ĐẠI HỌC THÁI NGUYÊN
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

Chủ biên: TS. Nguyễn Văn Núi,
Đồng chủ biên: ThS. Trần Hải Thanh,
Thành viên: TS. Tô Hữu Nguyên,
Thành viên: ThS. Phạm Thị Thương

GIÁO TRÌNH

LẬP TRÌNH JAVA

Nhà xuất bản…..

ĐẠI HỌC THÁI NGUYÊN
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

Chủ biên: TS. Nguyễn Văn Núi,
Đồng chủ biên: ThS. Trần Hải Thanh,
Thành viên: TS. Tô Hữu Nguyên,
Thành viên: ThS. Phạm Thị Thương

GIÁO TRÌNH

LẬP TRÌNH JAVA

Nhà xuất bản…..
Thái nguyên - 2024

Giáo Trình Lập trình Java

TS. Nguyễn Văn Núi (Chủ biên)
ThS. Trần Hải Thanh (Đồng chủ biên)
TS. Tô Hữu Nguyên
ThS. Phạm Thị Thương

Nhà xuất Bản:
Năm Xuất bản:
IBSN:

MỤC LỤC

DANH MỤC TỪ VIẾT TẮT VÀ THUẬT NGỮ 7
DANH MỤC HÌNH ẢNH 8
DANH MỤC BẢNG BIỂU 10
LỜI NÓI ĐẦU 11
Chương 1 TỔNG QUAN 12
1.1. GIỚI THIỆU PHƯƠNG PHÁP TIẾP CẬN TRONG LẬP TRÌNH 12
1.1.1. Lập trình tuyến tính 12
1.1.2. Lập trình hướng chức năng 13
1.1.3. Lập trình hướng đối tượng 16
1.2. CÁC KHÁI NIỆM CƠ BẢN CỦA LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG 16
1.2.1. Đối tượng 16
1.2.2. Lớp đối tượng 18
1.2.3. Trừu tượng hóa 19
1.2.4. Bao bọc và che giấu thông tin 20
1.2.5. Kế thừa 21
1.2.6. Đa hình 22
1.3. CÁCH TIẾP CẬN VÀ GIẢI QUYẾT BÀI TOÁN THEO PHƯƠNG PHÁP HƯỚNG ĐỐI TƯỢNG 23
1.3.1. Mô tả bài toán 23
1.3.2. Đặc tả yêu cầu 24
1.3.3. Trích chọn đối tượng 24
1.3.4. Mô hình hoá lớp đối tượng 25
1.3.5. Thiết kế tổng quát 26
1.3.6. Thiết kế chi tiết 26
1.4. MỘT SỐ NGÔN NGỮ LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG 31
1.5. GIỚI THIỆU MỘT SỐ EDITOR BIÊN SOẠN CHƯƠNG TRÌNH JAVA 32
1.5.1. Eclipse 32
1.5.2. NetBeans 35
1.5.3. IntelliJ IDEA 37
1.6. TỔNG KẾT CHƯƠNG 39

1.7. CÂU HỎI TRẮC NGHIỆM 40
Chương 2 GIỚI THIỆU NGÔN NGỮ LẬP TRÌNH JAVA 43
2.1. GIỚI THIỆU CHUNG 43
2.1.1. Lịch sử hình thành và phát triển của Java 43
2.1.2. Ngữ cảnh các ứng dụng của ngôn ngữ Java 43
2.2. MÔI TRƯỜNG LẬP TRÌNH JAVA 45
2.2.1. Máy ảo Java (JVM - Java Virtual Machine) 45
2.2.2. Bộ công cụ phát triển JDK 47
2.2.3. Java Runtime Environment (JRE) 48
2.2.4. Java Integrated Development Environment (IDE) 50
2.2.5. Java Enterprise Edition (Java EE) 51
2.2.6. Java Database Connectivity (JDBC) 53
2.3. CẤU TRÚC CỦA MỘT CHƯƠNG TRÌNH JAVA 54
2.4. BIÊN DỊCH VÀ THÔNG DỊCH CHƯƠNG TRÌNH JAVA 55
2.5. CÁC DẠNG CHƯƠNG TRÌNH ỨNG DỤNG CỦA JAVA 56
2.5.1. Phân loại theo cách vận hành, dịch và chạy chương trình 56
2.5.2. Phân loại theo dạng chương trình ứng dụng, mục đích sử dụng 58
2.8. TỔNG KẾT CHƯƠNG 60
2.9. CÂU HỎI TRẮC NGHIỆM 61
2.10. BÀI TẬP THỰC HÀNH 65
Chương 3 CÁC THÀNH PHẦN CƠ SỞ CỦA JAVA 67
3.1. CÁC PHẦN TỬ CƠ SỞ CỦA JAVA 67
3.2. CÁC KIỂU DỮ LIỆU TRONG JAVA 68
3.3. BIẾN, HẰNG, BIỂU THỨC VÀ CÁC PHÉP TOÁN CƠ BẢN 69
3.3.1. Biến (variable) 69
3.3.2. Hằng (constant) 71
3.3.3. Các phép toán cơ bản trong Java 71
3.4. CÁC CÂU LỆNH NHẬP XUẤT DỮ LIỆU CƠ BẢN QUA CÁC THIẾT BỊ VÀO RA CHUẨN 74
3.4.1. Nhập dữ liệu từ bàn phím 74
3.4.2. Xuất dữ liệu ra màn hình 74

3.5. CÁC CẤU TRÚC ĐIỀU KHIỂN CHƯƠNG TRÌNH TRONG JAVA 76
3.5.1. Câu lệnh rẽ nhánh 76
3.5.2. Các câu lệnh lặp 79
3.5.3. Các lệnh chuyển vị 83
3.6. CẤU TRÚC MẢNG, XÂU KÝ TỰ TRONG JAVA 86
3.6.1. Cấu trúc mảng 86
3.6.2. Xâu ký tự 90
3.7. NGOẠI LỆ VÀ XỬ LÝ NGOẠI LỆ TRONG JAVA 95
3.7.1. Cấu trúc phân cấp của các lớp xử lý ngoại lệ 96
3.7.2. Khối lệnh try, catch và finally 100
3.7.3. Ném ngoại lệ bằng lệnh 'throw' 102
3.7.4. Mệnh đề throws 103
3.7.5. Tự định nghĩa ngoại lệ 104
3.7.6. Lần vết ngoại lệ StackTrace 106
3.8. CASE STUDY 107
3.9. VÍ DỤ ÁP DỤNG 109
3.10. TỔNG KẾT CHƯƠNG 112
3.11. CÂU HỎI TRẮC NGHIỆM 112
3.12. BÀI TẬP THỰC HÀNH 117
Chương 4 LỚP ĐỐI TƯỢNG VÀ CÁC THÀNH PHẦN CỦA LỚP 118
4.1. LỚP ĐỐI TƯỢNG 118
4.1.1. Định nghĩa lớp đối tượng và các thành phần của lớp 118
4.1.2. Các thuộc tính dữ liệu của lớp 119
4.1.3. Định nghĩa hàm thành phần 119
4.1.4. Truyền tham số và lời gọi hàm 122
4.1.5. Toán tử tạo lập đối tượng 125
4.2. CÁC THUỘC TÍNH KIỂM SOÁT TRUY NHẬP CÁC THÀNH PHẦN CỦA LỚP 128
4.3. TẠO VÀ SỬ DỤNG ĐỐI TƯỢNG 130
4.4. CASE STUDY 132
4.5. VÍ DỤ ÁP DỤNG 133

4.6. TỔNG KẾT CHƯƠNG 138
4.7. CÂU HỎI TRẮC NGHIỆM 138
4.8. BÀI TẬP THỰC HÀNH 143
Chương 5 GÓI CÁC LỚP VÀ CÁC LỚP CƠ SỞ TRONG GÓI JAVA.LANG 145
5.1. GÓI 145
5.1.1 Tạo gói cho chương trình và cách sử dụng gói 145
5.1.2 Giới thiệu một số gói quan trọng của Java 146
5.2. CÁC LỚP CƠ SỞ TRONG GÓI JAVA.LANG 147
5.2.1. Lớp Object 147
5.2.2. Các lớp nguyên thủy 149
5.2.3. Lớp Math 152
5.2.4. Lớp String 154
5.2.5. Lớp StringBuffer 157
5.3. CASE STUDY 160
5.4. VÍ DỤ ÁP DỤNG 162
5.5. TỔNG KẾT CHƯƠNG 164
5.6. CÂU HỎI TRẮC NGHIỆM 164
5.7. BÀI TẬP THỰC HÀNH 169
Chương 6 KẾ THỪA 172
6.1. QUAN HỆ KẾ THỪA 172
6.2. THIẾT KẾ CÂY THỪA KẾ 174
6.3. VIẾT ĐÈ 176
6.4. QUAN HỆ “IS – A” VÀ “HAS – A” 178
6.4.1 Quan hệ “is – a” 178
6.4.2 Quan hệ “has – a” 179
6.5. CASE STUDY 181
6.6. Ví dụ áp dụng 181
6.7. TỔNG KẾT CHƯƠNG 185
6.8. CÂU HỎI TRẮC NGHIỆM 185
6.9. BÀI TẬP THỰC HÀNH 190
Chương 7 ĐA HÌNH VÀ LỚP TRỪU TƯỢNG 192

7.1. ĐA HÌNH 192
7.1.1. Đa hình tĩnh 192
7.1.2. Đa hình động 193
7.1.3. Đặc điểm của đa hình 195
7.2. LỚP TRỪU TƯỢNG 196
7.2.1. Khái niệm 196
7.2.2. Lớp trừu tượng và lớp cụ thể 197
7.2.3. Phương thức trừu tượng 198
7.2.4. Lớp Object 200
7.3. CASE STUDY 201
7.4. VÍ DỤ ÁP DỤNG 201
7.5. TỔNG KẾT CHƯƠNG 203
7.6. CÂU HỎI TRẮC NGHIỆM 204
7.7. BÀI TẬP THỰC HÀNH 208
Chương 8 GIAO DIỆN VÀ CÁC LỚP TRONG 210
8.1. GIAO DIỆN (INTERFACES) 210
8.1.1. Định nghĩa 210
8.1.2. Các tính chất của giao diện 213
8.1.3. Sử dụng giao diện 213
8.2. LỚP TRONG (INNER CLASSES) 216
8.2.1. Định nghĩa 216
8.2.2. Tính chất của lớp nội 216
8.2.3. Phân loại lớp nội 216
8.3. CASE STUDY 220
8.4. VÍ DỤ ÁP DỤNG 222
8.5. TỔNG KẾT CHƯƠNG 224
8.6. CÂU HỎI TRẮC NGHIỆM 225
8.7. BÀI TẬP THỰC HÀNH 230
Chương 9 MỘT SỐ CHỦ ĐỀ NÂNG CAO 231
9.1. MỘT SỐ CẤU TRÚC DỮ LIỆU TRONG JAVA 231
9.1.1. ArrayList 231

9.1.2. LinkedList 233
9.1.3. HashMap 235
9.1.4. TreeSet 236
9.1.5. Stack và Queue 238
9.1.6. Lớp Collection 240
9.2. CÁC LUỒNG VÀO/RA DỮ LIỆU TRONG JAVA 244
9.2.1. Khái niệm 244
9.2.2. Input Streams 245
9.2.2. Output Streams 247
9.2.3. Character Streams 248
9.2.4. Object Streams 251
9.3. TỔ CHỨC MÃ NGUỒN THEO MÔ HÌNH HƯỚNG ĐỐI TƯỢNG 254
9.3.1. Nhóm các lớp 254
9.3.2. Áp dụng các nguyên tắc hướng đối tượng 257
9.3.3. Sử dụng JavaDoc 259
9.4. CASE STUDY 267
9.5. VÍ DỤ ÁP DỤNG 269
9.6. TỔNG KẾT CHƯƠNG 272
9.7. CÂU HỎI TRẮC NGHIỆM 273
9.8. BÀI TẬP THỰC HÀNH 278
TÀI LIỆU THAM KHẢO 281

DANH MỤC TỪ VIẾT TẮT VÀ THUẬT NGỮ

TT	Từ viết tắt	Ý nghĩa
1	API	Application Programming Interface - tập hợp các quy tắc hoặc giao thức cho phép các ứng dụng phần mềm giao tiếp với nhau
2	AWT	Abstract Window Toolkit - một bộ công cụ phát triển các ứng dụng dựa trên GUI
3	CLI	Command Line Interface - Giao diện dòng lệnh
4	FR	Functional Requirements - Yêu cầu chức năng
5	GUI	Graphical User Interface - Giao diện đồ họa người dùng
6	IDE	Integrated Development Environment - Môi trường phát triển tích hợp
7	Java EE	Java Enterprise Edition - Công cụ phát triển phần mềm doanh nghiệp
8	Java ME	Java Micro Edition - Công cụ phát triển các ứng dụng trên điện thoại di động với Java
9	JavaSE	Java Standard Edition - Công cụ cơ bản phát triên phần mềm Java
10	JDBC	Java Database Connectivity - API chuẩn trong Java được sử dụng để kết nối và thực hiện các truy vấn với cơ sở dữ liệu
11	JDK	Java Develop Kit - Bộ công cụ phát triển ứng dụng Java
12	JRE	Java Runtime Environment - Môi trường thực thi cho các ứng dụng Java
13	JVM	Java Virtual Machine - Máy ảo Java
14	ODBC	Open Database Connectivity - giao thức tiêu chuẩn để kết nối và truy xuất dữ liệu từ các cơ sở dữ liệu
15	OOP	Object-oriented programming - Lập trình hướng đối tượng
16	PP	Procedural Programming - Lập trình thủ tục
17	SWT	Standard Widget Toolkit - một bộ công cụ đồ họa cho Java

