Bài 7
Lập trình Hướng Đối tượng

Module: BOOTCAMP PREPARATION

Kiểm tra bài trước

Hỏi và trao đổi về các khó khăn gặp phải trong bài “Lập trình hướng đối tượng”

Tóm tắt lại các phần đã học từ bài “Lập trình hướng đối tượng”

Mục tiêu

- Trình bày được mô hình lập trình hướng đối tượng
- Trình bày được các khái niệm lớp, đối tượng, phương thức, thuộc tính, phương thức khởi tạo
- Trình bày được cú pháp khái báo lớp
- Trình bày được cú pháp khởi tạo đối tượng
- Trình bày được cách truy xuất thuộc tính và phương thức của lớp
- Tạo và sử dụng được các đối tượng đơn giản
- Mô tả được lớp bằng biểu đồ

Thảo luận

Lập trình hướng đối tượng

Ngôn ngữ lập trình

- Ngôn ngữ lập trình cho phép Lập trình viên phát triển các phần mềm
- Các dòng ngôn ngữ lập trình phổ biến:
  - Ngôn ngữ máy (machine languages)
  - Ngôn ngữ assembly
  - Ngôn ngữ bậc cao (high-level languages)

Ngôn ngữ máy

- Được tạo thành từ các số 0 và số 1
- Là ngôn ngữ "native" của máy tính
- Khá khó để lập trình
- Ví dụ:

1110100010101 1110101011110

10111010110100 10100011110111

Ngôn ngữ assembly

- Ngôn ngữ assembly dễ lập trình hơn so với ngôn ngữ máy
- Bao gồm một tập các câu lệnh (command) cần thiết được quy định trong các bộ vi xử lý
- Ngôn ngữ assembly cần thiết được chuyển sang ngôn ngữ máy trước khi được thực thi
- Ví dụ:

  ADD 1001010, 1011010

Ngôn ngữ bậc cao

- Ngôn ngữ bậc cao rất dễ để lập trình so với ngôn ngữ assembly
- Cú pháp của ngôn ngữ bậc cao khá gần với tiếng Anh.
- Ngôn ngữ bậc cao có thể được phân loại thành 2 thể loại phổ biến:
  - Ngôn ngữ thủ tục (procedural languages)
  - Ngôn ngữ Lập trình Hướng đối tượng (OOP – Object Oriented Language)

Ngôn ngữ Thủ tục

- Các ngôn ngữ thủ tục thường là các ngôn ngữ bậc cao ra đời từ rất sớm
- Đặc trưng bởi các tập lệnh tuyến tính nối tiếp nhau
- Tập trung vào cấu trúc (structure) của chương trình
- Ví dụ: C, COBOL, Fortran, LISP, Perl, VBScript

Ngôn ngữ Hướng đối tượng (OOP)

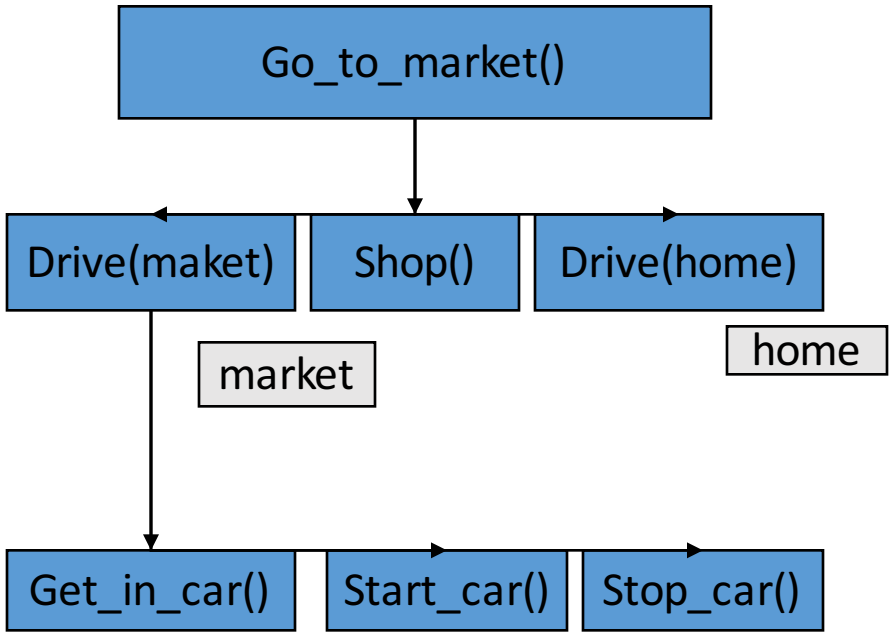
- Không tập trung vào cấu trúc như ngôn ngữ thủ tục, mà tập trung vào mô hình hoá dữ liệu (data modeling)
- Lập trình viên sử dụng các Lớp (class) khi lập trình
- Mô phỏng các đối tượng trong thế giới thực vào trong các chương trình.
- Class: khuôn mẫu của dữ liệu được mô hình hoá
- Ví dụ: C++, C#, Java, PHP, Javascript…

