



# **Môn: Lập trình Hướng đối tượng (Object Oriented Programming)**

## **Chương 2. Những khái niệm cơ bản của Lập trình HĐT**

# Nội dung

- 2.1. Khái niệm đối tượng
- 2.2. So sánh classes và structures
- 2.3. Mô tả thành phần Private và Public của classes
- 2.4. Định nghĩa các hàm của classes
- 2.5. Phương pháp sử dụng các đối tượng và các hàm thành viên của classes
- 2.6. Các ngôn ngữ lập trình hướng đối tượng thông dụng hiện nay
- 2.7. Cách viết class trong Java

## 2.1. Khái niệm đối tượng

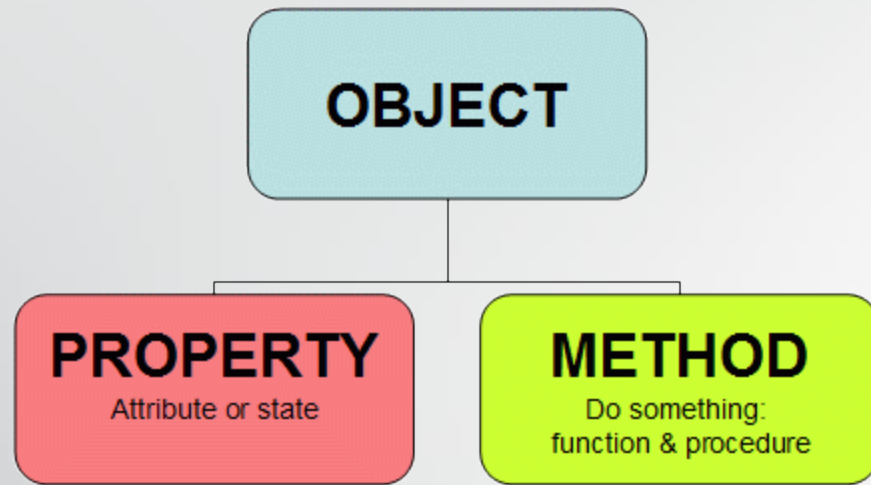
- 2.1.1. Định nghĩa và minh hoạ đối tượng
- 2.1.2. Biểu diễn đối tượng
- 2.1.3. Trừu tượng hoá đối tượng theo chức năng
- 2.1.4. Trừu tượng hoá đối tượng theo dữ liệu (Abstraction)
- 2.1.5. Khái niệm kế thừa (Inheritance)
- 2.1.6. Khái niệm đóng gói (Encapsulation)
- 2.1.7. Khái niệm đa hình (Polymorphism)



## 2.1.1. Định nghĩa và minh họa đối tượng

- Đối tượng là điểm cốt lõi để hiểu về công nghệ hướng đối tượng.
- Một đối tượng có các yếu tố: *identity* (định danh), *data members* (các trạng thái/thuộc tính) và *methods* (các phương thức/hành vi xử lý).
- Ví dụ:
  - Chó có tên cụ thể, có trạng thái (màu sắc, loại, tình trạng đói hay no) và hành vi (sủa, tha đồ vật đến, vẫy đuôi).
  - Xe đạp cũng có trạng thái (bánh răng, nhíp bàn đạp hiện tại, tốc độ hiện tại) và hành vi (thay đổi bánh răng, thay đổi nhíp bàn đạp, sử dụng thắng xe).
  - Việc xác định trạng thái và hành vi của các đối tượng trong thế giới thực là một cách thức tốt trong các khái niệm của lập trình hướng đối tượng.

Một số đối tượng này có thể chứa các đối tượng khác.

