

A detailed anatomical model of a human torso and head. The head is shown from the side, revealing the brain, eye, and ear structures. The neck area displays the trachea, esophagus, and major blood vessels. The chest cavity is exposed, showing the ribcage, lungs, heart, and surrounding muscles. The skin is a light beige color, and the internal organs are colored in various shades of red, blue, and yellow.

Encapsulation

Objectifs

- Notion d'encapsulation
- Modificateurs d'accès ***private / public*** (D'autres modificateurs seront vus plus loin dans le cours)

Encapsulation

- En programmation, l'encapsulation désigne le principe de regrouper des **données brutes** (état) avec un ensemble de routines permettant de les lire ou de les manipuler.
- Ce principe est souvent accompagné du **masquage** de ces **données brutes** afin de s'assurer que l'utilisateur ne contourne pas l'interface qui lui est destinée.
- L'ensemble se considère alors comme **une boîte noire ayant un comportement et des propriétés spécifiés**.

Encapsulation

- Protéger l'information
 - Accès aux informations pertinentes en lecture
 - Accès à des méthodes pour agir sur l'objet et donc le modifier (comportements)
- Cohérence
 - Un objet contient des données valides tout au long de sa vie
 - De l'initialisation jusqu'à la destruction



Modificateurs d'accès **private** / **public**

- Pour cacher la présence d'un membre d'une classe, on va utiliser le modificateur d'accès « **private** »
- Les membres privés ne sont accessibles qu'à partir des méthodes de la classe
- Ce modificateur peut aussi être appliqué sur une des méthodes des propriétés

```
public class DemoEncapsulation
{
    public List<Client> Clients { get; private set; }
}
```

- « **private** » est l'accès le plus restrictif



Modificateurs d'accès private / public

- Pour rendre public un membre d'une classe, on va utiliser le modificateur d'accès « **public** »
- Les membres public sont accessibles de partout, c'est-à-dire à l'intérieur de la classe et à l'extérieur
- « **public** » est l'accès le moins restrictif



Semblant d'encapsulation 1

Le solde n'est pas accessible directement mais au travers d'une propriété.

Ici, c'est comme si on donnait directement accès à la valeur.

Le seul avantage ici est de pouvoir modifier la façon de représenter l'état du solde.

```
public class DemoEncapsulation
{
    private decimal m_solde;

    public decimal Solde
    {
        get
        {
            return this.m_solde;
        }
        set
        {
            this.m_solde = value;
        }
    }
}
```



Semblant d'encapsulation 2

```
public class DemoEncapsulation
{
    private List<Client> m_clients;
    public List<Client> Clients
    {
        get
        {
            return this.m_clients;
        }
    }
    // ...
}
```

ou

```
public class DemoEncapsulation
{
    public List<Client> Clients { get; private set; }
    // ...
}
```

- La liste de clients ne peut pas être définie de l'extérieur de la classe.
- Par contre, en renvoyant la référence de la liste, un programmeur peut **modifier son contenu de l'extérieur**



Semblant d'encapsulation 2

```
public class DemoEncapsulation
{
    public List<Client> Clients { get; private set; }
    // ...
}

public static void DemoFuiteEncapsulation()
{
    DemoEncapsulation de = new DemoEncapsulation();
    List<Client> listeClients = de.Clients;
    // ...
}
```