DANH MỤC HÌNH ẢNH

Hình 1.1 Cấu trúc của chương trình theo hướng chức năng 13
Hình 1.2 Sơ đồ chức năng của Hệ thống quản lý thư viện 14
Hình 1.3 Quan hệ giữa dữ liệu và hàm trong lập trình hướng chức năng 15
Hình 1.4 Tập các đối tượng của hệ thống quản lý thư viện 16
Hình 1.5 Lớp đối tượng 19
Hình 1.6 Trừu tượng hoá 19
Hình 1.7 Sử dụng trừu tượng hoá để xây dựng lớp 20
Hình 1.8 Ví dụ kế thừa 22
Hình 1.9 Đa hình 22
Hình 1.10 Ví dụ đa hình 23
Hình 1.11 Tải Eclipse 33
Hình 1.12 Lựa chọn cài đặt Eclipse 34
Hình 1.13 Giao diện mở đầu của Eclipse 34
Hình 1.14 NetBeans 35
Hình 1.15 Cài đặt NetBeans 37
Hình 1.16 IntelliJ IDEA 37
Hình 1.17 Sử dụng IntelliJ IDEA 39
Hình 2.1 Chương trình java chạy trên mọi hệ điều hành 44
Hình 2.2 Kiến trúc máy ảo Java 46
Hình 2.3 Java Runtime Environment 48
Hình 2.4 Integrated Development Environment 50
Hình 2.5 Kiến trúc JDBC 53
Hình 2.6 Chạy và biên dịch chương trình java 56
Hình 3.1 Các kiểu dữ liệu nguyên thủy trong Java 69
Hình 3.2 Giá trị mặc định của các kiểu dữ liệu nguyên thuỷ 69
Hình 3.3 Mở rộng kiểu 73
Hình 3.4 Câu lệnh if đơn giản 76
Hình 3.5 Câu lệnh if … else 77
Hình 3.6 Câu lệnh switch 78
Hình 3.7 Mảng 86
Hình 3.8 Ngoại lệ 97
Hình 3.9 Quá trình xử lý ngoại lệ 99

Hình 3.10 Khối Try .... catch 100
Hình 3.11 Hoạt động của khối Try .... catch 101
Hình 4.1 Truyền các giá trị tham chiếu đối tượng 124
Hình 5.1 Lớp object 147
Hình 6.1 Cây kế thừa. 175
Hình 6.2 Xây dựng cây kế thừa 176
Hình 6.3 Quan hệ "is – a", hay quan hệ kế thừa 178
Hình 6.4 Ví dụ minh họa quan hệ "Association" 180
Hình 6.5 Quan hệ "Aggregation" 180
Hình 6.6 Ví dụ về quan hệ hợp thành (composition) 181
Hình 9.1 OutputStream và InputStream 245
Hình 9.2 InputStream Hierarchy 245
Hình 9.3 OutputStream Hierarchy 247
Hình 9.4 Character Streams Hierarchy 249
Hình 9.5 Ví dụ minh hoạ về sử dụng JavaDoc 264
Hình 9.6 Tạo JavaDoc với Eclipse IDE 265
Hình 9.7 Tạo JavaDoc với InteliJ IDE 266

DANH MỤC BẢNG BIỂU

Bảng 1.1 Bảng đặc tả yêu cầu của bài toán quản lý thư viện 24
Bảng 3.1 Các từ khóa của Java 68
Bảng 3.2 Các kí tự định dạng xuất dữ liệu 68
Bảng 3.3 Toán tử số học 71
Bảng 3.4 Toán tử quan hệ & logic 72
Bảng 3.5 Toán tử gán 72
Bảng 3.6 Dạng câu lệnh return 85
Bảng 3.7 So sánh for với foreach 89
Bảng 3.8 Ngoại lệ 98
Bảng 3.9 So sánh sự khác nhau giữa từ khóa throw và throws 104
Bảng 4.1 Thuộc tính kiểm soát truy nhập các thành phần của lớp 128
Bảng 5.1 Các phương thức quan trọng của lớp Object 148
Bảng 5.2 Một số phương thức tĩnh trong lớp Math 153
Bảng 6.1 Một số ví dụ về lớp cơ sở và lớp dẫn xuất 172
Bảng 9.1 Các phương thức quan trọng của ArrayList 231
Bảng 9.2 Các phương thức quan trọng của LinkList 233
Bảng 9.3 Các phương thức quan trọng của HashMap 235
Bảng 9.4 Các phương thức chính của TreeSet 237
Bảng 9.5 Các phương thức chính của Stack 238
Bảng 9.6 Các phương thức chính của Queue 239
Bảng 9.7 Các phương thức quan trọng trong Collection Framework 241
Bảng 9.8 Các thẻ JavaDoc phổ biến 261

LỜI NÓI ĐẦU

Giáo trình *"Lập trình Java"* được biên soạn bởi tập thể giảng viên thuộc Khoa Công nghệ thông tin - Trường Đại học Công nghệ thông tin và Truyền thông, nhằm phục vụ việc giảng dạy của giảng viên và học tập của sinh viên trong Trường. Nội dung giáo trình bám sát nội dung đề cương chi tiết của học phần *"Lập trình hướng đối tượng"* đào tạo ở trình độ đại học. Ngoài ra, giáo trình cũng cung cấp thêm các kiến thức mở rộng và nâng cao liên quan khác.

Nội dung giáo trình này tập trung vào việc trang bị những kiến thức cơ bản về lập trình hướng đối tượng, áp dụng với ngôn ngữ lập trình Java. Người đọc sẽ được trang bị các nguyên tắc cốt lõi, phương pháp giải quyết bài toán theo cách tiếp cận hướng đối tượng, cùng các bài thực hành thực tế để củng cố kiến thức. Đồng thời, giáo trình còn hướng dẫn cách khai thác và sử dụng các công cụ, công nghệ, và dịch vụ cơ bản của các IDE trong lập trình, từ đó giúp người học dễ dàng làm quen và áp dụng hiệu quả phương pháp hướng đối tượng trong lập trình. Những kiến thức trong giáo trình này sẽ là nền tảng quan trọng, hỗ trợ người học trên con đường trở thành kỹ sư lập trình chuyên nghiệp sau khi tốt nghiệp.

Nhằm đáp ứng mục tiêu đào tạo kết hợp sử dụng cách tiếp cận học và đánh giá theo quy trình khoa học, nội dung của cuốn giáo trình này được tổ chức thành một chuỗi gồm 9 chương, bám sát mục tiêu và các chuẩn đầu ra của học phần. Mỗi chương được thiết kế dưới dạng một bài học với cấu trúc tương đồng nhau, bao gồm các nội dung chính như sau:

1) Mục tiêu bài học;

2) Nội dung kiến thức trang bị nhằm đáp ứng mục tiêu bài học;

3) Tổng kết nội dung bài học;

4) Các câu hỏi củng cố, case study, ví dụ áp dụng hoặc bài tập thực hành.

Dù đã nỗ lực dành nhiều thời gian và tâm huyết để biên soạn, giáo trình chắc chắn khó tránh khỏi những thiếu sót nhất định. Chúng tôi rất mong nhận được những ý kiến đóng góp quý báu từ Quý Thầy/Cô và bạn đọc để tài liệu này ngày càng hoàn thiện hơn. Mọi ý kiến đóng góp xin gửi về địa chỉ email: nvnui@ictu.edu.vn. Xin chân thành cảm ơn!

Thái Nguyên, năm 2024

Đại diện nhóm tác giả

TS. Nguyễn Văn Núi

Chương 1 TỔNG QUAN

Chương đầu giới thiệu:

- Những đặc trưng cơ bản của hai cách tiếp cận trong lập trình: lập trình hướng chức năng và lập trình hướng đối tượng;
- Những khái niệm, thành phần cơ bản của phương pháp hướng đối tượng;
- Ngôn ngữ lập trình hướng đối tượng.

1.1. GIỚI THIỆU PHƯƠNG PHÁP 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. 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.

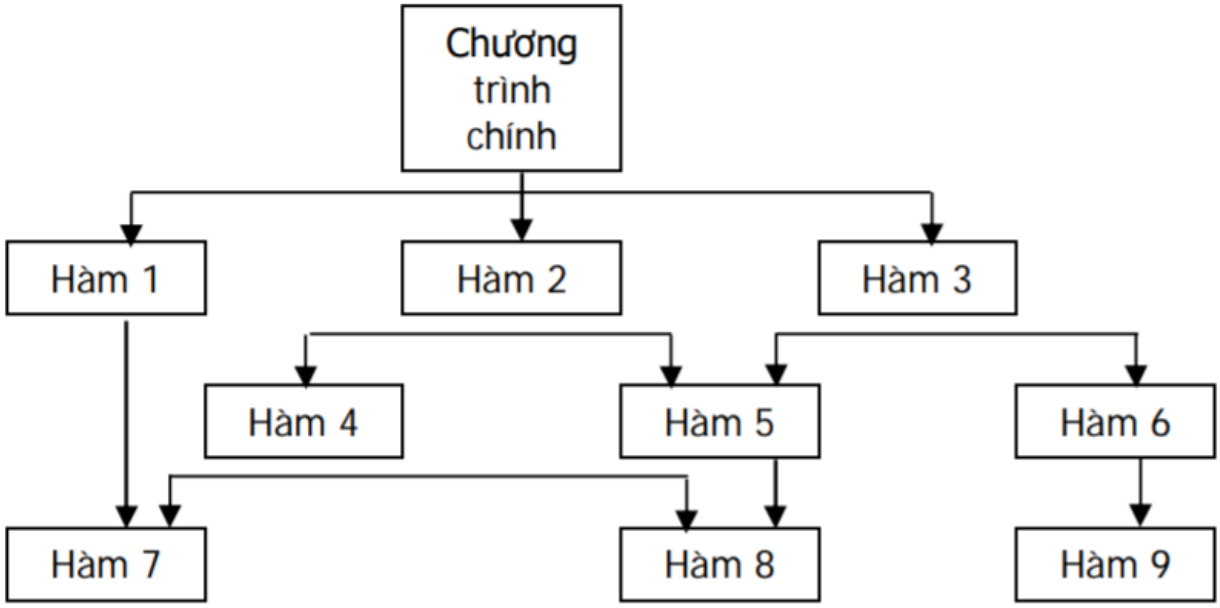
1.1.1. Lập trình tuyến tính

- Lập trình tuyến tính có 2 đặc trưng cơ bản sau:
  - + Đơn giản: chương trình được tiến hành đơn giản theo lối tuần tự, không phức tạp.
  - + Đơn luồng: Chỉ có một luồng công việc duy nhất, các công việc được thực hiện tuần tự trong các luồng đó.
- Ưu điểm: Chương trình đơn giản, dễ hiểu.
- Nhược điểm: Không thể dùng lập trình tuyến tính để giải quyết các bài toán phức tạp.

Ngày nay, lập trình tuyến tính chỉ tồn tại trong phạm vi các module nhỏ nhất của các phương pháp lập trình khác.

1.1.2. Lập trình hướng chức năng

Theo cách tiếp cận hướng chức năng thì một hệ thống phần mềm được xem như là là tập các công việc (chức năng, nhiệm vụ) cần tổ chức thực thi; 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*** dưới đây.



Hình 1.1 Cấu trúc của chương trình theo hướng chức năng

Cách tiếp cận theo hướng thủ tục dựa vào chức năng, nhiệm vụ là chính và phân rã chức năng làm mịn dần theo cách từ trên xuống (Top/Down). Các đơn thể chức năng trao đổi với nhau bằng cách truyền tham số hay sử dụng dữ liệu chung.

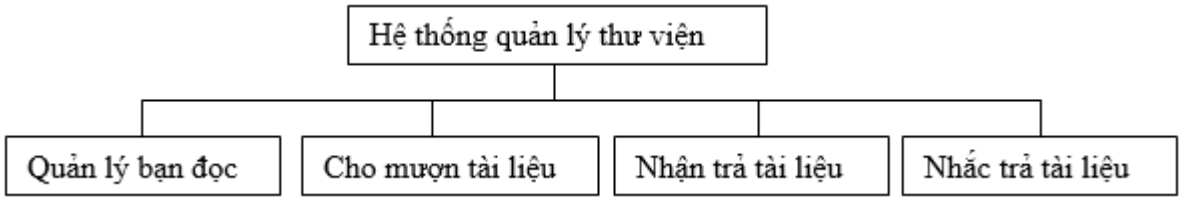
- ***Dựa vào chức năng nhiệm vụ***: Khi khảo sát, phân tích một hệ thống chúng ta thường tập trung vào các nhiệm vụ mà nó cần thực hiện. Chúng ta tập trung trước hết nghiên cứu các yêu cầu của bài toán để xác định các chức năng chính của hệ thống.

Ví dụ khi cần xây dựng "hệ thống quản lý thư viện" thì trước hết chúng ta thường đi nghiên cứu, khảo sát trao đổi và phỏng vấn xem những người thủ thư, bạn đọc cần phải thực hiện những công việc gì để phục vụ được bạn đọc và quản lý tốt được các tài liệu. Qua nghiên cứu "hệ thống quản lý thư viện", chúng ta xác định được các nhiệm vụ chính của hệ thống như: quản lý bạn đọc, cho mượn sách, nhận trả sách, thông báo nhắc trả sách, …

- ***Phân rã chức năng làm mịn từ trên xuống***: Khả năng của con người là có giới hạn khi khảo sát, nghiên cứu để hiểu và thực thi những gì mà hệ thống thực tế đòi hỏi. Để thống trị (quản lý được) độ phức tạp của những vấn đề phức tạp trong thực tế thường chúng ta phải sử dụng nguyên lý chia để trị (divide and conquer), nghĩa là phân tách nhỏ các chức năng chính thành các chức năng đơn giản hơn theo cách từ trên xuống. Quy trình này được lặp lại cho đến khi thu được những đơn thể chức năng tương đối đơn giản, hiểu được và thực hiện cài đặt chúng mà không làm tăng thêm độ phức tạp để liên kết chúng trong hệ thống.