## 2.1.1. Định nghĩa và minh họa đối tượng (tt)

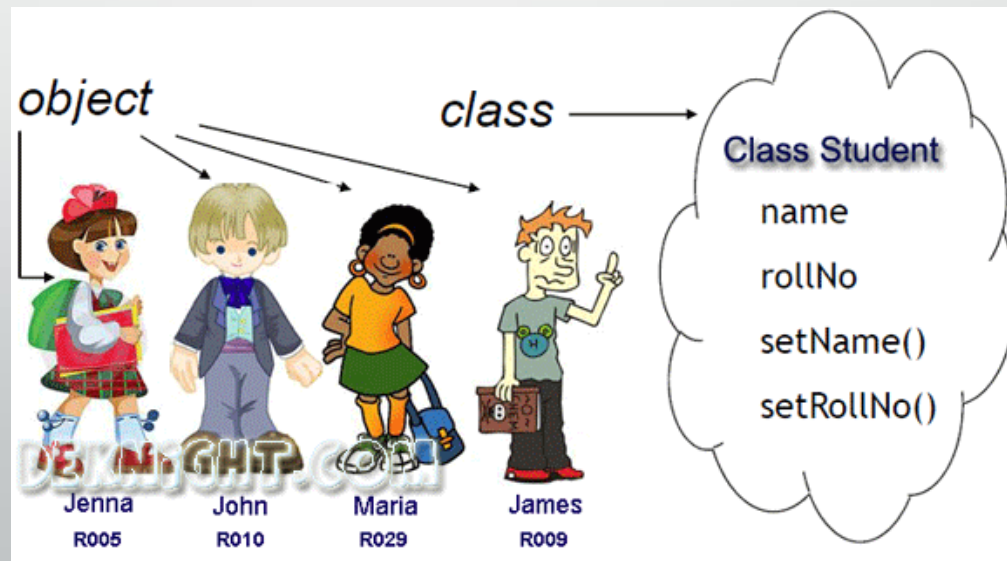


Name of Person : <u>Pritesh</u>	Name of Person : <u>Rani</u>
Skin Color : <u>White</u>	Skin Color : <u>White</u>
Gender : <u>Male</u>	Gender : <u>Female</u>
	
