

---

## Lập trình C#

### Mục tiêu:

Kết thúc nội dung thực hành này bạn nắm được các khái niệm:

*Classes and Methods*

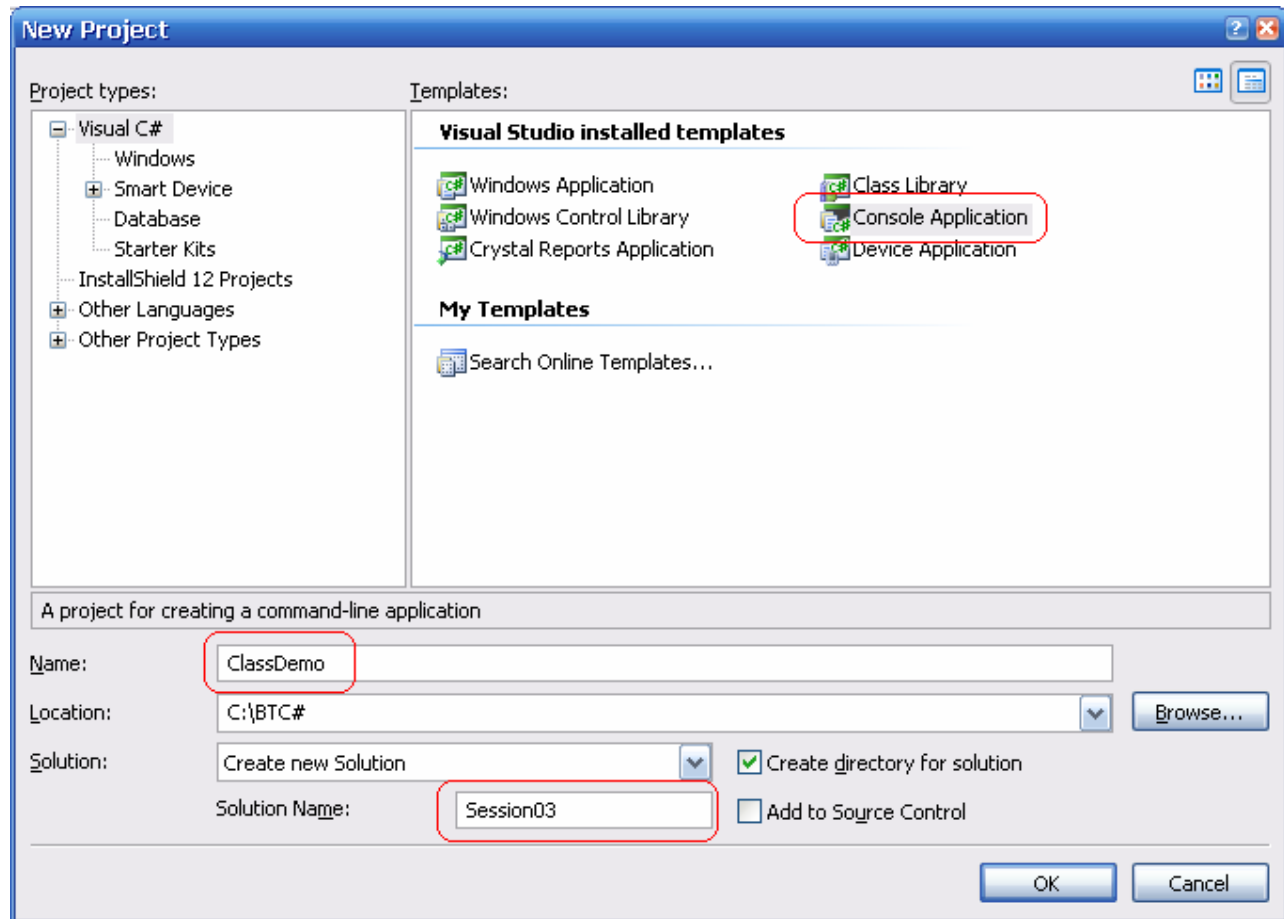
*Inheritance and Polymorphism*

## Phần I: Thực hành theo các bước – 45 phút

### Bài tập 1.1: Creating a class

Step 1: Open Visual Studio

Step 2: Select the menu File->New->Project to create console based project named 'ClassDemo' and Solution named Session03 as following



