informatique

ICT320

POO

PROGRAMMATION ORIENTEE OBJET

ANASS BENFARES &
XAVIER CARREL

ANASS BENFARES

Déroulement

- Accès à la Roadmap du cours : https://roadmap.sh/r/embed?id=66714f98c0f2325c34220bba

- 5 périodes de cours avec moi et des périodes de projet avec M.Melly.
- Théorie puis exercices pratiques.
- Informations concernant les évaluations à ce lien
- Ne pas oublier, les conventions de codage ETML !

Semaine 1 - Check

- Qu’a-t-on vu la dernière fois ?
- Quelles sont les éléments à retenir ?

Sujets

- Encapsulation des données
- Membres statiques
- Tests unitaires
- Game Engine

Encapsulation des données

La dernière fois nous avons vu les méthodes ou les attributs, nous intégrons maintenant un nouvel élément.

Les Propriétés

Encapsulation des données - Visibilité

Spécificité ?

```
class Car
{
    // Attributs
    private string _color;
    private int _doorNumber;


    /// <summary>
    /// Start a car
    /// </summary>
    public void start()
    {
        // TODO : create code
    }

    /// <summary>
    /// Stop a car
    /// </summary>
    public void stop()
    {
        // TODO : create code
    }

    /// <summary>
    /// Turn on or turn off headlights for a car
    /// </summary>
    /// <param name="toogle">Turn on or turn off headlights</param>
    public void toggleHeadlights(bool toogle)
    {
        // TODO : create code
    }
}
```

Encapsulation des données - Visibilité

Spécificité ?

```
class Car
{
    // Attributs
    public string _color;
    private int _doorNumber;


    /// <summary>
    /// Start a car
    /// </summary>
    public void start()
    {
        // TODO : create code
    }

    /// <summary>
    /// Stop a car
    /// </summary>
    public void stop()
    {
        // TODO : create code
    }

    /// <summary>
    /// Turn on or turn off headlights for a car
    /// </summary>
    /// <param name="toogle">Turn on or turn off headlights</param>
    public void toggleHeadlights(bool toogle)
    {
        // TODO : create code
    }
}
```

Encapsulation des données - Visibilité

```
Car porsche = new Car();
```

On peut accéder à un attribut de la classe **pour autant qu'il soit public**, de la sorte :

```
porsche._color = "Blue";
```

il est donc possible de modifier des éléments interne à la classe! -> ⚠ **Problème de Sécurité**

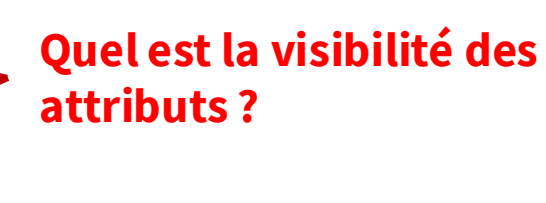
Encapsulation des données - Visibilité

Pour les méthodes et les attributs, il existe des niveaux de privilèges en fonction de la visibilité souhaitée.

Visibilité de la méthode / attribut	Influence
Private	Accessible depuis la class uniquement
Protected	Accessible depuis la class et les class qui héritent également (semaine 3)
Public	Accès depuis n'importe quelle autre partie du code, y compris à l'extérieur de la class ou du Namespace.

Encapsulation des données - Visibilité

```
using SnailJMY1;

var snail1 = new Snail();
snail1.Nickname = "steph";
snail1.Y = 1;

var snail2 = new Snail();
snail2.Nickname = "bob";
snail2.Y = 2;

snail1.BeReady();
snail2.BeReady();

for (int i = 0; i < 20; i++)
{
    snail1.MoveRight();
    snail2.MoveRight();

    Thread.Sleep(250);
}
```



Quel est la visibilité des attributs ?

Encapsulation #1

L’idée est de mettre des attributs privés et des méthodes publiques qui accède à l’attribut.

Cela pourrait être utile pour la création d’une validation via une méthode.

Get –> Utilisation d’un return pour obtenir la valeur de l’attribut en output de la méthode.

Set -> Modification de l’attribut via une variable placée en argument de la méthode.

Properties

Une propriété est une structure qui encapsule **un attribut.**

Elle permet de contrôler les accès à l'attribut qu'elle encapsule **via des accesseurs** et **mutateurs** (Get & Set)

C'est une sorte de méthode « masquée ».

Properties automatique

public string Color { get; set; } -> Propriété automatique

Encapsulation #2

L’idée est de mettre des attributs privés et des méthodes publiques qui accède à l’attribut.

Get –> Utilisation d’un return pour obtenir la valeur de l’attribut en output de la méthode.

Set -> Modification de l’attribut via une variable placée en argument de la méthode.

Membres statiques / Helpers

Les membres statiques sont des membres d'une class tout comme les attributs et les méthodes que nous avons déjà créé.

Ce sont des éléments qui sont **partagés** entre toutes les instances d'une classe sans avoir besoin de créer une instance dans cette classe.

Nous devons ajouté entre notre visibilité et notre nom de variable le terme « static ».

Membres statiques / Helpers

Exemple :

Le nombre total d’étudiants est un attribut statique.

La méthode qui va afficher le nombre total d’étudiant est une méthode statique.

On peut avoir des méthodes utilitaires dites statiques -> qui sont nommé Helpers.