Object : P2	Object : P2



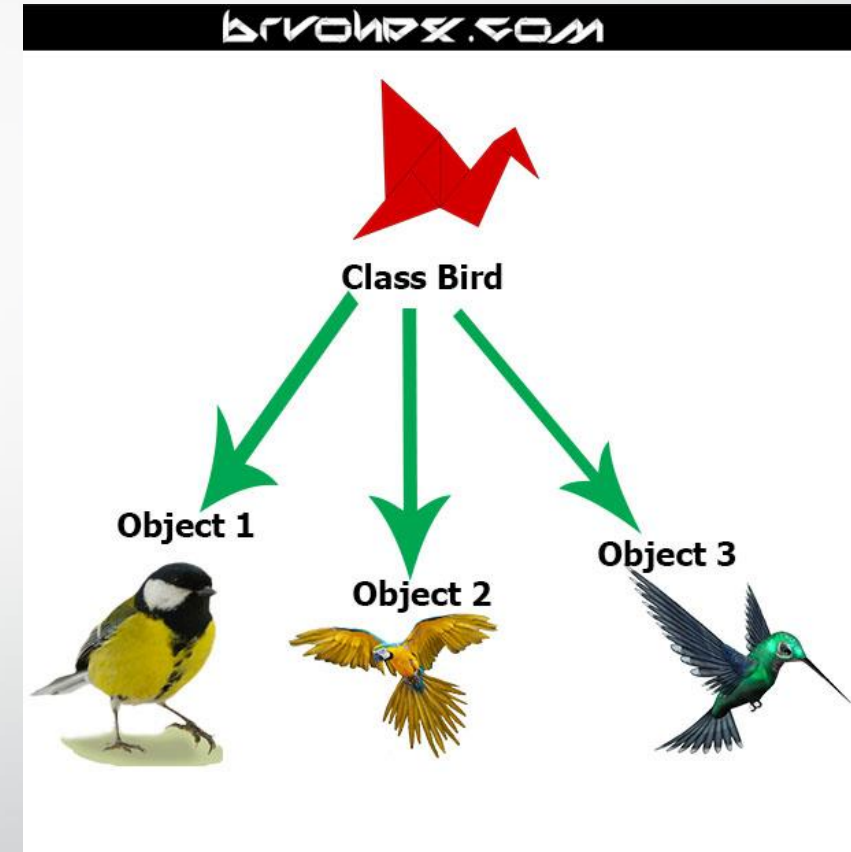
## 2.1.2. Biểu diễn đối tượng

- Lớp (class): đại diện của một tập các đối tượng (object) cùng loại → cùng mô tả, cùng hành vi.
- Để biểu diễn đối tượng thì cần phải *trừu tượng hóa đối tượng* thành *lớp đối tượng*.
- *Lớp là một khái niệm trừu tượng, dùng để chỉ tập hợp các đối tượng cùng kiểu.*
- Lớp có thuộc tính và phương thức mô tả đối tượng.



## 2.1.2. Biểu diễn đối tượng (tt)

- Trong thực tế, nhiều đối tượng thuộc cùng một loại. Có thể có hàng nghìn chiếc xe đạp cùng tồn tại, tất cả chúng đều giống nhau về cách sản xuất và mẫu mã.
- Mỗi chiếc xe đạp đã được tạo ra từ một tập thiết kế chung, vì thế chúng giống nhau về thành phần cấu tạo.
- Trong thuật ngữ hướng đối tượng, chiếc xe đạp là một *thể hiện* của một *lớp các đối tượng* có tên gọi là xe đạp.





## 2.1.2. Biểu diễn đối tượng (tt)

Lưu ý trong trường hợp thiết kế một lớp

- Cần biết những thông tin gì về một đối tượng thuộc lớp này. → Dữ liệu cần mô tả (thuộc tính).
- Cần (bên ngoài) thực sự xử lý gì (động từ) trên đối tượng → Hành vi giao tiếp (public).
- Để có được hành vi giao tiếp, có cần những xử lý thêm mà bên ngoài không cần biết hay không? → Hành vi nội (private).



## 2.1.3. Trừu tượng hoá đối tượng theo chức năng

- Mô hình hóa các đối tượng dựa trên các hành vi của đối tượng
- Quá trình trừu tượng hóa đối tượng theo chức năng:
  - Mô tả các hành vi có thể có của đối tượng
  - Gom các đối tượng có hành vi tương tự thành nhóm, loại bỏ các hành vi riêng biệt
  - Xây dựng lớp cho từng nhóm
  - Xây dựng phương thức cho hành vi chung của mỗi nhóm

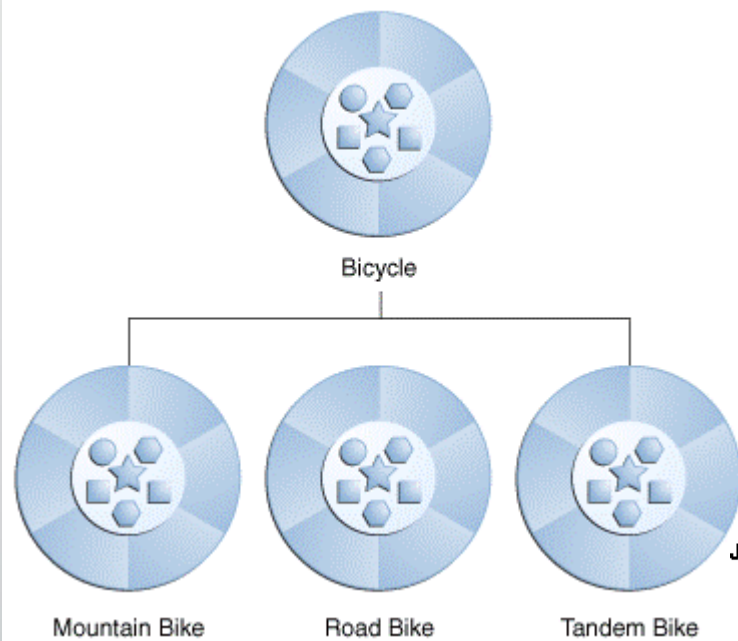
## 2.1.4. Trừu tượng hoá đối tượng theo dữ liệu

