Chương 6. Lập trình hướng đối tượng

Chương này giới thiệu những khái niệm cơ bản trong lập trình hướng đối tượng. Các khái niệm cơ bản như lớp, đối tượng, thuộc tính, phương thức, thông điệp, và quan hệ của chúng sẽ được thảo luận trong phần này. Thêm vào đó là sự trình bày của những đặc điểm quan trọng trong lập trình hướng đối tượng như tính bao gói, tính thừa kế, tính đa hình,.. nhằm giúp người học có cái nhìn tổng quát về lập trình hướng đối tượng.

6.1. Giới thiệu

Hướng đối tượng (object orientation) cung cấp một kiểu mới để xây dựng phần mềm. Trong kiểu mới này, các đối tượng (object) và các lớp (class) là những khối xây dựng trong khi các phương thức (method), thông điệp (message), và sự thừa kế (inheritance) cung cấp các cơ chế chủ yếu.

Lập trình hướng đối tượng (OOP- Object-Oriented Programming) là một cách tư duy mới, tiếp cận hướng đối tượng để giải quyết vấn đề bằng máy tính. Thuật ngữ OOP ngày càng trở nên thông dụng trong lĩnh vực công nghệ thông tin.

Khái niệm 6.1

> **Lập trình hướng đối tượng** (OOP) là một phương pháp thiết kế và phát triển phần mềm dựa trên kiến trúc lớp và đối tượng.

Nếu bạn chưa bao giờ sử dụng một ngôn ngữ OOP thì trước tiên bạn nên nắm vững các khái niệm của OOP hơn là viết các chương trình. Bạn cần hiểu được đối tượng (object) là gì, lớp (class) là gì, chúng có quan hệ với nhau như thế nào, và làm thế nào để các đối tượng trao đổi thông điệp (message) với nhau, vâng vâng.

OOP là tập hợp các kỹ thuật quan trọng mà có thể dùng để làm cho việc triển khai chương trình hiệu quả hơn. Quá trình tiến hóa của OOP như sau:

- Lập trình tuyến tính
- Lập trình có cấu trúc
- Sự trừu tượng hóa dữ liệu
- Lập trình hướng đối tượng

6.2. Trừu tượng hóa (Abstraction)

Trừu tượng hóa là một kỹ thuật chỉ trình bày những các đặc điểm cần thiết của vấn đề mà không trình bày những chi tiết cụ thể hay những lời giải thích phức tạp của vấn đề đó. Hay nói khác hơn nó là một kỹ thuật tập trung vào thứ cần thiết và phớt lờ đi những thứ không cần thiết.

Ví dụ những thông tin sau đây là các đặc tính gắn kết với con người:

- Tên
- Tuổi
- Địa chỉ
- Chiều cao
- Màu tóc

Giả sử ta cần phát triển ứng dụng khách hàng mua sắm hàng hóa thì những chi tiết thiết yếu là tên, địa chỉ còn những chi tiết khác (tuổi, chiều cao, màu tóc, ..) là không quan trọng đối với ứng dụng. Tuy nhiên, nếu chúng ta phát triển một ứng dụng hỗ trợ cho việc điều tra tội phạm thì những thông tin như chiều cao và màu tóc là thiết yếu.

Sự trừu tượng hóa đã không ngừng phát triển trong các ngôn ngữ lập trình, nhưng chỉ ở mức dữ liệu và thủ tục. Trong OOP, việc này được nâng lên ở mức cao hơn – mức đối tượng. Sự trừu tượng hóa được phân thành sự trừu tượng hóa dữ liệu và trừu tượng hóa chương trình.

Khái niệm 6.2

> **Trừu tượng hóa dữ liệu** (data abstraction) là tiến trình xác định và nhóm các thuộc tính và các hành động liên quan đến một thực thể đặc thù trong ứng dụng đang phát triển.
>
> **Trừu tượng hóa chương trình** (program abstraction) là một sự trừu tượng hóa dữ liệu mà làm cho các dịch vụ thay đổi theo dữ liệu.