Độ phức tạp liên kết các thành phần chức năng của hệ thống thường là tỉ lệ nghịch với độ phức tạp của các đơn thể. Vì thế một vấn đề đặt ra là có cách nào để biết khi nào quá trình phân tách các đơn thể chức năng hay còn gọi là quá trình làm mịn dần này kết thúc. Thông thường thì quá trình thực hiện phân rã các chức năng của hệ thống phụ thuộc nhiều vào độ phức hợp của bài toán ứng dụng và vào trình độ của những người tham gia phát triển phần mềm. Một hệ thống được phân tích dựa trên các chức năng hoặc quá trình sẽ được chia thành các hệ thống con và tạo ra cấu trúc phân cấp các chức năng.

Ví dụ 1: hệ thống quản lý thư viện có thể phân chia từ trên xuống như sau:

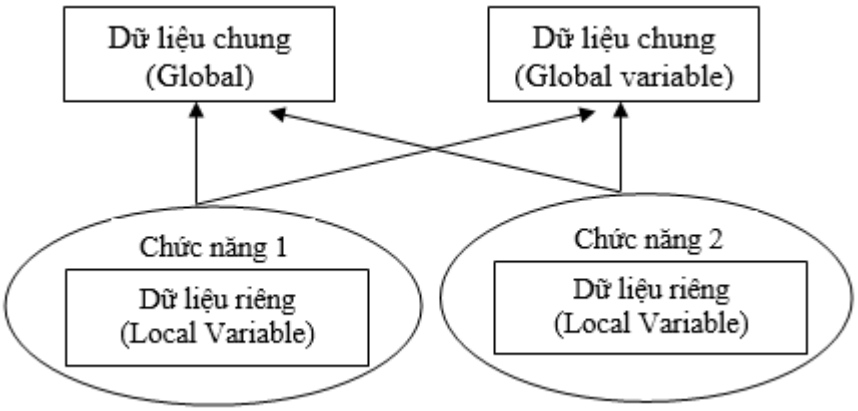


Hình 1.2 Sơ đồ chức năng của Hệ thống quản lý thư viện

Các chức năng của hầu hết các hệ thống thông tin quản lý đều có thể tổ chức thành sơ đồ chức năng theo cấu trúc phân cấp có thứ bậc.

- ***Các đơn thể chức năng trao đổi với nhau bằng cách truyền tham số hay sử dụng dữ liệu chung***: Một hệ thống phần mềm bao giờ cũng phải được xem như là một thể thống nhất, do đó các đơn thể chức năng phải có quan hệ trao đổi thông tin, dữ liệu với nhau. Trong một chương trình gồm nhiều hàm (thực hiện nhiều chức năng khác nhau) muốn trao đổi dữ liệu được với nhau thì nhất thiết phải sử dụng dữ liệu chung hoặc liên kết với nhau bằng cách truyền tham biến. Mỗi đơn thể chức năng không những chỉ thao tác, xử lý trên những dữ liệu cục bộ (Local Variables) mà còn phải sử dụng các biến chung, thường đó là các biến toàn cục (Global Variable).

Mối quan hệ giữa dữ liệu và hàm trong chương trình hướng chức năng được mô tả trong dưới đây.



Hình 1.3 Quan hệ giữa dữ liệu và hàm trong lập trình hướng chức năng

Nhận xét:

– Hệ thống được xây dựng dựa vào chức năng là chính mà trong thực tế thì chức năng, nhiệm vụ của hệ thống lại hay thay đổi. Để đảm bảo cho hệ thống thực hiện được công việc theo yêu cầu, nhất là những yêu cầu về mặt chức năng đó lại bị thay đổi là công việc phức tạp và rất tốn kém.

Ví dụ: giám đốc thư viện yêu cầu thay đổi cách quản lý bạn đọc hoặc hơn nữa, yêu cầu bổ sung chức năng theo dõi những tài liệu mới mà bạn đọc thường xuyên yêu cầu để đặt mua, v.v. Khi đó vấn đề bảo trì hệ thống phần mềm không phải là vấn đề dễ thực hiện. Nhiều khi có những yêu cầu thay đổi cơ bản mà việc sửa đổi không hiệu quả và vì thế đòi hỏi phải phân tích, thiết kế lại hệ thống thì hiệu quả hơn.

– Các bộ phận của hệ thống phải sử dụng biến toàn cục để trao đổi với nhau, do vậy khả năng thay đổi, mở rộng của chúng và của cả hệ thống là bị hạn chế. Như trên đã phân tích, những thay đổi liên quan đến các dữ liệu chung sẽ ảnh hưởng tới các bộ phận liên quan. Do đó, một thiết kế tốt phải rõ ràng, dễ hiểu và mọi sửa đổi chỉ có hiệu ứng cục bộ.
– Khả năng tái sử dụng (Reuse) bị hạn chế và không hỗ trợ cơ chế kế thừa (Inheritance). Để có độ thích nghi cao thì mỗi thành phần phải là tự chứa. Muốn là tự chứa hoàn toàn thì một thành phần không nên dùng các thành phần ngoại lai. Tuy nhiên, điều này lại mâu thuẫn với kinh nghiệm nói rằng các thành phần hiện có nên là dùng lại được. Vậy là cần có một sự cân bằng giữa tính ưu việt của sự dùng lại các thành phần (ở đây chủ yếu là các hàm) và sự mất mát tính thích ứng được của chúng. Các thành của hệ thống phải kết dính (Cohesion) nhưng phải tương đối lỏng để dễ thích nghi. Một trong cơ chế chính hỗ trợ để dễ có được tính thích nghi là kế thừa thì cách tiếp cận hướng chức năng lại không hỗ trợ. Đó là cơ chế biểu diễn tính tương tự của các thực thể, đơn giản hoá định nghĩa những khái niệm tương tự từ những sự vật đã được định nghĩa trước trên cơ sở bổ sung hay thay đổi một số các đặc trưng hay tính chất của chúng. Cơ chế này giúp chúng ta thực hiện được nguyên lý tổng quát hoá và chi tiết hoá các thành phần của hệ thống phần mềm.

1.1.3. Lập trình hướng đối tượng

Dựa trên nền tảng là các đối tượng: Đặt trọng tâm vào dữ liệu, xem hệ thống là tập các thực thể và tập các đối tượng.

- Đặt trọng tâm vào dữ liệu (thực thể). Khi khảo sát, phân tích một hệ thống chúng ta tìm hiểu xem nó gồm những thực thể nào. Thực thể hay còn gọi là đối tượng, là những gì như người, vật, sự kiện, v.v. mà chúng ta đang quan tâm, hay cần phải xử lý. Ví dụ, khi xây dựng "Hệ thống quản lý thư viện" thì trước hết chúng ta tìm hiểu xem nó gồm những lớp đối tượng hoặc những khái niệm nào.
- Xem hệ thống như là tập các thực thể, các đối tượng. Để hiểu rõ về hệ thống, chúng ta phân tách hệ thống thành các đơn thể đơn giản hơn. Quá trình này được lặp lại cho đến khi thu được những đơn thể tương đối đơn giản, dễ hiểu và thực hiện cài đặt chúng mà không làm tăng thêm độ phức tạp khi liên kết chúng trong hệ thống. Xét "Hệ thống quản lý thư viện", chúng ta có các lớp đối tượng sau:



Hình 1.4 Tập các đối tượng của hệ thống quản lý thư viện

- Các lớp đối tượng trao đổi với nhau bằng các thông điệp (Message). Theo nghĩa thông thường thì lớp (Class) là nhóm một số người, vật có những đặc tính tương tự nhau hoặc có những hành vi ứng xử giống nhau. Trong mô hình đối tượng, khái niệm lớp là cấu trúc mô tả hợp nhất các thuộc tính (Attributes), hay dữ liệu thành phần (Data Member) thể hiện các đặc tính của mỗi đối tượng và các phương thức (Methods), hay hàm thành phần (Member Function) thao tác trên các dữ liệu riêng và là giao diện trao đổi với các đối tượng khác để xác định hành vi của chúng trong hệ thống. Khi có yêu cầu dữ liệu để thực hiện một nhiệm vụ nào đó, một đối tượng sẽ gửi một thông điệp (gọi một phương thức) cho đối tượng khác. Đối tượng nhận được thông điệp yêu cầu sẽ phải thực hiện một số công việc trên các dữ liệu mà nó sẵn có hoặc lại tiếp tục yêu cầu những đối tượng khác hỗ trợ để có những thông tin trả lời cho đối tượng yêu cầu. Với phương thức xử lý như thế thì một chương trình hướng đối tượng thực sự có thể không cần sử dụng biến toàn cục nữa.

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

1.2.1. Đối tượng

- Đối tượng là thực thể của hệ thống được xác định thông qua định danh (tên gọi).
- Mỗi đối tượng bao gồm dữ liệu thành phần hay các thuộc tính mô tả các tính chất và phương thức thao tác trên dữ liệu để xác định hành vi của đối tượng.

Đối tượng = Thuộc tính (dữ liệu) + phương thức (hàm).

– Theo quan điểm của người lập trình đối tượng được xem như vùng nhớ được phân chia trong máy tính để lưu dữ liệu và tập các hàm tác động lên dữ liệu gắn với chúng.

Ví dụ 1: Quyển sách = thông tin về sách (khổ sách, độ dày, nội dung…) + Đọc (mở sách, lật trang, đọc 1 đoạn, đọc một trang, tìm mục lục…)

(đối tượng) = Thuộc tính + phương thức

Ta có thể đọc sách dưới ánh đèn.

Đèn = Thông tin trạng thái (sáng, tắt) + bật, tắt đèn.

Quyển sách chứa nội dung thông tin, nó cũng bao gồm các thông tin về trang thái (sách đang mở, đóng) và có thể chứa các đối tương nhỏ hơn (các trang sách) . Phương thức cúa sách cho phép truy cập đến trang và phương thức của trang truy cập đến thông tin trong trang đó.

– Đối tượng có thể là một thực thể trực quan (người, sự vật) hay là một khái niệm (phòng ban, bộ phận, đăng ký…).

Ta sẽ gọi tập hợp giá trị các thuộc tính của một đối tượng cụ thể tại một thời điểm nào đó là trạng thái (state) của nó.

– Một hành vi là cái mà đối tượng có thể thực hiện, và sẽ bằng cách nào đó tác động tới một hoặc nhiều thuộc tính của đối tượng (do đó ảnh hưởng đến trạng thái của đối tượng). *Xét một người có một thuộc tính tư thế - một trong các giá trị {đứng, ngồi, nằm, quỳ}¨ Một hành vi "đứng" có thể dẫn đến kết quả là người đó đứng dậy (nếu người đó đang không ở tư thế đứng)*
– Trao đổi thông điệp: Các đối tượng liên lạc với nhau bằng các thông điệp (message). Một thông điệp là một yêu cầu một đối tượng thực hiện một hành vi.

Ví dụ 2: nếu ta muốn một người nào đó đứng dậy, ta có thể gửi một thông điệp đến cho người đó để đề nghị người đó đứng dậy. Đối với con người, phần lớn việc "truyền thông điệp" (messagepassing) của ta được thực hiện qua đường tiếng nói. Ta sẽ thấy rằng cơ chế "truyền thông điệp" đó xảy ra trong lập trình hướng đối tượng qua các lời gọi hàm.

– Trong mô hình hướng đối tượng: các đối tượng tương tác với nhau để thực hiện nhiệm vụ.

Phân loại các đối tượng : Khả năng phân loại bẩm sinh của con người, khả năng trừu tượng hoá này cho phép ta đơn giản hoá và tổ chức cuộc sống của mình, và hiểu về một thế giới rất phức tạp. Có hai cách phân loại đối tượng mà ta hay dùng nhất: phân loại các đối tượng tương tự và phân loại các đối tượng theo thành phần.

– Phân loại các đối tượng tương tự: nhóm các đối tượng có cùng các thuộc tính và hành vi lại với nhau. Ví dụ, trong loại "ô tô", ta có thể đưa vào đó các đối tượng có các thuộc tính mầu sắc, nhãn hiệu, kiểu, và các hành vi lái, dừng, rẽ.

Lưu ý: phân loại theo "có hay không một thuộc tính", không phân loại theo giá trị của thuộc tính. Ở đây, ta không lập nhóm các xe ô tô có giá trị "đỏ" cho thuộc tính mầu và giá trị "Honda" cho thuộc tính nhãn hiệu đối tượng chỉ cần có các thuộc tính đó là đủ để thuộc nhóm.

– Phân loại theo cấu tạo: Thuộc tính của một lớp nào đó có thể có giá trị là một đối tượng thuộc một lớp nào đó.

– Phân loại đối tượng theo cấu tạo là một trong các kiểu *quan hệ* có thể tồn tại giữa các đối tượng. Kiểu quan hệ này thường được gọi là quan hệ chứa "has-a".

Mô hình hóa: Bước đầu tiên khi mô hình hoá một hệ thống gồm các lớp đối tượng, ta xác định xem:

– Có các lớp đối tượng nào?
– Chúng có các thuộc tính gì?
– Chúng có liên quan đến nhau như thế nào?
– Việc xác định hành vi của các lớp sẽ để sau

Ổn định tập thuộc tính cho mỗi lớp trước khi xác định các hành vi dùng để truy nhập các thuộc tính này.

