EZ Langage : Les classes

PLAN

1. Les constructeurs

Définition

Un constructeur est ce qui construit un objet et alloue de la mémoire. On peut le comparer à une fonction d'initialisation de la classe.

En EZ :

- Il n'y pas d'encapsulation : tous les attributs sont accessibles et par défaut public.
- Le constructeur par défaut et le constructeur paramétrique n'ont pas besoin d'être définis (ils sont créés implicitement).

Il y a plusieurs manières d'instancier un objet (voir exemples ci-dessous).

Syntaxe

```
c is MaClasse
// or
// Même ordre que la déclaration des attributs dans la classe
c is MaClass(val1,val2)
// or
c is MaClass(Att1="val1", Att3="val2")
```

Ci-dessous des exemples de déclaration

Exemples

Le constructeur par défaut

En EZ :

```
class Person

    nom is string
    prenom is string

end class

// Déclaration 1
p is Person
// Déclaration 2
p is Person("nom","prenom")
```

Traduction en C++ :

```
class Person
{
public:

   string nom;
   string prenom;
   Person(){
   }
};

// Déclaration 1
Person p;
// Déclaration 2
Person pp;
```

Le constructeur avec paramètres

En EZ :

```
class Person


        nom is string
        prenom is string
        age is integer

end class


// Déclaration 1
p is Person
// Déclaration 2
p is Person("nom", "prenom", 20)
// Déclaration 3:
p is Person(nom="nom", prenom="prenom")
// Déclaration 4:
p is Person(age=20)

// Remarque :
// Vous générez en C++ un constructeur qui contient tous les attributs de la classe, et un autre par défaut.
// Les attributs non initialisés par le constructeur de EZ doivent être par défaut à NULL.
```

Traduction en C++ :

```
class Person {

private:
   string m_nom;
   string m_prenom;
   int m_age;

public:

   Person( string nom, string prenom, int age )
   :m_nom(nom),m_prenom(prenom),m_age(age){}

};


// Déclaration 1
Person p("nom", "prenom", 20);
// Déclaration 2
Person * pp=new Person("nom", "prenom", 20);
```

2. Les destructeurs

Définition

Un destructeur est ce qui détruit un objet, son rôle principal est la libération de la mémoire allouée via le constructeur, aussi ce qui n'a pas été libéré durant la vie de l'objet.

Exemples

En EZ :

```
class MaClasse

        destruct
            print "destructeur de MaClasse"
        end destruct

end class
```

Traduction en C++ :

```
class MaClasse {
public:
  ~MaClasse(){
     cout << "destructeur de MaClasse"<<endl;
  }
};
```

3. fonction __init__

Définition

__init__ est la méthode qui va être appelée automatiquement après qu'un objet ait créé. Ce n'est pas un constructeur mais c'est un initialiseur.

Syntaxe

```
__init__(arg1,arg2,..)
  // Instruction
end

 // or
__init__()
  // Instruction
end init

 // Remarque: On peut terminer par "end"  ou "end init"
```

En EZ :

```
class Person


   counter is shared integer
    nom is string
    prenom is string

__init__()
     Person.counter++
  end init

end class
```

4. Les accesseurs (Getters)

Définition

Un accesseur est une fonction membre permettant de récupérer le contenu d'une donnée protégée.

En EZ il n y a pas de notion d’encapsulation, ainsi les accesseurs n’ont pas d’utilité.

Exemple

En EZ :

```
class Person
        m_nom is string
        m_prenom is string
        m_age is integer
end class

//classe main en ez

program main

procedure main()

         p is Person(“nom1”,”nom2”,20)
        print "nom "   ,  p.m_nom, ”\n”
        print "prenom " , p.m_prenom() , “\n”
        print "age ", p.m_age(), “\n”

end procedure
```

Traduction en C++ :

```
class Person {

public:

   string m_nom;
   string m_prenom;
   int m_age;

   Person(){}
```

```
  Person( string nom, string prenom, int age)
        :m_nom(nom),m_prenom(prenom),m_age(age)
  {}


};


int main()
{
  Person p("nom1","nom2",20);

  cout<<"nom " <<p.nom<<endl;
  cout<<"prenom "<<p.prenom<<endl;
  cout<<"age "<<p.age<<endl;

}
```

5. Les mutateurs (Setters)

Définition

Un mutateur est une fonction membre permettant de modifier le contenu d'une donnée membre protégée.

