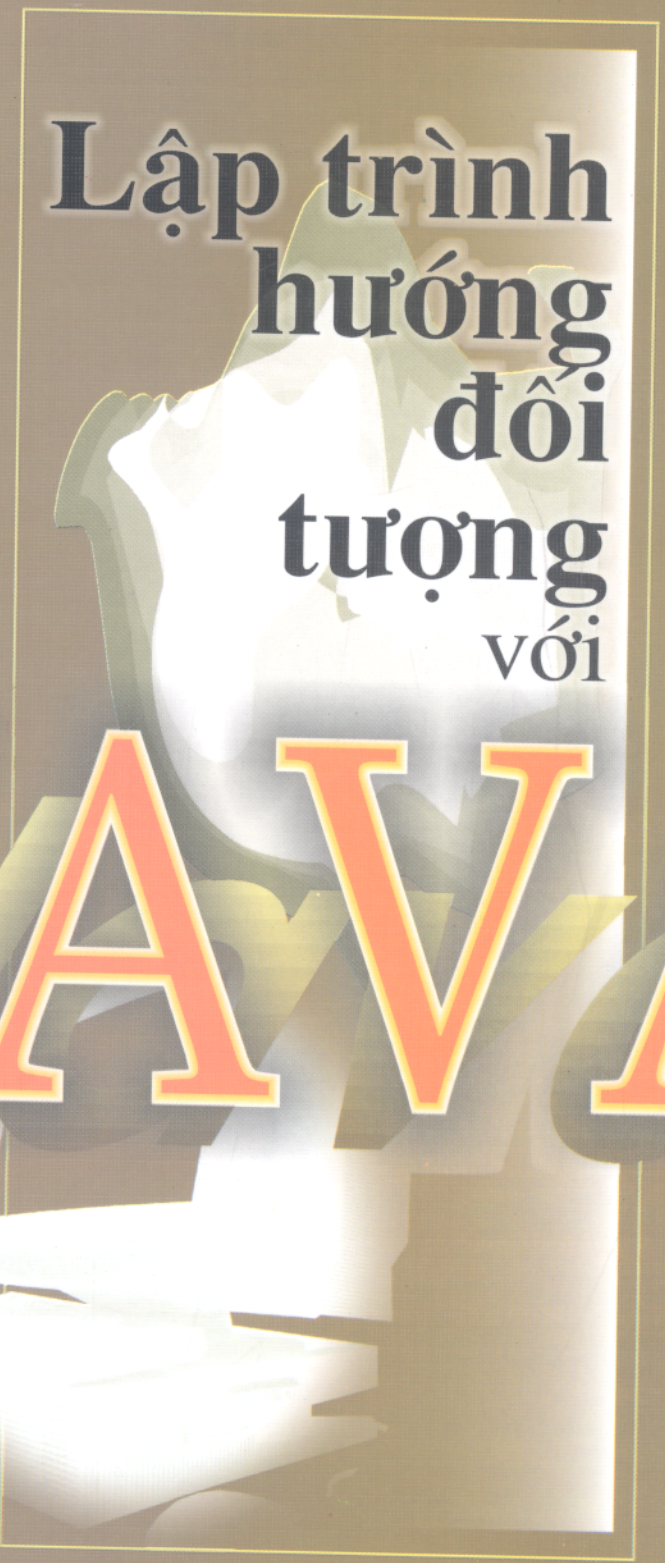
ĐOÀN VĂN BAN

Lập trình hướng đối tượng với JAVA



NHÀ XUẤT BẢN KHOA HỌC VÀ KỸ THUẬT

ĐOÀN VĂN BAN

LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG VỚI JAVA

(Tái bản lần thứ nhất: có hiệu chỉnh và bổ sung)

NHÀ XUẤT BẢN KHOA HỌC VÀ KỸ THUẬT
HÀ NỘI - 2005

LỜI NÓI ĐẦU

Các phương pháp hướng đối tượng, đặc biệt là *lập trình hướng đối tượng* được xây dựng dựa trên nhiều khái niệm mới và được hỗ trợ bởi nhiều công cụ, ngôn ngữ lập trình ([2], [8]) rất mạnh giúp cho việc tạo ra những phần mềm ứng dụng có chất lượng cao, ngày càng đáp ứng tốt hơn yêu cầu của người sử dụng.