Khi đã xác định được tất cả các lớp đối tượng cần thiết, các thuộc tính và quan hệ giữa chúng, bước tiếp theo là chỉ ra các hành vi của các lớp đối tượng đó.

– Có những hành vi nào?
– Các lớp đối tượng có thể cần những hành vi nào để có thể sử dụng các thuộc tính của mình?
– Các lớp đối tượng cần thực hiện những nhiệm vụ nào cho các đối tượng khác?

1.2.2. Lớp đối tượng

– Ta sẽ gọi một nhóm các đối tượng tương tự là một lớp đối tượng – object class. Một lớp là "một tập các đối tượng có cùng thuộc tính và hành vi"
– Mỗi đối tượng thuộc một lớp là một thể hiện (instance) của lớp đó. *Mỗi xe ô tô là một thể hiện của lớp ô tô.*
– Các thể hiện của một lớp phân biệt nhau bởi một số thuộc tính nhất định có giá trị duy nhất trong toàn lớp - định danh duy nhất ¨ *định danh của sinh viên là mã sinh viên – không có hai sinh viên nào có mã giống nhau đôi khi không dễ tìm được cách phân biệt giữa các thể hiện*
– Lớp định nghĩa kiểu dữ liệu và phương thức dùng để truy cập dữ liệu của đối tượng.

Ví dụ 3: *Lớp đơn xin việc = trường cần điền vào đơn + ghi/đọc đơn*

(đối tượng) = (dữ liệu) + (phương thức)

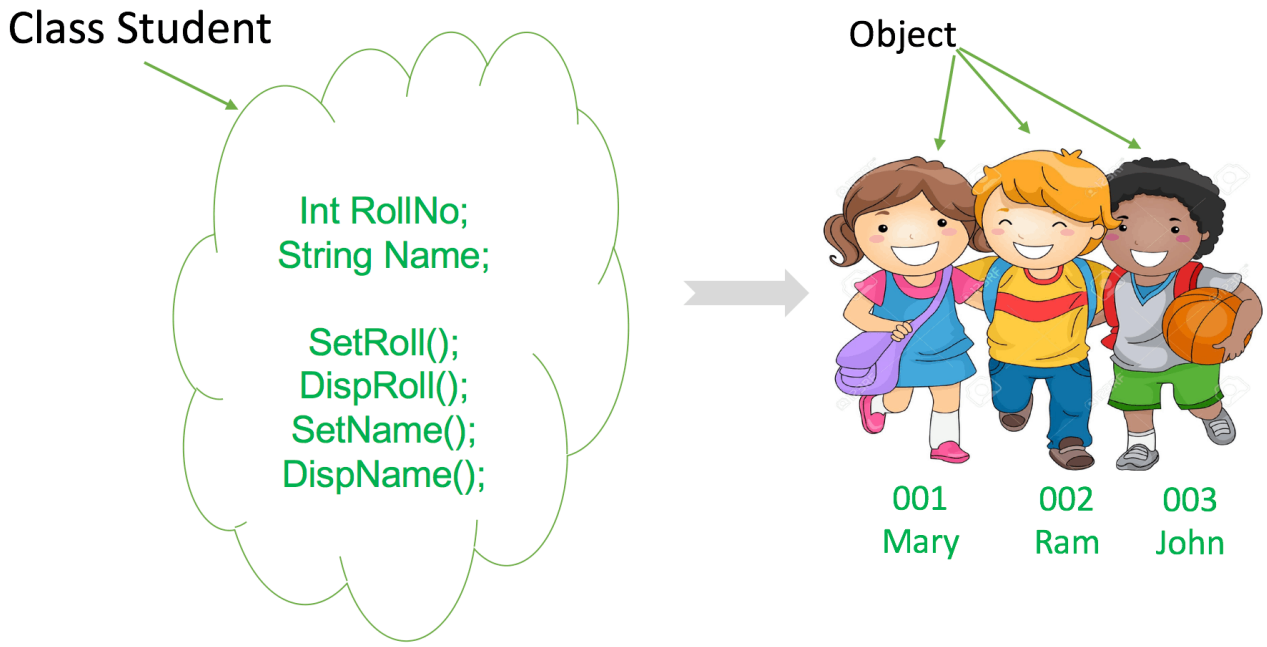
– Lớp mẫu đơn xin việc.

Những người đến xin việc họ phải được cung cấp một thực thể duy nhất của đơn, những thực thể tạo ra bằng cách lấy bản mẫu làm bản chính, sao chép tạo ra nhiều thực thể. Người xin việc phải điền các thông tin vào đơn bằng phương thức ghi đơn.

– Lớp được đặc tả bởi tên lớp: tập thuộc tính, tập các hàm.

Ví dụ 4: *lớp Student có 2 thuộc tính, 4 hàm (phương thức)*

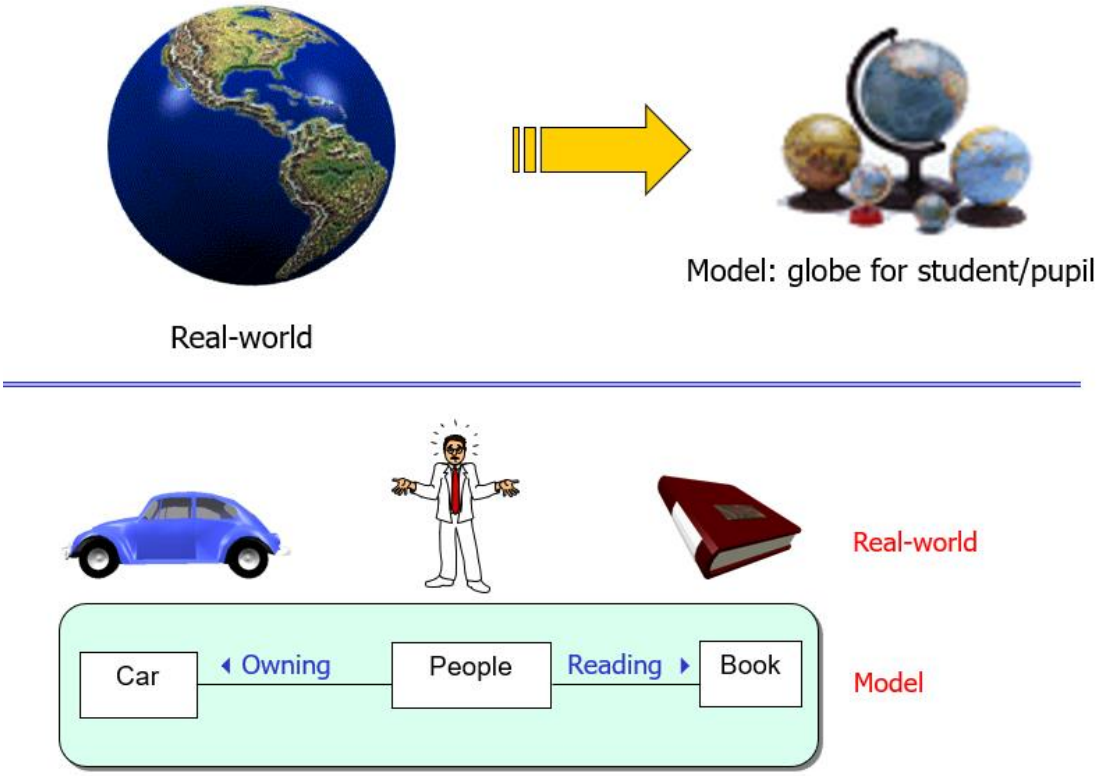
Đối tượng: *các đối tượng thuộc lớp Student nên nó mang đầy đủ thuộc tính và phương thức như lớp Student.*



Hình 1.5 Lớp đối tượng

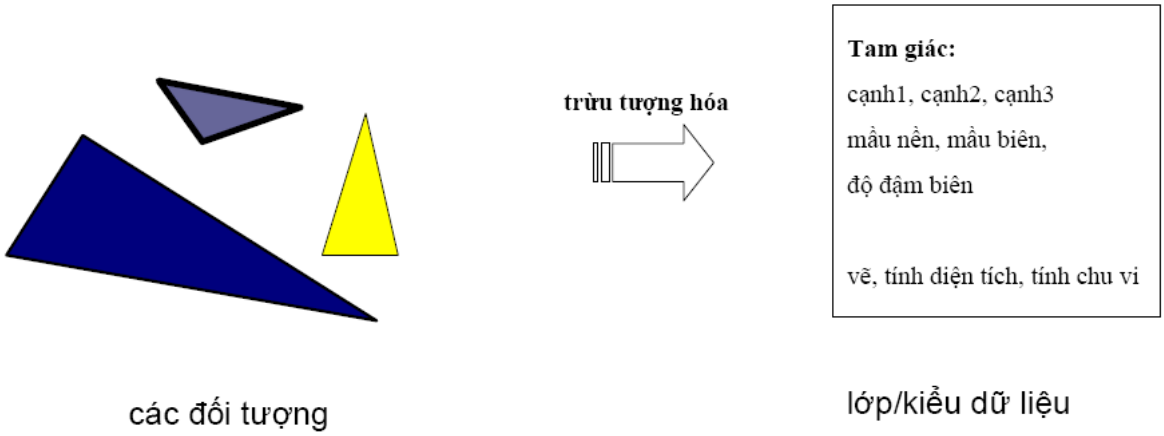
1.2.3. Trừu tượng hóa

Trừu tượng hóa là cách biểu diễn đặc tính chính và bỏ qua những chi tiết.



Hình 1.6 Trừu tượng hoá

Trừu tượng hóa là cách nhìn đơn giản hóa về một đối tượng mà trong đó chỉ bao gồm các đặc điểm được quan tâm và bỏ qua những chi tiết không cần thiết.



Hình 1.7 Sử dụng trừu tượng hoá để xây dựng lớp

- Trừu tượng hóa là sự mở rộng các khái niệm kiểu dữ liệu và cho phép định nghĩa những phép toán trừu tượng trên các kiểu dữ liệu trừu tượng.
- Để xác định lớp ta phải sử dụng khái niệm trừu tượng hóa.

1.2.4. Bao bọc và che giấu thông tin

- Đóng gói: Nhóm những gì có liên quan với nhau vào làm một, để sau này có thể dùng một cái tên để gọi đến: Các hàm/ thủ tục đóng gói các câu lệnh. Các đối tượng đóng gói dữ liệu của chúng và các thủ tục có liên quan.

Chi tiết về các thành viên của một đối tượng cần được đóng gói bên trong đối tượng, để cách duy nhất truy nhập đến chúng là qua các hành vi của đối tượng. Tương tự, chi tiết về cách thực hiện các hành vi của một đối tượng cũng nên được đóng gói để bên ngoài không biết chi tiết.

Tại sao đóng gói đối tượng? Đóng gói bắt chước thế giới thực: "*nếu tôi muốn anh đứng dậy (thay đổi thuộc tính "vị trí") khả năng lớn là tôi sẽ đề nghị anh đứng dậy thay vì đến kéo anh dậy*." điều này làm đơn giản hoá cuộc sống, ta thường không quan tâm đến chi tiết. Trong lập trình hướng đối tượng, lợi ích của việc đóng gói: khi che dấu chi tiết cài đặt, người dùng bên ngoài không phải bận tâm đến chuyện cái gì được làm như thế nào. Điều này đem lại lợi ích đáng kể trong việc bảo trì phần mềm - do người dùng không bao giờ biết chi tiết bên trong đối tượng, ta có thể thay đổi các chi tiết đó mà họ không cần biết (miễn là giao diện với bên ngoài không đổi)

- Xác định các vùng: riêng (private) công khai (public) hay bảo vệ (protected) nhằm điều khiển, hay han chế những truy nhập tùy tiện của những đối tượng khác.
- Việc đóng gói dữ liệu và hàm vào cùng một đơn vị cấu trúc (lớp) được xem là nguyên tắc bao bọc thông tin.
- Nguyên tắc bao bọc dữ liệu để cấm sự truy nhập trực tiếp gọi là sự che dấu thông tin.

Che dấu thông tin: đóng gói để che một số thông tin và chi tiết cài đặt nội bộ để bên ngoài không nhìn thấy. mục tiêu là để khách hàng của ta (thường là các lập trình viên khác) coi các đối tượng của ta là các hộp đen.

1.2.5. Kế thừa

– *Kế thừa*: Là nguyên lý kế thừa cho phép các đối tượng của lớp này được quyền sử dụng lại một số tính chất (thuộc tính và phương thức) của lớp khác.

Ví dụ 5: Nếu ta có một mô hình kế thừa giữa 2 lớp A và B; trong đó: A - lớp cha; B - lớp con.

➔ Khi đó:

+ Lớp B sẽ kế thừa các thuộc tính và các hàm có khai báo public hoặc protected của lớp A.
+ Trong lớp B có thể bổ sung thêm một số tính chất, phương thức truy cập đẻ thu hẹp phạm vi xác định đối tượng trong lớp mới.
+ *Những thuộc tính hay hàm được khai báo private thì không kế thừa lại được.*

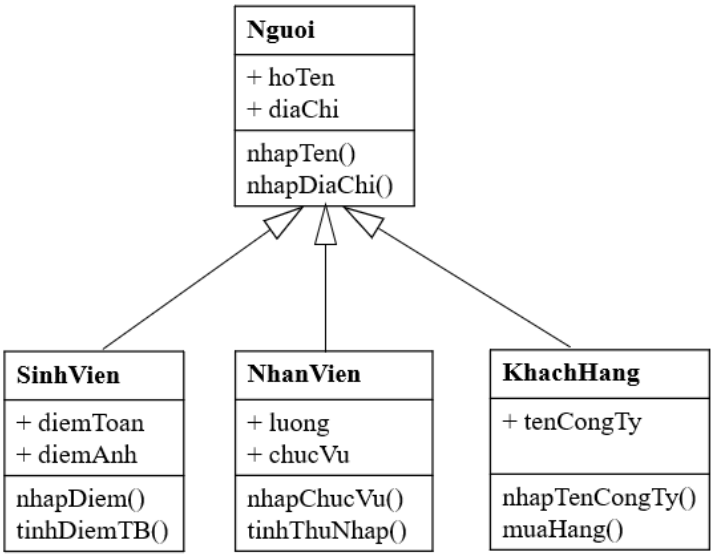
Lưu ý: Java chỉ hỗ trợ một cơ chế kế thừa *(giữa các lớp)* duy nhất, đó là kế thừa đơn, *không có kế thừa bội*. Để tận dụng lợi ích của kế thừa bội, Java xây dựng khái niệm và mở rộng kế thừa bội đối với các interface *(giao diện)*.

