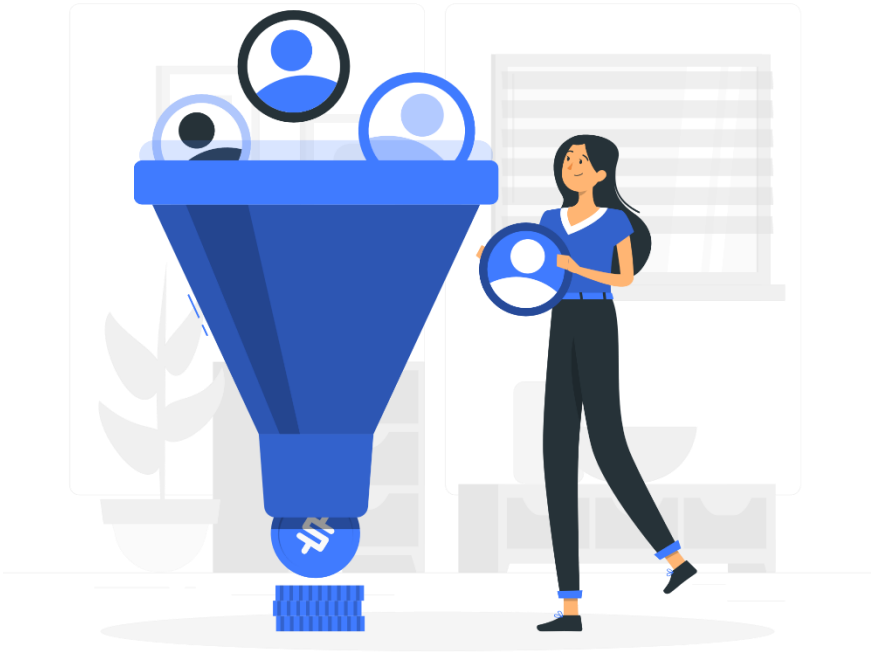
GREEN ACADEMY
IT·DESIGN·KOREAN ACADEMY
VIETNAM

Khoá học: Game Developer

Bài 03: Lớp (class) và đối tượng (object)



GREEN ACADEMY
IT·DESIGN·KOREAN ACADEMY
VIETNAM

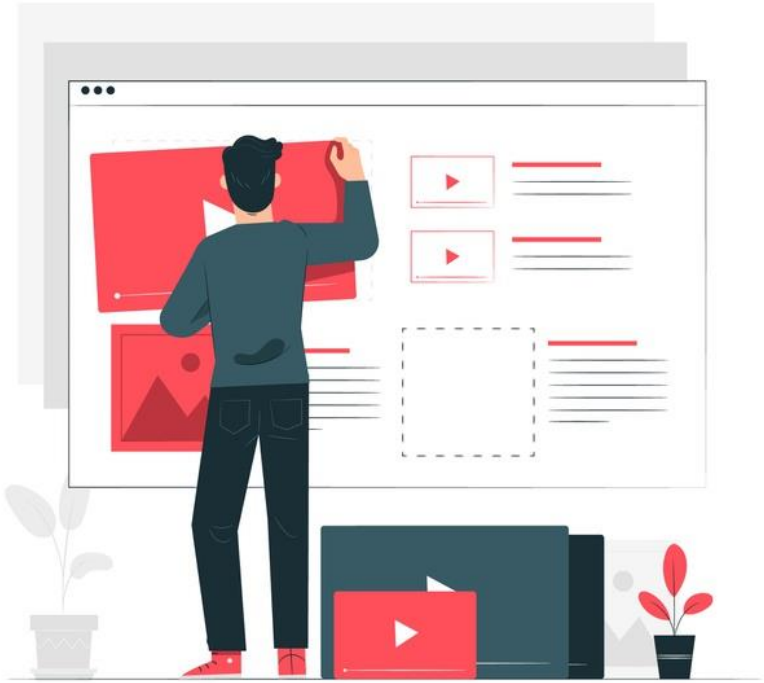
Mục tiêu bài học

Kết thúc bài học này, bạn có thể

- Hiểu rõ khái niệm đối tượng và lớp
- Mô hình hoá lớp và đối tượng
- Định nghĩa lớp và tạo đối tượng
- Định nghĩa trường, phương thức
- Hiểu cách sử dụng hàm khởi tạo
- Hiểu đặc tính che dấu
- Sử dụng các đặc tả truy xuất

GREEN ACADEMY
IT·DESIGN·KOREAN ACADEMY
VIETNAM

Nội dung



1. Khái niệm lập trình Hướng đối tượng
2. Khai báo lớp, biến thành viên và phương thức
3. Định nghĩa hàm khởi tạo
4. Tạo và sử dụng đối tượng