Step 3: Rename the class file 'program.cs' to 'ClassDemo.cs'

Step 4: Replace code in 'ClassDemo.cs' with given code

```
using System;
class Car
{
    // declare the fields
    public string make;
    public string
    model; public
    string color;
    public int
    yearBuilt;

    // define the
    methods public void
    Start()
    {
        System.Console.WriteLine(model + " started");
    }

    public void Stop()
    {
        System.Console.WriteLine(model + " stopped");
    }
}
class ClassDemo
{
    public static void Main()
    {
        // declare a Car object reference named myCar
        Car myCar;

        // create a Car object, and assign its address to myCar
        System.Console.WriteLine("Creating a Car object and assigning " +
            "its memory location to myCar");
        myCar = new Car();

        // assign values to the Car object's fields using myCar
        myCar.make = "Toyota";
        myCar.model = "MR2";
        myCar.color = "black";
        myCar.yearBuilt =
        1995;

        // display the field values using myCar
        System.Console.WriteLine("myCar details:");
        System.Console.WriteLine("myCar.make = " + myCar.make);
        System.Console.WriteLine("myCar.model = " + myCar.model);
        System.Console.WriteLine("myCar.color = " + myCar.color);
        System.Console.WriteLine("myCar.yearBuilt = " + myCar.yearBuilt);
        // call the methods using
```

```
myCar myCar.Start();
myCar.Stop();
// declare another Car object reference and
// create another Car object
System.Console.WriteLine("Creating another Car object and " +
    "assigning its memory location to redPorsche");
Car redPorsche = new Car();
redPorsche.make = "Porsche";
redPorsche.model =
    "Boxster";
redPorsche.color = "red";
redPorsche.yearBuilt = 2000;
System.Console.WriteLine("redPorsche is a " + redPorsche.model);

// change the object referenced by the myCar object reference
// to the object referenced by redPorsche
System.Console.WriteLine("Assigning redPorsche to
myCar"); myCar = redPorsche;
System.Console.WriteLine("myCar details:");
System.Console.WriteLine("myCar.make = " + myCar.make);
System.Console.WriteLine("myCar.model = " + myCar.model);
System.Console.WriteLine("myCar.color = " + myCar.color);
System.Console.WriteLine("myCar.yearBuilt = " + myCar.yearBuilt);
// assign null to myCar (myCar will no object) longer reference an

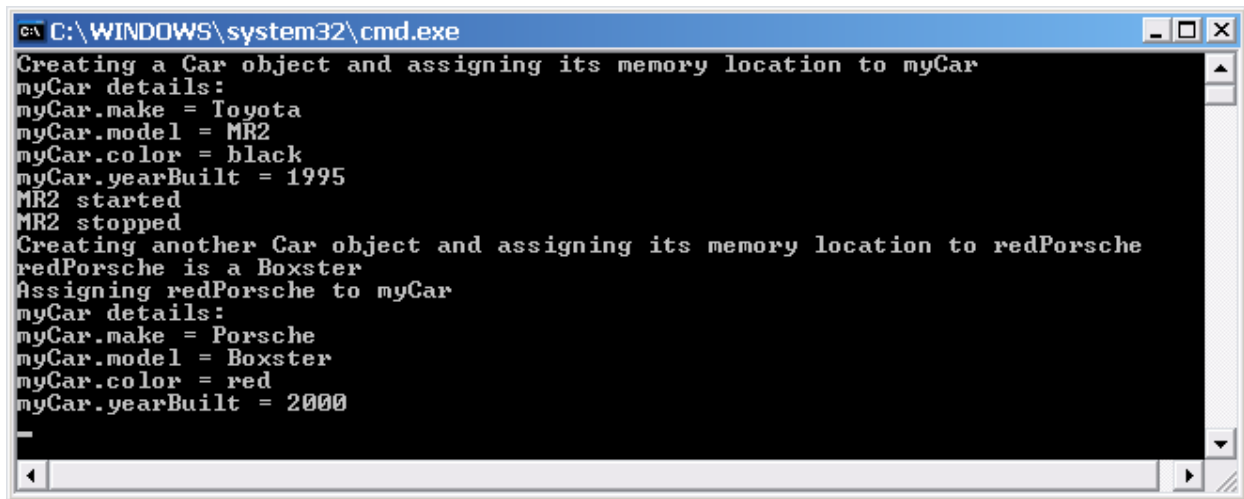
myCar = null; Console.ReadLine();
}
```

