# Hướng đối tượng với C#

# Các khái niệm chính yếu 01

- **Encapsulation** - Bao bọc
  - *Thao tác với dữ liệu thông qua phương thức*
- **Inheritance** - Kế thừa
  - *Tái sử dụng lại khả năng của lớp khác*
- **Polymorphism** - Đa hình
  - *Nhiều lớp cùng chữ kí hàm nhưng xử lí cụ thể khác nhau*
- **Abstraction** - Tính trừu tượng
  - *Che giấu một số thông tin cụ thể nhất định*

# Các khái niệm chính yếu 02

- Class & Object, UML Class diagram
- Override & Overloading
- Lớp trừu tượng
- Giao diện

# Class - Lớp

Lớp là **bản thiết kế** (blueprint) để tạo ra đối tượng

```
class Employee {
    /* Attributes */
    public string FirstName;     // Họ
    public string LastName;      // Tên
    public string Tel;           // Số điện thoại
}
```

# Object - Đối tượng

Đối tượng là một **thể hiện** cụ thể (instance) tạo ra từ lớp

```
static void Main() {
  Employee alice; // Khai báo một nhân viên Alice, chưa được cấp phát bộ nhớ
  alice = new Employee();

  // Thao tác với các thuộc tính của alice
  alice.FirstName = "Alice";
  alice.LastName = "Maive";
  alice.Tel = "0909111254";

  // Khai báo và cấp phát bộ nhớ luôn
  Employee bob = new Employee() {
      FirstName = "Bob",
      LastName = "Tayson",
      Tel = "0911273812"
  };
}
```

Có bao nhiêu thể hiện của lớp Employee được tạo ra trong hàm main?

# Hàm hủy - Destructor

Không cần thiết vì C# có **Garbage Collector** (bộ thu gom rác) tự động

# Encapsulation - Getter & Setter

**Tránh** thao tác **trực tiếp** với thuộc tính (attribute)

```
class Employee {
    /* Attributes */
    public string FirstName;     // Họ
    public string LastName;      // Tên
    public string Tel;           // Số điện thoại
}
```

```
class Employee {
    /* Attributes / Backup field */
    private string _firstName;     // Họ
    private string _lastName;      // Tên
    private string _tel;           // Số điện thoại

    /* Properties */
    public string FirstName {
        get {
            return _firstName;
        }
        set {
            _firstName = value;
        }
    }
```

# Rút gọn encapsulation

```
class Employee {
    /* Attributes / Backup field */
    private string _firstName;     // Họ
    private string _lastName;      // Tên
    private string _tel;           // Số điện thoại

    /* Properties */
    public string FirstName {
        get {
            return _firstName;
        }
        set {
            _firstName = value;
        }
    }
```

```
class Employee {
    public string FirstName {get; set;}
    public string LastName {get; set;}
    public string Tel {get; set;}
}
```

# Tại sao phải bao bọc với getter và setter?

1. **Tiện thêm logic xử lí sau này**
   - ❏ Ví dụ: tài khoản 20.000, trừ 30.000 sẽ không được
   - ❏ Cô lập lỗi, tiện bảo trì
2. **Lập trình giao diện: property mới databinding được!**

# Thuộc tính phái sinh - Derivative property

```
class Employee {
    public string FirstName {get; set;}
    public string LastName {get; set;}
    public string Tel {get; set;}

    public string FullName {
        get {
            return FirstName + " " + LastName;
        }
    }
}
```

# Sử dụng lambda expression

```
class Employee {
    public string FirstName {get; set;}
    public string LastName {get; set;}
    public string Tel {get; set;}

    public string FullName => $"{FirstName} {LastName}";
}
```

# Hàm tĩnh - Static function

```
class Point2D {
    public int X { get; set;}
    public int Y { get; set; }

    public static float CalcDistance(Point2D a, Point2D b) {
        int dx = a.X - b.X;
        int dy = a.Y - b.Y;

        return Math.Sqrt(dx * dx + dy * dy);
    }
}
```