GREEN ACADEMY
IT·DESIGN·KOREAN ACADEMY
VIETNAM

Các phương pháp lập trình

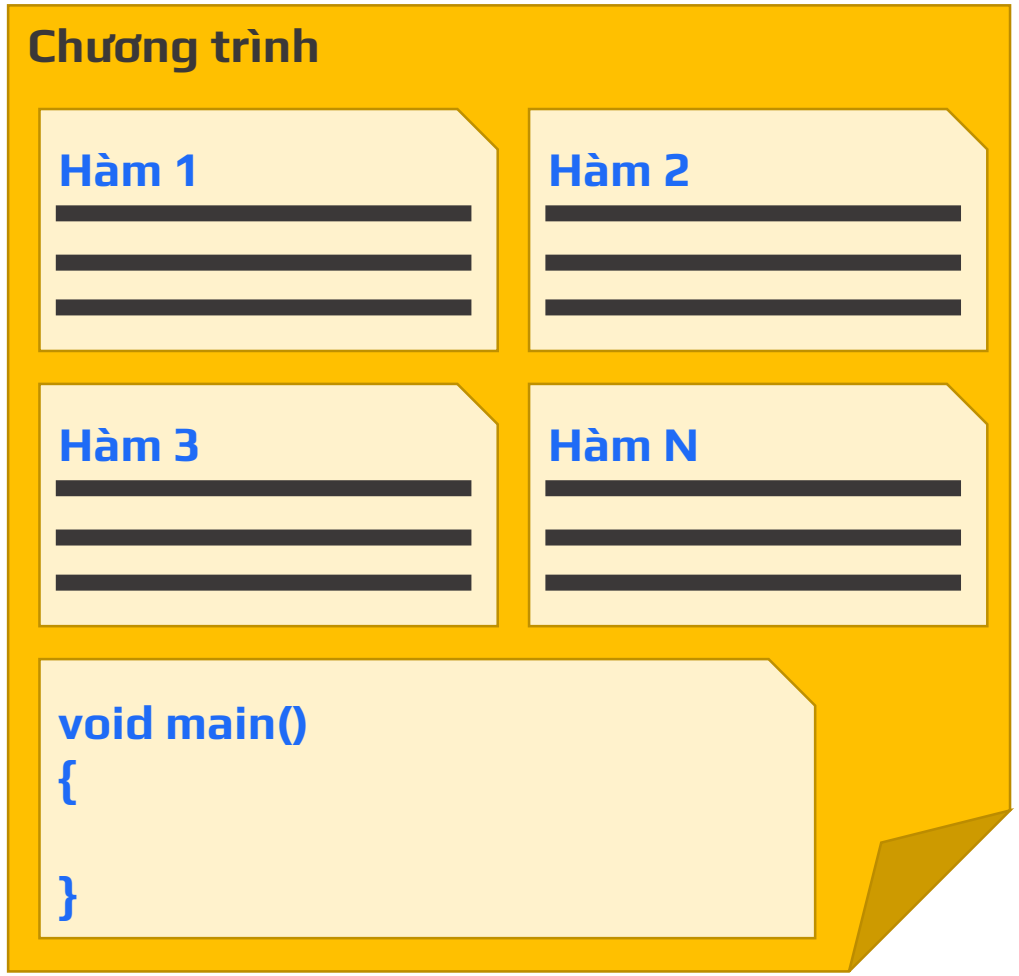
Lập trình hướng lệnh

- Chương trình là tập hợp các lệnh.
- Việc viết chương trình là xác định xem chương trình gồm những lệnh nào, thứ tự thực hiện của các lệnh ra sao.