Step 5: Select menu File -> Save to save the file

Step 6: Select Build -> Build ClassDemo option to build the project Step

7: Select Debug -> Start without Debugging to execute the program The

output of the program as following



```
C:\WINDOWS\system32\cmd.exe
Creating a Car object and assigning its memory location to myCar
myCar details:
myCar.make = Toyota
myCar.model = MR2
myCar.color = black
myCar.yearBuilt = 1995
MR2 started
MR2 stopped
Creating another Car object and assigning its memory location to redPorsche
redPorsche is a Boxster
Assigning redPorsche to myCar
myCar details:
myCar.make = Porsche
myCar.model = Boxster
myCar.color = red
myCar.yearBuilt = 2000
```

## Bài tập 1.2: Nested Class

- Step 1: Add a console based project 'NestedClass' to the solution
- Step 2: Right click on project NestedClass -> set as Startup project
- Step 3: Rename the class file 'Program.cs' to 'NestedClass.cs'
- Step 4: Replace the code in 'NestedClass.cs' with the given code

```
using System;
class Car
{
    // declare the Engine
    class public class Engine
    {
        // declare the Engine
        fields public int
        cylinders;
        public int horsepower;

        // define the Engine
        method public void
        Start()
        {
            System.Console.WriteLine("Engine started");
        }
    }
}
```

```
    }

    // declare the Car
    fields public string
    make;
    public Engine engine;    // Car has an Engine

    // define the Car
    method public void
    Start()
    {
        engine.Start();
    }
}

class ClassDemo2
{
    public static void Main()
    {
        // declare a Car object reference named myCar
        System.Console.WriteLine("Creating a Car
        object"); Car myCar = new Car();
        myCar.make = "Toyota";

        // Car objects have an Engine object
        System.Console.WriteLine("Creating an Engine
        object"); myCar.engine = new Car.Engine();
        myCar.engine.cylinders = 4;
        myCar.engine.horsepower = 180;

        // display the values for the Car and Engine object fields
        System.Console.WriteLine("myCar.make = " + myCar.make);
        System.Console.WriteLine("myCar.engine.cylinders = " +
        myCar.engine.cylinders);
        System.Console.WriteLine("myCar.engine.horsepower = " +
        myCar.engine.horsepower);

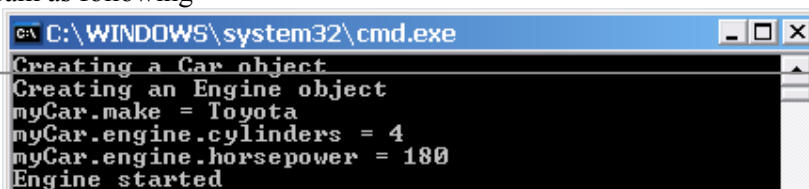
        // call the Car object's Start()
        method myCar.Start();
        Console.ReadLine();
    }
}
```

Step 5: Select menu File -> Save to save the file

Step 6: Select Build -> Build 'NestedClass' option to build the project

Step 7: Select Debug -> Start without Debugging to execute the program

The output of program as following



```
C:\WINDOWS\system32\cmd.exe
Creating a Car object
Creating an Engine object
myCar.make = Toyota
myCar.engine.cylinders = 4
myCar.engine.horsepower = 180
Engine started
```

## Bài tập 1.3: Static member

Step 1: Add a console based project 'StaticMember' to the solution

Step 2: Right click on project StaticMember -> set as Startup project

Step 3: Rename the class file 'Program.cs' to 'StaticMember.cs'