Ngôn ngữ Hướng đối tượng (OOP)

- OOP (Object Oriented Programming) là một triết lý thiết kế chương trình
- Có nhiều ngôn ngữ lập trình hỗ trợ OOP
- Tất cả mọi thứ trong OOP đều là "đối tượng". Một chương trình phần mềm được coi như là một thế giới bao gồm các đối tượng tương tác với nhau
- Mã lệnh và dữ liệu được kết hợp trong một thể thống nhất – đó là *đối tượng*.
- Đối tượng bao gồm:
  - Thuộc tính: các dữ liệu, tính chất của đối tượng
  - Hành vi: các khả năng, hành động mà đối tượng có thể thực hiện
- Các đối tượng có thể có quan hệ với nhau

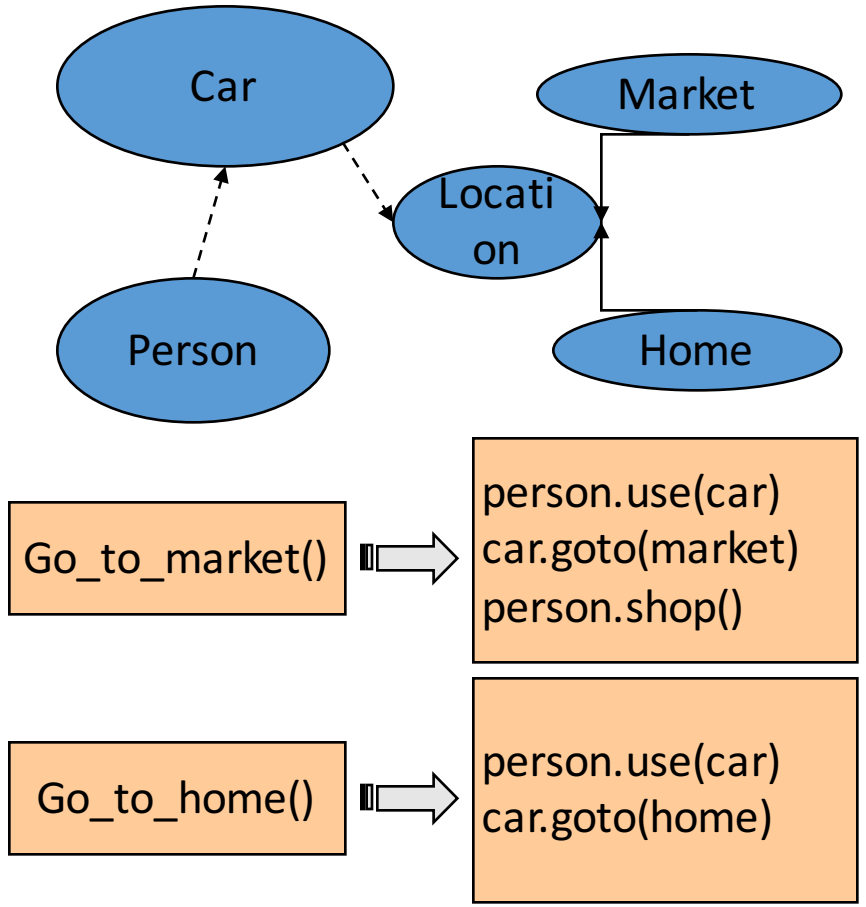
So sánh Lập trình Thủ tục và OOP - 1

- Lập trình hướng cấu trúc:
  - Hướng tiếp cận: từ trên xuống (top down)
  - Chia nhỏ bài toán thành các module chức năng.
  - Dữ liệu và mã lệnh phân tán.
- Hạn chế:
  - Tính ổn định giảm khi hệ thống phát triển
