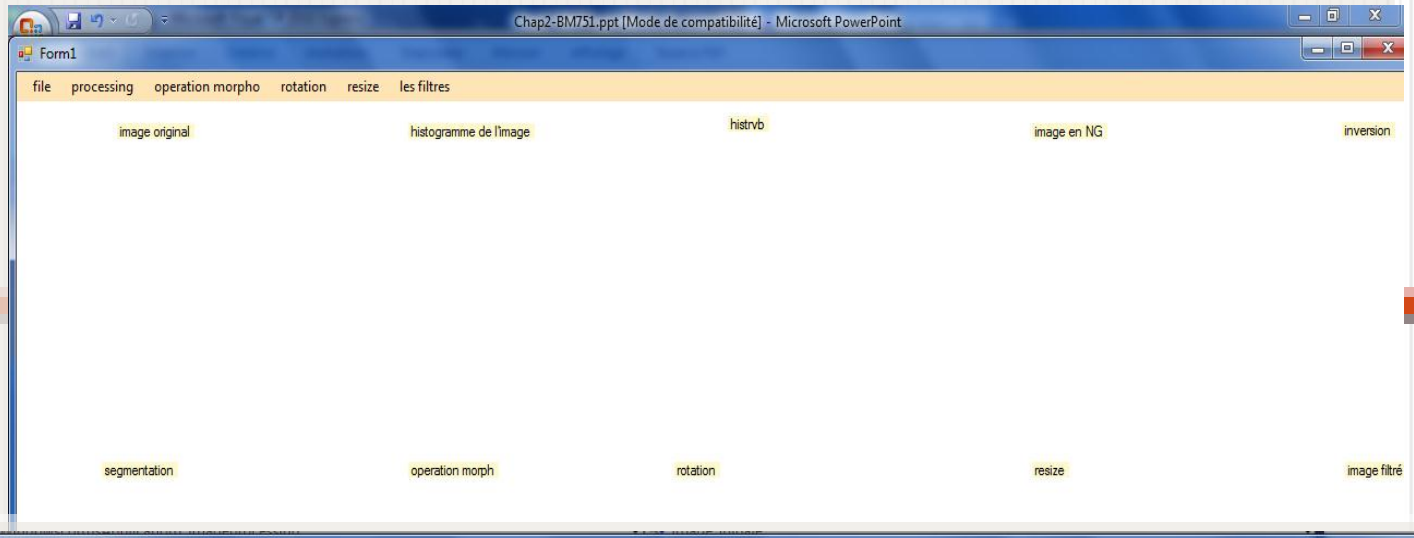
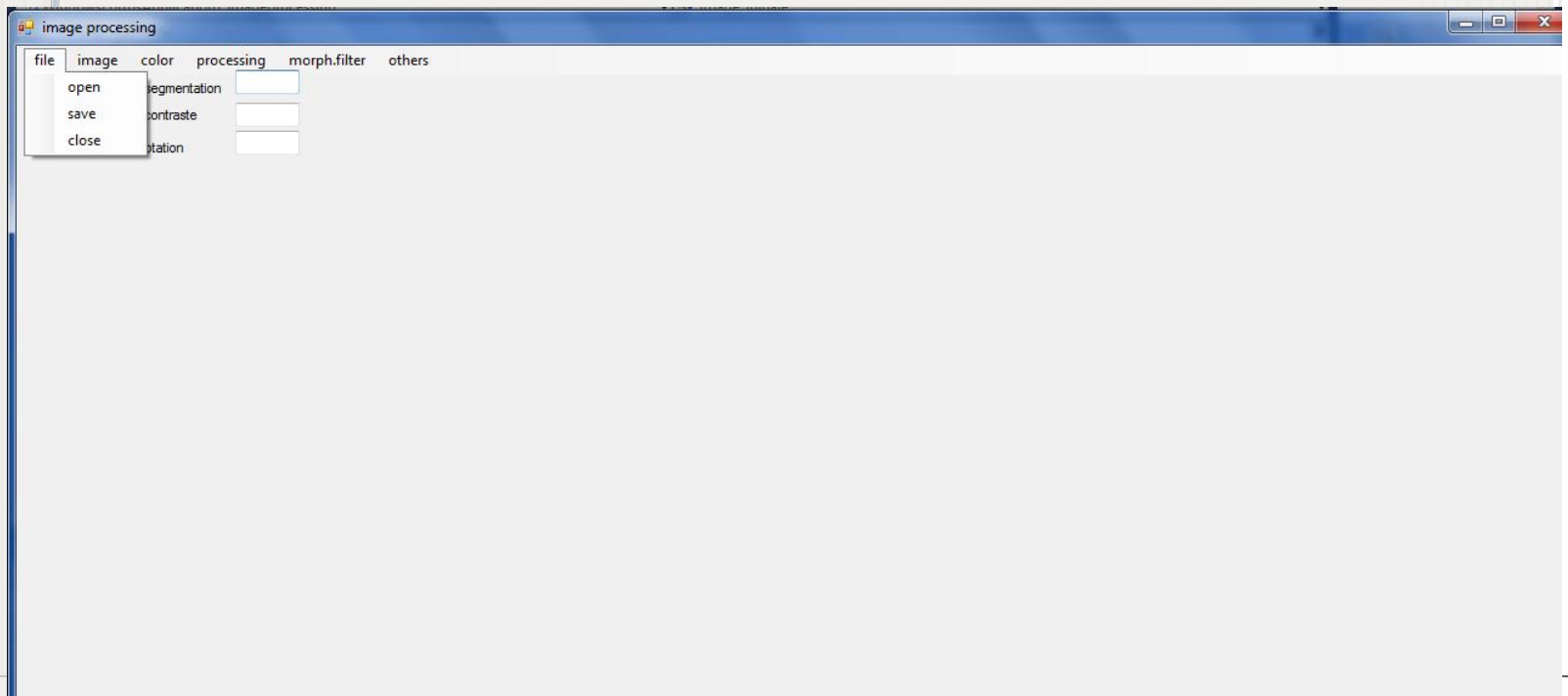
BM751 : Chapter 2