Ngôn ngữ Java do gãng Sun (http://java.sun.com) phát triển từ đầu những năm 90 đã trở thành ngôn ngữ lập trình hướng đối tượng rất được ưa chuộng trong những năm gần đây nhờ một số đặc điểm hết sức thích hợp với mạng Internet, hiện đã và đang được dùng phổ biến trên toàn thế giới nhằm đáp ứng các yêu cầu phát triển các ứng dụng trên mạng phục vụ cho nhiều người sử dụng với những môi trường thực hiện phần mềm khác nhau, Java là một ngôn ngữ lập trình hoàn chỉnh được thiết kế theo cách tiếp cận hướng đối tượng và kế thừa, sử dụng lại có nâng cấp của những ngôn ngữ lập trình trước nó.

Về mặt cú pháp, Java rất giống với C++, một ngôn ngữ lập trình hướng đối tượng dùng phổ biến nhất hiện nay, nhưng loại đi một số tính khả dụng (*facilities*) quá mạnh nhưng khó và ít dùng, hoặc thừa về mặt ngôn ngữ [1] như: loại bỏ kế thừa nhiều lớp, vì chúng có thể tạo ra những lược đồ dữ liệu dạng đồ thị và là nguyên nhân chính có thể gây ra sự phức tạp và không đảm bảo tính nhất quán, tính đúng đắn trong quan hệ thông tin ([10], [11]); Java không cho phép thao tác số học trên kiểu con trỏ vì đây là nguồn gốc của những "con bọ" rất khó phát hiện ra khi biên dịch, v.v... Mục đích chính của Java là: *đơn giản, thân thiện, hướng đối tượng và cách tân nhằm tạo ra những phần mềm ứng dụng độc lập với môi trường sử dụng* ([7], [8]).

Cuốn sách này giới thiệu về lập trình hướng đối tượng sử dụng ngôn ngữ lập trình Java. Nội dung chính của cuốn sách được trình bày trong mười chương.

Chương I trình bày khái quát cách tiếp cận hướng chức năng và lập trình hướng đối tượng. Ngôn ngữ mô hình hóa hệ thống UML [2] được sử dụng để đặc tả các khái niệm cơ bản của lập trình hướng đối tượng. Chương II giới thiệu chu trình phát triển của các chương trình Java, quá trình dịch, thông dịch với JVM (Java Virtual Machine) và các thể loại chương trình ứng dụng, nhất là ứng dụng nhúng (*applet*) của Java. Chương III và chương IV trình bày những khái niệm cơ sở nhất của một ngôn ngữ lập trình và nêu cách xây dựng, tổ chức lớp các đối tượng trong các chương trình ứng dụng. Các lệnh điều khiển dòng thực hiện chương trình, đặc biệt là cơ chế xử lý ngoại lệ hỗ trợ để tạo ra những chương trình hoạt động tốt trong mọi tình huống, thích ứng được với mọi điều kiện trên cơ sở kiểm soát được các lỗi, các tình

huống có thể xảy ra, được giới thiệu chi tiết ở chương V. Chương VI đề cập đến một số lớp cơ sở nhất của Java và các kiểu cấu trúc dữ liệu phổ dụng như kiểu tuyển tập (*Collection*), kiểu tập hợp (Set), kiểu danh sách (List), v.v. Vấn đề phát triển những ứng dụng *applet* và sử dụng giao diện đồ họa của Windows (AWT) dưới dạng các trang Web với nhiều ví dụ minh họa được giới thiệu ở chương VII. Chương VIII giới thiệu các lớp xử lý các luồng dữ liệu vào/ra chuẩn và những vấn đề có tổ chức, đọc, ghi lên các loại tệp dữ liệu. Chương IX trình bày vấn đề kết nối các cơ sở dữ liệu với JDBC nhằm tạo ra những hệ thống phần mềm tích hợp từ nhiều loại hệ thống thông tin khác nhau trên mạng. Lần tái bản này được bổ sung thêm chương X giới thiệu các thành phần của Swing, cho phép tạo ra những phần mềm mà bạn "thấy và cảm nhận được". Trong các chương có nhiều ví dụ là những chương trình hoàn chỉnh, minh họa cho cách sử dụng những khái niệm đã nêu ở trên.