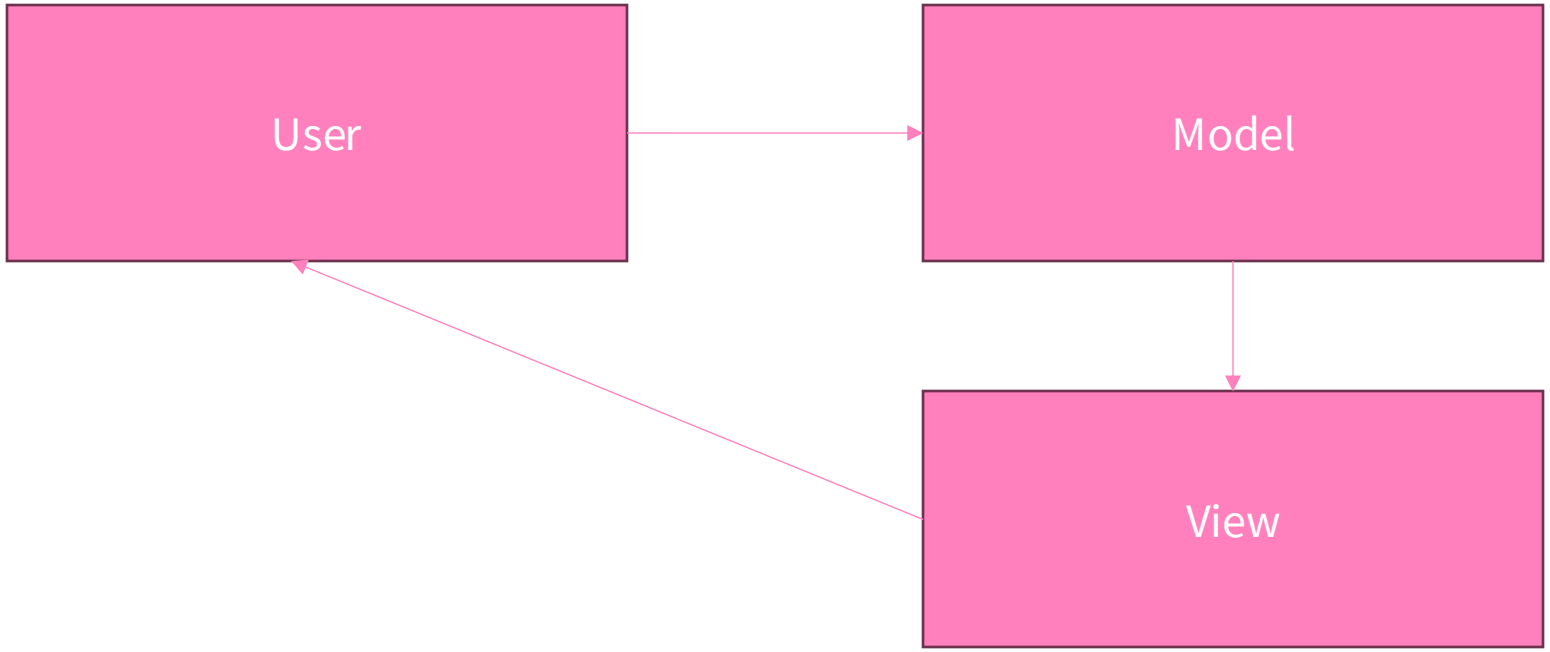
Membres statiques

```
class Student
{
    // Attributs / Proprietes
    2 références
    private int _idStudent;
    2 références
    private string _firstName;
    2 références
    private string _lastName;
    //Static attribut
    3 références
    public static int _studentTotal = 0;

    /// <summary>
    /// Constructeur pour initialiser un nouvel élève
    /// <param name=idEleve'>Student ID</param>
    /// </summary>
    2 références
    public Student(int id,string prenom, string nom)
    {
        this._idStudent = id;
        this._lastName = nom;
        this._firstName = prenom;
        _studentTotal++;
    }

    // Méthode statique pour afficher le nombre total d'élèves
    0 références
    public static void ShowStudentTotal()
    {
        Console.WriteLine("Nombre total d'élèves dans l'école : " + _studentTotal);
    }

    // Méthode non-statique pour afficher les détails d'un élève
    2 références
    public void ShowStudentDetails()
    {
        Console.WriteLine("ID de l'élève : " + this._idStudent+" " +this._firstName +" " +this._lastName);
    }
}
```

```
static void Main()
{
    // Création de deux élèves
    Student eleve1 = new Student(1,"John","Doe");
    Student eleve2 = new Student(2,"Jane", "Darc");

    // Affichage des détails des élèves
    eleve1.ShowStudentDetails(); // Affiche "ID de l'élève : 1"
    eleve2.ShowStudentDetails(); // Affiche "ID de l'élève : 2"
    Console.WriteLine(Student._studentTotal);


}
```

Création des objets dans le Main

Game engine



Tests unitaires

On doit effectuer des tests définis à l'avance

Selon trois grands principes:

- Arrange
- Act
- Assert

Pour définir les tests, on utilise un Framework nommé MSTest

Tests unitaires

On va vérifier le comportement de nos méthodes à l'aides des Tests unitaires.

Pour cela on va recréer des méthodes spécifiques.

Exercices pratiques

- Snail
- Parachutes
- Drones