Ví dụ 6: Trong một bài toán có 3 lớp cần quản lý: Sinh Viên, Nhân Viên, Khách Hàng. Rõ ràng rằng cả 3 lớp này đều có chung các thông tin của một người (họ tên, địa chỉ); vì vậy, để tối ưu hoá viết code và linh hoạt khi sử dụng sau này, chúng ta thường xây dựng mô hình kế thừa như sau:

– Lớp **Nguoi** *(lớp cha)*: Có các thuộc tính và phương thức/hàm như sau:
  + Thuộc tính: hoTen, diaChi
  + Phương thức: nhapTen(), nhapDiaChi()
– Lớp **SinhVien** *(lớp con)*: Có các thuộc tính và phương thức/hàm như sau:
  + Thuộc tính: *hoTen, diaChi, diemToan, diemAnh*
  + Phương thức/hàm: *nhapTen*(), *nhapDiaChi(), nhapDiem(), tinhDiemTB()*
– Lớp **NhanVien** *(lớp con)*: Có các thuộc tính và phương thức/hàm như sau:
  + Thuộc tính: *hoTen, diaChi, luong, chucVu*
  + Phương thức/hàm: *nhapTen(), nhapDiaChi(), nhapChucVu(), tinhThuNhap()*
– Lớp **KhachHang** *(lớp con)*: Có các thuộc tính và phương thức/hàm như sau:
  + Thuộc tính: *hoTen, diaChi, tenCongTy*
  + Phương thức/hàm: *nhapTen(), nhapDiaChi*(), *nhapTenCongTy(), muaHang()*

➔ Mô hình kế thừa và khung mã lệnh tương ứng sử dụng ngôn ngữ lập trình Java được thể hiện ở Hình 1.8

A) Mô hình kế thừa



B) Codes Java tương ứng

```
class Nguoi{
    private String hoTen;
    private String diaChi;

    public void nhapTen() {}
    public void nhapDiaChi() {}
}
class SinhVien extends Nguoi{
    private double diemToan;
    private double diemAnh;

    public void nhapDiem() {}
    public void tinhDiemTB() {}
}
class NhanVien extends Nguoi{
    private double luong;
    private String chucVu;

    public void nhapChucVu() {}
    public void tinhThuNhap() {}
}
class KhachHang extends Nguoi{
    private String tenCongTy;

    public void nhapTenCongTy() {}
    public void muaHang() {}
}
```

Hình 1.8 Ví dụ kế thừa

1.2.6. Đa hình

Đa hình: Là kỹ thuật sử dụng để mô tả khả năng gửi một thông điệp chung tới nhiều đối tượng mà mỗi đối tượng lại có cách xử lý riêng theo ngữ cảnh của mình.

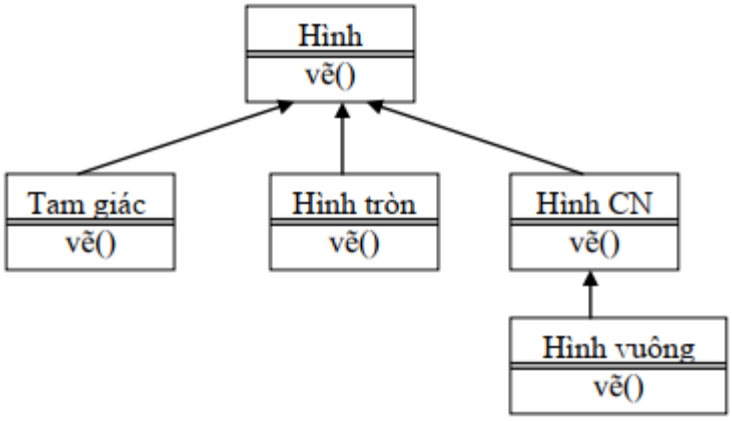
Khả năng của đối tượng có nhiều phương thức cùng tên nhưng nội dung thực hiện khác nhau. Một thông điệp (lời gọi hàm) được hiểu theo các cách khác nhau tuỳ theo danh sách tham số của thông điệp. Ví dụ: nhận được cùng một thông điệp "nhảy", một con kangaroo và một con cóc nhảy theo hai kiểu khác nhau: chúng cùng có hành vi "nhảy" nhưng các hành vi này có nội dung khác nhau.



chú ý rằng kangaroo và cóc thuộc hai nhánh trong cây phả hệ động vật

Hình 1.9 Đa hình

Ví dụ 7: Hàm vẽ là hàm đa trị. Nó được xác định tùy ngữ cảnh khi sử dụng. Khi sử dụng làm vẽ với đối tượng tam giác thì thực hiện việc vẽ tam giác. Khi sử dụng hàm vẽ với đối tượng hình chữ nhật thì thực hiện việc vẽ hình chữ nhật...



Hình 1.10 Ví dụ đa hình

Nạp chồng: Là trường hợp đặc biệt của đa trị. Dùng một tên hàm cho nhiều định nghĩa hàm, nhưng khác nhau ở kiểu của tham số hoặc số lượng tham số thì khi đó nội dung thực trong mỗi hàm là khác nhau.

Ví dụ 8: *hàm cộng()*

```
int cong(int, int)
float(float, float)
string cong(string, string)
int cong(int a, int b, int c)
int cong(int a[]) // danh sách đầu vào là một mảng
```

1.3. CÁCH TIẾP CẬN VÀ GIẢI QUYẾT BÀI TOÁN THEO PHƯƠNG PHÁP HƯỚNG ĐỐI TƯỢNG

1.3.1. Mô tả bài toán

Mô tả bài toán là bước đầu tiên giúp chúng ta hiểu rõ mục tiêu của bài toán chúng ta cần giải quyết. Khi mô tả bài toán, chúng ta trả lời các câu hỏi cơ bản sau:

- Vấn đề là gì?: Xác định vấn đề cần giải quyết.
- Mục tiêu là gì?: Xác định mục tiêu cuối cùng mà hệ thống cần đạt được.
- Các thành phần chính của bài toán là gì?: Xác định các yếu tố hoặc thành phần chính của bài toán (có thể là người dùng, dữ liệu, các sự kiện chính, v.v.).
- Phạm vi bài toán: Xác định phạm vi bài toán, bao gồm những gì cần xây dựng và những gì nằm trong phạm vi của bài toán.

Bước mô tả bài toán đóng vai trò quan trọng giúp định hướng cho các bước tiếp theo. Nếu không mô tả rõ ràng, có thể dẫn đến việc phát triển hệ thống thiếu chính xác hoặc không đáp ứng được nhu cầu của người dùng.

Ví dụ 1: Ta có bài toán xây dựng một hệ thống quản lý thư viện. Mục tiêu là hỗ trợ nhân viên thư viện quản lý sách và người dùng, cùng với chức năng mượn và trả sách.

Các chức năng chính yêu cầu:

- Quản lý sách: Hệ thống cần có khả năng thêm, sửa, xóa thông tin về các cuốn sách.
- Quản lý người dùng: Cung cấp tính năng đăng ký và xóa tài khoản người dùng.
- Quản lý mượn trả: Ghi nhận và theo dõi tình trạng mượn trả sách, kiểm tra sách có sẵn hay không.

Lưu ý: Cần đặc biệt quan tâm đến các quy tắc nghiệp vụ. Ví dụ: một người dùng không được phép mượn nhiều hơn một số sách tối đa hoặc sách phải được trả trước khi mượn tiếp.

1.3.2. Đặc tả yêu cầu

Đặc tả yêu cầu là một bước phân tích chi tiết, giúp xác định rõ các yêu cầu của hệ thống. Các chức năng mà hệ thống cần phải thực hiện. Đây là những hành vi cụ thể mà hệ thống phải đáp ứng.

Ví dụ 2: Bảng đặc tả yêu cầu của bài toán quản lý thư viện

Bảng 1.1 Bảng đặc tả yêu cầu của bài toán quản lý thư viện

Mã yêu cầu	Yêu cầu	Loại yêu cầu	Mô tả
FR01	Thêm sách	Chức năng	Hệ thống cho phép quản trị viên thêm sách vào thư viện
FR02	Xóa sách	Chức năng	Hệ thống cho phép quản trị viên xóa sách khỏi thư viện
FR03	Sửa thông tin sách	Chức năng	Hệ thống cho phép quản trị viên cập nhật thông tin sách
FR04	Đăng ký người dùng	Chức năng	Hệ thống cho phép người dùng đăng ký tài khoản
FR05	Xóa người dùng	Chức năng	Hệ thống cho phép quản trị viên xóa tài khoản người dùng
FR06	Mượn sách	Chức năng	Người dùng có thể mượn sách nếu sách đang có sẵn
FR07	Trả sách	Chức năng	Người dùng có thể trả sách và cập nhật lại trạng thái sách

Việc đặc tả yêu cầu giúp hiểu rõ những gì hệ thống phải thực hiện, đảm bảo rằng mọi chức năng và tính chất mong muốn của hệ thống đều được liệt kê đầy đủ, từ đó giảm thiểu rủi ro sai sót trong quá trình phát triển.

1.3.3. Trích chọn đối tượng

Trích chọn đối tượng là quá trình xác định các đối tượng trong hệ thống. Các đối tượng là những thành phần có khả năng lưu trữ trạng thái và thực hiện các hành vi cụ thể. Trong một bài toán hướng đối tượng, mỗi đối tượng đại diện cho một khái niệm thực hoặc trừu tượng trong hệ thống, ví dụ như "Người dùng", "Sách", "Giỏ hàng".

Các tiêu chí trích chọn đối tượng:

- Đối tượng phải là các thực thể hoặc khái niệm có ý nghĩa độc lập trong bối cảnh bài toán.

- Đối tượng phải có trạng thái và hành vi rõ ràng: Ví dụ, trong hệ thống thư viện, đối tượng "Sách" có trạng thái (tiêu đề, tác giả, trạng thái mượn) và hành vi (cập nhật trạng thái, kiểm tra tính sẵn sàng).

Ví dụ 3: Các đối tượng chính trong hệ thống quản lý thư viện

- **Sách (Book)**: Là thực thể đại diện cho một cuốn sách trong thư viện, với các thuộc tính như tiêu đề, tác giả, trạng thái (có sẵn, đang mượn).
- **Người dùng (User)**: Đại diện cho người dùng của hệ thống (có thể là thành viên hoặc quản trị viên), với các thuộc tính như tên, số điện thoại.
- **Quản lý mượn trả (BorrowManager)**: Đối tượng này đại diện cho chức năng quản lý các thao tác mượn và trả sách, giúp ghi nhận và kiểm tra tình trạng sách.

Bước trích chọn đối tượng giúp xác định các thành phần chính của hệ thống và cách chúng tương tác với nhau, từ đó tạo nền tảng cho các bước mô hình hóa và thiết kế lớp đối tượng sau này.

1.3.4. Mô hình hoá lớp đối tượng

Mô hình hóa lớp đối tượng là quá trình xác định các lớp (Classes) trong hệ thống và xây dựng các thuộc tính (Attributes) và phương thức (Methods) của chúng. Một lớp đại diện cho một nhóm đối tượng có cùng tính chất và hành vi.

- Lớp (Class): Là một mô hình hoặc bản thiết kế cho các đối tượng. Một lớp xác định các thuộc tính và phương thức mà các đối tượng của lớp đó sẽ có.
- Thuộc tính (Attributes): Là các đặc điểm, trạng thái của lớp hoặc đối tượng. Ví dụ, lớp "Sách" có các thuộc tính như "Tựa sách", "Tác giả".
- Phương thức (Methods): Là các hành vi hoặc hành động mà lớp hoặc đối tượng có thể thực hiện. Ví dụ, lớp "Sách" có thể có phương thức "kiểm tra tính sẵn sàng".

Ví dụ 4: Các lớp đối tượng trong hệ thống quản lý thư viện

Lớp **Book** (Sách)

- **Thuộc tính**:
  - bookID: Mã định danh duy nhất của mỗi cuốn sách.
  - title: Tiêu đề sách.
  - author: Tác giả của sách.
  - status: Trạng thái của sách (có sẵn, đang mượn).
- **Phương thức**:
  - addBook(): Thêm sách mới vào thư viện.
  - updateBookInfo(): Cập nhật thông tin sách.
  - checkAvailability(): Kiểm tra xem sách có sẵn hay không.

Lớp **User** (Người dùng)

- **Thuộc tính**:
  - userID: Mã định danh của người dùng.

+ name: Tên người dùng.
+ contactInfo: Thông tin liên hệ của người dùng.

- **Phương thức**:
  + register(): Đăng ký tài khoản người dùng.
  + deleteAccount(): Xóa tài khoản người dùng khỏi hệ thống.

