants : **nom**, **dateCreation**, **abonnes[]**, **nouvelles[]** et **modérateur**
2. Créer une classe **Modérateur** et **Abonné** héritants d'une classe abstraite **Utilisateur** contenant les attributs suivants : **prenom**, **nom**, **age**.
L'**abonné** peut **ajouter** et **consulter** une nouvelle.
Le **modérateur** peut **supprimer une nouvelle**, **bannir un abonné**, **ajouter un abonné** ou **lister les abonnés et les nouvelles**.
3. Créer une classe **Nouvelle** contenant les attributs suivant : **sujet**, **descriptif**
4. Réaliser une **IHM** pour tester l'application

Exemple

```

C:\Users\gharr\source\repos  x  +  v
=== Premiere étape : Création du Forum ===

Quel est le nom de ce forum ? Test
Combien d'abonnés ce forum aura-t-il ? 10
Combien de nouvelles ce forum aura-t-il ? 10
Ce forum aura-t-il un modérateur ? Y/nY
=== Affectation d'un modérateur au forum ===

Quel est le nom du modérateur ? Modo
Quel est le prénom du modérateur ?
Quel est l'âge du modérateur ? 29
=== Menu Principal ===

1. Voir les abonnés
2. Ajouter un abonné
3. Bannir un abonné
4. Voir les nouvelles
5. Consulter une nouvelle
6. Ajouter une nouvelle
7. Répondre à une nouvelle
8. Supprimer une nouvelle
0. Quitter le programme
|

```

8) Figure

Sujet

1. Créer une classe **Point** possédant comme attributs **posX: double** et **posY: double** ainsi qu'une méthode **ToString()**
2. Créer une classe **abstraite Figure** possédant un attribut **origine** de type **Point**
Les classes créées ensuite **hériteront** de **Figure**
3. Créer une interface **IDeplacable** contenant la méthode **Deplacement(double, double)** permettant de déplacer l'**origine** de la figure
La classe **Figure** l'implémentera
4. Créer une classe **Carré** ayant comme nouvel attribut son **côté**
5. Créer une classe **Rectangle** ayant comme nouveaux attributs sa **longueur** et sa **largeur**
6. Créer une classe **Triangle** ayant comme attributs sa **base** et sa **hauteur** (ce triangle sera **isocèle**)
7. Toutes les Figures auront une méthode **ToString** et la méthode **Deplacement** implémentée
8. Réaliser une **IHM** pour tester l'application

Exemple

Coordonnées du carré ABCD (Côté = 2) :

```
A = 2;4
B = 4;4
C = 4;2
D = 2;2
```

Coordonnées du rectangle ABCD (Longueur = 3, Largeur = 5) :

```
A = 2;4
B = 5;4
C = 5;-1
D = 2;-1
```

Coordonnées du triangle ABCD (Base = 4, Hauteur = 5) :

```
A = 2;4
B = 4;-1
C = 0;-1
```

Deplacement du carré par (1,3) :