  - Khó bảo trì và tái sử dụng
  - Chi phí phát triển cao



So sánh Lập trình Thủ tục và OOP - 2

- Hướng đối tượng
  - Hướng tiếp cận đa dạng gần với thực tế.
  - Tìm và phân tích mối quan hệ giữa các đối tượng trong bài toán
  - Mã lệnh và dữ liệu liên kết trong thể thống nhất.
- Ưu thế:
  - Khả năng tái sử dụng cao
  - Ổn định và dễ bảo trì
  - Chi phí giảm dần.



Các khái niệm

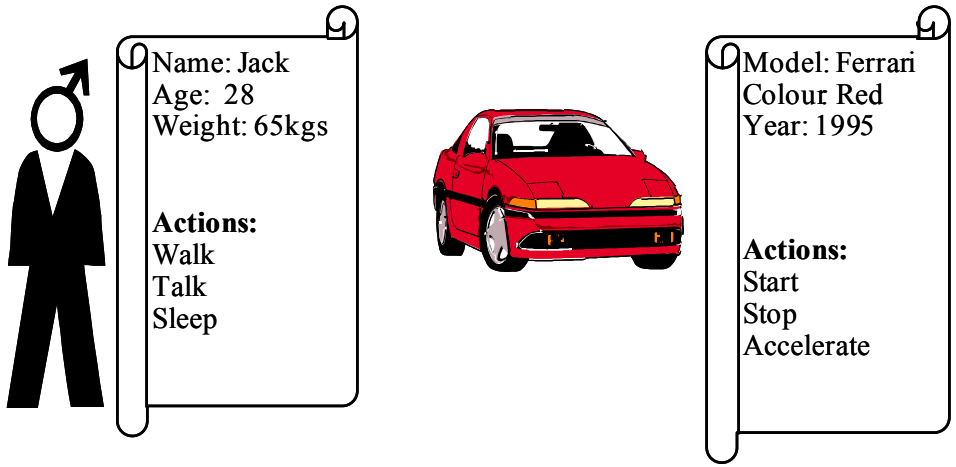
- Đối tượng (Object)
- Lớp (Class)
- Thuộc tính (Property)
- Phương thức (Method)/Hành vi (Behavior)/Hành động (Action)/Khả năng (Capability)
- Các đặc tính cơ bản:
  - Tính bao gói
  - Tính kế thừa
  - Tính trừu tượng
  - Tính đa hình

Đối tượng

- Mỗi đối tượng có những **thuộc tính** hay những đặc điểm mô tả và những **hành vi** riêng nhằm phân biệt nó với các đối tượng khác.