Informatics Tools

General Syntax of C#

Introduction





Introduction to C# programming

In this chapter, we will discover what is C#, its history.

We will see how to create small computer applications,!!!!!

C# is a programming language, created by Microsoft in 2002.

Introduction:

- How to create programs?
- C# compared to C and C++ very simple.
- C# was compiled, we find it in the * .exe form, and it is also possible to create libraries in *.dll form.
- The development of a program in C# must be written in *.cs file, and we will find the **Program.cs file**, which is the file was generated by Visual C# during the creation of the project

- C# allows to develop of applications : executables.
- It is possible to create libraries.

An Overview of Programming in "C# , C++"

Console application

WriteLine method, allows to write a character string to the console.

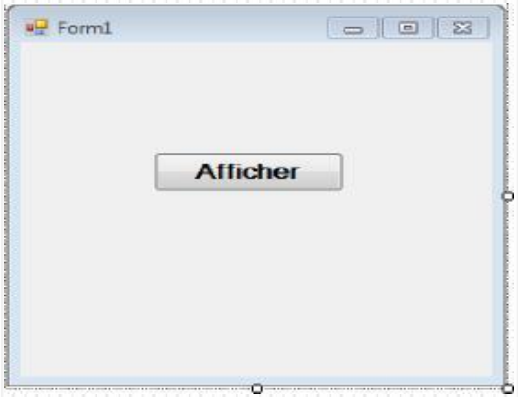
```
static void Main(string[] args)
{
    Console.WriteLine("Hello World !!");
}
```

Comments
It is possible to put comments, starting a comment with /* and ending with */;
Or using //

Exemple 1: Example 1: Creation of an application based on a "MessageBox"

- Add a button in the development interface.
- Select the button and choose to the properties window.
- In the "text" property, change "button1 by "Show".
- Note: It is important to distinguish between the "Name" property and the "Text" property.
- Name of this button will be used in programming.

Example 1: Creation of an application based on a "MessageBox"




```
        private void button1_Click(object sender, EventArgs e)
        {
            MessageBox.Show("Bismi Allah");
        }
    }
}
```

Example 2: Display a message in a "TextBox"

Double click on the button and write the following code: (attention to the name of the TextBox or TextBox1 component).

```
private void button1_Click(object sender, EventArgs e)
{

    // MessageBox.Show("Bismi allah");
    textBox.Text = "hamd lah";
}
```

Example 2: Display a message in a "TextBox"

Insert a second button and name it "Delete". Double-click on the button and write the following code:

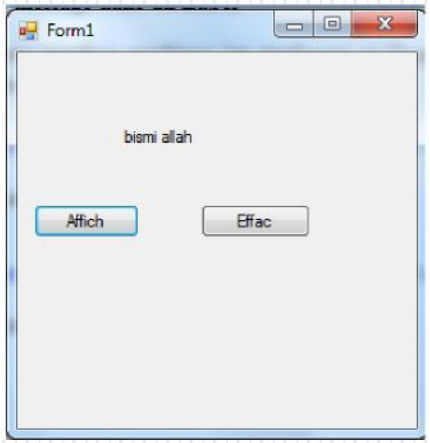
```
private void button2_Click(object sender, EventArgs e)
{
    textBox1.Clear();
}
```

Example 3: Display a message in a Label

Insert a Label in your Graphical interface, then write the following code :

```
private void button1_Click(object sender, EventArgs e)
{
    label1.Text = "bismi allah";
}

private void button2_Click(object sender, EventArgs e)
{
    label1.ResetText();
}
```



Variables

Like all programming languages, C# stores data using variables :

```
int age;
    age = 30
```

We can replace the previous code with :

```
int age = 30;
```

We can at any time seek the value contained in the age variable, for example :

```
        int age = 30;
Console.WriteLine(age);
```

It is possible to modify the value of the age variable at any time :

```
int age = 30;
Console . WriteLine (age ); // affiche 30
 age = 20;
 Console . WriteLine (age ); // affiche 20
```

For example, we can store a character string using the string.

```
string prenom = " Mohammed ";
```

Or a boolean (which represents a true or false value) with :

```
bool estVrai = true ;
```

Type	Description
`byte`	Entier de 0 à 255
`short`	Entier de -32768 à 32767
`int`	Entier de -2147483648 à 2147483647
`long`	Entier de -9223372036854775808 à 9223372036854775807
`float`	Nombre simple précision de -3,402823e38 à 3,402823e38
`double`	Nombre double précision de -1,79769313486232e308 à 1,79769313486232e308
`decimal`	Nombre décimal convenant particulièrement aux calculs financiers (en raison de ses nombres significatifs après la virgule)
`char`	Représente un caractère
`string`	Une chaîne de caractère
`bool`	Une valeur booléenne (vrai ou faux)

- **Math.Sqrt** method is used to calculate the square root of number.

- Example:

```
int i = 100000 ;
short s = i;
```

We will obtain a compilation error.

Example :

However, explain the following code:

```
double prix = 125.55;
 int valeur = prix ;
  //solution : compilation error
```

and :

```
 double prix = 125.55;
 int valeur = (int ) prix ;
// solution = 125
```

Assignments, operations,

We can do these operations :

```
int age1 = 20;
int age2 = 30;
int average = ( age1 + age2 ) / 2;
```

The + operator can be used to add strings :

```
string codePostal = " 13000 ";
string city = " Tlemcen ";
string address = codePostal + " " +city ;
Console . WriteLine (address);
```

The ++ operator allows to increment by 1,

or the -- operator allows to realize a decrement by 1.

```
Int   age = 20;
age = age + 10; // age contains 30 ( addition )
age = age ++; // age contains 31 ( increment  by 1)
age = age --; // age contains 30 (decrement by 1)
age += 10; // age = age + 10 (age contains 40)
age /= 2; // age = age / 2 => ( age contains 20)
```

Exercise: compare results

```
int average = 5.0 / 2.0;
Console .WriteLine (average );
```

and

```
double average = 5.0 / 2.0;
Console .WriteLine (average )
```

Solution : 1st case impossible (result=2), 2nd case it is possible (result=2.5)

Special characters, for example the following code:

```
string Name = " My  name is \" MMMMM\"";
Console . WriteLine ( Name);
```

We can use the special character "\n" --→ go to :

```
Console . WriteLine (" word1 \n word2\n word3\ word4\n\n\n");
```

To read data directly from the keyboard using the following instruction :

```
int a =
Convert.ToInt32(Console.ReadLine());
```

The following code will display the dash (-) :

```
Console . WriteLine (" Choose to do :");
Console . WriteLine ("\t - Water the plants");
Console . WriteLine ("\t - Wash the car ");
```

Opérateur	Description
==	Égalité
!=	Différence
>	Supérieur à
<	Inférieur à
>=	Supérieur ou égal
<=	Inférieur ou égal
&&	ET logique
||	OU logique

The conditional statements

if loop:

The if loop allows to execute code if a condition is true. For example :

```
double note = 12;
 if ( note >= 10)
 Console .WriteLine (" successful ");
```

The conditional statements :

If loop:

Or

```
decimal Bankaccount = 300;
if (Bankaccount >= 0)
 Console . WriteLine (" positive");
 if ( compteEnBanque < 0)
 Console . WriteLine (" negative");
```