Step 4: Replace the code in 'StaticMember.cs' with the given code

```
using System;
// declare a Cat
class class Cat
{
    // a private static member to keep
    // track of how many Cat objects have
    // been created
    private static int instances = 0;
    private int weight;
    private String name;

    // cat constructor
    // increments the count of Cats
    public Cat(String name, int
weight)
    {
        instances++;
        this.name = name;
        this.weight =
weight;
    }

    // Static method to retrieve
    // the current number of Cats
    public static void
HowManyCats()
```

```
        {
            Console.WriteLine("{0} cats
                               adopted", instances);
        }
        public void TellWeight()
        {
            Console.WriteLine("{0} is {1} pounds",
                               name, weight);
        }
    }

class ClassDemo3
{
    public void Run()
    {
        Cat.HowManyCats();
        Cat frisky = new Cat("Frisky",
                               5); frisky.TellWeight();
        Cat.HowManyCats();
        Cat whiskers = new Cat("Whisky",
                               7); whiskers.TellWeight();
        Cat.HowManyCats();
    }

    public static void Main()
    {
        ClassDemo3 t = new ClassDemo3();
        t.Run();

        Console.ReadLine();
    }
}
```

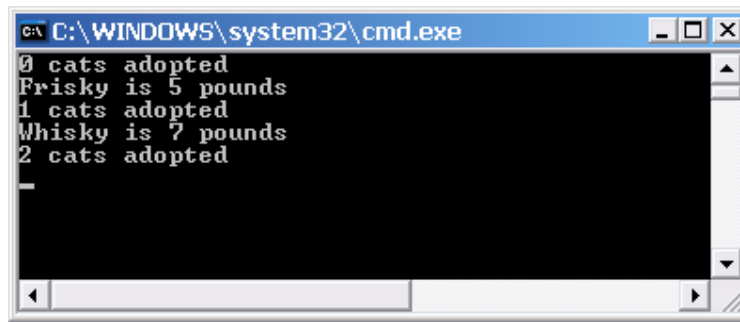
Step 5: Select menu File -> Save to save the file

Step 6: Select Build -> Build 'StaticMember' option to build the project

Step 7: Select Debug -> Start without Debugging to execute the program

The output of program as following





```
C:\WINDOWS\system32\cmd.exe
0 cats adopted
Frisky is 5 pounds
1 cats adopted
Whisky is 7 pounds
2 cats adopted
-

```

## Bài tập 1.4: Using “virtual” and “override” keyword

Step 1: Add a console based project ‘OverrideMethod’ to the solution

Step 2: Right click on project Polymorphism -> set as Startup project

Step 3: Rename the class file ‘Program.cs’ to ‘OverrideMethod.cs’ Step

4: Replace the code in ‘OverrideMethod.cs’ with the given code

```
public virtual void Display()
{
    Console.WriteLine("Class B's Display Method");
}
public class C : B
{
    public override void Display()
    {
        Console.WriteLine("Class C's Display Method");
    }
}

public class ContainedClass
{
    int MyInt = 0;
}

public class D : C
{
    public ContainedClass MyClass = new ContainedClass();
    public override void Display()
    {
        Console.WriteLine("Class D's Display Method");
    }
}
```

```
class ClassDemol
{
    static void Main(string[] args)
    {
        B MyB = new
        D(); D MyD =
        new D();

        //Both result in in D's instance of Display being //called
        MyB.Display(
        );
        MyD.Display(
        );

        Console.ReadLine();
    }
}
```

```
using System;
public class B
{
```

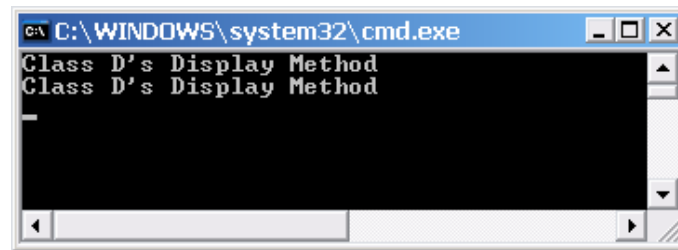
---

Step 5: Select menu File -> Save to save the file

Step 6: Select Build -> Build 'OverrideMethod' option to build the project

Step 7: Select Debug -> Start without Debugging to execute the program

The output of program as following



```
C:\WINDOWS\system32\cmd.exe
Class D's Display Method
Class D's Display Method
```

The image shows a screenshot of a Windows command prompt window. The title bar indicates the path 'C:\WINDOWS\system32\cmd.exe'. The command prompt displays two lines of output: 'Class D's Display Method' on the first line and 'Class D's Display Method' on the second line. The text is in a monospaced font, and the window has a standard Windows XP-style border with minimize, maximize, and close buttons.