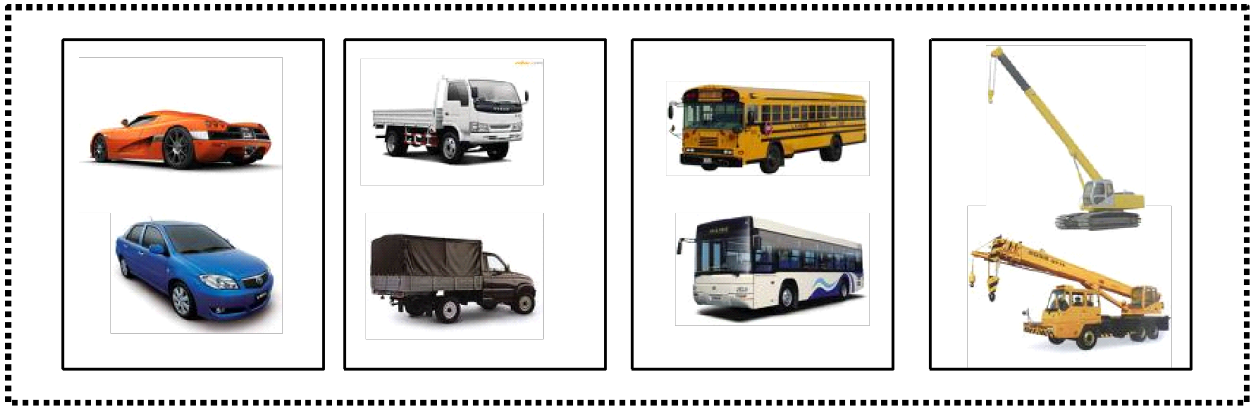
Thuộc tính và Hành vi

- Thuộc tính là những đặc điểm đặc trưng của đối tượng, thể hiện thông qua những giá trị cụ thể.
- Hành vi là những cách thức mà qua đó đối tượng thể hiện sự hoạt động hay chức năng của chúng.