- Mô hình hóa đối tượng dựa vào thuộc tính của các đối tượng
- Quá trình trừu tượng hóa đối tượng theo chức năng:
  - Mô tả các hành vi có thể có của đối tượng
  - Gom các đối tượng có hành vi tương tự thành nhóm, loại bỏ các hành vi riêng biệt
  - Xây dựng lớp cho từng nhóm
  - Xây dựng phương thức cho hành vi chung của mỗi nhóm

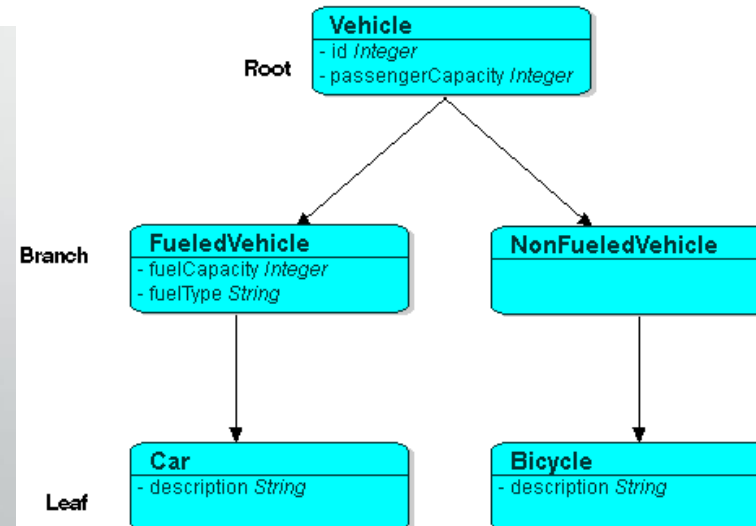
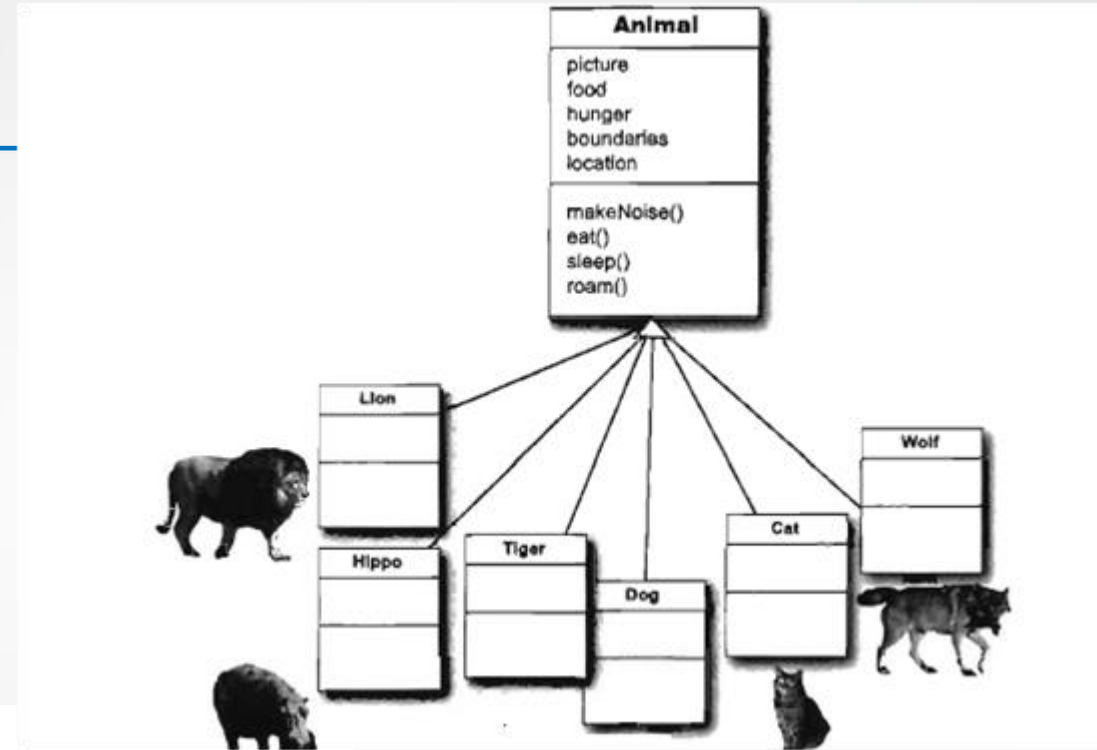
## 2.1.5. Khái niệm kế thừa