It is possible to use multiple conditions using the combination of else and if.

```
if ( civilite == " Mrs")
 Console .WriteLine (" Vous êtes une femme ");
 else if ( civilite == " Miss")
 Console .WriteLine (" Miss");
 else if ( civilite == "M.")
 ……….;
 else
……….;
```

Example;

In the following example, we display the welcome message only if the login “Image” AND the password “test” are correct.

```
string login = " Image ";
 string password = " test ";
 if ( login == " Image" && password == " test ")
 Console .WriteLine (" Welcome");
else
Console .WriteLine (" Login incorrect ");
```

Other logical operators exist, we have in particular the operator | | which corresponds to the logical “OR”:

Exercise ; If you are “Mrs” or “Miss” “ a woman”; otherwise “a man”;

```
if ( MMMM == " Mrs " || MMMM== " Miss ")
 Console . WriteLine (" woman ");
 else
 Console . WriteLine (" man");
```

Switch statement

The switch statement can be used when a variable takes many values..

```
string civilite = "M.";
 switch ( civilite )
 {
 case "M." :
 Console . WriteLine (" hello  X1");
 break ;
 case " Mrs ":
 Console . WriteLine (" hello  X2 ");
 break ;
 case " Miss ":
 Console . WriteLine (" hello X3");
 break ; }
```

switch begins by evaluating the variable. With the case word, we list the different possible cases for the variable and we execute the corresponding instructions up to the break word "means that we exit the switch".

Create a method

The aim of the method is to organize the code in order to avoid repeating the same program:

I define a method called "AffichageBienvenue" by Instruction :

static void AffichageBienvenue ()

Void word: Signify that the method returns nothing (), indicate that the method has no parameters.

Static word : Indicat that the method is still available and ready to use.

It is possible to use several parameters in the method. For example :

```
static void Hello (string name, int age,)
{
Console .WriteLine (" Hello " + name);
Console .WriteLine ("You have " + age + " year");
}
```

The Hello method takes as parameters a character string "name" and an integer "age".

We can call this method from the Main() method:

```
static void Main ( string [] args )
{
Hello (" Mohamed ", 30);
Hello ("Samir", 20);

}
```

Result
Hello Mohammad
You have 30 years old
Hello Samir
You have 20 years old

Arrays:

```
string [] Days = new string [] { "Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday," Sunday " };
        Console.WriteLine(Days[1]);
```

We define here an array of character strings which contains 7 character strings: the days of the week. The new operator is used to create the array.

Exemple

```
Console .WriteLine (Days [3]); // to Display the day of Thursday
Console .WriteLine (Days [0]); // to Display the day of Monday
Console .WriteLine (Days [10 ]); // error because index does not
exist
```

We use Days.Length to find out the Length of the array.

It is possible to display all the elements of the array:

```
string [] Days = new string [] { "Monday”, “Tuesday”, “Wednesday”,
“Thursday”, “Friday”, “Saturday,” Sunday " };
        for ( int i = 0; i < Days.Length ; i++)
 {
 Console . WriteLine (Days [i]);
 }
```

We can use the Array.Sort() method to sort :

```
Array . Sort ( jours );
```

The List :

```
List <int > numbers = new List <int >(); / / list creation
 numbers.Add (8); / / The list contains the number 8
 numbers .Add (9); / / The list contains the number 8, 9
 numbers .Add (4); / / The list contains the number 8, 9, 4
  numbers. RemoveAt (1); / / The list contains the number 8, 4
```

Use the .NET framework

The .NET framework is a toolbox that contains many methods to build all applications. We will need to use elements of the .NET framework to create our applications.

Using Instruction

We will search for the current date. To do this, we will use the instruction :

```
Console . WriteLine ( DateTime . Now);
```

We search our application to display the Now property of the DateTime object..

or we should write :

```
System . Console . WriteLine ( System . DateTime .Now);
```

However, it is possible to search in the System namespace directly :

```
using System
```

Add references

To declare complex numbers, you must use the following reference :

System.Numerics.dll

Then, we can use the following code to read the complex number :

```
Complex c = Complex . One;
Console . WriteLine (c);
Console . WriteLine (" Partie ré elle : " + c. Real );
Console . WriteLine (" Partie imaginaire : " + c. Imaginary );
```

Exercise: :

The aim of this application is to create a small application to display a message based on the user's name and the time of day:

– "Hello X" for the 9 a.m. --- 6 p.m. Mondays, Tuesdays, Wednesdays, Thursdays and Fridays;

– "Good evening X" for the 6 p.m. --- 9 a.m. Mondays, Tuesdays, Wednesdays, Thursdays;

– "weekend X"

Solution

```
static void Main ( string [] args )
{
 if ( DateTime .Now . DayOfWeek == DayOfWeek . Saturday ||
DateTime .Now . DayOfWeek == DayOfWeek . Sunday )
  {
   DisplayWeekEnd ();
  }
  else
  {
```

```
 if ( DateTime .Now . DayOfWeek == DayOfWeek . Monday &&
DateTime .Now . Hour < 9)
 {
??????????????????????????
 }
 else
 {
 if ( DateTime .Now . Hour >= 9 && DateTime .Now. Hour <18)
 {
??????????????????????????
}
```

```
if ( DateTime .Now . DayOfWeek == DayOfWeek . Friday
&&  DateTime .Now . Hour >= 18)
 {

 DisplayWeekEnd(); / / it's Friday evening
```

For loop

The for loop is used to repeat a piece of code as long as a condition is true.

```
int compteur ;
for ( compteur = 0; compteur < 50; compteur ++)
 {
 Console . WriteLine (" Bonjour C#");
 }
```

In general, we use the for loop to browse an array.

```
string [] Days = new string [] { "Monday”, “Tuesday”, “Wednesday”,
“Thursday”, “Friday”, “Saturday,” Sunday " };
int ii ;
 for ( ii = 0; ii < 7; ii ++)
 {
 Console . WriteLine (Days [ ii ]);
 }
```

Exercise :

From a following vector : [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]

Show next list : [9 7 5 3 1].

Solution :

```
int intTemp = Convert.ToInt32(Console.ReadLine());
int [ ] numbers = new int [ ] { 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 };
  // for ( int i = 0; i <9; i++) // classic case
 for ( int i = 9; i > 0; i -= 2)
       {
 Console . WriteLine ( numbers [i]);
 }
```

The foreach loop

C# has a special operator: foreach;

```
string [] Days = new string [] { "Monday", "Tuesday", "Wednesday",
"Thursday", "Friday", "Saturday," Sunday " };
foreach ( string Day in Days)
 {
 Console .WriteLine (Day );
 }
```

This loop will browse all the elements of the "days" array. At each iteration, the loop will create a string day which will contain the current element of the array.

While loop

The while loop is generally less used than for loop. The while loop will repeat a condition until it is not verified.

Example :

```
int i = 0;
while (i < 50)
{
Console . WriteLine (" Hello C#");
i++;
}
```

Exercise

Calculate the sum of integers, for example: calculate the sum of numbers from 1 to 10, i.e. 1 + 2 + 3 + 4 + 5 + 6 + 7 + 8 +9 + 10.

Solution

```
int  result  = 0;
for (int i = 1; i <= 5; i++)
        {
              result +=i;
        }
        Console.WriteLine(result);
```

The break and continue instructions

Example: It is possible to exit a loop using the break instruction. Execution of the program continues with the instructions located after the loop.

```
int i = 0;
while ( true )
{
if (i >= 50)
{
break ; }
Console . WriteLine (" Hello C#");
i++; }
```

This code produces an infinite loop. The break word allows to exit the loop as soon as the value of i will achieve 50.

It is possible to go to the next iteration using of the continue word, the following example::

```
for ( int i = 0; i < 20; i++)
{
if (i % 2 == 0)
{
continue ;
}
Console .WriteLine (i);
}
```

The % operator is called “modulo” and allows to obtain the remainder of the division. As soon as an **even number** is found, the program go to the next iteration by use the continue word.

Exercise ;

Calculate the sum of integers " 1 + 2 + 3 + 4 + 5 + 6 + 7 + 8 + 9 + 10", use a method contains parameters "1, and 10".

Console .WriteLine (CalculateSumNumber (1, 10));

Solution :

```
static CalculateSumNumber (int borneMin , int borneMax )
{
int result = 0;
for ( int i = borneMin ; i <= borneMax ; i++)
{
result += i;
}
return result ; }
```

```
Console . WriteLine (CalculateSumNumber(1, 10));
```

Read keyboard in console

Today, there are many communication interfaces that can be used on a computer (keyboard, mouse, . . .). We will see how to use them in a console application.