Cuốn sách được biên soạn dựa trên kinh nghiệm giảng dạy giáo trình phân tích, thiết kế và lập trình hướng đối tượng của tác giả trong nhiều năm tại các khóa cao học, đại học của ĐH Quốc Gia Hà Nội, ĐH Bách Khoa Hà Nội, ĐH Khoa Học Huế, v.v. Cuốn sách có thể dùng làm giáo trình học tập, tài liệu tham khảo cho sinh viên các hệ kỹ sư, cử nhân, học viên cao học CNTT và các bạn quan tâm đến vấn đề lập trình hướng đối tượng để phát triển những ứng dụng độc lập với môi trường, hay để xây dựng các Web Site trên mạng.

Tác giả bày tỏ lòng biết ơn chân thành tới các bạn đồng nghiệp trong Phòng các Hệ thống phần mềm tích hợp, đặc biệt cám ơn TS. Đặng Thành Phu, Viện Công nghệ Thông tin, TT KHTN & CNQG, PTS.TS. Đỗ Đức Giáo, Khoa Công nghệ, ĐH QG HN đã động viên, góp ý và giúp đỡ để hoàn chỉnh nội dung cuốn sách này. Xin cám ơn Nhà xuất bản Khoa học và Kỹ thuật đã hỗ trợ và tạo điều kiện để cuốn sách được tái bản.

Mặc dù rất cố gắng nhưng tài liệu này chắc chắn không tránh khỏi những sai sót. Chúng tôi rất mong nhận được các ý kiến đóng góp của bạn đọc để có chỉnh lý kịp thời.

Thư góp ý xin gửi về: Nhà xuất bản Khoa học và Kỹ thuật 70 Trần Hưng Đạo Hà Nội.

Hà Nội, tháng 5 năm 2005

Tác giả

CHƯƠNG I

GIỚI THIỆU VỀ LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG

1.1. CÁC CÁCH TIẾP CẬN TRONG LẬP TRÌNH

Phương pháp lập trình truyền thống chúng ta vẫn áp dụng đó là *lập trình có cấu trúc*. Phương pháp lập trình này thực hiện theo cách tiếp cận *hướng chức năng dựa chủ yếu vào việc phân tách các chức năng chính* của bài toán thành những chức năng đơn giản hơn và thực hiện làm mịn dần từ trên xuống (*top-down*). Theo cách tiếp cận hướng chức năng, chương trình được xem như là một tập các hàm chức năng, trong đó dữ liệu và các hàm là tách rời nhau. Đối với những hệ thống lớn, phức hợp, độ phức tạp của chương trình tăng lên, sự phụ thuộc của nó vào các kiểu dữ mà nó xử lý cũng tăng theo. Các kiểu dữ liệu được xử lý trong nhiều chức năng bên trong chương trình có cấu trúc, và khi có sự thay đổi về kiểu dữ liệu thì cũng phải thực hiện thay đổi ở mọi nơi mà dữ liệu đó được sử dụng. Một nhược điểm nữa của lập trình hướng chức năng là khi có nhiều người tham gia xây dựng chương trình, mỗi người được giao viết một số chức năng (hàm) riêng biệt nhưng lại phải sử dụng chung dữ liệu. Khi có nhu cầu cần thay đổi dữ liệu sẽ ảnh hưởng rất lớn đến công việc của nhiều người, v.v.

Phương pháp lập trình mà ngày nay chúng ta nghe nói tới nhiều đó là *lập trình hướng đối tượng*. Lập trình hướng đối tượng dựa trên nền tảng là các *đối tượng*. Đối tượng (thực thể) được xây dựng trên cơ sở gắn dữ liệu với các phép toán sẽ thể hiện được đúng cách mà chúng ta suy nghĩ, bao quát về chúng trong thế giới thực [3], [5]. Chẳng hạn, ô tô có bánh xe, di chuyển được và hướng của nó thay đổi được bằng cách thay đổi tay lái. Tương tự, cây là loại thực vật có thân gỗ và lá. Cây không phải là ô tô và những gì thực hiện được với ô tô sẽ không làm được với cây.