GREEN ACADEMY
IT·DESIGN·KOREAN ACADEMY
VIETNAM

Các phương pháp lập trình

Lập trình hướng thủ tục



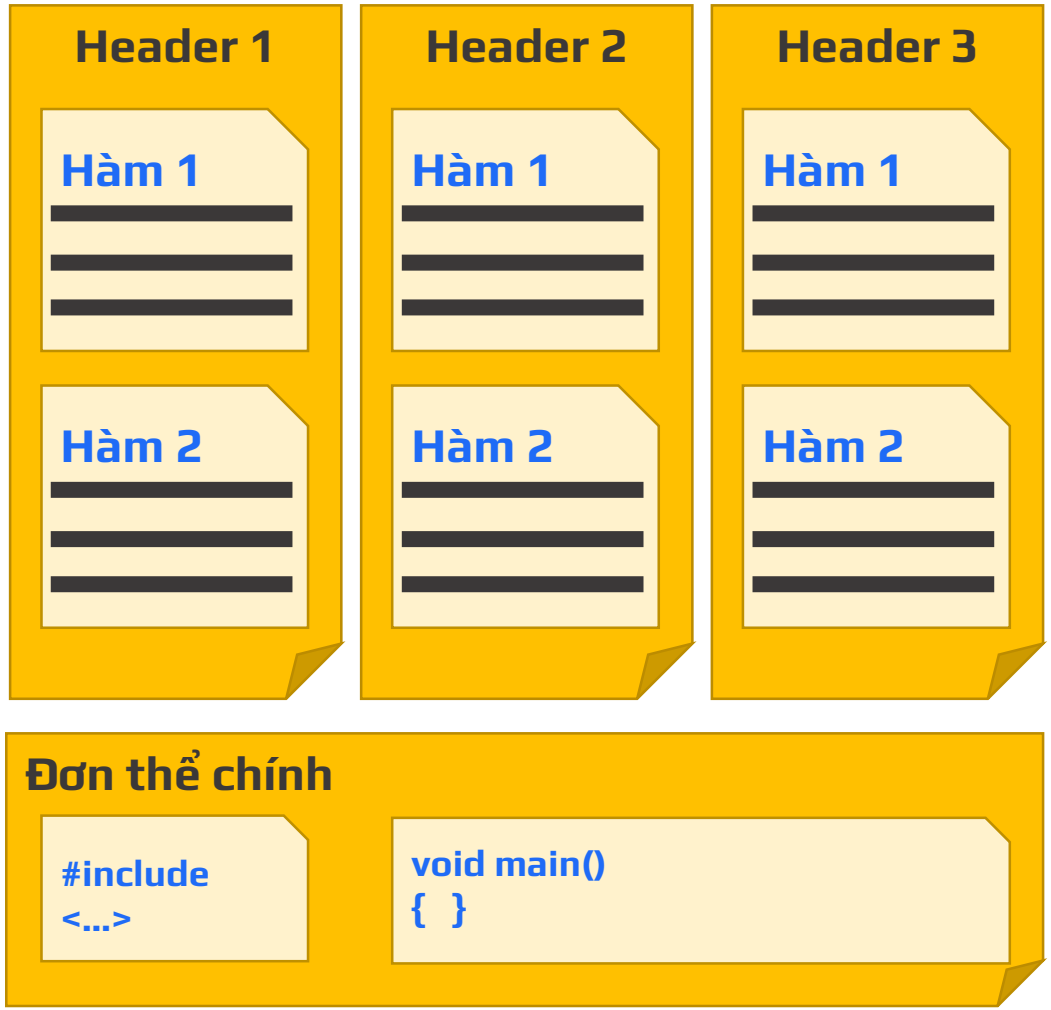
(Procedure Oriented Programming)

- Phương pháp này người ta xem chương trình là một hệ thống các thủ tục và hàm.
- Mỗi thủ tục/hàm là một dãy các lệnh được sắp thứ tự.
- Việc viết chương trình là xác định chương trình gồm các thủ tục và hàm nào, và mối quan hệ giữa các hàm.

GREEN ACADEMY
IT·DESIGN·KOREAN ACADEMY
VIETNAM

Các phương pháp lập trình

Lập trình hướng đơn thể



- Phương pháp này người ta xem chương trình là 1 hệ thống đơn thể.
- Mỗi đơn thể là 1 hệ thống các thủ tục và hàm.
- Việc viết chương trình là xác đinh xem chương trình gồm những đơn thể nào.
- Có 2 loại đơn thể:
  - Đơn thể hướng dữ liệu
  - Đơn thể hướng chức năng

GREEN ACADEMY
IT·DESIGN·KOREAN ACADEMY
VIETNAM

Đối tượng

Object

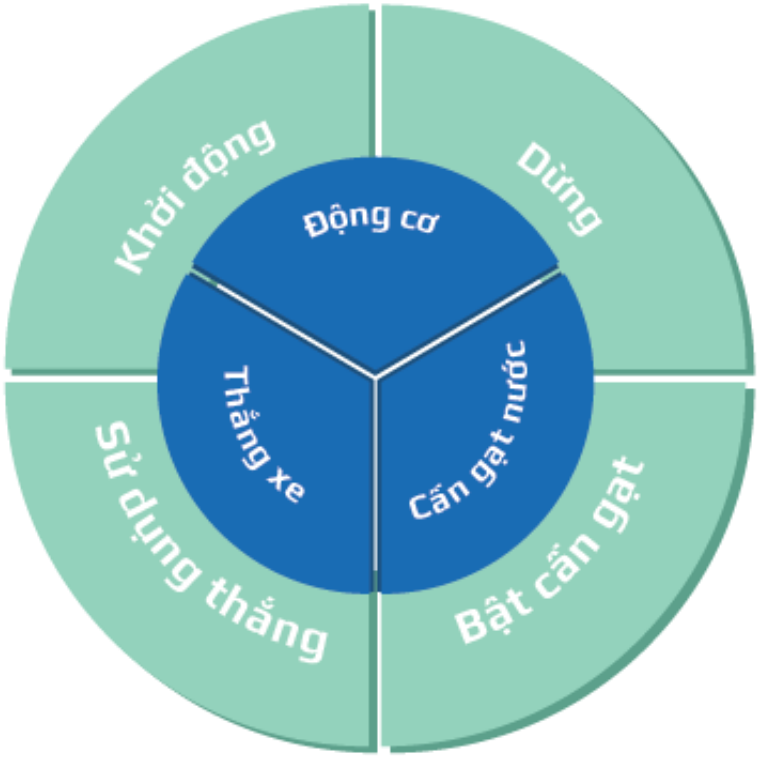
- **Công nghệ hướng đối tượng** dựa trên khái niệm Đối tượng trong thế giới thực.
- Mỗi **đối tượng** ngoài đời thực tồn tại các **thuộc tính** và **hành vi** riêng của nó.