6.3. Đối tượng (object)

Các đối tượng là chìa khóa để hiểu được kỹ thuật hướng đối tượng. Bạn có thể nhìn xung quanh và thấy được nhiều đối tượng trong thế giới thực như: con chó, cái bàn, quyển vở, cây viết, tivi, xe hơi ...Trong một hệ thống hướng đối tượng, mọi thứ đều là đối tượng. Một bảng tính, một ô trong bảng tính, một biểu đồ, một bảng báo cáo, một con số hay một số điện thoại, một tập tin, một thư mục, một máy in, một câu hoặc một từ, thậm chí một ký tự, tất cả chúng là những ví dụ của một đối tượng. Rõ ràng chúng ta viết một chương trình hướng đối tượng cũng có nghĩa là chúng ta đang xây dựng một mô hình

của một vài bộ phận trong thế giới thực. Tuy nhiên các đối tượng này có thể được biểu diễn hay mô hình trên máy tính.

Một đối tượng thế giới thực là một thực thể cụ thể mà thông thường bạn có thể sờ, nhìn thấy hay cảm nhận được. Tất cả các đối tượng trong thế giới thực đều có **trạng thái** (state) và **hành động** (behaviour). Ví dụ:

	Trạng thái	**Hành động**
Con chó	▪ Tên ▪ Màu ▪ Giống ▪ Vui sướng	▪ Sủa ▪ Vẫy tai ▪ Chạy ▪ Ăn
Xe đạp	▪ Bánh răng ▪ Bàn đạp ▪ Dây xích ▪ Bánh xe	▪ Tăng tốc ▪ Giảm tốc ▪ Chuyển bánh răng

Các **đối tượng phần mềm** (software object) có thể được dùng để biểu diễn các đối tượng thế giới thực. Chúng được mô hình sau khi các đối tượng thế giới thực có cả trạng thái và hành động. Giống như các đối tượng thế giới thực, các đối tượng phần mềm cũng có thể có trạng thái và hành động. Một đối tượng phần mềm có biến (variable) hay trạng thái (state) mà thường được gọi là **thuộc tính** (attribute; property) để duy trì trạng thái của nó và **phương thức** (method) để thực hiện các hành động của nó. Thuộc tính là một hạng mục dữ liệu được đặt tên bởi một định danh (identifier) trong khi phương thức là một chức năng được kết hợp với đối tượng chứa nó.

OOP thường sử dụng hai thuật ngữ mà sau này Java cũng sử dụng là thuộc tính (attribute) và phương thức (method) để đặc tả tương ứng cho trạng thái (state) hay biến (variable) và hành động (behavior). Tuy nhiên C++ lại sử dụng hai thuật ngữ **dữ liệu thành viên** (member data) và **hàm thành viên** (member function) thay cho các thuật ngữ này.

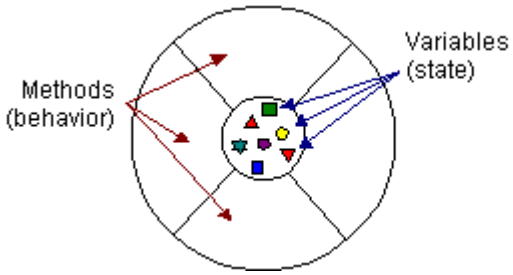
Xét một cách đặc biệt, chỉ một đối tượng riêng rẽ thì chính nó không hữu dụng. Một chương trình hướng đối tượng thường gồm có hai hay nhiều hơn các đối tượng phần mềm tương tác lẫn nhau như là sự tương tác của các đối tượng trong trong thế giới thực.

Khái niệm 6.3

> **Đối tượng** (object) là một thực thể phần mềm bao bọc các thuộc tính và các phương thức liên quan.

Kể từ đây, trong giáo trình này chúng ta sử dụng thuật ngữ **đối tượng** (object) để chỉ một đối tượng phần mềm. Hình 6.1 là một minh họa của một đối tượng phần mềm:

Hình 6.1 Một đối tượng phần mềm



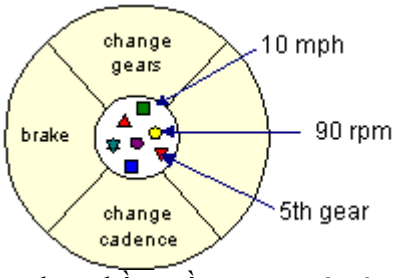
Mọi thứ mà đối tượng phần mềm biết (trạng thái) và có thể làm (hành động) được thể hiện qua các thuộc tính và các phương thức. Một đối tượng phần mềm mô phỏng cho chiếc xe đạp sẽ có các thuộc tính để xác định các trạng thái của chiếc xe đạp như: tốc độ của nó là 10 km trên giờ, nhịp bàn đạp là 90 vòng trên phút, và bánh răng hiện tại là bánh răng thứ 5. Các thuộc tính này thông thường được xem như **thuộc tính thể hiện** (instance attribute) bởi vì chúng chứa đựng các trạng thái cho một đối tượng xe đạp cụ thể. Trong kỹ thuật hướng đối tượng thì một đối tượng cụ thể được gọi là một **thể hiện** (instance).

Khái niệm 6.4

Một đối tượng cụ thể được gọi là một **thể hiện** (instance).

Hình 6.2 minh họa một xe đạp được mô hình như một đối tượng phần mềm:

Hình 6.2 Đối tượng phần mềm xe đạp



Đối tượng xe đạp phần mềm cũng có các phương thức để thắng lại, tăng nhịp đạp hay là chuyển đổi bánh răng. Nó không có phương thức để thay đổi tốc độ vì tốc độ của xe đạp có thể tính ra từ hai yếu tố số vòng quay và bánh răng hiện tại. Những phương thức này thông thường được biết như là các **phương thước thể hiện** (instance method) bởi vì chúng tác động hay thay đổi trạng thái của một đối tượng cụ thể.

6.4. Lớp (Class)

Trong thế giới thực thông thường có nhiều loại đối tượng cùng loại. Chẳng hạn chiếc xe đạp của bạn chỉ là một trong hàng tỉ chiếc xe đạp trên thế giới. Tương tự, trong một chương trình hướng đối tượng có thể có nhiều đối tượng cùng loại và chia sẻ những đặc điểm chung. Sử dụng thuật ngữ hướng đối tượng, chúng ta có thể nói rằng chiếc xe đạp của bạn là một thể hiện của lớp xe đạp. Các xe đạp có một vài trạng thái chung (bánh răng hiện tại, số vòng quay hiện tại, hai bánh xe) và các hành động (chuyển bánh răng, giảm tốc). Tuy nhiên, trạng thái của mỗi xe đạp là độc lập và có thể khác với các trạng thái của các xe đạp khác. Trước khi tạo ra các xe đạp, các nhà sản xuất thường thiết lập một bảng thiết kế (blueprint) mô tả các đặc điểm và các yếu tố cơ bản của xe đạp. Sau đó hàng loạt xe đạp sẽ được tạo ra từ bản thiết kế này. Không hiệu quả nếu như tạo ra một bản thiết kế mới cho mỗi xe đạp được sản xuất.

Trong phần mềm hướng đối tượng cũng có thể có nhiều đối tượng cùng loại chia sẻ những đặc điểm chung như là: các hình chữ nhật, các mẫu tin nhân viên, các đoạn phim, … Giống như là các nhà sản xuất xe đạp, bạn có thể tạo ra một bảng thiết kế cho các đối tượng này. Một bảng thiết kế phần mềm cho các đối tượng được gọi là **lớp** (class).

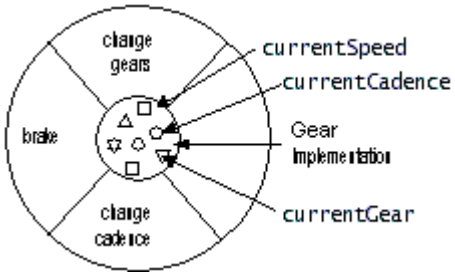
Khái niệm 6.5

> **Lớp** (class) là một thiết kế (blueprint) hay một mẫu ban đầu (prototype) định nghĩa các thuộc tính và các phương thức chung cho tất cả các đối tượng của cùng một loại nào đó.
>
> Một đối tượng là một thể hiện cụ thể của một lớp.