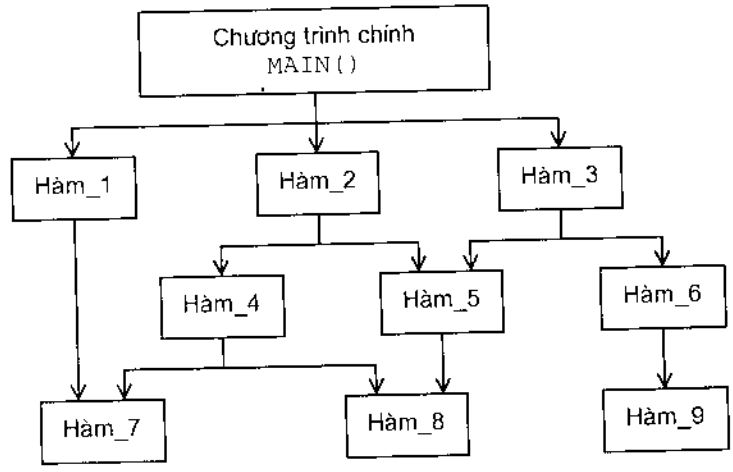
Lập trình hướng đối tượng cho phép chúng ta kết hợp những tri thức bao quát về các quá trình thực tế với những khái niệm trừu tượng được sử dụng trong máy tính. Chương trình hay hệ thống hướng đối tượng được xem như là tập các lớp đối tượng tương tác với nhau để thực hiện các yêu cầu của bài toán đặt ra.

Như vậy, hiện nay chúng ta có hai cách tiếp cận cơ bản để phát triển các hệ thống phần mềm: đó là cách tiếp cận hướng chức năng (*Function-Oriented*) và hướng đối tượng (*Object-Oriented*). Cả hai cách tiếp cận này đều áp dụng một nguyên lý chung để mô hình hóa hệ thống (để hiểu hệ thống) là thực hiện "*chia để trị*", phân chia bài toán thành những đơn vị tương đối đơn giản mà có thể dễ dàng quản lý được chúng một cách có hiệu quả.

Để hiểu rõ và áp dụng hiệu quả những phương pháp lập trình cần lựa chọn, chúng ta cần phân biệt những đặc trưng cơ bản nhất, đánh giá những mặt mạnh, mặt yếu của chúng.

1.1.1. LẬP TRÌNH HƯỚNG CHỨC NĂNG (THỦ TỤC)

Những ngôn ngữ lập trình bậc cao truyền thống như COBOL, FORTRAN, PASCAL, C v.v..., được gọi chung là ngôn ngữ lập trình *hướng chức năng*. Theo cách tiếp cận hướng chức năng thì một hệ thống phần mềm được xem như là dãy các công việc (chức năng) cần thực hiện như nhập dữ liệu, tính toán, xử lý, lập báo cáo và in ấn kết quả v.v... Mỗi công việc đó sẽ được thực hiện bởi một số hàm nhất định. Như vậy *trọng tâm của cách tiếp cận này là các hàm chức năng*. Cấu trúc của chương trình được xây dựng theo cách tiếp cận hướng chức năng có dạng như hình 1.1.

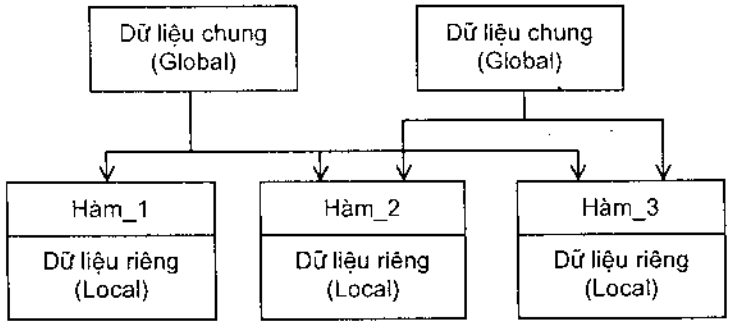


Hình 1.1. Cấu trúc của chương trình hướng chức năng.