GREEN ACADEMY
IT·DESIGN·KOREAN ACADEMY
VIETNAM

Đặc điểm và Hành vi

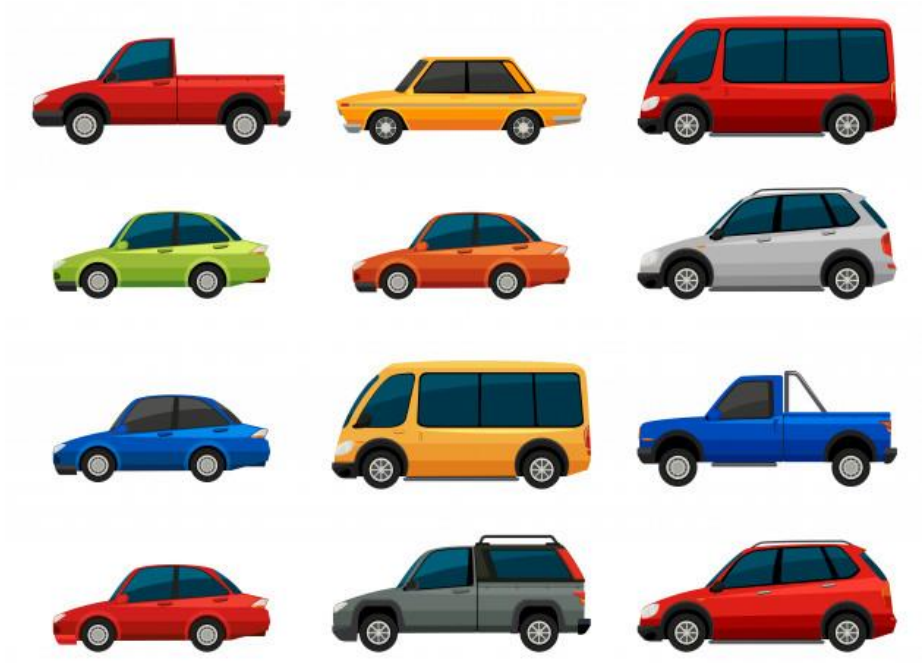
Object

- **Đặc điểm**
  - Hãng sản xuất
  - Model
  - Năm
  - Màu sắc
- **Hành vi (Chức năng)**
  - Khởi động
  - Dừng
  - Thắng
  - Bật cấn gạt nước



GREEN ACADEMY
IT·DESIGN·KOREAN ACADEMY
VIETNAM

Lớp
Class



Car

Nhóm các **Xe ô tô**

Animal

Nhóm các loài **Động vật**

GREEN ACADEMY
IT·DESIGN·KOREAN ACADEMY
VIETNAM

Lớp
Class

- **Lớp** là một khuông mẫu được sử dụng để mô tả các đối tượng cùng loại.
- Một lớp bao gồm các **thuộc tính** (trường dữ liệu) và các **phương thức** (hàm thành viên).

Objects

John
3A
study()

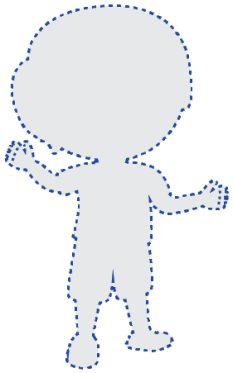
Susan
4C
play()

Tim
5A
play()

Sam
1B
doHomework()

Class

Attributes	**Methods**
name	study()
grade	play()
	doHomeWork()