- Các loại đối tượng khác nhau thường có một số đặc điểm giống nhau
  - Ví dụ như xe đạp leo núi, xe đạp đường trường, và xe đạp đôi đều có những đặc tính của xe đạp (*tốc độ hiện tại, nhịp đạp hiện tại, bánh răng hiện tại*).
  - Tuy vậy mỗi loại lại có thêm các tính năng bổ sung để phân biệt giữa chúng: Xe đạp đôi có hai chỗ ngồi và hai bộ tay lái; xe đạp đường trường có tay lái cong xuống dưới; một số xe đạp leo núi có bộ vòng xích bổ sung, nhằm làm giảm tỉ lệ của các bánh răng.
- Lập trình hướng đối tượng cho phép các lớp thừa kế các thuộc tính và hành vi chung từ các lớp khác.
- Không giới hạn số lớp con thừa kế từ lớp cha.
- Các quan hệ is-a, has-a

## 2.1.5. Khái niệm kế thừa

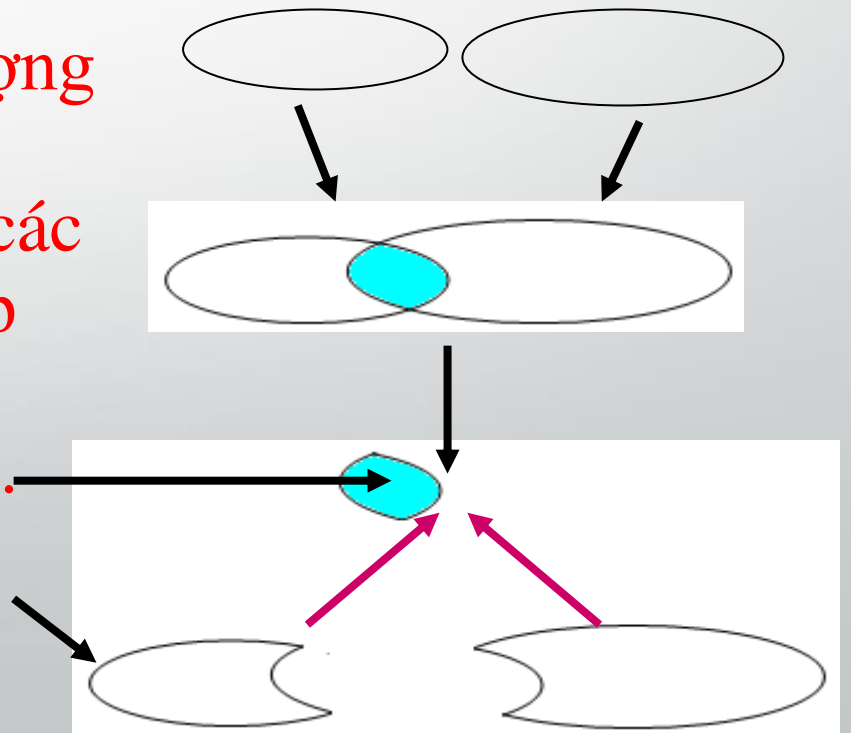


Java Inheritance Hierarchy:



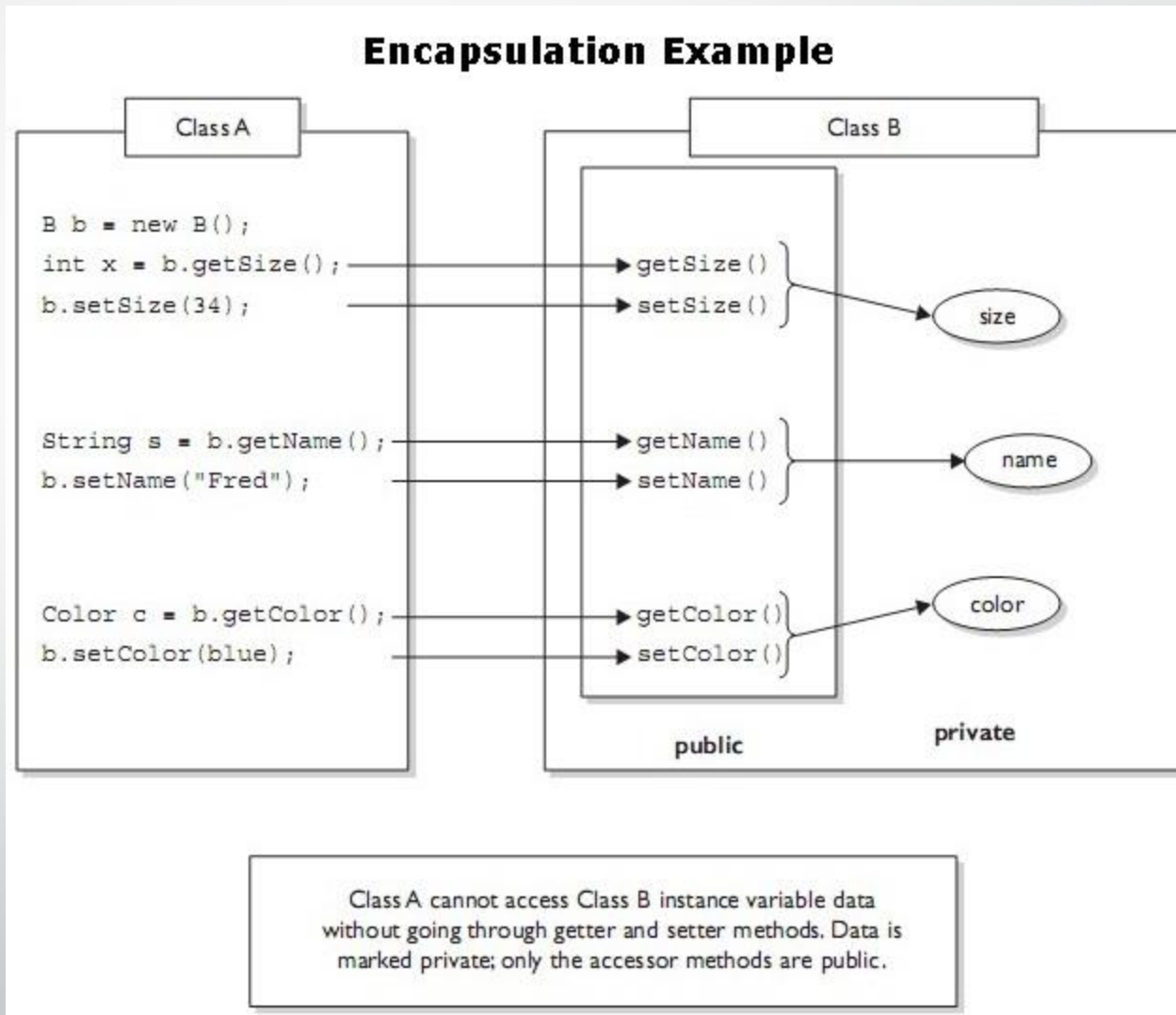
## 2.1.5. Khái niệm kế thừa (tt)