Lập trình hướng chức năng (LTHCN) sử dụng *kỹ thuật phân rã các chức năng (hàm) theo cách tiếp cận top-down để tạo ra cấu trúc phân cấp. Chương trình* được xây dựng theo cách tiếp cận hướng chức năng thực chất là *tập các chương trình con* (có thể xem như là các hàm) mà theo đó máy tính cần thực hiện để hoàn thành những nhiệm vụ đặt ra của hệ thống.

Khi đặt trọng tâm vào các hàm thì dữ liệu, những cái mà các hàm sử dụng để thực hiện công việc của mình lại trở thành thứ yếu. Cái gì sẽ xảy ra đối với dữ liệu và gắn

dữ liệu với các hàm như thế nào? cùng nhiều vấn đề khác cần phải giải quyết khi chúng ta muốn xây dựng các phương pháp phù hợp để phát triển những hệ thống phần mềm giải quyết những yêu cầu đặt ra của thực tế. Hơn nữa, hệ thống luôn là một thể thống nhất, do vậy *các hàm trong một chương trình phải có liên hệ, trao đổi được với nhau*. Như chúng ta đã biết, các hàm (các bộ phận chức năng) chỉ có thể trao đổi được với nhau *thông qua các tham số, nghĩa là phải sử dụng biến chung (global)*. Mỗi hàm có thể có vùng dữ liệu riêng còn gọi là *dữ liệu cục bộ (local)*. Mối quan hệ giữa dữ liệu và hàm trong chương trình hướng chức năng được mô tả trong hình 1.2.



Hình 1.2. ***Quan hệ giữa dữ liệu và hàm trong LTHCN.***

Nhiều hàm có thể truy nhập, *sử dụng dữ liệu chung, làm thay đổi giá trị của chúng và vì vậy rất khó kiểm soát*. Nhất là đối với các chương trình lớn, phức tạp thì vấn đề càng trở nên khó khăn hơn. Khi chúng ta muốn thay đổi, bổ sung cấu trúc dữ liệu dùng chung cho một số hàm thì chúng ta phải thay đổi hầu như tất cả các hàm liên quan đến dữ liệu đó. Nhất là đối với những chương trình, dự án tin học lớn, phức tạp đòi hỏi nhiều người, nhiều nhóm tham gia thì những thay đổi của những biến dữ liệu chung sẽ *ảnh hưởng tới tất cả những ai có liên quan tới chúng*.

Một đặc tính nữa của cách tiếp cận hướng chức năng dễ nhận thấy là *tính mở (open) của hệ thống kém*. Thứ nhất, vì *dựa chính vào chức năng mà trong thực tế thì nhiệm vụ của hệ thống lại hay thay* đổi nên khi đó muốn cho hệ thống đáp ứng các yêu cầu thì phải thay đổi lại cấu trúc của hệ thống, nghĩa là phải thiết kế, lập trình lại hệ thống. Thứ hai, việc sử dụng các biến dữ liệu chung trong chương trình làm cho các nhóm chức năng phụ thuộc vào nhau về cấu trúc dữ liệu nên cũng hạn chế tính mở của hệ thống. Trong thực tế, cơ cấu tổ chức của mọi tổ chức thường ít thay đổi hơn là chức năng, nhiệm vụ phải thực hiện.

Mặt khác, cách tiếp cận hướng chức năng lại *tách dữ liệu khỏi chức năng xử lý* nên vấn đề che giấu, bảo vệ thông tin trong hệ thống là kém, nghĩa là vấn đề *an toàn, an ninh dữ liệu* là rất phức tạp.

Ngoài ra cách tiếp cận hướng chức năng cũng không hỗ trợ *việc sử dụng lại và kế*

thừa nên chất lượng và giá thành của các phần mềm rất khó được cải thiện. Những trở ngại mà chúng ta đã nêu ở trên sẽ làm cho mô hình được xây dựng theo cách tiếp cận hướng chức năng không mô tả được đầy đủ, trung thực hệ thống trong thực tế.