Lớp **BorrowManager** (Quản lý mượn trả)

- **Thuộc tính**:
  + borrowID: Mã định danh của phiên mượn.
  + userID: Mã người dùng mượn sách.
  + bookID: Mã sách đang được mượn.
  + borrowDate: Ngày mượn sách.
  + returnDate: Ngày trả sách.
- **Phương thức**:
  + borrowBook(): Thao tác mượn sách.
  + returnBook(): Thao tác trả sách.
  + checkStatus(): Kiểm tra tình trạng mượn trả của sách.

Mô hình hóa lớp giúp hệ thống hóa các đối tượng và mối quan hệ giữa chúng. Bước này giúp dễ dàng hình dung cấu trúc của hệ thống, đồng thời tạo cơ sở cho việc triển khai mã nguồn.

1.3.5. Thiết kế tổng quát

Thiết kế tổng quát là bước xác định cách các đối tượng và lớp sẽ tương tác với nhau trong hệ thống. Thông qua thiết kế tổng quát, chúng ta mô tả cách các đối tượng phối hợp với nhau để thực hiện các chức năng chính của hệ thống.

Thiết kế tổng quát giúp hình dung cách mà hệ thống sẽ hoạt động trong thực tế, cung cấp bản vẽ chi tiết về dòng dữ liệu và logic xử lý. Đây là bước chuyển tiếp giữa lý thuyết và triển khai thực tế, đảm bảo các lớp và đối tượng có thể thực hiện các chức năng mong muốn khi tương tác.

Ví dụ 5: Luồng hoạt động chính cho chức năng mượn sách

1. Người dùng gửi yêu cầu mượn sách đến *BorrowManager*.
2. *BorrowManager* gọi phương thức *checkAvailability()* của lớp *Book* để kiểm tra tình trạng sách.
3. Nếu sách có sẵn, *BorrowManager* ghi nhận thông tin mượn và thay đổi trạng thái sách thành "đang mượn".
4. Nếu sách không sẵn sàng, hệ thống sẽ thông báo cho người dùng.

1.3.6. Thiết kế chi tiết

Thiết kế chi tiết là quá trình xác định chi tiết các thuật toán, cách thức lưu trữ và truy xuất dữ liệu, và các mô hình tương tác bên trong của từng lớp. Ở bước này, các phương thức và thuộc tính được triển khai một cách cụ thể dựa trên các yêu cầu đã xác định.

Các thành phần của thiết kế chi tiết:

- Thuật toán: Mô tả các bước cụ thể để hoàn thành một tác vụ hoặc chức năng.
- Chi tiết triển khai phương thức: Mỗi phương thức trong từng lớp được mô tả chi tiết về logic xử lý, cách thức thực hiện các thao tác.
- Chi tiết cấu trúc dữ liệu: Lựa chọn các kiểu dữ liệu phù hợp để lưu trữ thông tin một cách tối ưu.

Thiết kế chi tiết là nền tảng để lập trình viên chuyển từ ý tưởng sang mã nguồn thực tế. Đây là bước quyết định để đảm bảo các đối tượng và lớp hoạt động chính xác và hiệu quả.

Ví dụ 6: Thiết kế chi tiết cho bài toán quản lý thư viện

1. Lớp *Book*

Lớp *Book* sẽ đại diện cho một cuốn sách và chứa các thuộc tính cùng phương thức liên quan.

```
public class Book {
  private String bookID;
  private String title;
  private String author;
  private boolean isAvailable; // true nếu có sẵn, false nếu đang mượn
  public Book(String bookID, String title, String author) {
        this.bookID = bookID;
        this.title = title;
        this.author = author;
        this.isAvailable = true; // Mới thêm vào thư viện thì sách sẽ có sẵn
  }
  // Phương thức kiểm tra tính khả dụng của sách
  public boolean isAvailable() {
        return isAvailable;
  }
  // Phương thức cập nhật trạng thái sách
  public void updateStatus(boolean status) {
        this.isAvailable = status;
  }
  // Các phương thức getter
  public String getBookID() {
        return bookID;
  }
  public String getTitle() {
```

```
        return title;
    }
    public String getAuthor() {
        return author;
    }
}
```

2. Lớp *User*

Lớp *User* sẽ đại diện cho người dùng trong hệ thống.

```
public class User {
    private String userID;
    private String name;
    public User(String userID, String name) {
        this.userID = userID;
        this.name = name;
    }
    // Getter
    public String getUserID() {
        return userID;
    }
    public String getName() {
        return name;
    }
}
```

3. Lớp *BorrowManager*

Lớp *BorrowManager* sẽ quản lý các thao tác mượn và trả sách.

```
import java.util.HashMap;
import java.util.Map;
public class BorrowManager {
    private Map<String, Book> bookCollection; // Bảng chứa các cuốn sách
    private Map<String, String> borrowedBooks; // Bảng chứa các sách đã mượn (userID -> bookID)
    public BorrowManager() {
        bookCollection = new HashMap<>();
        borrowedBooks = new HashMap<>();
```

```
    }
    // Phương thức thêm sách vào thư viện
    public void addBook(Book book) {
        bookCollection.put(book.getBookID(), book);
    }
    // Phương thức kiểm tra và mượn sách
    public String borrowBook(User user, String bookID) {
        // Kiểm tra xem sách có trong bộ sưu tập không
        Book book = bookCollection.get(bookID);
        if (book == null) {
            return "Sách không tồn tại trong thư viện.";
        }
        // Kiểm tra tính khả dụng của sách
        if (book.isAvailable()) {
            // Cập nhật trạng thái sách thành không có sẵn
            book.updateStatus(false);
            // Lưu bản ghi mượn sách
            borrowedBooks.put(user.getUserID(), bookID);
            return "Mượn sách thành công: " +book.getTitle();
        } else {
            return "Sách đã được mượn.";
        }
    }
    // Phương thức trả sách
    public String returnBook(User user, String bookID) {
        // Kiểm tra xem người dùng có bản ghi mượn sách không
        if (borrowedBooks.containsKey(user.getUserID()) &&
borrowedBooks.get(user.getUserID()).equals(bookID)) {
            // Cập nhật trạng thái sách thành có sẵn
            Book book = bookCollection.get(bookID);
            if (book != null) {
                book.updateStatus(true);
            }
            // Xóa bản ghi mượn sách
```

```
                borrowedBooks.remove(user.getUserID());
                return "Trả sách thành công: " + book.getTitle();
            } else {
                return "Không tìm thấy bản ghi mượn cho sách này.";
            }
        }
    }
```

4. Sử dụng các lớp

Dưới đây là ví dụ về cách sử dụng các lớp đã được định nghĩa ở trên để thực hiện việc mượn và trả sách.

```
    public class LibrarySystem {
        public static void main(String[] args) {
            // Tạo đối tượng quản lý mượn sách
            BorrowManager borrowManager = new BorrowManager();
            // Tạo và thêm sách vào thư viện
            Book book1 = new Book("B001", "Lập trình Java cơ bản", "Nguyễn Văn
A");
            Book book2 = new Book("B002", "Hệ thống thông tin", "Trần Thị B");
            borrowManager.addBook(book1);
            borrowManager.addBook(book2);
            // Tạo người dùng
            User user1 = new User("U001", "Nguyễn Văn C");
            // Mượn sách
            System.out.println(borrowManager.borrowBook(user1, "B001")); // Mượn
thành công
            System.out.println(borrowManager.borrowBook(user1, "B001")); // Sách đã
được mượn
            // Trả sách
            System.out.println(borrowManager.returnBook(user1, "B001")); // Trả thành
công
            System.out.println(borrowManager.returnBook(user1, "B001")); // Không
 tìm thấy bản ghi mượn
        }
    }
```

1.4. MỘT SỐ NGÔN NGỮ LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG

Các ngôn ngữ lập trình thời kỳ đầu như Basic, Fortran... không có cấu trúc và cho phép viết những đoạn mã rối rắm (spaghetti code). Lập trình viên sử dụng các lệnh "goto" và "gosub" để nhảy đến mọi nơi trong chương trình. Đoạn trình trên khó theo dõi, khó hiểu, dễ gây lỗi, khó sửa đổi.

Đặc điểm của lập trình hướng chức hay lập trình thủ tục (Procedural Programming- PP) là:

- Sử dụng các cấu trúc vòng lặp: *for, while, repeat, do-while*
- Chương trình là một chuỗi các hàm/ thủ tục
- Mã chương trình tập trung thể hiện thuật toán: làm như thế nào

Hạn chế của lập trình thủ tục:

- Dữ liệu và phần xử lý tách biệt
- Dữ liệu thụ động, xử lý chủ động
- Không đảm bảo được tính nhất quán và các ràng buộc của dữ liệu
- Khó cấm mã ứng dụng sửa dữ liệu của thư viện
- Khó bảo trì code
- Phần xử lý có thể nằm rải rác và phải hiểu rõ cấu trúc dữ liệu

Lập trình hướng đối tượng: *Lập trình hướng đối tượng* cho phép khắc phục các hạn chế nói trên

Các ngôn ngữ lập trình hướng đối tượng không mới: Simula (1967) là ngôn ngữ đầu tiên, có lớp, thừa kế, liên kết động (hay còn gọi là hàm ảo). Nhưng các ngôn ngữ hướng đối tượng chậm hơn các ngôn ngữ thời kỳ đầu nên chúng chỉ được dùng rộng rãi khi máy tính bắt đầu chạy nhanh (khoảng thời gian chiếc máy Pentium đầu tiên ra đời).

Dựa vào khả năng đáp ứng các khái niệm về hướng đối tượng, ta chia ra làm hai loại:

* Ngôn ngữ lập trình dựa trên đối tượng (object-based):

Lập trình dựa trên đối tượng là kiểu lập trình hỗ trợ chính cho việc bao bọc, che giấu thông tin và định danh các đối tượng. Lập trình dựa trên đối tượng có những đặc tính sau:

- Bao bọc dữ liệu,
- Cơ chế che giấu và hạn chế truy nhập dữ liệu,
- Tự động tạo lập và huỷ bỏ các đối tượng,
- Tính đa xạ.

Ngôn ngữ hỗ trợ cho kiểu lập trình trên được gọi là ngôn ngữ lập trình dựa trên đối tượng. Ngôn ngữ trong lớp này không hỗ trợ cho việc thực hiện kế thừa và liên kết động. Ada là ngôn ngữ lập trình dựa trên đối tượng.

* Ngôn ngữ lập trình hướng đối tượng (object-oriented)

Lập trình hướng đối tượng là kiểu lập trình dựa trên đối tượng và bổ sung thêm nhiều cấu trúc để cài đặt những quan hệ về kế thừa và liên kết động. Vì vậy đặc tính của LTHĐT có thể viết một cách ngắn gọn như sau:

Các đặc tính dựa trên đối tượng + kế thừa + liên kết động.

Ngôn ngữ hỗ trợ cho những đặc tính trên được gọi là ngôn ngữ lập trình hướng đối tượng, ví dụ như Java, C++, Smalltalk, Object Pascal hay Eiffel, v.v...

1.5. GIỚI THIỆU MỘT SỐ EDITOR BIÊN SOẠN CHƯƠNG TRÌNH JAVA

1.5.1. Eclipse

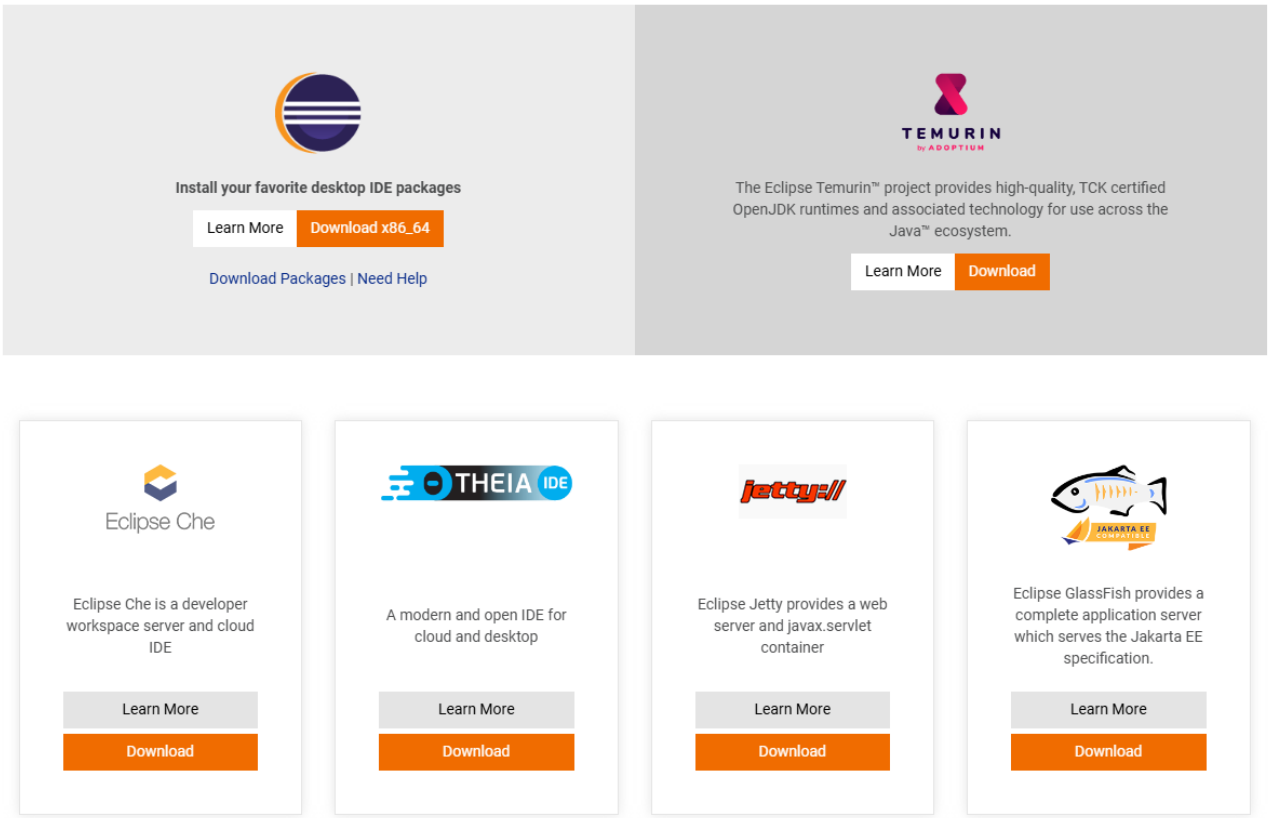
Eclipse là một công cụ lập trình mã nguồn mở mạnh mẽ, được thiết kế như một môi trường phát triển tích hợp chủ yếu cho Java, nhưng cũng hỗ trợ lập trình bằng nhiều ngôn ngữ lập trình khác như C/C++, PHP, Ruby,... Cùng với khả năng mở rộng mã nguồn thông qua việc tích hợp các plugin cho dự án, Eclipse là sự lựa chọn linh hoạt cho các nhà phát triển. Điều đặc biệt là Eclipse hoàn toàn miễn phí và đi kèm với nhiều plugin hữu ích.