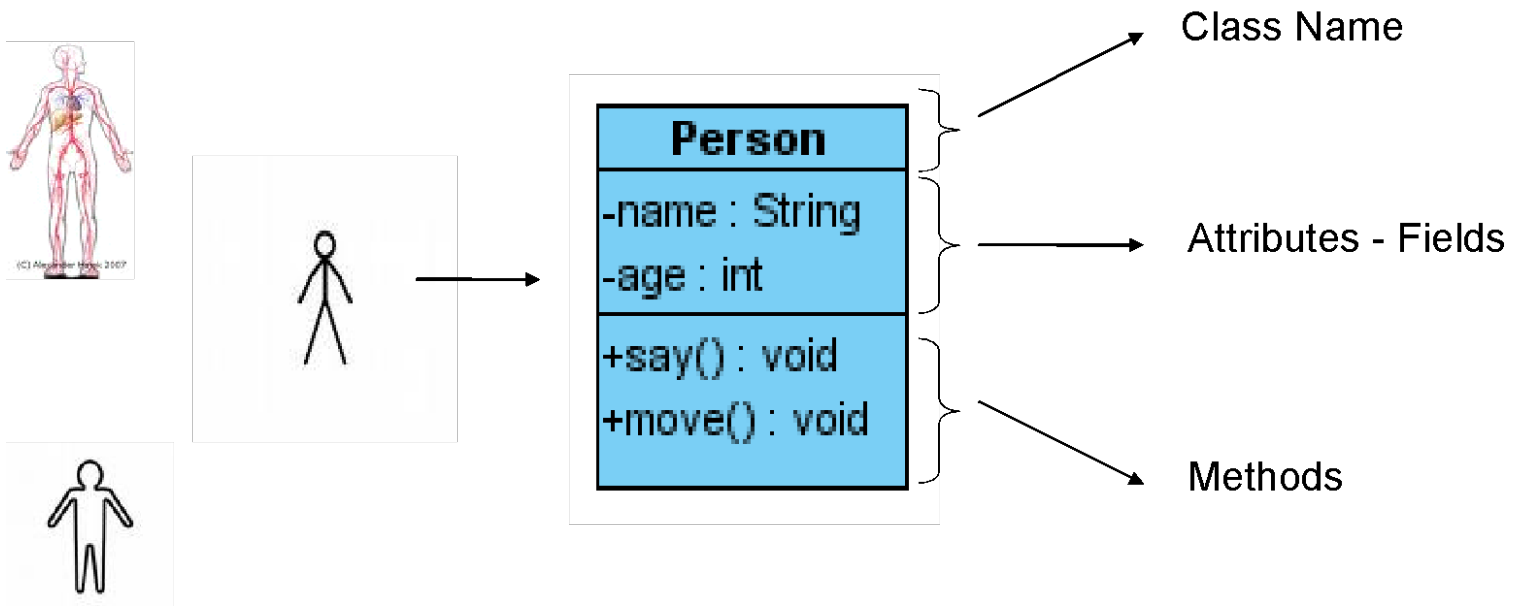
Lớp

- **Lớp** là khái niệm dùng để mô tả một **loại đối tượng** có những thuộc tính, hành vi và những mối quan hệ thông thường tương tự nhau
- Thuật ngữ lớp có thể hiểu là cách nói vắn tắt của cụm từ "lớp các đối tượng".
- Như vậy mỗi đối tượng được coi như là một thể hiện của lớp với những giá trị thuộc tính cũng như cách thức hoạt động đặc đặc trưng.



Sơ đồ mô tả lớp

- Sơ đồ lớp mô tả những đặc điểm khái quát nhất về lớp bao gồm: Tên lớp, danh sách các thuộc tính (tên và loại dữ liệu) và các phương thức



Demo

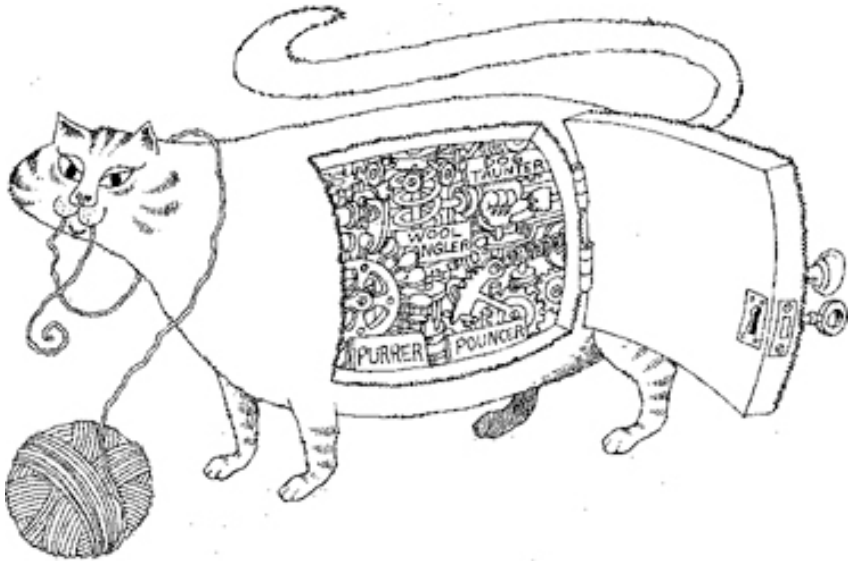
Phân tích các thuộc tính và phương thức của đối tượng