1.1.2. LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG

Ngược lại, lập trình hướng đối tượng (LTHĐT) *đặt trọng tâm vào các đối tượng*, yếu tố quan trọng trong quá trình phát triển chương trình và nó không cho phép dữ liệu tách biệt, chuyển động tự do trong hệ thống. Dữ liệu được gắn chặt với các hàm thành phần và chúng được tổ chức, quản lý truy nhập theo nhiều mức khác nhau.

LTHĐT cho phép chúng ta phân tích bài toán thành *tập các thực thể* được gọi là các *lớp đối tượng*, sau đó xây dựng các dữ liệu thành phần cùng với các hàm thành phần thao tác trên các dữ liệu đó và trao đổi với những đối tượng khác để thực hiện những nhiệm vụ được giao.

Một chương trình, một hệ thống được xem như là một tập các lớp đối tượng và các đối tượng đó trao đổi với nhau thông qua việc gửi và nhận các thông điệp (message), do vậy một chương trình hướng đối tượng thực sự có thể hoàn toàn không cần sử dụng biến chung [1], [3].

Lập trình hướng đối tượng *dựa chủ yếu vào các đối tượng*, nên khi có nhu cầu thay đổi thì chỉ cần thay đổi ở một số lớp có liên quan, hoặc có thể bổ sung một số lớp mới trên cơ sở kế thừa và sử dụng lại nhiều nhất có thể. Mặt khác, như trên đã phân tích, một chương trình hướng đối tượng có thể không sử dụng hoặc hạn chế sử dụng các biến chung, do vậy dễ dàng tạo ra được những hệ thống có *tính mở* cao hơn.

Cơ chế bao bọc, che giấu thông tin của phương pháp hướng đối tượng giúp tạo ra được những *hệ thống an toàn, an ninh cao* hơn cách tiếp cận hướng chức năng.

Hơn nữa, phương pháp hướng đối tượng còn hỗ trợ rất mạnh nguyên lý *sử dụng lại nhiều nhất có thể và tạo ra mọi khả năng để kế thừa* những lớp đã được thiết kế, lập trình tốt để nhanh chóng tạo ra được những phần mềm có chất lượng, giá thành rẻ hơn và đáp ứng các yêu cầu của người sử dụng.

1.2. NHỮNG KHÁI NIỆM CƠ BẢN CỦA LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG

Để tiện lợi và phù hợp với xu thế phát triển hiện nay của công nghệ thông tin, chúng ta sẽ sử dụng UML (*Unified Modelling Language* [2], [7]) để đặc tả những khái niệm cơ bản của lập trình hướng đối tượng thông qua *Rational Rose* (http://www.rational.com) . UML là *ngôn ngữ mô hình hóa hình thức, thống nhất và*

trực quan. Phần lớn các thông tin trong mô hình được thể hiện bởi các ký hiệu đồ họa, biểu đồ thể hiện mối quan hệ giữa các thành phần của hệ thống một cách thống nhất và có logic chặt chẽ.

UML được sử dụng để *đặc tả, xây dựng và làm tài liệu cho các tác phẩm (Artifacts), các kết quả của của các pha phân tích, thiết kế và lập trình hướng đối tượng* dưới dạng các biểu đồ, bản mẫu hay các trang Web.

UML là ngôn ngữ chuẩn công nghiệp để lập kế hoạch chi tiết phát triển phần mềm. Hiện nay nhiều hãng sản xuất phần mềm lớn như: Microsoft, IBM, HP, Oracle, Digital Equipment Corp., Texas Instruments, Rational Software, v.v., sử dụng UML như là chuẩn cho ngôn ngữ mô hình hóa hệ thống phần mềm.

Phương pháp hướng đối tượng được xây dựng dựa trên một tập các khái niệm cơ sở:

1. Đối tượng (object),
2. Lớp đối tượng (class),
3. Trừu tượng hóa dữ liệu (Data Abstraction),
4. Bao bọc và che giấu thông tin (Encapsulation and Information Hiding),
5. Mở rộng, kế thừa giữa các lớp (Inheritance),
6. Đa xạ và nạp chồng (Polymorphism and Overloading),
7. Liên kết động (Dynamic Binding),
8. Truyền thông điệp (Message Passing).

