SEANCE 1 - CLASSE PERSONNE

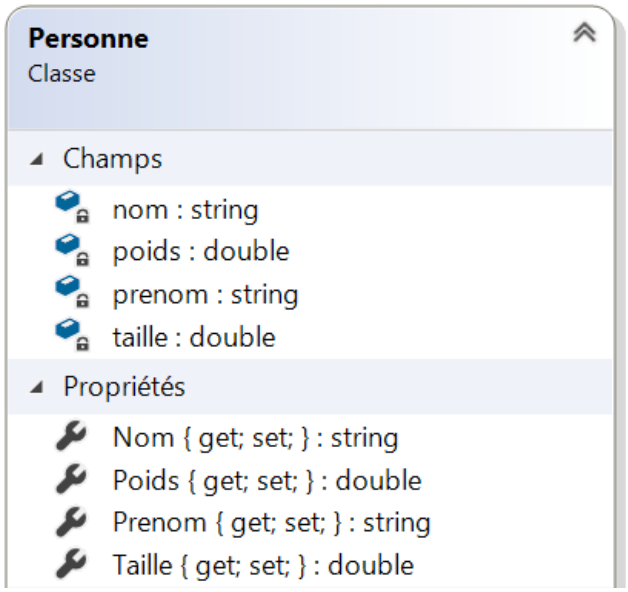
OBJECTIFS

- Assimiler le vocabulaire du paradigme (monde) « objet »
- Comprendre l'organisation d'une classe
- Découvrir le mécanisme de création d'une classe à travers l'IDE : acquérir les bons réflexes
- Comprendre le principe d'encapsulation

EXO : CLASSE PERSONNE

1. Créez un répertoire « **Sequence_6_Classes** » . Dans le quel, vous allez créer un répertoire « TD16-Classe-Personne »

2. Puis un projet « TD16 » dans lequel vous ajouterez la classe « **Personne »**. Définissez 4 champs **privés** : nom, prenom, taille, poids.



3. **Paramétrez visual studio editor : suivez les diapos 22 et 23 du cours sur les classes.**

4. Créez les **propriétés associées : sélection des champs puis** ctrl R + ctrl E ou clic sur l'ampoule/ encapsuler les champs.

5. Dans le main, **instanciez** un objet Personne.
   « Personne p = new Personne( ) ; »

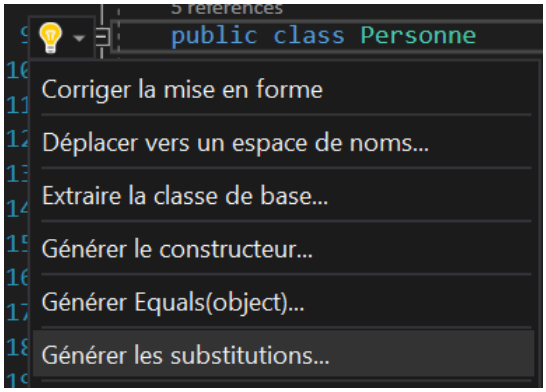
   p est une instance de la classe Personne, dès que vous manipulerez une propriété ou une méthode liée à p, le mot clef this dans la définition de la classe sera en fait p, this correspond à l'objet en cours d'utilisation, car une classe est une définition, son code ne s'exécute qu'à travers ses objets.

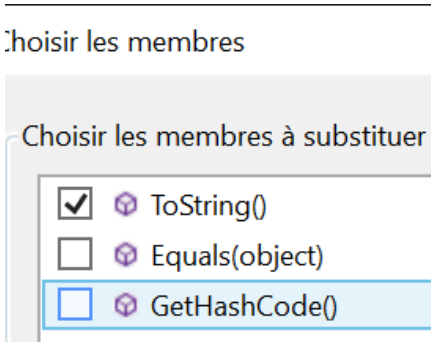
6. Dans le main, assignez des valeurs aux 4 champs de votre objet à l'aide des propriétés en mettant directement des données valides sans faire de saisies utilisateurs. Ex : p.Prenom="Marc" ;