Trở lại ví dụ về xe đạp chúng ta thấy rằng một lớp Xedap là một bảng thiết kế cho hàng loạt các đối tượng xe đạp được tạo ra. Mỗi đối tượng xe đạp là một thể hiện của lớp Xedap và trạng thái của nó có thể khác với các đối tượng xe đạp khác. Ví dụ một xe đạp hiện tại có thể là ở bánh răng thứ 5 trong khi một chiếc khác có thể là ở bánh răng thứ 3.

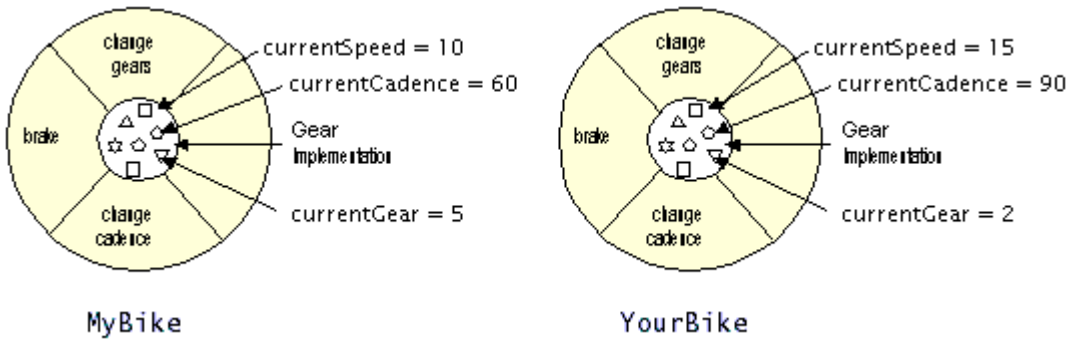
Lớp Xedap sẽ khai báo các thuộc tính thể hiện cần thiết để chứa đựng bánh răng hiện tại, số vòng quay hiện tại, .. cho mỗi đối tượng xe đạp. Lớp Xedap cũng khai báo và cung cấp những thi công cho các phương thức thể hiện để cho phép người đi xe đạp chuyển đổi bánh răng, phanh lại, chuyển đổi số vòng quay, .. như Hình 6.3.

Hình 6.3 Khai báo cho lớp Xedap



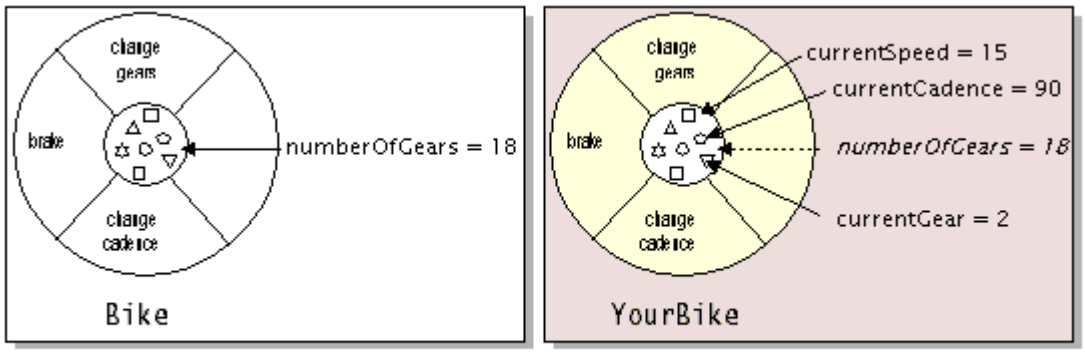
Sau khi bạn đã tạo ra lớp xe đạp, bạn có thể tạo ra bất kỳ đối tượng xe đạp nào từ lớp này. Khi bạn tạo ra một thể hiện của lớp, hệ thống cấp phát đủ bộ nhớ cho đối tượng và tất cả các thuộc tính thể hiện của nó. Mỗi thể hiện sẽ có vùng nhớ riêng cho các thuộc tính thể hiện của nó. Hình 6.4 minh họa hai đối tượng xe đạp khác nhau được tạo ra từ cùng lớp Xedap:

Hình 6.4 Hai đối tượng của lớp Xedap



Ngoài các thuộc tính thể hiện, các lớp có thể định nghĩa các **thuộc tính lớp** (class attribute). Một thuộc tính lớp chứa đựng các thông tin mà được chia sẻ bởi tất cả các thể hiện của lớp. Ví dụ, tất cả xe đạp có cùng số lượng bánh răng. Trong trường hợp này, định nghĩa một thuộc tính thể hiện để giữ số lượng bánh răng là không hiệu quả bởi vì tất cả các vùng nhớ của các thuộc tính thể hiện này đều giữ cùng một giá trị. Trong những trường hợp như thế bạn có thể định nghĩa một thuộc tính lớp để chứa đựng số lượng bánh răng của xe đạp.Tất cả các thể hiện của lớp Xedap sẽ chia thuộc tính này. Một lớp cũng có thể khai báo các **phương thức lớp** (class methods). Bạn có thể triệu gọi một phương thức lớp trực tiếp từ lớp nhưng ngược lại bạn phải triệu gọi các phương thức thể hiện từ một thể hiện cụ thể nào đó.

Hình 6.5 Lớp và thể hiện của lớp



Khái niệm 6.6

> **Thuộc tính lớp** (class attribute) là một hạng mục dữ liệu liên kết với một lớp cụ thể mà không liên kết với các thể hiện của lớp. Nó được định nghĩa bên trong định nghĩa lớp và được chia sẻ bởi tất cả các thể hiện của lớp.
>
> **Phương thức lớp** (class method) là một phương thức được triệu gọi mà không tham khảo tới bất kỳ một đối tượng nào. Tất cả các phương thức lớp ảnh hưởng đến toàn bộ lớp chứ không ảnh hưởng đến một lớp riêng rẽ nào.

6.5. Thuộc tính (Attribute)

Các thuộc tính trình bày trạng thái của đối tượng. Các thuộc tính nắm giữ các giá trị dữ liệu trong một đối tượng, chúng định nghĩa một đối tượng đặc thù.

Khái niệm 6.7

> **Thuộc tính** (attribute) là dữ liệu trình bày các đặc điểm về một đối tượng.

Một thuộc tính có thể được gán một giá trị chỉ sau khi một đối tượng dựa trên lớp ấy được tạo ra. Một khi các thuộc tính được gán giá trị chúng mô tả một đối tượng. Mọi đối tượng của một lớp phải có cùng các thuộc tính nhưng giá trị của các thuộc tính thì có thể khác nhau. Một thuộc tính của đối tượng có thể nhận các giá trị khác nhau tại những thời điểm khác nhau.

6.6. Phương thức (Method)

Các phương thức thực thi các hoạt động của đối tượng. Các phương thức là nhân tố làm thay đổi các thuộc tính của đối tượng.

Khái niệm 6.8

> **Phương thức** (method) có liên quan tới những thứ mà đối tượng có thể làm. Một phương thức đáp ứng một chức năng tác động lên dữ liệu của đối tượng (thuộc tính).

Các phương thức xác định cách thức hoạt động của một đối tượng và được thực thi khi đối tượng cụ thể được tạo ra.Ví dụ, các hoạt động chung của một đối tượng thuộc lớp Chó là sủa, vẫy tai, chạy, và ăn. Tuy nhiên, chỉ khi một đối tượng cụ thể thuộc lớp Chó được tạo ra thì các phương thức sủa, vẫy tai, chạy, và ăn mới được thực thi.

Các phương thức mang lại một cách nhìn khác về đối tượng. Khi bạn nhìn vào đối tượng Cửa ra vào bên trong môi trường của bạn (môi trường thế giới thực), một cách đơn giản bạn có thể thấy nó là một đối tượng bất động không có khả năng suy nghỉ. Trong tiếp cận hướng đối tượng cho phát triển hệ thống, Cửa ra vào có thể được liên kết tới phương thức được giả sử là có thể được thực hiện. Ví dụ, Cửa ra vào có thể mở, nó có thể đóng, nó có thể khóa, hoặc nó có thể mở khóa. Tất cả các phương thức này gắn kết với đối tượng Cửa ra vào và được thực hiện bởi Cửa ra vào chứ không phải một đối tượng nào khác.