Ưu điểm của Eclipse

- Sử dụng cho hai hệ điều hành là Windows và Linux.
- Hỗ trợ nhiều ngôn ngữ lập trình khác như Java, C/C++, PHP.
- Dễ dàng làm việc với cả GUI và non-GUI.
- Tốc độ tải nhanh với SWT/JFace.
- Đa dạng plugin hữu ích như Testing, Code formatter...
- Phù hợp cho các dự án lớn và triển khai trên máy tính cũng như đám mây lữu trữ.
- Mã nguồn mở giúp cá nhân hóa và tạo cấu trúc đầy đủ tính năng cho các công cụ tích hợp.

Tải và cài đặt Eclipse

- Truy cập trang tải Eclipse: https://www.eclipse.org/downloads/
- Chọn phiên bản Eclipse IDE phù hợp với nhu cầu lập trình của bạn và nhấn vào download.



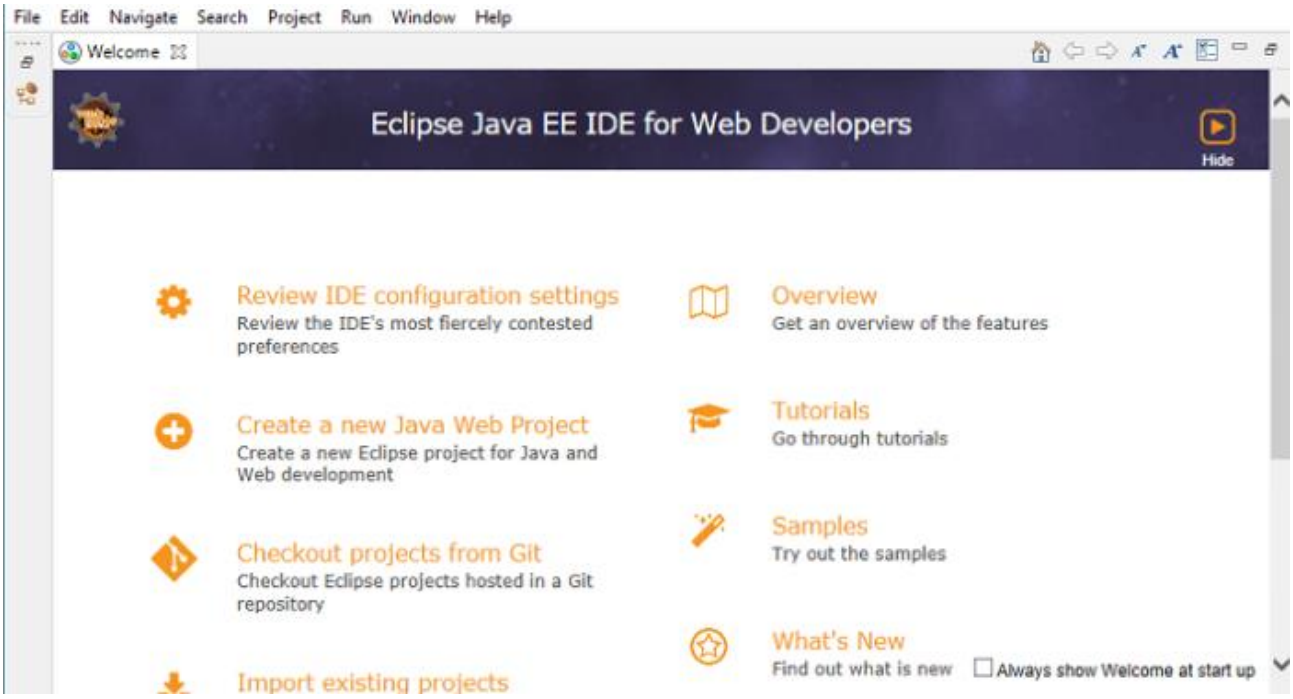
Hình 1.11 Tải Eclipse

– Tải về và mở file cài đặt Eclipse.
– Lựa chọn cài đặt Eclipse
  + Tùy chọn 1: Cài đặt phiên bản Eclipse cho lập trình viên Java với các chức năng cơ bản.
  + Tùy chọn 2: Eclipse cho Java EE, thích hợp cho các dự án web sử dụng Java.
  + Tùy chọn 3: Cho phép lập trình C/C++ trên Eclipse.



Hình 1.12 Lựa chọn cài đặt Eclipse

– Sau khi cài đặt xong, nhấn "Launch" để mở Eclipse.
– Giao diện mở đầu của Eclipse sẽ hiện ra, sẵn sàng cho việc lập trình Java của bạn.



Hình 1.13 Giao diện mở đầu của Eclipse

1.5.2. NetBeans

NetBeans là một môi trường phát triển tích hợp (Integrated Development Environment - IDE) mã nguồn mở, ra đời từ năm 1996 dưới tên gọi Xelfi, ban đầu là dự án IDE Java được phát triển bởi sinh viên dưới sự hướng dẫn của Khoa Toán - Lý tại Đại học Charles, Prague. Sau đó, vào năm 1997, Roman Staněk đã thành lập một công ty quanh dự án này, và sau đó bán phiên bản thương mại của NetBeans cho đến khi Sun Microsystems mua lại vào năm 1999.

Sau khi mua lại, Sun Microsystems mở mã nguồn của NetBeans vào tháng 6 năm 2000, từ đó NetBeans trở thành một dự án mã nguồn mở phát triển dưới sự hỗ trợ của cộng đồng người dùng. Oracle Corporation mua lại Sun vào năm 2010, khiến NetBeans trở thành một phần của Oracle. Tuy nhiên, vào tháng 9/2016, Oracle đã đề xuất quyên tặng dự án NetBeans cho Apache Software Foundation, để mở ra mô hình quản trị mới.



Hình 1.14 NetBeans

NetBeans không chỉ hỗ trợ việc phát triển ứng dụng bằng nhiều ngôn ngữ lập trình như Java, PHP, C/C++, JavaScript và các ngôn ngữ khác, mà còn cung cấp một loạt các tính năng như gỡ lỗi, tự động hoàn thành mã, quản lý phiên bản và tích hợp với các công cụ hỗ trợ phát triển phổ biến. IDE này cũng cho phép người dùng mở rộng chức năng, thông qua việc cài đặt các plugin từ cộng đồng lập trình viên.

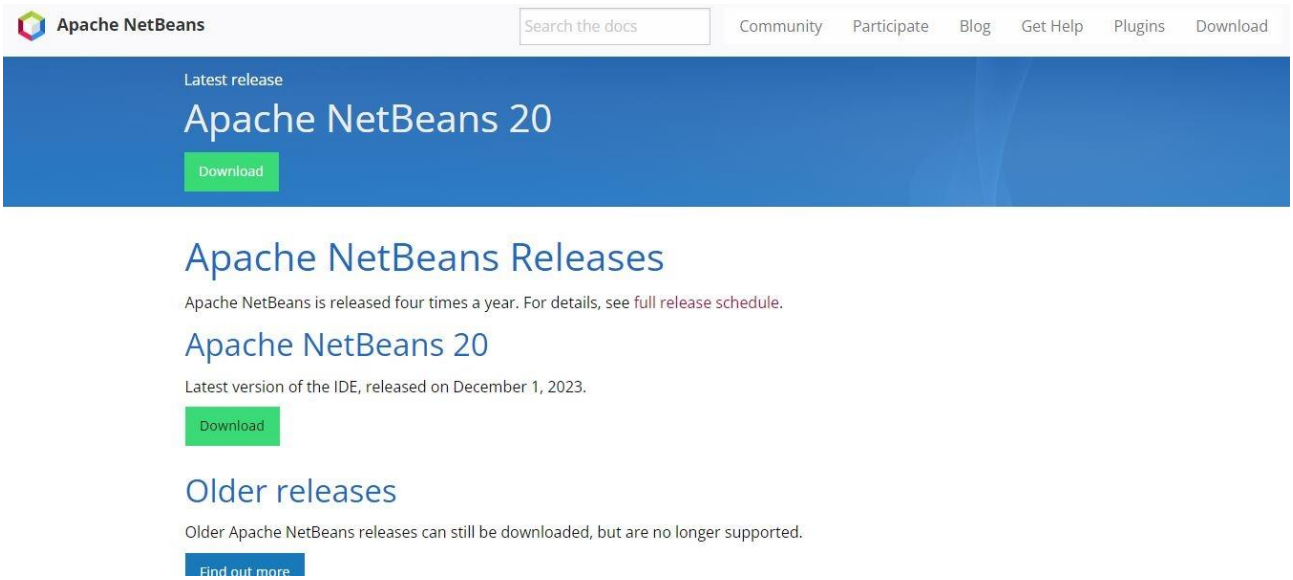
Nhờ khả năng hỗ trợ đa ngôn ngữ và giao diện người dùng thân thiện, NetBeans đã trở thành một công cụ lý tưởng cho việc phát triển ứng dụng đa nền tảng, và hỗ trợ nhu cầu của nhiều dự án khác nhau. Từ người mới bắt đầu đến những lập trình viên có kinh nghiệm, NetBeans đều cung cấp cho họ sự tiện ích, hỗ trợ rất nhiều trong quá trình phát triển phần mềm.

Các tính năng chính của NetBeans

NetBeans IDE đem đến sự thuận tiện và hiệu suất cao trong việc phát triển ứng dụng Java cũng như các ngôn ngữ lập trình khác, với nhiều tính năng nổi bật:

- Tích hợp với Apache: Được phát triển và hỗ trợ bởi Apache Software Foundation, NetBeans cho phép dễ dàng tích hợp các công nghệ và plugin từ Apache, tăng hiệu suất làm việc của người dùng.
- Hỗ trợ Git: Tích hợp sẵn hệ thống quản lý phiên bản Git, giúp các lập trình viên quản lý mã nguồn và làm việc nhóm một cách dễ dàng.
- Tích hợp Plugin: Hỗ trợ một loạt các plugin từ cộng đồng lập trình viên, mở rộng tính năng và tùy chỉnh IDE theo nhu cầu cụ thể của dự án.
- Hỗ trợ phát triển web: IDE hỗ trợ các framework phổ biến cho phát triển ứng dụng web như JavaServer Faces (JSF), Struts và Spring MVC.
- Tích hợp server ứng dụng: NetBeans có tích hợp các server ứng dụng phổ biến như Apache Tomcat, GlassFish và WildFly, giúp người dùng triển khai ứng dụng dễ dàng.
- Debug và Profiling: Cung cấp tính năng gỡ lỗi và đánh giá hiệu năng, giúp người dùng xác định, sửa lỗi trong mã nguồn nhanh chóng và hiệu quả.
- Tích hợp trình biên dịch: Tích hợp trình biên dịch cho nhiều ngôn ngữ, giúp kiểm tra và chạy ứng dụng mà không cần thoát khỏi môi trường IDE.
- Xác định mã nguồn thông minh: Cung cấp tính năng gợi ý mã, hoàn thành mã tự động và xác định lỗi cú pháp, tăng năng suất, giảm lỗi trong quá trình phát triển ứng dụng.
- Có thiết kế giao diện người dùng: NetBeans đi kèm với trình thiết kế giao diện Swing và JavaFX, để thiết kế giao diện người dùng một cách dễ dàng.
- Hỗ trợ dự án lớn: Có khả năng hỗ trợ các dự án lớn có cấu trúc phức tạp, quản lý mã nguồn hiệu quả.
- Hỗ trợ đa ngôn ngữ: Hỗ trợ phát triển ứng dụng trong nhiều ngôn ngữ lập trình như Java, PHP, C/C++, JavaScript, HTML, CSS và các ngôn ngữ khác.
- Giao diện người dùng thân thiện: Giao diện người dùng dễ sử dụng, phù hợp cho cả người mới bắt đầu và người có kinh nghiệm, giúp tăng hiệu suất làm việc.