Coordonnées du carré ABCD (Côté = 2) :

```
A = 3;7
B = 5;7
C = 5;5
D = 3;5
```

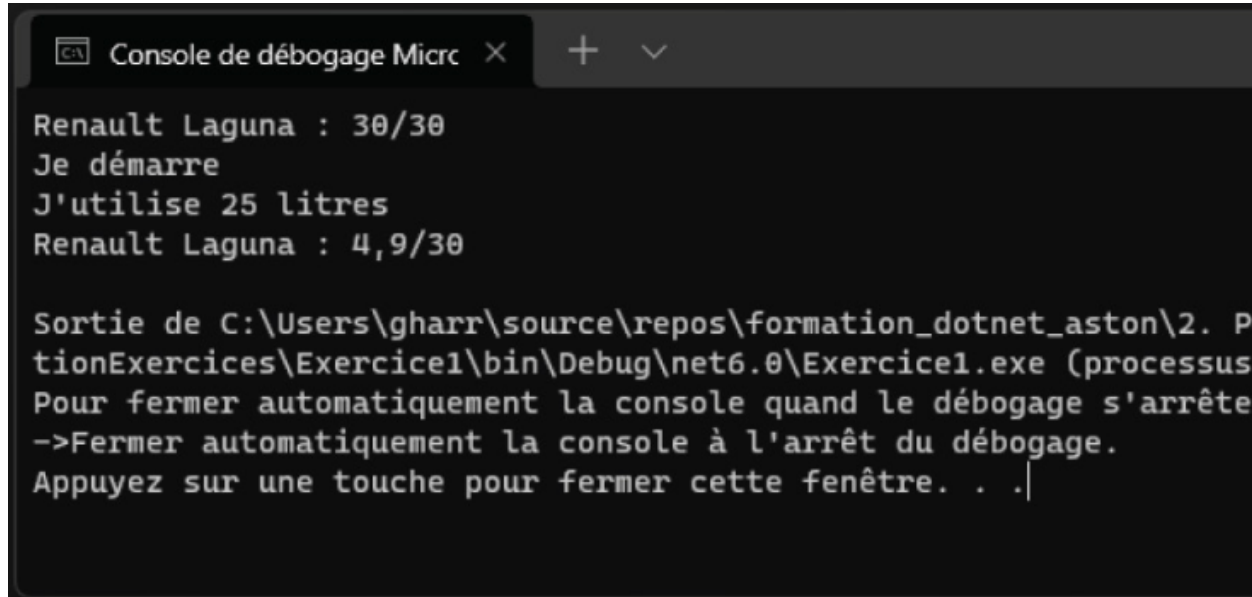
9) Voiture

Sujet

1. Créer une classe abstraite **Véhicule** contenant les attributs suivants et les initialiser à l'aide du constructeur : marque et modèle
Ajouter les méthodes suivantes à la classe Véhicule :
démarrer(): bool, arrêter(): void, faireLePlein(double): void
2. Créer une classe **Moteur** contenant les attributs suivants : volumeReservoir, volumeTotal, estDemarre.
Cette classe possédera les méthodes de véhicule.
démarrer() consommera cependant 1/10 de litre
3. Créer une classe abstraite héritant de Voiture nommée **VéhiculeAMoteur** ayant une propriété moteur qui servira à déléguer les trois méthodes héritées de Véhicule

4. Créer une classe héritant de VéhiculeAMoteur nommée **Voiture** qui servira à la construction d'une Peugeot 206
5. Créer une IHM pour tester le fonctionnement de l'application

Exemple



```

CA Console de débogage Micrc x + v
Renault Laguna : 30/30
Je démarre
J'utilise 25 litres
Renault Laguna : 4,9/30

Sortie de C:\Users\gharr\source\repos\formation_dotnet_aston\2. P
tionExercices\Exercice1\bin\Debug\net6.0\Exercice1.exe (processus
Pour fermer automatiquement la console quand le débogage s'arrête
->Fermer automatiquement la console à l'arrêt du débogage.
Appuyez sur une touche pour fermer cette fenêtre. . .|
  
```

10) L'hôtel

Sujet

1. Créer une classe **Client** possédant : un **identifiant**, un **nom**, un **prénom** et un **numéro de téléphone**
2. Créer une classe **Chambre** ayant : un **numéro**, un **statut** (libre/occupé/en nettoyage de type enum), un **nombre de lits** et un **tarif**.
3. Créer une classe **Réservation** possédant : un **identifiant**, un **statut** (prévu/en cours/fini/annulé), une **liste de chambres** et un **client**
4. Créer une classe **Hotel** comportant : une **liste de clients**, une **liste de chambres** et une **liste de réservations**
5. Créer une **IHM** pour tester l'application

Exemple

```

C:\Users\gharr\source\repos x + v
Quel est le nom de l'hôtel ? Excelsior
Excelsior créé avec succès !
=== Menu Principal ===

1. Ajouter un client
2. Afficher la liste des clients
3. Afficher les réservations d'un client
4. Ajouter une réservation
5. Annuler une réservation
6. Afficher la liste des réservations
0. Quitter
Votre choix : 1
=== Ajout d'un client ===

Quel est le nom du client ? MARTIN
Quel est le prénom du client ? Albert
Quel est le téléphone du client ? 0123456789
Client ajouté avec succès !

=== Menu Principal ===

1. Ajouter un client
2. Afficher la liste des clients
3. Afficher les réservations d'un client
4. Ajouter une réservation
5. Annuler une réservation
6. Afficher la liste des réservations
0. Quitter
Votre choix : |

```

11) La pile

Sujet

1. Créer une classe **Pile<T>** contenant un attribut `T[] elements`
2. Ajouter une méthode permettant d'**empiler** un nouvel élément
3. Ajouter une méthode permettant de **dépiler** le dernier élément empilé
4. Ajouter une méthode permettant de **recupérer** un élément par son index et ainsi de le **retirer** de la pile
5. Créer une IHM séparée en 3 parties : une pour une pile de **string**, une pour une pile de **decimal** et une pour une pile d'objet **Personne**(nom, prenom, age) (on pourra utiliser des méthodes génériques pour faciliter la saisie)

LIFO : Last In First Out => Pile/Stack

FIFO : First In First Out => File/Queue

Exemple

```

C:\Users\gharr\source\repos x + v

=== Menu Principal ===

1. Empiler
2. Dépiler
3. Récupérer à X
0. Quitter
Votre choix : 1

Valeur à empiler : Test
Test a été ajoutée à la pile !

=== Menu Principal ===

1. Empiler
2. Dépiler
3. Récupérer à X
0. Quitter
Votre choix : 3

Veuillez donner un indice :1
La valeur trouvée à l'indice 1 est : Test

=== Menu Principal ===

1. Empiler
2. Dépiler
3. Récupérer à X
0. Quitter
Votre choix : |
    
```