- Thừa kế đơn (single inheritance): Một lớp chỉ có thể có tối đa *một lớp cha*.
- Thừa kế bội (đa thừa kế, multi-inheritance): Một lớp có thể có nhiều *lớp cha*.
- Kỹ thuật phân cấp thừa kế
  - (1) Liệt kê đặc điểm của các loại đối tượng cần quan tâm.
  - (2) Tìm tập giao của các tính chất giữa các lớp, tách tập giao này để xây dựng lớp cha.
  - (3) Đặt 1 tên gọi có ý nghĩa cho lớp cha.
  - (4) Phần còn lại sau khi tách tập giao là các lớp con.



## 2.1.6. Khái niệm đóng gói

- Che dấu/Ẩn dữ liệu (private/protected).
  - Ẩn dữ liệu bên trong phương thức
  - Ẩn dữ liệu bên trong đối tượng.
- Bên ngoài chỉ tương tác được với đối tượng qua một số phương thức (public).



## 2.1.7. Khái niệm đa hình

- Thể hiện thông qua việc gửi các thông điệp (message). Việc gửi các thông điệp này có thể so sánh như việc gọi các hàm bên trong của một đối tượng.
- Các phương thức dùng trả lời cho một thông điệp sẽ tùy theo đối tượng mà thông điệp đó được gửi tới sẽ có phản ứng khác nhau.
- Người lập trình có thể định nghĩa một phương thức cho một loạt các *đối tượng gần nhau* nhưng khi thi hành thì dùng cùng một tên gọi mà sự thi hành của mỗi đối tượng sẽ tự động xảy ra tương ứng theo đặc tính của từng đối tượng mà không bị nhầm lẫn.
- Thí dụ khi định nghĩa hai đối tượng *Hình vuông* và *Hình tròn* thì có một phương thức chung là *chu vi*. Khi gọi phương thức này thì nếu đối tượng là *Hình vuông* nó sẽ tính theo công thức khác với khi đối tượng là *Hình tròn*.



## 2.1.7. Khái niệm đa hình (tt)

- *Tính đa hình là khả năng một ngôn ngữ xử lý các đối tượng hữu quan theo cùng một cách.*
- Tính đa hình thể hiện dưới nhiều hình thức

### **Kết nối trễ - Late Binding**

- Là khả năng cho phép người lập trình gọi trước một phương thức của đối tượng, tuy chưa xác định đối tượng có phương thức muốn gọi hay không.
- Khi thực hiện, chương trình mới xác định được đối tượng và gọi phương thức tương ứng của đối tượng đó.
- Kết nối trễ giúp chương trình được uyển chuyển hơn, chỉ yêu cầu đối tượng cung cấp đúng phương thức cần thiết.

## 2.1.7. Khái niệm đa hình (tt)

### **Nạp chồng – Overloading (trong cùng 1 lớp)**

- Là khả năng cho phép một lớp có nhiều thuộc tính, phương thức cùng tên nhưng với các tham số khác nhau về loại cũng như về số lượng.
- Khi được gọi, dựa vào tham số truyền vào, phương thức tương ứng sẽ được thực hiện.

### **Ghi chồng – Overriding (trong kế thừa lớp cha và con)**

- Hình thức này áp dụng cho *lớp con* đối với *lớp cha*
- *Lớp con* được phép có một phương thức cùng tên, cùng số tham số có kiểu dữ liệu như phương thức của *lớp cha* hoặc những lớp trước đó nữa (lớp phát sinh ra lớp cha ...) với cài đặt khác đi.
- Lúc thực thi, nếu *lớp con* không có phương thức riêng, phương thức của *lớp cha* sẽ được gọi, ngược lại nếu có, phương thức của *lớp con* được gọi.