6.7. Thông điệp (Message)

Một chương trình hay ứng dụng lớn thường chứa nhiều đối tượng khác nhau. Các đối tượng phần mềm tương tác và giao tiếp với nhau bằng cách gởi các **thông điệp** (message). Khi đối tượng A muốn đối tượng B thực hiện các phương thức của đối tượng B thì đối tượng A gởi một thông điệp tới đối tượng B.

Ví dụ đối tượng người đi xe đạp muốn đối tượng xe đạp thực hiện phương thức chuyển đổi bánh răng của nó thì đối tượng người đi xe đạp cần phải gởi một thông điệp tới đối tượng xe đạp.

Đôi khi đối tượng nhận cần thông tin nhiều hơn để biết chính xác thực hiện công việc gì. Ví dụ khi bạn chuyển bánh răng trên chiếc xe đạp của bạn thì bạn phải chỉ rõ bánh răng nào mà bạn muốn chuyển. Các thông tin này được truyền kèm theo thông điệp và được gọi là các **tham số** (parameter).

Một thông điệp gồm có:

- Đối tượng nhận thông điệp
- Tên của phương thức thực hiện
- Các tham số mà phương thức cần

Khái niệm 6.9

> Một **thông điệp** (message) là một lời yêu cầu một hoạt động.
>
> Một thông điệp được truyền khi một đối tượng triệu gọi một hay nhiều phương thức của đối tượng khác để yêu cầu thông tin.

Khi một đối tượng nhận được một thông điệp, nó thực hiện một phương thức tương ứng. Ví dụ đối tượng xe đạp nhận được thông điệp là chuyển đổi bánh răng nó sẽ thực hiện việc tìm kiếm phương thức chuyển đổi bánh răng tương ứng và thực hiện theo yêu cầu của thông điệp mà nó nhận được.

6.8. Tính bao gói (Encapsulation)

Trong đối tượng xe đạp, giá trị của các thuộc tính được chuyển đổi bởi các phương thức. Phương thức changeGear() chuyển đổi giá trị của thuộc tính currentGear. Thuộc tính speed được chuyển đổi bởi phương thức changeGear() hoặc changRpm().

Trong OOP thì các thuộc tính là trung tâm, là hạt nhân của đối tượng. Các phương thức bao quanh và che giấu đi hạt nhân của đối tượng từ các đối tượng khác trong chương trình.Việc bao gói các thuộc tính của một đối tượng bên trong sự che chở của các phương thức của nó được gọi là sự **đóng gói** (encapsulation) hay là đóng gói dữ liệu.

Đặc tính đóng gói dữ liệu là ý tưởng của các nhà thiết các hệ thống hướng đối tượng. Tuy nhiên, việc áp dụng trong thực tế thì có thể không hoàn toàn như thế. Vì những lý do thực tế mà các đối tượng đôi khi cần phải phơi bày ra một vài thuộc tính này và che giấu đi một vài phương thức kia. Tùy thuộc vào các ngôn ngữ lập trình hướng đối tượng khác nhau, chúng ta có các điều khiển các truy xuất dữ liệu khác nhau.

Khái niệm 6.10

> **Đóng gói** (encapsulation) là tiến trình che giấu việc thực thi chi tiết của một đối tượng.

Một đối tượng có một giao diện chung cho các đối tượng khác sử dụng để giao tiếp với nó. Do đặc tính đóng gói mà các chi tiết như: các trạng thái

được lưu trữ như thế nào hay các hành động được thi công ra sao có thể được che giấu đi từ các đối tượng khác. Điều này có nghĩa là các chi tiết riêng của đối tượng có thể được chuyển đổi mà hoàn toàn không ảnh hưởng tới các đối tượng khác có liên hệ với nó. Ví dụ, một người đi xe đạp không cần biết chính xác cơ chế chuyển bánh răng trên xe đạp thực sự làm việc như thế nào nhưng vẫn có thể sử dụng nó. Điều này được gọi là che giấu thông tin.

Khái niệm 6.11

> **Che giấu thông tin** (information hiding) là việc ẩn đi các chi tiết của thiết kế hay thi công từ các đối tượng khác.

6.9. Tính thừa kế (Inheritance)

Hệ thống hướng đối tượng cho phép các lớp được định nghĩa kế thừa từ các lớp khác. Ví dụ, lớp xe đạp leo núi và xe đạp đua là những lớp con (subclass) của lớp xe đạp. Như vậy ta có thể nói lớp xe đạp là lớp cha (superclass) của lớp xe đạp leo núi và xe đạp đua.