Thảo luận

Tính bao gói
Tính kế thừa
Tính trừu tượng
Tính đa hình

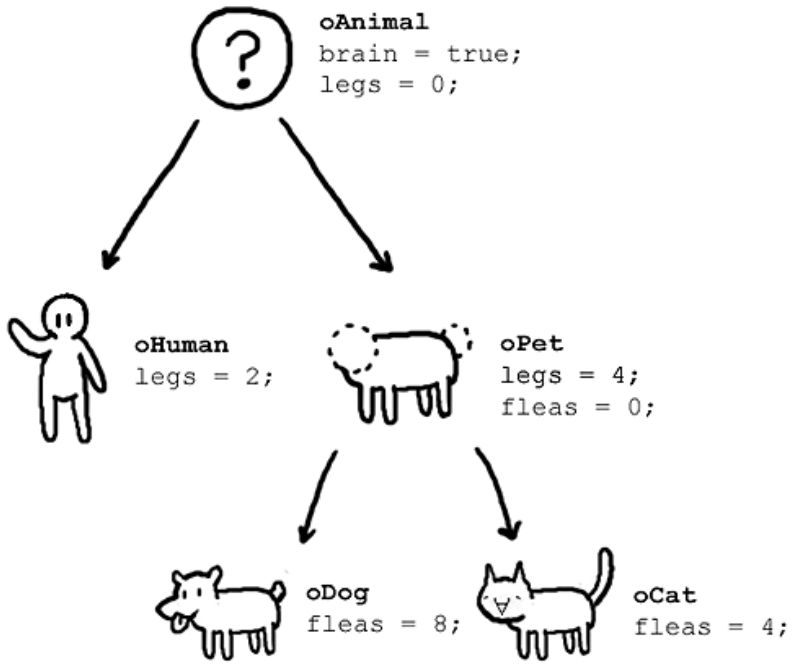
Tính bao gói (encapsulation)

- Khả năng truy suất vào các thành phần của một đối tượng trong khi vẫn đảm bảo che giấu các đặc tính riêng tư bên trong đối tượng được gọi là tính bao gói.



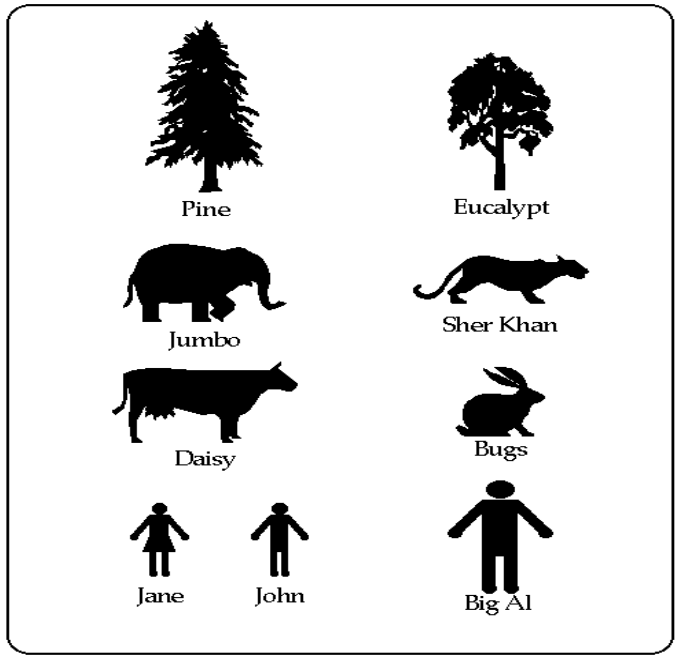
Tính kế thừa

- Tính kế thừa cho phép các đối tượng có thể chia sẻ hay mở rộng các thuộc tính hoặc phương thức mà không phải tiến hành định nghĩa lại.



Tính trừu tượng

- Loại bỏ những thuộc tính và hành vi không quan trọng của đối tượng, chỉ giữ lại những thuộc tính và hành vi có liên quan đến vấn đề đang giải quyết



Tính đa hình

- Tính đa hình thể hiện khi với cùng một phương thức nhưng có thể có cách ứng xử khác nhau ở những lớp cùng giao diện

Thảo luận

Lớp và Đối tượng

Khởi tạo đối tượng

- Cú pháp

```
var objectName = {};
```

hoặc

```
var objectName = { property1:value1, property2:value2 };
```

```
var person = {
    firstName:"John",
    lastName:"Doe",
    age:50,
    eyeColor:"blue"
};
```

Định nghĩa lớp

- Tạo lớp mới với thuộc tính và phương thức

```
function person(firstName, lastName, age, eyeColor) {
    this.firstName = firstName;
    this.lastName = lastName;
    this.age = age;
    this.eyeColor = eyeColor;
    this.changeName = function (name) {
        this.lastName = name;
    };
}
```