## Bài tập 1.5: Polymorphism

Step 1: Add a console based project ‘**Polymorphism**’ to the solution

Step 2: Right click on project Polymorphism -> set as Startup project

Step 3: Rename the class file ‘Program.cs’ to ‘**Polymorphism.cs**’ Step

4: Replace the code in ‘**Polymorphism.cs**’ with the given code

```
using System;
class Window
{
    // constructor takes two integers to
    // fix location on the console
    public Window(int top, int
left)
    {
        this.top = top;
        this.left = left;
    }

    // simulates drawing the window
    public virtual void
DrawWindow()
    {
        Console.WriteLine("Window: drawing Window at {0}, {1}",
            top, left);
    }

    // these members are protected and thus visible
    // to derived class methods. We'll examine this
    // later in the
    chapter protected int
top; protected int
left;
}

// ListBox derives from Window
class ListBox : Window
{
    // constructor adds a
    parameter public ListBox(int
top,
        int left, string contents):base(top, left)    // call base
constructor
    {

        listBoxContents = contents;
    }
}
```

```
// an overridden version (note keyword) because in the
// derived method we change the
behavior public override void
DrawWindow()
{
    base.DrawWindow(); // invoke the base method
    Console.WriteLine("Writing string to the listbox: {0}",
        listBoxContents);
}

private string listBoxContents; // new member variable
}

class Button : Window
{
    public Button(int top, int left): base(top, left)
    {
    }

    // an overridden version (note keyword) because in the
    // derived method we change the
    behavior public override void
    DrawWindow()
    {
        Console.WriteLine("Drawing a button at {0}, {1}\n",top, left);
    }
}

class Polymorphism
{
    static void Main(string[] args)
    {
        Window win = new Window(1, 2);
        ListBox lb = new ListBox(3, 4, "Stand alone list box");
        Button b = new Button(5, 6);
        win.DrawWindow();
        lb.DrawWindow();
        b.DrawWindow();

        Window[] winArray = new Window[3];
        winArray[0] = new Window(1, 2);
        winArray[1] = new ListBox(3, 4, "List box in array");
        winArray[2] = new Button(5, 6);

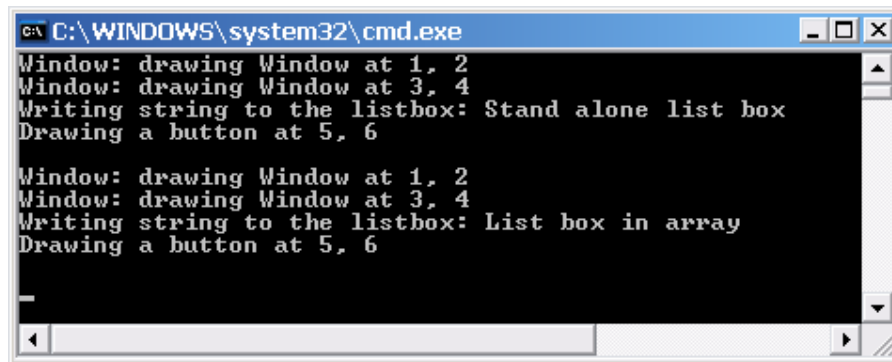
        for (int i = 0; i < 3; i++)
        {
            winArray[i].DrawWindow();
        }
        Console.ReadLine();
    }
}
```

```
}
```

Step 5: Select menu File -> Save to save the file

Step 6: Select Build -> Build 'OverrideMethod' option to build the project

Step 7: Select Debug -> Start without Debugging to execute the program The output of program as following



```
C:\WINDOWS\system32\cmd.exe
Window: drawing Window at 1, 2
Window: drawing Window at 3, 4
Writing string to the listbox: Stand alone list box
Drawing a button at 5, 6

Window: drawing Window at 1, 2
Window: drawing Window at 3, 4
Writing string to the listbox: List box in array
Drawing a button at 5, 6
```