GREEN ACADEMY
IT·DESIGN·KOREAN ACADEMY
VIETNAM

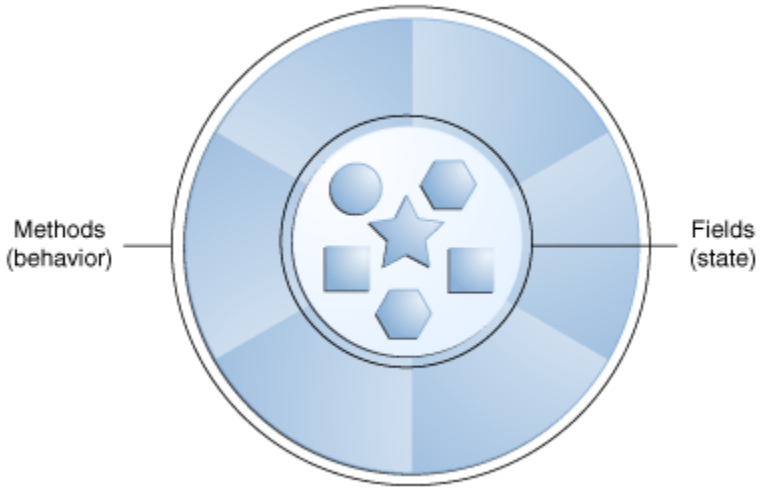
Thuộc tính và Phương thức

Fields & Methods

- **Thuộc tính**
  - Hãng sản xuất
  - Model
  - Năm
  - Màu sắc
- **Phương thức**
  - Khởi động
  - Dừng
  - Thắng
  - Bật cần gạt nước

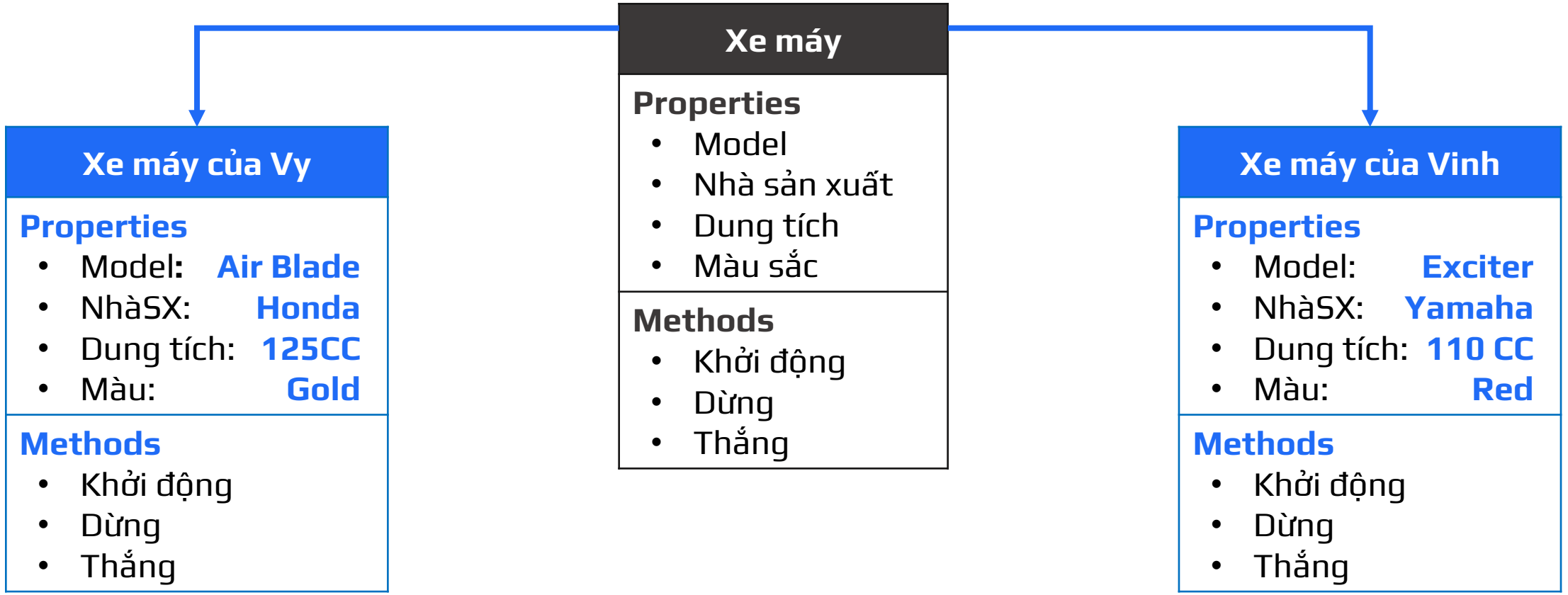
Danh từ
(noun)

Động từ
(verb)