- Tạo đối tượng

```
var myFather = new person("John", "Doe", 50, "blue");
var myMother = new person("Sally", "Rally", 48, "green");
```

Sử dụng thuộc tính

- Truy xuất giá trị của thuộc tính của đối tượng

```
objectName.property          // person.age
```

- Sử dụng câu lệnh for-in để truy xuất đến tất cả các thuộc tính của đối tượng

```
objectName["property"]       // person["age"]
```

```
//myFather.age
//myFather["age"]
```

```
var person = {fname:"John", lname:"Doe", age:25};

for (x in person) {
    txt += person[x];
}
```

Sử dụng phương thức

- Gọi một phương thức của đối tượng

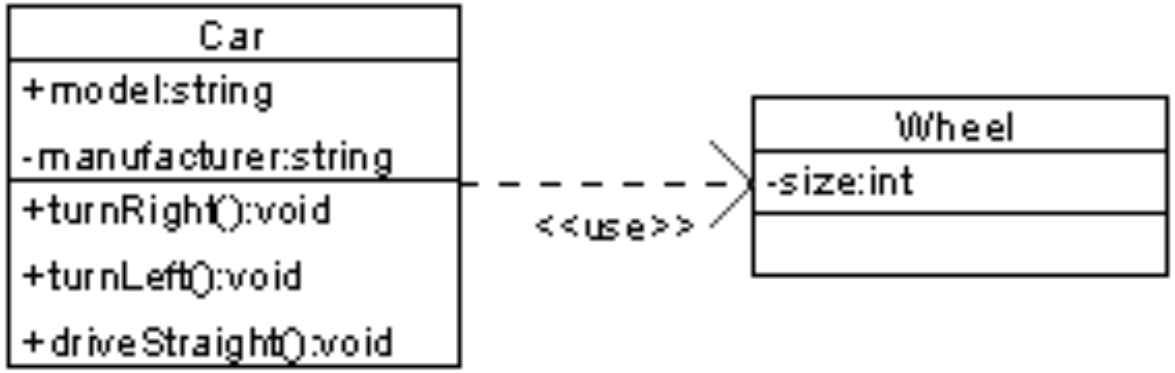
```
objectName.methodName(parameters);
```

- Truyền tham số khi gọi phương thức của đối tượng

```
myMother.changeName("Doe");
```

Mô tả lớp bằng biểu đồ

- Có thể sử dụng những cách khác nhau để mô tả lớp
- Các ký hiệu UML là cách thông dụng để mô tả lớp và nhiều ký hiệu khác
- Để mô tả một lớp cơ bản, cần xác định:
  - Tên lớp
  - Các thuộc tính
  - Các phương thức
  - Các hàm khởi tạo



Demo: Tạo và sử dụng lớp Hình Tròn

Hãy xây dựng lớp hình tròn theo sơ đồ lớp sau:

Circle
- radius: double - color: string
+ Circle() + getRadius(): double + getArea(): double

Lớp Circle (hình tròn) gồm:

- Thuộc tính:
  - bán kính (radius) sẽ nhận vào giá trị dạng số thực
  - màu sắc (color) sẽ nhận vào giá trị dạng chuỗi.
- Phương thức:
  - Circle() là phương thức khởi tạo để tạo đối tượng không tham số.
  - getRadius() là phương thức trả về bán kính của hình tròn
  - getArea() là phương thức trả về diện tích hình tròn theo công thức S = Math.PI * radius * radius

Demo: Tạo và sử dụng lớp Hình Tròn

```
var Circle = function(radius) {
    this.getRadius = function() {
        return radius;
    }
    this.getArea = function() {
        return Math.PI * radius * radius;
    }
};
var circle = new Circle(2);
circle.getRadius(); // 2
circle.getArea(); // 12.566370614359172
```