En EZ il n'y a pas de notion d'encapsulation, ainsi les mutateurs n'ont pas d' utilité.

Exemples

En EZ :

```
class Person

        m_nom is string
        m_prenom is string
        m_age is integer
end class

//classe main en ez
program main

procedure main()

         p is Person
         p.m_nom=”nom1”
         p.m_prenom=”prenom1”
         p.m_age=20

end procedure
```

Traduction en C++ :

```
class Person {
public:
  string m_nom;
  string m_prenom;
  int m_age;
};

int main()
{
  Person p;

  p.m_nom="nom1";
  p.m_prenom="prenom1";
  p.m_age=20;

  return EXIT_SUCCESS;
}
```

6. La surcharge des opérateurs

Définition

La redéfinition d'un opérateur se fait en déclarant et définissant une méthode ayant pour nom operator suivi de l'opérateur.

Exemples

Exemple 1

En EZ :

```
operator== (c is C)  return  bool


end
```

Traduction en C++ :

```
bool operator==(C const & c) const{

}
```

Exemple 2

En EZ :

```
operator< (c is C)  return  bool

end
```

Traduction en C++ :

```
bool operator<(C const & c) const{
}
```

Exemple 3

En EZ :

```
operator= (c is C)  return  C


end
```

Traduction en C++ :

```
C const & operator=(C const & c){

}
```

7. La surcharge des méthodes

Il est possible de déclarer et définir plusieurs fonctions ayant le même nom, à condition que leurs arguments soient différents.

Exemple

```
//Méthode 1
function add() return integer
      v is integer
      v = 1 + 2
      return v
end function
//Méthode 2

function add(a,b are integer) return integer
      v is integer
      v = a + b
      return v
end function

//Méthode 3

function add(a,b,c are integer) return integer
      v is integer
      v = a + b + c
      return v
end function
```

8. La surcharge de la fonction d'affichage "print"

Définition

Le mot-clé print permet de faire une sortie d'affichage. Il est toutefois possible de le redéfinir dans une classe selon la sortie souhaitée.

Exemples

En EZ :

```
class Person


   nom is string
   prenom is string
   age is integer

   procedure print()
      print “nom: ” , nom,  “  prenom:”, prenom, “  age: ”, age
   end procedure

end class

//main

program main

procedure main()

    p is Person(“dupont”,”laurent”,25)
    print p

end procedure

//ouput
nom: dupont  prenom: laurent  age: 25
```

9. Les données et fonctions membres statiques

Définition

Une fonction membre déclarée static a la particularité de pouvoir être appelée sans devoir instancier la classe.
Elle ne peut utiliser que des variables et des fonctions membres static.

Exemples

Exemple 1

En EZ :

```
class MaClasse
        a is shared integer = 0
end class
```

Traduction en C++ :

```
class Exemple {
public:
   static int a;
};

// dans le fichier .cpp
int Exemple::a = 0;
```

Exemple 2

En C++ :

```
class A
{
public:
  // non static
  int var1;
  void f1() {};
  //static
  static int var2;
  static void f2() {};
};
// initialisation de la variable static
int A::var2 = 0;
int main()
{
  // non static
  A a;
  a.var1 = 1;
  a.f1();

  //static
  A::var2 = 1;
  A::f2();
}
```

En EZ :

```
class A
   // non static
    var1 is integer
    procedure f1()
    end procedure

   //static
 var2 is shared integer = 0

 shared procedure f2()
 end procedure

end class

// function main
procedure main()

   // non static
    a is A
    a.var1 = 1
    a.f1()
   //static
    A.var2 = 1;
    A.f2();

end procedure
```