Khái niệm 6.12

> **Thừa kế** (inheritance) nghĩa là các hành động (phương thức) và các thuộc tính được định nghĩa trong một lớp có thể được thừa kế hoặc được sử dụng lại bởi lớp khác.

Khái niệm 6.13

> **Lớp cha** (superclass) là lớp có các thuộc tính hay hành động được thừa hưởng bởi một hay nhiều lớp khác.
>
> **Lớp con** (subclass) là lớp thừa hưởng một vài đặc tính chung của lớp cha và thêm vào những đặc tính riêng khác.

Các lớp con thừa kế thuộc tính và hành động từ lớp cha của chúng. Ví dụ, một xe đạp leo núi không những có bánh răng, số vòng quay trên phút và tốc độ giống như mọi xe đạp khác mà còn có thêm một vài loại bánh răng vì thế mà nó cần thêm một thuộc tính là gearRange (loại bánh răng).

Các lớp con có thể định nghĩa lại các phương thức được thừa kế để cung cấp các thi công riêng biệt cho các phương thức này. Ví dụ, một xe đạp leo núi sẽ cần một phương thức đặc biệt để chuyển đổi bánh răng.

Các lớp con cung cấp các phiên bản đặc biệt của các lớp cha mà không cần phải định nghĩa lại các lớp mới hoàn toàn. Ở đây, mã lớp cha có thể được sử dụng lại nhiều lần.

6.10.Tính đa hình (Polymorphism)

Một khái niệm quan trọng khác có liên quan mật thiết với truyền thông điệp là **đa hình** (polymorphism). Với đa hình, nếu cùng một hành động (phương thức) ứng dụng cho các đối tượng thuộc các lớp khác nhau thì có thể đưa đến những kết quả khác nhau.

Khái niệm 6.14

> **Đa hình** (polymorphism) nghĩa là "nhiều hình thức", hành động cùng tên có thể được thực hiện khác nhau đối với các đối tượng/các lớp khác nhau.

Chúng ta hãy xem xét các đối tượng Cửa Sổ và Cửa Cái. Cả hai đối tượng có một hành động chung có thể thực hiện là đóng. Nhưng một đối tượng Cửa Cái thực hiện hành động đó có thể khác với cách mà một đối tượng Cửa Sổ thực hiện hành động đó. Cửa Cái khép cánh cửa lại trong khi Cửa Sổ hạ các thanh cửa xuống. Thật vậy, hành động đóng có thể thực hiện một trong hai hình thức khác nhau. Một ví dụ khác là hành động hiển thị. Tùy thuộc vào đối tượng tác động, hành động ấy có thể hiển thị một chuỗi, hoặc vẽ một đường thẳng, hoặc là hiển thị một hình.

Đa hình có sự liên quan tới việc truyền thông điệp. Đối tượng yêu cầu cần biết hành động nào để yêu cầu và yêu cầu từ đối tượng nào. Tuy nhiên đối tượng yêu cầu không cần lo lắng về một hành động được hoàn thành như thế nào.

Bài tập cuối chương 6

6.1 Trình bày các định nghĩa của các thuật ngữ:

- Lập trình hướng đối tượng
- Trừu tượng hóa
- Đối tượng
- Lớp
- Thuộc tính
- Phương thức
- Thông điệp

6.2 Phân biệt sự khác nhau giữa lớp và đối tượng, giữa thuộc tính và giá trị, giữa thông điệp và truyền thông điệp.

6.3 Trình bày các đặc điểm của OOP.

6.4 Những lợi ích có được thông qua thừa kế và bao gói.

6.5 Những thuộc tính và phương thức cơ bản của một cái máy giặt.

6.6 Những thuộc tính và phương thức cơ bản của một chiếc xe hơi.

6.7 Những thuộc tính và phương thức cơ bản của một hình tròn.

6.8 Chỉ ra các đối tượng trong hệ thống rút tiền tự động ATM.

6.9 Chỉ ra các lớp có thể kế thừa từ lớp điện thoại, xe hơi, và động vật.