Demo

Tạo lớp và đối tượng

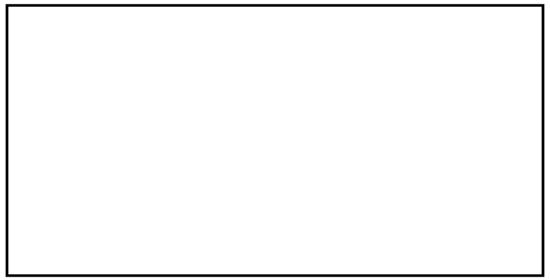
Thảo luận

Sử dụng Canvas

Vẽ trên Canvas

- Canvas được sử dụng để vẽ các hình ảnh trong trang web
- Tạo canvas

```
<canvas id="myCanvas" width="200" height="100" style="border:1px solid #000000;">
</canvas>
```



  - Trong đó:
    - width và height là các thuộc tính để quy định kích thước của canvas
    - id là thuộc tính để chúng ca có thể truy xuất được đối tượng canvas này từ mã Javascript (thông qua phương thức document.getElementById()).

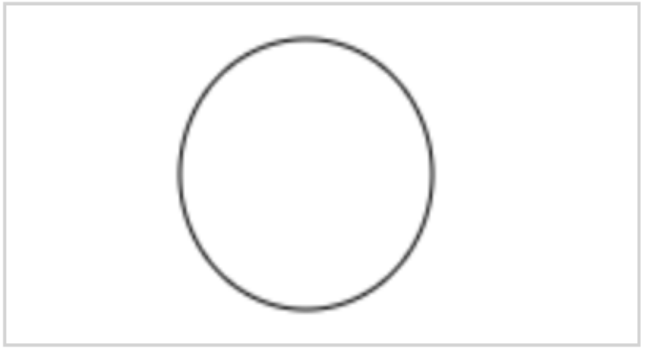
Thuộc tính để vẽ trên canvas

- Các thuộc tính về màu sắc, kiểu dáng (style) và đổ bóng (shadow):
  - fillStyle
  - strokeStyle
  - shadowColor
- Các thuộc tính về đường, văn bản, thao tác điểm ảnh
  - lineWidth
  - font
  - width
  - height

Vẽ hình tròn

- Sử dụng phương thức arc()

```
var c = document.getElementById("myCanvas");
var ctx = c.getContext("2d");
ctx.beginPath();
ctx.arc(95,50,40,0,2*Math.PI);
ctx.stroke();
```



Vẽ hình chữ nhật

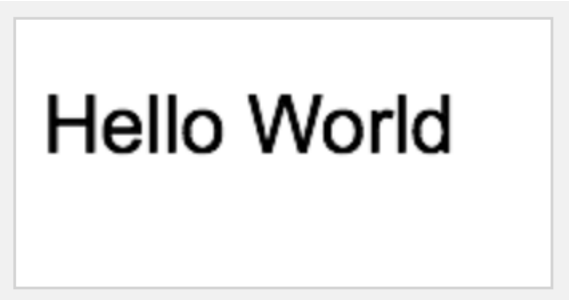
- Sử dụng phương thức fillRect()

```
var canvas = document.getElementById("myCanvas");
var ctx = canvas.getContext("2d");
ctx.fillStyle = "#FF0000";
ctx.fillRect(0,0,150,75);
```

Vẽ chữ

- Sử dụng phương thức fillText()

```
var canvas = document.getElementById("myCanvas");
var ctx = canvas.getContext("2d");
ctx.font = "30px Arial";
ctx.fillText("Hello World",10,50);
```



Vẽ hình ảnh

- Canvas được sử dụng để vẽ các hình ảnh trong trang web

```
var canvas = document.getElementById('myCanvas');
var context = canvas.getContext('2d');
var imageObj = new Image();
imageObj.onload = function() {
        context.drawImage(imageObj, 69, 50);
};
imageObj.src = 'http://www.html5canvastutorials.com/demos/assets/darth-vader.jpg';
```

Demo

Sử dụng canvas

Hướng dẫn

Hướng dẫn làm bài thực hành và bài tập

Chuẩn bị bài tiếp theo: *Lập trình hướng đối tượng*