Université Internationale de Casablanca

UNIVERSITÉ RECONNUE PAR L'ÉTAT

POO: Langage C# (Suite)

Pr. EL OUKKAL Sanae

Constante

- Une constante nommée est associée à une valeur pour toute la durée de l'application.
- Sa valeur ne peut changer.

- Déclaration:
  - La syntaxe est similaire à celle de la déclaration d'une variable, excepté que le mot clé ***const*** précède la déclaration, et que l'initialisation à la déclaration est obligatoire :

const *type nom = expression ;*

Constante

- Une constante est implicitement statique. Il est donc inutile d'ajouter le mot-clé *static*.

- Portée et durée de vie d'une constante:
  - Les mêmes règles que celles pour les variables s'appliquent également aux constantes.

Valeur par Défaut & Obtention du Type

- Valeur par défaut:
  - L'opérateur default retourne la valeur par défaut du type spécifié. Il s'agit de la valeur quand une variable de ce type n'est pas initialisée (0 pour les nombres, null pour les types références).

- Obtenir le type:
  - L'opérateur *typeof* retourne une instance de la classe System. **Type** pour le type spécifié entre parenthèses.
  - Exemple:

```
Type t=typeof(int);
```

Université Internationale de Casablanca

UNIVERSITÉ RECONNUE PAR L'ÉTAT

NOTIONS DE BASE DE LA POO

POO

- POO est une méthode de programmation qui est basée sur l'évolution des objets.

- Ce qui permet:
  - Partage d'objet,
  - Gain de temps,

- Deux notions sont traitées dans la PO:
  - Classe
  - Objet.

Avantages POO

- Facilité l'organisation grâce aux objets.
- Méthode plus intuitive car plus proche de la réalité, principe d'héritage
- Gestion de projet plus efficace : la sécurisation via encapsulation permet d'avoir plusieurs développeurs travaillant sur un même projet, chacun ne travaille que sur les implémentations le concernant.
- Le code est "factorisé" : il est plus facilement lisible et donc corrigible, et est moins lourd.

Classe

- La classe est l'élément fondamental contenant les éléments du programme:
  - Une déclaration de données,
  - Une méthode ou une instruction appartenant à une classe.

- Une classe est déclarée avec le mot- clé *class*.

- Le concept de classe correspond à la généralisation de la notion de type des programmes structurés.

Classe

- Créer un nouveau type de données, c'est modéliser de la manière la plus juste un objet, à partir des possibilités offertes par un langage de programmation.

- Une classe est modèle, dite aussi définition.

- Les membres d'une classe:
  - Les attributs, privés
  - Les méthodes

Classe

- Il faudra donc énumérer toutes les propriétés de cet objet et toutes les fonctions qui vont permettre de définir son comportement.

- Exemple:
  - Une voiture c'est une classe.
  - Et les objets ce sont les différents modèles de voiture qui existent.
  - Leurs fonctionnement est d'avoir la possibilité d'accélérer, de freiner, ….

Classe:
Membre d'une classe

- Les membres d'une classe sont de différents types :
  - Les fonctions, appelées méthodes, traitent les données, appellent d'autres méthodes, retournent éventuellement une valeur ;
  - Les variables, appelées attributs, stockent les données ;
  - Les propriétés sont des méthodes spéciales utilisées comme des variables.

Classe: Attributs

- Un attribut d'une classe se déclare avec un spécificateur d'accès qui précise son niveau de visibilité.

- Les attributs sont déclarés privés, c'est à dire inaccessibles en dehors de la classe produite:
  - Ceci est une règle fondamentale de la programmation objet

Classe: Méthodes

- Le rôle d'une méthode est d'effectuer un traitement en utilisant les données.
- Les méthodes peuvent être publiques ou privées.
- L'ensemble des méthodes publiques représente l'interface de la classe, sa partie "utilisable" à l'extérieur.