1.2.1. ĐỐI TƯỢNG

Đối tượng là khái niệm cơ sở, quan trọng nhất của cách tiếp cận hướng đối tượng. Đối tượng là thực thể của hệ thống, của CSDL và được xác định thông qua định danh ID (*IDentifier*) của chúng. Mỗi *đối tượng có tập các đặc trưng bao gồm* cả các phần tài sản thường là *các dữ liệu thành phần* hay các thuộc tính mô tả các tính chất và *các phương thức, các thao tác trên các dữ liệu* để xác định hành vi của đối tượng đó.

Đối tượng là những thực thể được xác định trong thời gian hệ thống hướng đối tượng hoạt động. Như vậy đối tượng có thể biểu diễn cho người, vật, hay một bảng dữ liệu hoặc bất kỳ một hạng thức nào đó cần xử lý trong chương trình. Đối tượng cũng có thể là các dữ liệu được định nghĩa bởi người sử dụng (người lập trình) như *vector*, danh sách, các *record* v.v... Nhiệm vụ của LTHĐT là phân tích bài toán thành các đối tượng và xác định được bản chất của sự trao đổi thông tin giữa chúng. Đối tượng trong chương trình cần phải được chọn sao cho nó thể hiện được một cách gần nhất với những thực thể có trong hệ thống thực.

Như vậy, đối tượng được định nghĩa một cách trừu tượng như là một khái niệm, một

cấu trúc gộp chung cả phần dữ liệu (thuộc tính) với các hàm (phương thức) thao tác trên những dữ liệu đó và có thể trao đổi với những đối tượng khác. Theo quan điểm của người lập trình [3], *đối tượng được xem như là vùng bộ nhớ được phân chia trong máy tính để lưu trữ dữ liệu và tập các hàm tác động trên dữ liệu gắn với chúng*. Bởi vì các vùng phân hoạch bộ nhớ là độc lập với nhau nên các đối tượng có thể sử dụng bởi nhiều chương trình khác nhau mà không ảnh hưởng lẫn nhau.

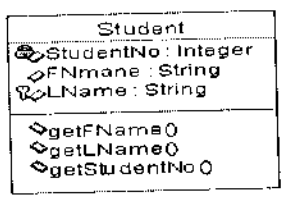
1.2.2. LỚP ĐỐI TƯỢNG

Lớp là bản mẫu hay một kiểu chung cho tất cả những đối tượng có những *đặc trưng giống nhau*, nghĩa là có các thuộc tính và hành vi giống nhau. Đối tượng chính là thể hiện (cá thể) của một lớp xác định. Trong lập trình hướng đối tượng, lớp được xem là đồng nhất với kiểu dữ liệu trừu tượng (ADT - Abstract Data Type được Barbara đề xuất vào những năm 70).

Phương pháp lập trình *hướng đối tượng* là cách phân chia chương trình thành các đơn thể (các lớp) bằng cách tạo ra các vùng bộ nhớ cho cả dữ liệu lẫn hàm và chúng sẽ được sử dụng như các mẫu để tạo ra bản sao từng đối tượng khi chúng được tạo ra trong hệ thống.

Nhiệm vụ của người lập trình là tìm cách xác định đầy đủ, chính xác danh sách các lớp đại diện cho tất cả các thực thể trong hệ thống và mối quan hệ giữa các lớp đối tượng đó [3], [5].

Như vậy, lớp chính là tập các đối tượng có cùng các thuộc tính và hành vi giống nhau. Các thành phần của lớp có thể chia thành ba vùng quản lý chính: công khai (*public*), được bảo vệ (*protected*) và vùng riêng (*private*). Trong ngôn ngữ mô hình hóa thống nhất UML, cấu trúc của lớp thường được đặc tả bởi: tên của lớp, tập các thuộc tính và tập các hàm.



Hình 1.3. ***Lớp* Student *trong UML.***

Lớp có tên là *Student*. Lớp có ba thuộc tính: *StudentNo* có kiểu *Integer* là sở hữu riêng (*private*), bị khóa; thuộc tính *FName* có kiểu *String* là công khai (public) và thuộc tính *LName* là được bảo vệ (*protected*). Lớp *Student* có ba hàm : *getStudentNo(), getFName(), getLName()* đều là công khai.