## Phần II: Tự thực hành – 60 phút

### Bài tập 2.1: Thiết kế lớp Atom

Thiết kế và viết mã một lớp có tên là Atom chứa thông tin về một nguyên tử. Đặt định nghĩa lớp của bạn trong tệp có tên Atom.cs. Bao gồm các chức năng thành viên sau trong thiết kế của bạn:

`boolean accept ()` - lời nhắc và chấp nhận từ đầu vào chuẩn

- một số nguyên chứa số nguyên tử,
- một chuỗi giữ biểu tượng nguyên tử,
- một chuỗi chứa tên đầy đủ của nguyên tử và
- một giá trị dấu phẩy động giữ trọng lượng nguyên tử.

Nếu bất kỳ đầu vào nào không hợp lệ, hàm của bạn sẽ từ chối đầu vào đó và yêu cầu dữ liệu mới.

`void display ()` - hiển thị thông tin nguyên tử trên đầu ra tiêu chuẩn.

Thiết kế và viết mã một chương trình chính chấp nhận thông tin cho tối đa 10 nguyên tố nguyên tử và hiển thị thông tin nguyên tử ở dạng bảng.

Đầu ra chương trình có thể trông giống như sau:

```
Atomic Information
=====
Enter atomic number : 3
Enter symbol : Li
Enter full name : lithium
Enter atomic weight : 6.941

Enter atomic number : 20
Enter symbol : Ca
Enter full name : calcium
Enter atomic weight : 40.078

Enter atomic number : 30
Enter symbol : Zn
Enter full name : zinc
Enter atomic weight : 65.409

Enter atomic number : 0

No Sym Name
Weight
```

```
-----  
3  L  lithium          6.941  
  i  
2  C  calcium         40.078  
0  a  
3  Z  zinc            65.409  
0  n
```



## Bài tập 2.2: Thiết kế lớp Employee

Viết một lớp **Employee** để ghi lại các thuộc tính và hành vi sau đây cho một Nhân viên

khai báo các biến sau

- string firstName
- string lastName
- string address
- long sin
- double salary

Tạo **constructor** để khởi tạo tất cả các biến thành viên từ các tham số truyền vào

Override phương thức **ToString** method in thông tin nhân viên ở định dạng dễ nhìn

Định nghĩa một phương thức để tính thưởng ( salary x percentage trong đó percentage được đưa ra dưới dạng tham số)

Viết một chương trình Test để kiểm tra tất cả các hành vi của lớp **Employee**.

## Bài tập 2.3: Giải phương trình theo lập trình hướng đối tượng

Xây dựng chương trình giải phương trình (bậc nhất, bậc hai) trên cơ sở lập trình hướng đối tượng.

- Lớp là **Phương trình, Số**. Các thuộc tính và phương thức học viên tự định nghĩa
- Nhập được số hữu tỷ, số phức, đa dạng hóa hiển thị, bắt lỗi mã nguồn chặt, chương trình thân thiện, mã nguồn trong sáng,...)
- Sử dụng mã nguồn làm rõ các khái niệm public, private, protected

## Bài tập 2.4: Tính diện tích và chu vi của một hình theo LTHĐT

Viết chương trình tính chu vi và diện tích của các hình sau: đường tròn, hình chữ nhật, hình thang, tam giác.

- Lớp là hình (Shape) và các lớp kế thừa là Đường tròn, hình chữ nhật, hình thang, tam giác
- Có Enum là các loại hình
- Có phương thức là tính chu vi, tính diện tích
- Có các thuộc tính tương ứng với các hình

## **Bài tập 2.5: Xây dựng chương trình mô phỏng tham chiếu tham trị**

Xây dựng chương trình mô phỏng được:

- Kiểu dữ liệu tham chiếu, tham trị
- Chuyển đổi kiểu dữ liệu tham chiếu sang tham trị và ngược lại.
- Trường hợp truyền tham số tham trị mà lại thay đổi giá trị
- Trường hợp truyền tham trị mà không bị thay đổi giá trị thì làm như thế nào