GREEN ACADEMY
IT·DESIGN·KOREAN ACADEMY
VIETNAM

Mô hình lớp và Đối tượng



GREEN ACADEMY
IT·DESIGN·KOREAN ACADEMY
VIETNAM

Định nghĩa Class

Class definition

Khai báo các trường
(Properties)

Khai báo các phương thức
(Methods)

```
public class Student {
   //Properties
   public string name;
   public string grade;

   //Methods
   public void whoAmI() {
      Console.WriteLine("What's your name: ");
      name = Console.ReadLine();
      Console.WriteLine("Which grade are you studying: ");
      grade = Console.ReadLine();
   }
   public void introduceMyself() {
      Console.WriteLine("My name is " + name + "I am studying in grade " +
      grade);
   }
}
```

Lớp **Student** có **2 thuộc tính (fields):** *name, grade*, và **2 phương thức (methods)**: *whoAmI(), introduceMyself()*

GREEN ACADEMY
IT·DESIGN·KOREAN ACADEMY
VIETNAM

Tạo đối tượng

Using class to create an object

```
public static void main(String[] args) {
  Student std = new Student();
  std.whoAmI();
  std.introduceMyself();
}
```

- Toán tử **new** được sử dụng để tạo đối tượng.
- Biến **std** chứa tham chiếu tới đối tượng.
- Sử dụng **dấu chấm (.)** để truy xuất các thành viên của lớp (fields và methods)

GREEN ACADEMY
IT·DESIGN·KOREAN ACADEMY
VIETNAM

Định nghĩa phương thức

Methods definition

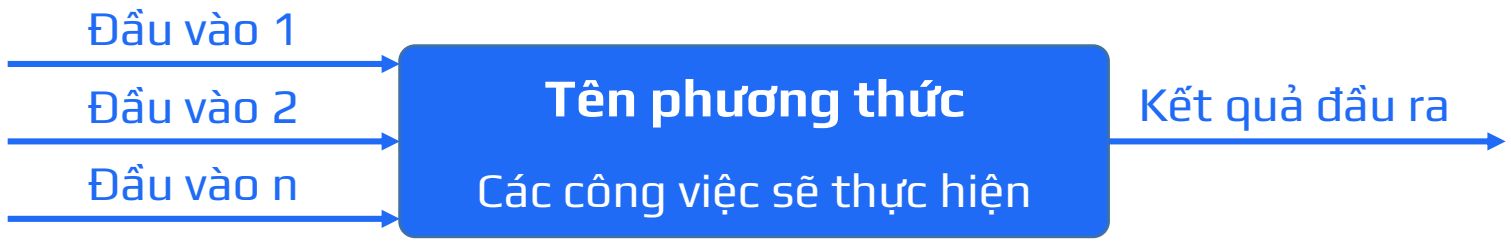
- Phương thức là một hàm chứa đoạn mã thực hiện một công việc cụ thể nào đó.
- Phương thức có thể không/có một/nhiều tham số.
- Phương thức có thể có/không có kiểu trả về.

```
<kiểu trả về> <tên phương thức> ([danh sách tham số])
{
    //Thân phương thức chứa khối câu lệnh thực thi
    [return <giá trị>;]
}
```

