# Constantes, variables et méthodes de classe

## AP2 - programmation objet en C++

Semestre 2, année 2009-2010

Département d'informatique
IUT Bordeaux 1

Février 2010

# Le projet

Situation : une classe `Gadget`

- un *gadget* possède une *couleur*
- chacun a un *numéro de série* différent.

## Exemple

```
Gadget maserati("jaune");
..
cout << "le gadget  " << maserati.getColor()
     << " porte le numéro de série "
     << maserati.getSerialNumber();
```

Le numéro de série est attribué automatiquement

# Les constructeurs

- sans paramètre
  - lui attribuer un nouveau numéro de série
  - lui affecter la couleur par défaut
- avec une couleur en paramètre :
  - lui attribuer un nouveau numéro de série
  - lui affecter la couleur indiquée
- constructeur par copie
  - lui attribuer un nouveau numéro de série
  - copier la couleur

# Affectation

- Affectation `g1 = g2`
  - change la couleur de `g1`
  - laisse son numéro de série inchangé

# Première esquisse

## Gadget.h

```
class Gadget
{
private:
    string my_color;
    int    my_serialNumber;

public:
    Gadget();
    ..
    string getColor()          const;
    int    getSerialNumber()   const;
}
```

### Remarque

- Les données `my_color`, `my_serialNumber` et les accesseurs `getColor()`, `getSerialNumber()` sont liés à un objet Gadget existant.

On parle de variables ou méthodes d'instance.

Le constructeur sans paramètres utilise

- une constante `defaultColor` (chaîne)
- une fonction `makeNewSerialNumber()`

**Pseudo-code**

```
méthode makeNewSerialNumber() retourne entier :
    incrémenter lastSerialNumber;
    retourner   lastSerialNumber;
```

- une variable `lastSerialNumber`

Ces éléments sont associés à la classe, mais ne sont pas liés à une instance

constantes, variables et méthodes de classe

# Déclarations

## Gadget.h

```
class Gadget
{
private:
  ...
  static const string defaultColor;
  static       int    lastSerialNumber;
  static       int    makeNewSerialNumber();
  ...
}
```

# Initialisation

En général, se fait dans le code source de la classe

## Gadget.cc

```
#include "Gadget.h"
#include <string>
using namespace std;

const string Gadget::defaultColor      = "black";
int          Gadget::lastSerialNumber  =  2010000;
```

Exception : pour les constantes de classe de type entier, l'initialisation peut se faire directement dans le fichier d'entêtes.

# Raison technique

Le compilateur

- matérialise la déclaration d'une constante de classe

```
class C {
  static const T  var;
};
```

par la réservation d'une zone mémoire statique associé au nom `C::var`

- traduit l'initialisation

```
T C::var = valeur;
```

par une séquence d'instructions qui sera exécutée au démarrage.

- une seule séquence d'initialisation par zone.

# Initialisation dans l'entête

- Conséquence : si l'initialisation était déclarée dans le fichier d'entête,
  il y aurait autant de séquences d'initialisation
  que de fichiers sources qui incluent l'entête.
- Erreur à l'édition des liens si l'entête est citée plusieurs fois (définition multiple).

Exception pour les constantes entières qui peuvent "expansées" par le compilateur là où elles sont employées (pas de réservation d'espace mémoire).

# Méthodes de classe

Les méthodes de classe

- ne sont pas liées à des instances
- n'ont donc pas accès aux variables et méthodes d'instance

# Visibilité

Les variables, constantes et méthodes de classe peuvent être appelées de l'extérieur (si elles sont publiques).
On les préfixe par le nom de la classe.

## Exemple

```
int n = Gadget::numberOfGadgets();
```