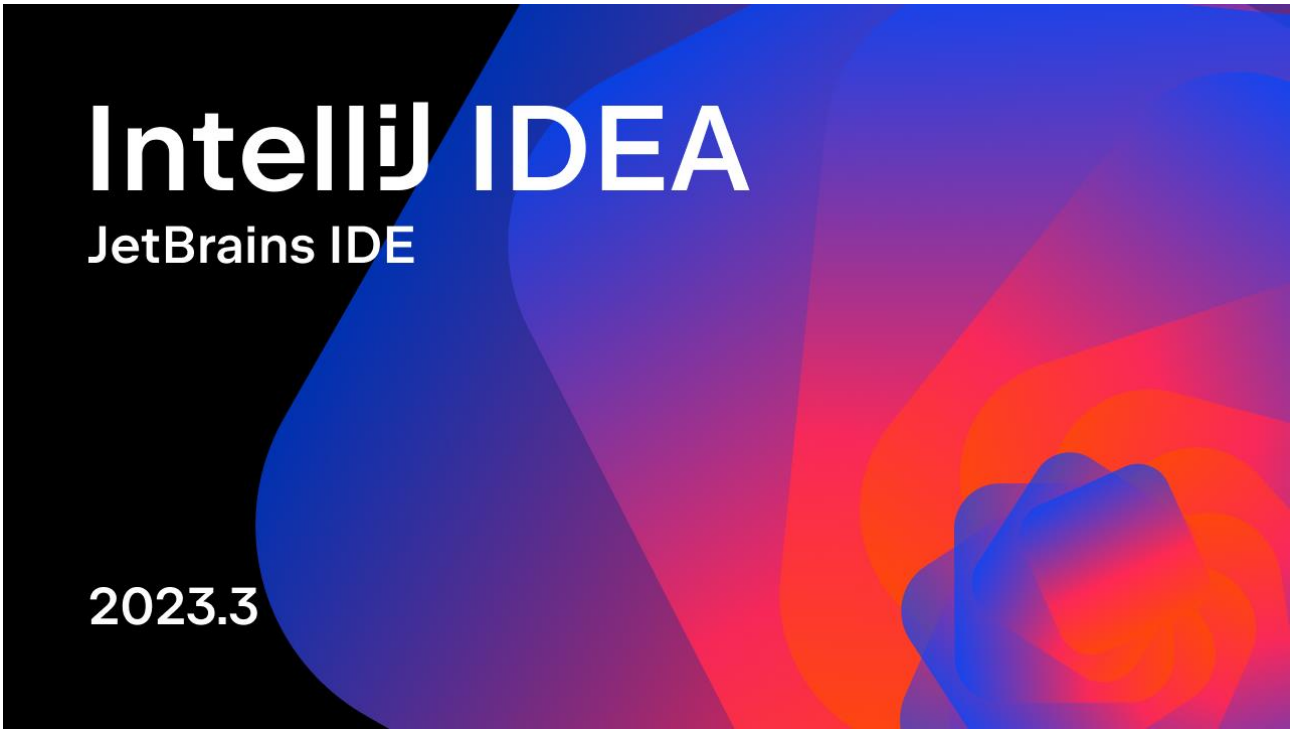
Cài đặt NetBeans: Truy cập trang web chính thức của NetBeans IDE theo đường link: https://netbeans.apache.org/front/main/download/index.html và làm theo hướng dẫn



Hình 1.15 Cài đặt NetBeans

1.5.3. IntelliJ IDEA

IntelliJ IDEA là một môi trường phát triển tích hợp (IDE) mạnh mẽ được thiết kế đặc biệt cho lập trình Java, và cũng có khả năng hỗ trợ một số ngôn ngữ khác như Node.js, Python hay nhiều ngôn ngữ khác. Phiên bản đầu tiên của IntelliJ IDEA được phát hành vào tháng 1/2001, từ đó đến nãy nó đã trở thành một trong những IDE hàng đầu trong cộng đồng phát triển phần mềm.



Hình 1.16 IntelliJ IDEA

Mặc dù ban đầu được biết đến chủ yếu là một IDE Java, IntelliJ IDEA đã mở rộng sự hỗ trợ của mình cho nhiều ngôn ngữ và framework khác nhau. Nó cung cấp một loạt các tính năng thông minh như trình soạn thảo mã thông minh, công cụ phân tích mã và bộ công cụ refactorings giúp tối ưu hóa quy trình phát triển phần mềm.

IntelliJ IDEA có hai phiên bản chính: phiên bản miễn phí (community) và phiên bản trả phí (ultimate). Phiên bản trả phí cung cấp nhiều tính năng mở rộng hơn như hỗ trợ JavaScript, TypeScript, các plugin GWT, Vaadin, kiểm tra đoạn mã trùng lặp cùng các tính năng nổi trội khác. Ngoài ra, IntelliJ IDEA cũng là nền tảng cho nhiều sản phẩm IDE khác của JetBrains như PyCharm, PhpStorm, RubyMine,… giúp đáp ứng nhu cầu phát triển phần mềm đa dạng của các nhà phát triển.

Lợi ích khi sử dụng Intellij IDEA:

- Tập trung quản lý tác vụ: IntelliJ IDEA cung cấp một môi trường tích hợp để các nhà phát triển có thể tập trung vào phát triển, quản lý tất cả các tác vụ hàng ngày của họ một cách hiệu quả.
- Phát triển và gỡ lỗi mã một cách linh hoạt: Nhờ vào các công cụ mạnh mẽ của IntelliJ IDEA, nhà phát triển có thể viết, gỡ lỗi, tái cấu trúc, kiểm tra và tìm hiểu mã của bạn một cách linh hoạt mà không bị ảnh hưởng.
- Hỗ trợ nhiều ngôn ngữ và công nghệ: IntelliJ IDEA xử lý một cơ sở mã hỗn hợp của nhiều ngôn ngữ lập trình như Java, Ruby, Groovy, Python và Scala, đồng thời hỗ trợ nhiều framework và công nghệ khác nhau.
- Duy trì chất lượng mã tự động: Công cụ tự động của IntelliJ IDEA giúp duy trì chất lượng mã thông qua việc theo dõi và sửa lỗi trên tất cả các cấp độ, từ câu lệnh đến kiến trúc tổng thể.
- Tạo mã sạch và nhanh chóng: IntelliJ IDEA được thiết kế để tạo ra mã sạch và có thể thực hiện mã trong thời gian ngắn nhất, giúp tăng hiệu suất làm việc của nhà phát triển.
- Đa quy mô: IntelliJ IDEA không chỉ phù hợp cho các nhà phát triển cá nhân mà còn đối với các doanh nghiệp với mọi quy mô, từ nhỏ đến lớn.
- Hỗ trợ hệ thống điều khiển phiên bản và CI/CD: IntelliJ IDEA tích hợp tốt với các hệ thống điều khiển phiên bản phổ biến và server tích hợp liên tục như TeamCity, giúp tạo ra một quy trình phát triển phần mềm mạnh mẽ và linh hoạt.

Cài đặt IntelliJ IDEA: Truy cập vào trang web chính thức của IntelliJ IDEA tại địa chỉ https://www.jetbrains.com/idea/download và làm theo hướng dẫn cài đặt.



Hình 1.17 Sử dụng IntelliJ IDEA

1.6. TỔNG KẾT CHƯƠNG

Chương 1 cung cấp một cái nhìn tổng quát về các phương pháp tiếp cận trong lập trình, từ nền tảng lý thuyết đến ứng dụng thực tế. Mục tiêu chính của chương là giúp người đọc nắm vững các khái niệm cơ bản trong lập trình và làm quen với lập trình hướng đối tượng, từ đó tạo nền tảng cho việc giải quyết các bài toán phức tạp hơn.

1. **Giới thiệu các phương pháp tiếp cận trong lập trình:** Ba phương pháp lập trình chính là lập trình tuyến tính, lập trình hướng chức năng, và lập trình hướng đối tượng được giới thiệu. Mỗi phương pháp có ưu điểm riêng và phù hợp với từng loại bài toán khác nhau, từ đó đặt nền tảng để hiểu rõ vai trò của lập trình hướng đối tượng.
2. **Các khái niệm cơ bản của lập trình hướng đối tượng:** Các khái niệm chính của lập trình hướng đối tượng bao gồm:
   - **Đối tượng:** Thành phần cơ bản trong lập trình hướng đối tượng, đại diện cho các thực thể có tính chất và hành vi riêng.
   - **Lớp đối tượng:** Định nghĩa khuôn mẫu của đối tượng, giúp tổ chức và tái sử dụng mã lệnh.

- **Trừu tượng hóa:** Quá trình đơn giản hóa đối tượng để tập trung vào các đặc điểm chính.
- **Bao bọc và che giấu thông tin:** Giữ cho dữ liệu trong đối tượng an toàn và không bị can thiệp trực tiếp từ bên ngoài.
- **Kế thừa:** Cho phép một lớp con kế thừa các thuộc tính và phương thức của lớp cha, giúp tái sử dụng mã và mở rộng chức năng.
- **Đa hình:** Khả năng dùng cùng một phương thức để thực hiện các hành động khác nhau, tùy thuộc vào ngữ cảnh.

3. **Cách tiếp cận và giải quyết bài toán theo phương pháp hướng đối tượng:** Bao gồm các bước từ mô tả bài toán, đặc tả yêu cầu, trích chọn đối tượng, mô hình hoá lớp đối tượng, thiết kế tổng quát, và thiết kế chi tiết. Các bước này giúp cấu trúc và triển khai bài toán hiệu quả, đảm bảo rõ ràng và dễ bảo trì.
4. **Ngôn ngữ lập trình hướng đối tượng phổ biến:** Một số ngôn ngữ như Java, C++, Python... được giới thiệu, nêu rõ ưu điểm và ứng dụng của từng ngôn ngữ trong lập trình hướng đối tượng.
5. **Các Editor biên soạn chương trình phổ biến:** Giới thiệu về các công cụ hỗ trợ lập trình như Eclipse, NetBeans, và IntelliJ IDEA, giúp lập trình viên làm việc hiệu quả và tối ưu hoá quy trình phát triển.

1.7. CÂU HỎI TRẮC NGHIỆM

1. Lập trình tuyến tính tập trung vào cách:

A. Sử dụng đối tượng và lớp để giải quyết vấn đề

B. Tổ chức các lệnh theo trình tự

C. Xây dựng các chức năng độc lập, không có trạng thái

D. Mô hình hóa các thực thể trong thực tế

2. Lập trình hướng chức năng tập trung vào:

A. Tính kế thừa và đa hình

B. Tính bao bọc và che giấu thông tin

C. Tổ chức chương trình thành các hàm và chức năng

D. Sử dụng các lớp và đối tượng để mô hình hóa vấn đề

3. Lập trình hướng đối tượng chủ yếu sử dụng:

A. Các cấu trúc tuyến tính

B. Các hàm toán học và phương thức

C. Các lớp và đối tượng

D. Các mảng và biến số học

4. Trong lập trình hướng đối tượng, "lớp" là:

A. Đối tượng đại diện cho một thực thể trong thế giới thực

B. Cách tổ chức dữ liệu dưới dạng bảng

C. Khuôn mẫu của đối tượng, chứa các thuộc tính và phương thức chung

D. Một biến đơn giản dùng để lưu trữ dữ liệu

5. Kỹ thuật che giấu thông tin (Encapsulation) trong lập trình hướng đối tượng nhằm mục đích:

A. Đảm bảo dữ liệu không bị truy cập trực tiếp từ bên ngoài đối tượng

B. Cho phép đối tượng kế thừa các thuộc tính của đối tượng khác

C. Tạo ra các bản sao khác nhau của cùng một đối tượng

D. Thực hiện tính toán trên các hàm toán học

6. Khái niệm kế thừa (Inheritance) trong lập trình hướng đối tượng là:

A. Một phương thức cho phép đối tượng tồn tại độc lập

B. Cách đối tượng nhận thuộc tính và phương thức từ lớp cha

C. Cách để chia nhỏ chương trình thành các phần chức năng độc lập

D. Mô hình hóa các dữ liệu tuyến tính

7. Trong lập trình hướng đối tượng, "đa hình" (Polymorphism) có nghĩa là:

A. Khả năng một đối tượng tồn tại dưới nhiều dạng khác nhau

B. Khả năng chia nhỏ chương trình thành các mô-đun độc lập

C. Khả năng các lớp chia sẻ cùng một phương thức

D. Khả năng tổ chức dữ liệu thành các mảng và chuỗi

8. Quá trình đơn giản hóa đối tượng để chỉ tập trung vào các đặc điểm quan trọng gọi là:

A. Bao bọc

B. Đa hình

C. Trừu tượng hóa

D. Kế thừa

9. Một lớp trong lập trình hướng đối tượng có thể chứa các thành phần nào?

A. Biến và hàm

B. Biến và phương thức

C. Đối tượng và phương thức

D. Mảng và chuỗi

10. Lợi ích chính của lập trình hướng đối tượng là:

A. Chia nhỏ chương trình thành các mô-đun độc lập

B. Cho phép tạo ra các đối tượng với thuộc tính và hành vi cụ thể

C. Tạo ra các lệnh thực thi theo trình tự

D. Giúp chương trình chạy nhanh hơn

11. Bước đầu tiên trong quy trình giải quyết bài toán theo hướng đối tượng là:

A. Trích chọn đối tượng

B. Mô tả bài toán

C. Thiết kế chi tiết

D. Xác định lớp đối tượng

12. Phần mềm nào dưới đây là một IDE hỗ trợ lập trình hướng đối tượng?

A. Microsoft Word

B. Eclipse

C. Adobe Photoshop

D. VLC Media Player

13. Trong lập trình hướng đối tượng, đối tượng được định nghĩa là:

A. Một thực thể đơn giản lưu trữ dữ liệu

B. Một bản sao của lớp

C. Một thuộc tính của lớp

D. Một phương thức cho việc chia nhỏ chương trình

14. Ngôn ngữ lập trình nào dưới đây không phải là ngôn ngữ hướng đối tượng?

A. Java

B. C++

C. Python

D. C

15. Trong lập trình hướng đối tượng, phương thức là:

A. Thuộc tính của lớp

B. Hàm được định nghĩa trong lớp và dùng để thực hiện hành vi của đối tượng

C. Một biến toàn cục dùng cho mọi đối tượng

D. Phần mở rộng của đối tượng