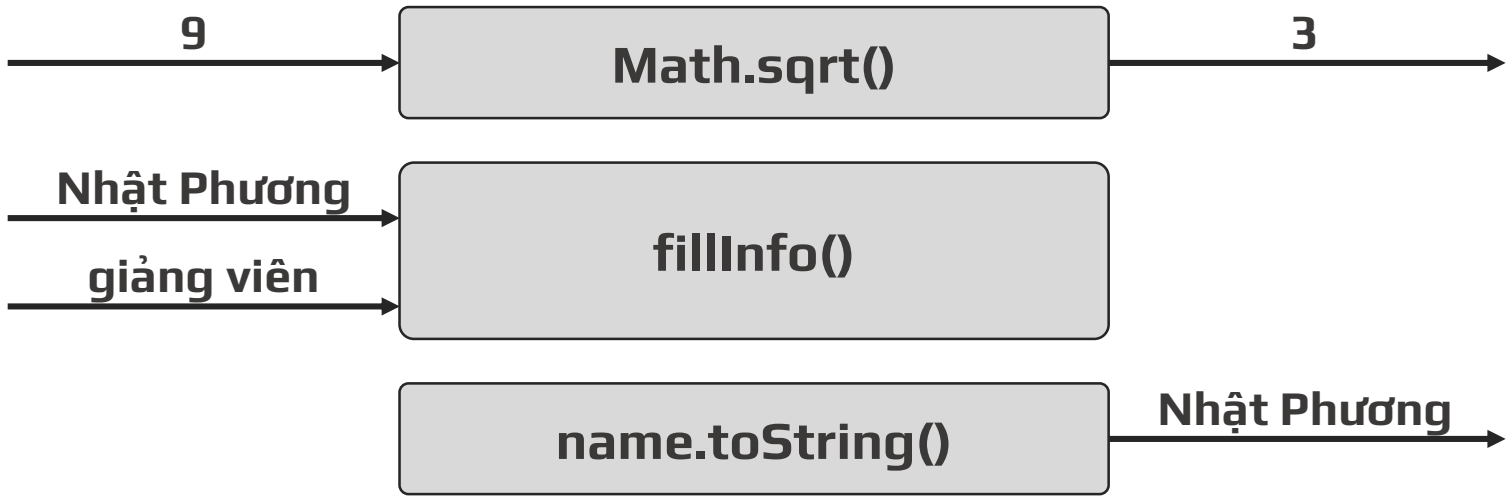
GREEN ACADEMY
IT·DESIGN·KOREAN ACADEMY
VIETNAM

Mô hình phương thức

A. Mô hình chung



B. Ví dụ



GREEN ACADEMY
IT·DESIGN·KOREAN ACADEMY
VIETNAM

Phương thức nạp chồng

Overloading Methods

- Xét trường hợp overload sau

```
class MayTinh {
    int tong(int a, int b) {return a + b;}
    int tong(int a, int b, int c) {return a + b + c;}
}
```

- Với lớp trên, bạn có thể sử dụng để tính tổng 2 hoặc 3 số nguyên

```
MayTinh mt = new MayTinh();
int t1 = mt.tong(5, 7);
int t2 = mt.tong(5, 7, 9);
```

GREEN ACADEMY
IT·DESIGN·KOREAN ACADEMY
VIETNAM

Phương thức nạp chồng

Overloading Methods

- Trong một lớp có thể có nhiều phương thức trùng tên nhưng khác nhau về chữ ký (kiểu dữ liệu, số lượng và trật tự các tham số)
- Trong lớp MyClass có 4 phương thức trùng tên nhưng khác nhau về chữ ký (signatures)

```
public class MyClass {
  void method() {…}
  void method(int x) {…}
  void method(float x) {…}
  void method(int x, double y) {…}
}
```

GREEN ACADEMY
IT·DESIGN·KOREAN ACADEMY
VIETNAM

Phương thức khởi tạo

Constructor

- **Constructor** là một phương thức đặc biệt được sử dụng để tạo đối tượng.
- Đặc điểm:
  - Tên constructor trùng tên lớp
  - Không có kiểu trả về

```
public class Baby {
   string name;
   int birthYear;
   public Baby(string name, int birthYear) {
      this.name = name;
      this.birthYear = birthYear;
   }
}
```

```
Baby b1 = new Baby (“Hữu Vinh”, 1995);
Baby b2 = new Baby (“Tấn Toàn”, 2005);
```

GREEN ACADEMY
IT·DESIGN·KOREAN ACADEMY
VIETNAM

Phương thức khởi tạo

Constructor

- Trong một lớp có thể định nghĩa **nhiều constructor** với danh sách tham số khác nhau. Mỗi constructor cung cấp 1 cách tạo đối tượng.
- Khi không khai báo constructor thì Java tự động cung cấp **default constructor** (0 tham số).

```
public class Baby {
    string name;
    int birthYear;
    public Baby(string name, int birthYear) {
        this.name = name;
        this.birthYear = birthYear;
    }
    public Baby(int birthYear) {
        this.name = “No name”;
        this.birthYear = birthYear;
    }
}
```

```
Baby b1 = new Baby (“Hữu Vinh”, 1995);
Baby b2 = new Baby (2005);
```

GREEN ACADEMY
IT·DESIGN·KOREAN ACADEMY
VIETNAM

this

Keyword

- **this** được sử dụng để đại diện cho đối tượng hiện tại.
- **this** được sử dụng trong lớp để tham chiếu tới các thành viên của lớp (field, method)
- **this.field** để phân biệt field với các biến cục bộ hoặc tham số của phương thức.

```
public class MyClass {
  int field;
  void method(int field) {
    this.field = field;
  }
}
```

GREEN ACADEMY
IT·DESIGN·KOREAN ACADEMY
VIETNAM

Tầm vực truy xuất

Access modifier

- Tầm vực truy xuất được sử dụng để khai báo khả năng truy xuất đến các thành viên của lớp.
- Có 3 đặc tả trong C#, mức độ che dấu đối tượng tăng theo chiều mũi tên.

public

- Công khai hoàn toàn

protected

- Tương tự private nhưng cho phép kế thừa dù lớp cha và con.