7. **Exécutez ces quelques lignes en mode pas à pas F11 pour comprendre et suivre l'ordre et le cheminement des instructions.**

8. Dans le main, affichez p.
   « Console.WriteLine(p) ; »
   Rappel : « Console.WriteLine(p) ; » ⇔: « Console.WriteLine(p.ToString()) ; »
   Toutes les classes ont une méthode ToString par défaut, celle définie dans la classe Object qui fait le strict minimum dont elle est capable : concaténation du namespace et de la classe.

9. Cliquez sur Personne (1 seul clic) internal class Personne puis utilisez le tournevis pour générer les substitutions . Ne substituez que ToString pour le moment (car la substitution de Equals et GetHashCode n'est pas optimale depuis ce menu). Cela va générer le ToString par défaut qui fait appel au ToString hérité d'Object. Remplacez complétement le code généré par une concaténation des valeurs contenues dans les 4 champs à l'aide des propriétés (Ex : **this.Nom**) afin de retourner une chaîne descriptive de l'objet. Puis relancez votre programme.





10. Ajoutez les unités de mesure :

```
Public static readonly string UNITE_MESURE_POIDS = "kg";
public static readonly string UNITE_MESURE_TAILLE = "m";
```

Et utilisez-les dans votre ToString : **Personne.**UNITE_MESURE_POIDS

REM il est possible de ne pas préfixer par Personne, car on est dans la même classe. Mais cela permet de différencier l'usage :

- **D'une variable d'instance qui appartient à l'objet et dépend de l'objet préfixée par this**
- **D'une variable de classe préfixée par la classe (Personne)**

11. Exécutez ces quelques lignes en mode pas à pas F11 pour comprendre et suivre l'ordre et le cheminement des instructions.

12. Au sein des propriétés , programmez des vérifications de la value qu'on veut affecter aux champs pour assurer l'intégrité des données. Pour :
    a. **Prenom** : vérifiez que la donnée n'est pas nulle et pas vide (méthode IsNullOrEmpty() de la classe String). De plus, assurez-vous de stocker la donnée au format « nom propre » : 1ere lettre en majuscule puis minuscule. Puis dans le main, testez en modifiant le prénom avec des chaines vides, nulles ou tout en majuscules, tout en minuscule.

Donnée en entrée p.Prenom	**Résultat attendu**
MARC	Marc
marc	Marc
" "	Exception
null	Exception

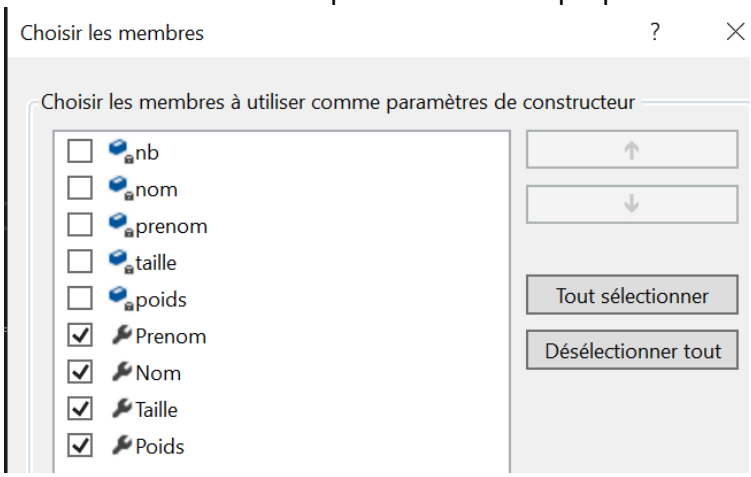
b. **Poids** : vérifiez que la donnée est un nombre positif qui n'excède pas 1 000k. Testez en suite avec un poids négatif puis > 1 000 pour vous assurer du déclenchement d'exception. Créez une constante pour stocker le POIDS_MAX :
public static readonly double POIDS_MAX = 1000;

Pour l'utiliser , veillez à la précéder par le nom de la classe : Personne.POIDS_MAX

Rappel : il s'agit d'une info stockée une seule fois au sein de la classe, elle est partagée par tous les objets Personne.

c. **Taille** : vérifiez que la donnée est un nombre positif qui n'excède pas 5 m. Créez une nouvelle constante. Testez en suite avec une taille négatif puis > 5 pour vous assurer du déclenchement d'exception.