# Thuộc tính tĩnh

```
class Point2D {
    public static int InstanceCount {get; set;} = 0;

    public int X { get; set;}
    public int Y { get; set; }

    public static float CalcDistance(Point2D a, Point2D b) {
        int dx = a.X - b.X;
        int dy = a.Y - b.Y;

        return (float) Math.Sqrt(dx * dx + dy * dy);
    }

    public Point2D() {
        InstanceCount++;
    }
}
```

```
static void Main() {
    Point2D a = new Point2D() { X = 1, Y = 1};
    Console.WriteLine($"InstanceCount: {Point2D.InstanceCount}");

    Point2D b = new Point2D() { X = 2, Y = 2};
    Console.WriteLine($"InstanceCount: {Point2D.InstanceCount}");

    float distance = Point2D.CalcDistance(a, b);
    Console.WriteLine($"Distance from a to b is: {distance}");
```

# Kế thừa

```
abstract class Vehicle  // Base class (parent)
{
    public string Brand {get; set;}

    public void Run() {
        Console.WriteLine("Running");
    }

    public abstract void Honk();
}

class Car : Vehicle  // Derived class (child)
{
    public override void Honk() {
        Console.WriteLine("Tuut Tut");
    }
}
```

```
static void Main() {
    // Create a myCar object
    Car myCar = new Car() {Brand = "Mercedes"};

    // Call the honk() method (From the Vehicle class) on the myCar object
    mycar.Run();
    myCar.Honk();

    // Display the value of the brand field (from the Vehicle class) and the
    Console.WriteLine(myCar.Brand);
```

# Cho phép lớp con nạp đè hành động của cha

```
abstract class Vehicle  // Base class (parent)
{
    public string Brand {get; set;}

    public virtual void Run() { // Hành xử mặc định
        Console.WriteLine("Running");
    }

    public abstract void Honk();
}
```

```
class Car : Vehicle  // Derived class (child)
{
    public override void Honk() {
        Console.WriteLine("Tuut Tut");
    }

    public override void Run() { // Con có cách hành xử khác
        Console.WriteLine("Car is starting the engine.");
    }
}
```

# Đa xạ - Thêm vào lớp xe mui trần

```
class Car : Vehicle  // Derived class (child)
{
    public override void Honk() {
        Console.WriteLine("Tuut Tut");
    }

    public override void Run() { // Con có cách hành xử khác
        Console.WriteLine("Car is starting the engine.");
    }
}

class Cabriolet: Vehicle {
    public override void Honk() {
        Console.WriteLine("Bruh Bruh");
    }

    public override void Run() { // Con có cách hành xử khác
        Console.WriteLine("Removing the hood");
        Console.WriteLine("Cabriolet is starting the engine.");
    }
}
```

```
static void Main() {
    // Create a myCar object
    Vehicle car01 = new Car()       { Brand = "Mercedes"};
    Vehicle car02 = new Cabriolet() { Brand = "Ford"};

    car01.Run();
    car02.Run();
```

# Mảng đối tượng

```
static void Main() {
    List<Vehicle> vehicles = new List<Vehicle>();

    vehicles.Add(new Car(){ Brand = "Mercedes"});
    vehicles.Add(new Cabriolet() { Brand = "Ford"});

    vehicles[0].Run();
    vehicles[1].Run();
```

# Bài tập

Thông tin một sinh viên gồm: Mã số sinh viên, Họ và tên, Điểm trung bình, Địa chỉ, Số điện thoại.

- Tự phát sinh ngẫu nhiên 10 sinh viên
- Cho biết điểm trung bình tổng thể của 10 sinh viên này
- Cho biết tên, số điện thoại của sinh viên có điểm trung bình thấp nhất và cao nhất

# Đọc thêm

https://www.w3schools.com/cs/cs_oop.asp