المحاضرة الأولى

The important features of OOP related to C#:

- **Encapsulation**
- **Polymorphism**
- **Inheritance**
- **Interface**
- **Data Hiding**

Encapsulation:

The wrapping up (تغليف) of data and function into a single unit (class).

Polymorphism:

Means the ability to take more than one form, the behavior depends upon the types of data used in the operation.

Inheritance:

Any class may inherit (also known as derived) from another (parent or base class), which means that it will have all the members that the class it inherits from has.

Inheritance allows you to extend or create more specific classes from a single, more generic base class.

Interface :

Is a collection of public methods and properties that are grouped together to encapsulate specific functionality.

Data Hiding:

The insulation of the data from direct access by the program.

Structure of program in the C# language:

```
using System;
using
System.Collections.Gene
ric;
using System.Text;
namespace
ConsoleApplication1
{
```

Variables

Bool	int	Ushort
Byte	Long	String
Char	Sbyte	
Decimal	Short	
Double	Uint	
Float	Ulong	

EX:

Console .writeline (9>10)

EX:

Decimal x=2.5 ; Warrning

Decimal x=2.5m; or 2.5M;

Decimal y=2;

Float F=1.5; Warrning

Float F=1.5f or 1.5F;

KEYWORD

Byte

Decimal

Event

if

EX:

```
Int @if
For (@if=0; @if<10 ;  @if++)
     Concole.Writeline(@if);
```

EX:

```
static void Main(string[] args)
        {
            int myinteger;
            string mystring;
            myinteger = 17;
            mystring = "\"myinteger\"is";
            Console .WriteLine ("{0}
{1}",mystring,myinteger);

        }
```

"myinteger"is 17

Variable Naming:

Myname
VAR
_test
} ok

9A
namespace
It is
} not ok

EX:

int xsize,ysize;

int age=23;

int xsize=24, ysize=78;

int xsize=23, ysize;

Expressions:

- Unary
- Binary

Mathematical Operators:

+	va1=var2 + var3
-	va1=var2 - var3
*	va1=var2 * var3
/	va1=var2 / var3
%	va1=var2 % var3

+	var1= + var2
-	var1= - var2

For string:

+	var1=var2 + var3

++	var1=++var2
--	var1= -- var2
++	var1=var2++
--	var1=var2--

EX:

```
static void Main(string[] args)
        {
            int var1, var2 = 5, var3 = 6;
            var1 = var2++ * --var3;
            Console.WriteLine("{0}", var1);
        }
```

EX:

Var1=++6 ; error

Int var2=6;

Var1=++var2; } **ok**

Assignment Operations:

=	var1=var2
+=	var1+=var2
-=	var1 -=var2
*=	var1 *=var2
/=	var1 /=var2
%=	var1 %=var2

Boolean Logic:

Relational operation

==	var1==var2==var3
!=	var1=var2!=var3
<	var1=var2< var3
>	var1=var2 > var3
<=	var1=var2<= var3
>=	var1=var2>= var3

Ex:

```
static void Main(string[] args)
        {
            int myval = 11;
            bool ISLESSTHAN10;
            ISLESSTHAN10 = myval < 10;
            Console.WriteLine(ISLESSTHAN10);
        }
```

EX:

```
static void Main(string[] args)
        {
            string mystring = "RAMI";
            bool ISRAMI;
            ISRAMI = mystring == "RAMI";
            Console.WriteLine(ISRAMI);
        }
```

!	var1 !=var2
&- &&	var1=var2 & var3

	- ||	var1=var2 | var3
^	var1=var2 ^ var3

Bitwise Operations

Ex:

```
static void Main(string[] args)
        {
            int result, op1, op2;
            op1 = 4;
            op2 = 5;
            result = op1 & op2;
            Console.WriteLine(result );
        }
```

Bitwise Shift Operators:

>>	var1=var2>>var3
<<	var1=var2<<var3

EX:

```
static void Main(string[] args)
        {
            int var1, var2 = 10;
            var1 =var2 >> 1;
            Console.WriteLine(var1 );
        }
```

5

```
static void Main(string[] args)
        {
            int var1, var2 = 10;
            var1 =var2 >> 2;
            Console.WriteLine(var1 );
        }
```

2