13. Cliquez sur Personne (1 seul clic) internal class Personne puis utilisez le tournevis pour générer le **constructeur…** Sélectionnez les **propriétés** afin que ce constructeur **réutilise** les contrôles que vous venez de mettre en place au sein des propriétés.



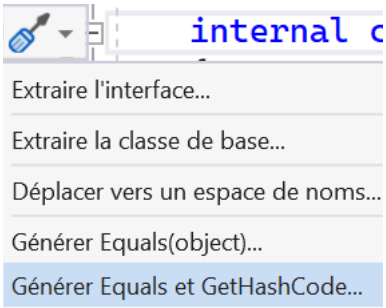
14. Que se passe-t-il dans votre main ? une erreur
Aide : le mécanisme de constructeur par défaut n'est plus mis en place.

15. Générez le **constructeur par défaut : Personne(). Rem : ce constructeur n'assure aucun contrôle des valeurs, il peut être dangereux de l'utiliser.**

16. Modifiez votre main pour construire un objet Personne avec le constructeur ayant 4 paramètres. Ainsi vous n'aurez plus besoin d'assigner les valeurs une par une.

17. Pour rendre le nom optionnel, surchargez le constructeur avec la signature suivant : Personne (String prenom, double taille, double poids**).**

18. Pour que votre classe soit complète, générez Equals, GetHashCode ainsi que les opérateurs, en prenant bien soin de sélectionner les propriétés. **Ainsi, il sera possible de tester si 2 personnes sont identiques !** cliquez sur Personne (un seul clic !) puis utilisez le tournevis.



19. Instanciez 2 personnes : vous et votre voisin. Testez et indiquez si vous êtes identiques.

20. Inspirez-vous de la signature des surcharges de == et != pour surcharger les opérateurs < et >, <= et >= : ils compareront la taille. Testez pour afficher le plus grand des 2.

21. Définissez dans la classe Personne les méthodes puis utilisez-les dans le main :
    a. public double CalculeImc() : calcule et renvoie l'IMC = Poids / Taille²
    b. public String DonneIndicationImc() : réutilise CalculeImc( ) puis renvoie le texte correspondant à l'imc.

REMARQUE : on appelle cela des méthodes d'instance (elles ne sont pas statiques) : elles s'appuient sur une instance (=objet) . Ici il n'y a pas de paramètres, car toutes les données nécessaires sont dans la classe (et donc dans l'objet).

IMC (kg.m-2)	Interprétation (d'après l'OMS)
Moins de 18,5	Maigreur
18,5 à 25	Corpulence normale
25 à 30	Surpoids
Plus de 30	Obésité

22. Améliorez votre main en instanciant, cette fois-ci, un objet Personne **à l'aide de saisies utilisateurs. Vous pouvez mettre en commentaire ce que vous avez fait précédemment dans le main.**

POUR LES PLUS RAPIDES

23. Ajoutez un champ dateNaissance : la propriété doit s'assurer que la date < date du jour

24. Définissez la méthode CalculeAge ( ) : int – elle renvoie l'âge de la personne. Aide : la propriété DayOfYear des objets DateTime peut être utile.

25. CalculeAge peut être transformée en propriété en lecture seule. Faites le nécessaire.

26. Définissez la méthode EstPlusJeune(Personne p) : bool – elle renvoie true si la personne est plus jeune que celle passée en paramètre, false sinon.