## 2.2. So sánh classes và structures

	Structures (cấu trúc)	Classes (lớp)
Khái niệm	Mô tả dữ liệu theo hướng lập trình cấu trúc	Mô tả dữ liệu và hành vi của đối tượng theo hướng OOP
Mục đích và chức năng	Nhóm các dữ liệu có liên quan thành một đơn vị thống nhất. Có thể gắn hàm đi kèm với cấu trúc để xử lý dữ liệu.	Nhóm các dữ liệu có liên quan thành một lớp, có phương thức để thực hiện hành vi của đối tượng.
Ưu điểm và nhược điểm	<ul style="list-style-type: none"><li>• Làm cho chương trình dễ đọc theo hướng thuật toán</li><li>• Không hạn chế truy cập</li></ul>	<ul style="list-style-type: none"><li>• Làm cho chương trình gọn gàng, xử lý đồng bộ và thống nhất</li><li>• <i>Hạn chế truy cập</i></li><li>• <i>Đóng gói</i></li><li>• <i>Thừa kế</i></li></ul>

## 2.3. Mô tả thành phần Private và Public của classes

- Phạm vi truy cập/bảo vệ dữ liệu từ việc truy cập từ bên ngoài
- Phân loại phạm vi truy cập:
  - Private
    - Không được truy cập dữ liệu từ bên ngoài
    - Được sử dụng nội bộ lớp: phương thức <-> thuộc tính, phương thức <-> phương thức
    - Các dữ liệu riêng tư hoặc các phương thức dùng nội bộ cần được bảo vệ
  - Public
    - Chia sẻ với các đối tượng bên trong lẫn bên ngoài lớp
    - Các phương thức set/get dữ liệu hoặc các chức năng chương trình
  - Protected: Chia sẻ với các đối tượng thuộc lớp con

## 2.4. Định nghĩa các hàm của classes

- *Các phương thức/hàm* của một lớp định nghĩa lớp có thể làm những gì. Có hai loại phương thức trong ngôn ngữ Java:
  - Hàm khởi tạo
  - Các phương thức/hàm khác
- Cả hai đều có định tố truy cập (access specifier) để chỉ ra những đối tượng nào có thể sử dụng chúng) và phần thân (giữa cặp ngoặc nhọn), có chứa một hay nhiều câu lệnh.

## 2.4. Định nghĩa các hàm của classes (tt)

*accessSpecifier ClassName( arguments ) { constructor statement(s) }*

*accessSpecifier returnValueDataType methodName ( arguments ) { statement(s) }*

- *public*: Bất kỳ đối tượng nào trong bất kỳ package nào đều có thể thấy phương thức này.
- *protected*: Bất kỳ một cá thể nào của lớp, lớp con trong cùng một package và bất kỳ lớp nào không phải là lớp con nhưng nằm trong cùng một package thì có thể thấy phương thức này. Lớp con trong các package khác không thể thấy nó.
- *private*: Không một đối tượng nào ngoài các hàm của lớp có thể thấy được biến phương thức này, thậm chí cả lớp con.

## 2.5. Phương pháp sử dụng các đối tượng và các hàm thành viên của classes

- Khởi tạo đối tượng và gán tham chiếu
- Gọi phương thức

Ví dụ:

```
Student s = new Student();  
s.setFirstName("AA");  
/*nếu phạm vi truy cập của setFirstName(String  
s) là private thì s không gọi được  
setFirstName("AA")  
// lệnh s.setFirstName("AA"); sẽ báo lỗi  
System.out.print(s.getFirstName());
```



## 2.6. Các ngôn ngữ lập trình hướng đối tượng thông dụng hiện nay

---