Classe: Méthodes

- On peut les classer en trois groupes selon leurs fonctionnalités:
  - Les fonctions de création:
    - Ces fonctions qui permettent de créer des objets.
  - Les méthodes accesseurs/modificateurs:
    - Elles permettent :

      Soit de donner l'état de l'objet sans le modifier (get);

      Soit de modifier l'état de l'objet (set).
  - Les méthodes agissant sur le comportement ou l'état de l'objet. Ce sont toutes les autres méthodes.

Objet

- C’est le fait d’instancier une classe,
- C’est la réalisation concrète de la structure de l’objet.
- Cet objet a toutes les fonctionnalités déclarées dans la classe, les propriétés et les éléments qui le composent.

Objet:
Le type object

- La classe *System.Object* est équivalente au type object.

- Il s'agit de la classe de base de toutes les autres classes.

- Créer un nouvel objet est appelé *instancier une classe* et utilise l'opérateur *new* :

  classe variable=new classe(arguments...);

Constructeur et Opérateur new

- On retrouve dans une classe, des méthodes spéciales appelées constructeurs qui permettent d'instancier un objet.

- Un constructeur permet de créer une instance de la classe, en donnant une valeur initiale à toutes ses variables membre.

- L'objet est créé dans un certain état.

Constructeur et Opérateur new

- Le constructeur est une méthode qui porte le même nom que la classe mais qui n'a pas de type.

- Le constructeur est exécuté lorsqu'un objet est créé à l'aide de l'opérateur new.

- Il existe 3 types de constructeurs:
  - Un constructeur par défaut/sans arguments,
  - Un constructeur avec arguments/paramétré,
  - Un constructeur par recopie.

Règles sur Constructeur

- Les constructeurs ont une signature particulière:
  - Ils permettent la construction d'objet,
  - Ils portent le nom de la classe.
  - Ils ne retourne rien, pas même void.
  - Toujours public.
  - Possibilité de surcharge.

Constructeur sans Arguments

- Dans ce constructeur , nous effectuons l'initialisation de valeurs.
- Au lieu de le faire pour chaque instance de notre classe, nous le faisons dans le constructeur.
- L'initialisation sera faite automatiquement pour chaque instance.

Constructeur sans Arguments

- Lorsqu'une classe ne définit aucun constructeur, un constructeur par défaut sans aucun paramètre est implicitement créé. Il n'a aucun comportement mais son existence permet d'instancier des objets de cette classe.

- En revanche, toute définition explicite d'un constructeur dans une classe « désactive » le constructeur par défaut.

Constructeur sans Arguments

- Exemple:

```
class Personne{
        /* attributs */
// constructeur par défaut ou sans paramètres:
        public Personne ( ) {
}
```

Constructeur Paramétré

- C’est le cas contraire du constructeur par défaut.
- Dans ce constructeur , nous allons donner aux objets instanciés des valeurs précises.
- On peut avoir plusieurs constructeurs avec arguments dans une même classe. Ce qui différencie les unes des autres est le nombre de paramètre.

Constructeur Paramétré

- Exemple:

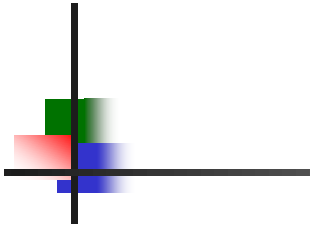
```
class Personne{
/* attributs */
//constructeur paramétré

public Personne ( paramètres){
//instructions
}
```

Constructeur par Copie

- Lorsqu'un objet est initialisé par un autre objet de la même classe, on parle du constructeur par copie.

- Les deux objets crées par ces constructeurs (constructeur paramétré et constructeur par copie) sont deux instances distinctes, mais elles ont les mêmes valeurs.

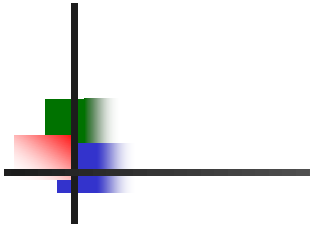
- Le constructeur par copie permet d'initialiser une instance en copiant les attributs d'une autre instance du même type.