Read a sentence

When a user inputs information through their keyboard, the input process concludes upon the user's confirmation with the Enter key.

```
Console . WriteLine (" Read a sentence");
string saisie = Console . ReadLine ();
Console . WriteLine ("You wrote : " + saisie );
```

Read a character

Enter just one character, for this we will use the following method :

Console.ReadKey().

Example :

```
Console . WriteLine («  Do you want to continue (Y/N) ?");
 if ( saisie .Key == ConsoleKey .Y)
 {
Console . WriteLine (“ Continue. …");…….
……..……
```

Create my first object

Objects had characteristics; it's about properties, doing actions...

We can give a name to this class: Car.

Visual C# Express generates the following code:

```
using System ;
using System . Collections . Generic ;
using System . Linq ;
 namespace MaPremiereApplication
 {
 Class Car
{
.....................
}
```

We find the word class followed by the name of our Car class and bracket in order to delimit it.
Note also that this class is part of the MyFirstApplication namespace..

A class is a way to represent an object.

A class was generated by Visual C# Express, ---- class Program.

```
class Program
    {
static void Main ( string [ ] args )
 {
……..
 }
}
```

The special method **Main()** is the application input.

The **class word** allows to define a class, -------- the structure of an object.

Methods

We have created our Car object, and our object is able to perform actions (i.e. honk).

```
class Car
{
void honk ()
{
Console .WriteLine (" Message !");
} }
```

- The public keyword indicate that our method is accessible from other classes. Without this keyword, It is impossible to use this method by other objects.
- We can use the private keyword to make the method inaccessible.
- The void keyword means that the method returns nothing.
- The static keyword indicates that the method is always available.

C#, an object-oriented language

What Does Object-Oriented Language Mean?

What Does Object Mean?

Object-oriented language (OOL) is a high-level computer programming language that implements objects and their associated procedures within the programming context to create software programs.

C#, an object-oriented language

For example: If we take a table, a car, etc…

– They have properties (the table has four legs, blue color, etc.).

– These objects can achieve actions (Car can drive, honk, etc.).

– They can interact between them (Driver object, Car object, Steering wheel object, etc.).

The object is characterized by its properties, its actions etc.

We can have several car objects.

Encapsulation :

- Encapsulation is a way to restrict the direct access to some components of an object, so users cannot access state values for all of the variables of a particular object. Encapsulation can be used to hide both data members and data functions or methods associated with an instantiated class or object.

- Encapsulation protects the object's data and its internal functioning

Inheritance (object-oriented programming)

in object-oriented programming, inheritance is the mechanism of basing an object or class upon another object (prototype-based inheritance) or class (class-based inheritance), retaining similar implementation.

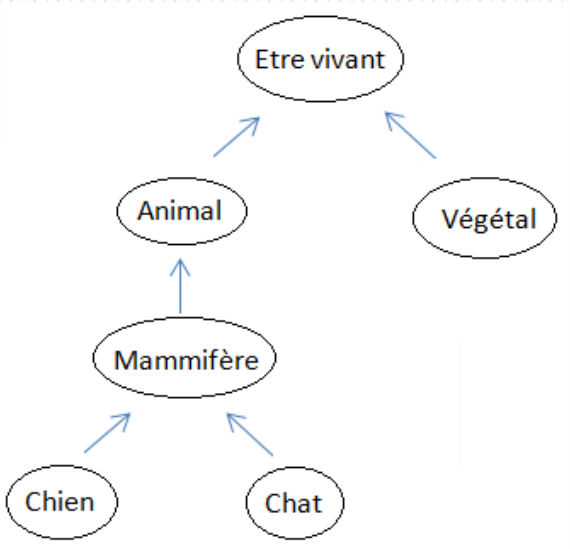
Also defined as deriving new classes (sub classes) from existing ones such as base class. In most class-based object-oriented languages like C++, an object created through inheritance, a "child object", acquires all the properties and behaviors of the "parent object", with the exception of: constructors, destructors, overloaded operators and friend functions of the base class. Inheritance allows programmers to create classes that are built upon existing classes, the relationships of objects or classes through inheritance give rise to a directed acyclic graph.

Inheritance

- An inheritance relationship : An object called “Parent ” can transmit certain of its characteristics to another object called “child”.

- We say that the child object is a specialization of the Parent object.

The following diagram, each arrow represents inheritance relationship between objects.



Polymorphism

- Polymorphism is a feature of object-oriented programming languages that allows a specific routine to use variables of different types at different times.
- Polymorphism is the ability for an object to perform the same action with different types of participants.
- For example: our Application object realize Action 1, Action 2, …. action n.

Thank You