private

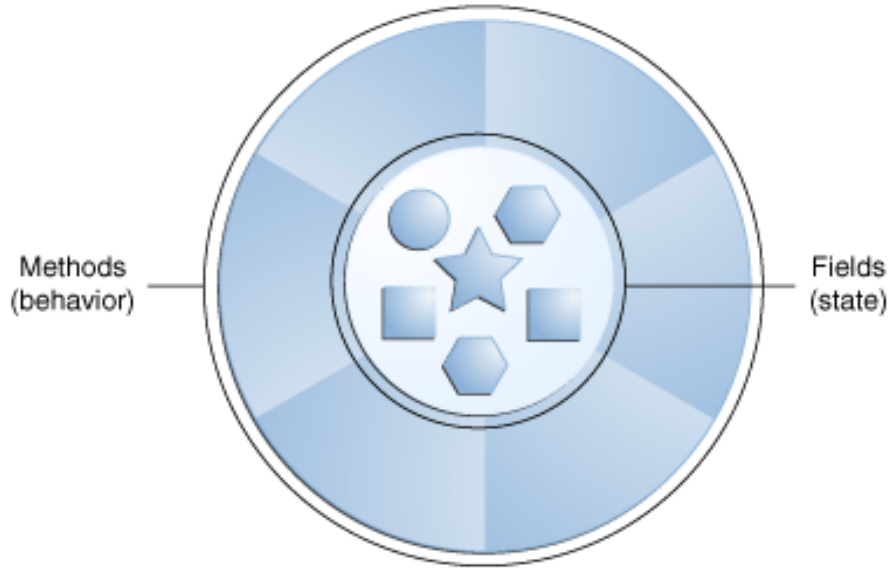
- Chỉ được phép sử dụng nội bộ trong class

GREEN ACADEMY
IT·DESIGN·KOREAN ACADEMY
VIETNAM

Tính bao đóng

Encapsulation

- Mọi dữ liệu của lớp nên được che dấu từ bên ngoài. Tính bao đóng (đóng gói) sử dụng phương thức để truy xuất các trường dữ liệu trong hướng đối tượng
- Mục đích của sự che dấu là để:
  - Bảo vệ dữ liệu
  - Tăng cường khả năng mở rộng



GREEN ACADEMY
IT·DESIGN·KOREAN ACADEMY
VIETNAM

Chuyện gì xảy ra nếu không bao đóng?

Non-Encapsulation

- Giả sử định nghĩa lớp **SinhVien** và công khai trường **Họ tên** và **điểm**.
- Khi sử dụng, người dùng có thể gán dữ liệu cho các trường một cách tuỳ tiện.

```
public class SinhVien {
  public String hoTen;
  public double diem;
}
```

```
public class MyClass {
  public static void main(String[] args) {
      SinhVien sv = new SinhVien();
      sv.hoTen = “Nguyễn Văn A”;
      sv.diem = 20.5;
  }
}
```

GREEN ACADEMY
IT·DESIGN·KOREAN ACADEMY
VIETNAM

Tính bao đóng

Encapsulation

- Để che dấu thông tin, sử dụng private cho các trường dữ liệu.
- Bổ sung các thuộc tính:
  - **Getter**: lấy dữ liệu cho các trường đã che dấu. Thuộc tính chỉ đọc, không được phép chỉnh sửa dữ liệu.

```
private double diem;
//Setter
public double Diem
{
      get
      {
            return diem;
      }
}
```

GREEN ACADEMY
IT·DESIGN·KOREAN ACADEMY
VIETNAM

Tính bao đóng

Encapsulation

- Để che dấu thông tin, sử dụng private cho các trường dữ liệu.
- Bổ sung các thuộc tính:
  - **Setter**: thiết lập dữ liệu cho các trường đã che dấu. Thuộc tính chỉ ghi, không được phép hiển thị dữ liệu.

```
private double diem;
//Setter
public double Diem
{
      set
      {
            diem = value;
      }
}
```

GREEN ACADEMY
IT·DESIGN·KOREAN ACADEMY
VIETNAM

Tính bao đóng

Encapsulation

```
public class SinhVien {
   private String hoTen;
   private double diem;
   public string HoTen {
        set { hoTen = value;}
        get { return hoTen.toUpper();}
   }
   public double Diem {
        set {
        if(diem < 0 || diem >10) diem=0;
        else diem = value;
        }
        get { return diem;}
   }
}
```

Thêm đoạn mã lệnh vào **setter** để xử lý các dữ liệu không hợp lệ.

Thêm đoạn mã lệnh vào **getter** để thay đổi dữ liệu ban đầu.

Nếu trường dữ liệu không có nhu cầu trên thì tham số setter được xem như là biến trung gian trao đổi giữa bên trong với bên ngoài.

```
public class MyClass {
  public static void main(String[] args) {
      SinhVien sv = new SinhVien();
      sv.setHoTen("Nguyễn Văn A");
      sv.setDiem(20.5);
  }
}
```