LE MOT CLÉ THIS

This

- Dans toute méthode, vous disposez d'une référence vers l'objets servant de contexte à l'exécution de la méthode, cette référence s'appelle ***this***.



L'ENCAPSULATION

L'encapsulation

- Un des grands principes de la POO est *l'encapsulation*.

- L'encapsulation permet de cacher l'implémentation d'un objet (les informations liées à la structure de l'objet).

- Protéger l'information contenue dans notre objet et le rendre manipulable que par ses actions ou propriétés.

L'encapsulation

- Ainsi, notre objet est protégé et il fonctionne comme une boite noire.
- Par conséquent, un utilisateur n'ayant pas les droits requis, ne peut accéder aux données que par les interfaces(notions qu'on verra par la suite).
- Cela permet d'assurer l'intégrité des données.

L'encapsulation

- L'encapsulation est l'un des principes fondamentaux de la POO.

- L'objectif de l'encapsulation est de ne laisser accessible que le strict nécessaire pour que la classe soit utilisable.

Accesseurs

- L'encapsulation des attributs a permis d'interdire toute modification (accidentelle ou volontaire) des données d'un objet.

- Par conséquent, on aimerait pouvoir accéder aux données de la classe, tout en maintenant un certain niveau de contrôle. Cela est possible en ajoutant des *accesseurs* à la classe.

Accesseurs

- Un accesseur est une méthode le plus souvent publique qui permet d'accéder à un attribut privé:
  - Un accesseur en lecture (getter) permet de lire la valeur d'un attribut.
  - Un accesseur en écriture (mutateur ou setter) permet de modifier la valeur d'un attribut.

Accesseurs (méthodes)

- Syntaxe définition de la méthode get:

```
public  typeattribut   getNomAttribut(){
                return NomAttribut;
}
```

- Syntaxe de l'appel:

```
typeattrinut  nomVar=nominstance.getNomAttribut();
```

Accesseurs (méthodes)

- Syntaxe définition de la méthode set:

```
public void setNomAttribut(typeattrubut NomParam){
        NomAttribut = NomParam;
}
```

- Syntaxe de l'appel:

```
nominstance.setNomAttribut(valeur);
```

Accesseurs(Propriété)

- En C#, les accesseurs prennent la forme de propriétés.

- Une propriété se manipule comme un champ, mais il s'agit en réalité d'un couple d'accesseurs *get* et *set*.

Accesseurs(propriété)

- Syntaxe:

```
public typeattribut  NomPourAttribut{
        get { return nomAttribut; }
        set { nomAttribut = value; }
}
```

Accesseur(propriété): get

- **getNomAttribut**{ **return** ..... ;}:

  cet accesseur indique que la propriété est en lecture et doit renvoyer un résultat dont le type doit être le même que celui de l'attribut.

- Exemple:

```
public string getHello( ) {
        return hello ;
}
```

Accesseur(propriété): set

- **setNomAttribut** { ....}:

  cet accesseur indique que l'attribut est en écriture et sert à initialiser ou à modifier la propriété.

- Exemple:

```
public void setHello(string nvHello) {
        hello = nvHello;
}
```

Accesseurs(Propriété): Appel

- Syntaxe d'appel de la propriété:

Pour considérer le get, on utilise la syntaxe suivante:

- typeattribuut nomvar=nominstance.NomPropriétéPourAttribut;

Pour considérer le set, on utilise la syntaxe suivante:

- nominstance.NomPropriétéPourAttribut=valeur;

Exercice 1:

- Définir une classe *Point* caractérisée par son abscisse et son ordonné.
- Définir à l'aide des *getters* et les *setters* les méthodes d'accès aux attributs de la classe.
- Définir le constructeur sans paramètre et d'initialisation de la classe.
- Définir le constructeur avec argument qui permet de créer des points dans le repère.
- Définir le constructeur par recopie de la classe point
- Définir la méthode *distance ()* qui retourne la distance entre l'origine du repère et le point en cours.
- Écrire un programme permettant de tester la classe.

BONNE CHANCE