

ỦY BAN NHÂN DÂN TP. HỒ CHÍ MINH
TRƯỜNG CAO ĐẲNG CÔNG NGHỆ THỦ ĐỨC
KHOA CÔNG NGHỆ THÔNG TIN

GIÁO TRÌNH

HỌC PHẦN: LẬP TRÌNH JAVA
NGÀNH/NGHỀ: CÔNG NGHỆ THÔNG TIN
TRÌNH ĐỘ: CAO ĐẲNG

Tái bản có bổ sung chỉnh sửa

TP. Hồ Chí Minh, năm 2019

LỜI GIỚI THIỆU

Học phần Lập trình Java là học phần bắt buộc của ngành Công nghệ thông tin. Đây là học phần cơ bản về ngôn ngữ lập trình Java, là học phần nền tảng cho các học phần sau như lập trình Android và học phần Phát triển ứng dụng Java. Ở các học phần trước, sinh viên đã được làm quen về kỹ thuật lập trình, cấu trúc dữ liệu thì qua học phần Lập trình Java, sinh viên sẽ áp dụng các kiến thức kỹ thuật lập trình đã học và kết hợp kiến thức về ngôn ngữ lập trình Java để giải quyết các bài toán ứng dụng Java đơn giản.

Giáo trình này được biên soạn dựa theo đề cương học phần “Lập trình Java” của Khoa Công nghệ thông tin – Trường Cao đẳng Công nghệ Thủ Đức. Dù đã rất cố gắng, song sẽ không tránh khỏi những thiếu sót, rất mong nhận được sự góp ý chân thành từ các quý đọc giả để giáo trình được hoàn thiện hơn.

Tp. Hồ Chí Minh, ngày ... tháng ... năm

Tác giả biên soạn

Nguyễn Thị Hồng Mỹ

MỤC LỤC

_Toc12999111

CHƯƠNG 1. TỔNG QUAN VỀ JAVA.....1

1.1. Giới thiệu ngôn ngữ lập trình Java.....	2
1.1.1. Lịch sử phát triển của Java	2
1.1.2. Đặc trưng của Java	2
1.1.3. Các loại ứng dụng Java.....	3
1.2. Môi trường lập trình Java, máy ảo Java, JDK	4
1.2.1. Môi trường phát triển Java	4
1.2.2. Máy ảo Java (JVM - Java Virtual Machine)	4
1.2.3. JDK	9
1.3. IDE để lập trình Java.....	14
1.4. Cấu trúc chương trình Java đơn giản	17
1.5. Các quy tắc cơ bản của ngôn ngữ Java	18
1.5.1. Cú pháp comment trong Java	18
1.5.2. Quy tắc đặt tên trong Java	18
1.6. Biên dịch và thực thi chương trình Java	20
Case Study Quản Lý Sinh Viên	23
Bài tập	24

CHƯƠNG 2. NỀN TẢNG JAVA CƠ BẢN27

2.1. Cú pháp Java cơ bản	28
2.1.1. Kiểu dữ liệu và biến.....	28
2.1.2. Toán tử.....	31
2.1.3. Cấu trúc điều khiển.....	32
2.1.4. Vòng lặp	34
2.1.5. Chuỗi (String) trong Java	35
2.1.6. Mảng một chiều, Mảng hai chiều	37
2.1.7. Một số thư viện thường dùng	37
2.1.8. Định dạng Formating.....	38
2.2. Lớp và đối tượng trong Java	40

2.2.1. Lớp - Class.....	40
2.2.2. Đối tượng - Object	42
2.2.3. Từ khóa static.....	43
2.2.4. Từ khóa final.....	44
2.2.5. Package	44
2.2.6. Phạm vi truy xuất.....	45
2.3. Đặc điểm hướng đối tượng trong Java.....	45
2.3.1. Tính trừu tượng (Abstraction)	46
2.3.2. Tính đóng gói (Encapsulation)	45
2.3.3. Tính kế thừa (Inheritance)	45
2.3.4. Tính đa hình (Polymorphism).....	49
2.4. JUnit Test	50
Case Study Quản Lý Sinh Viên	54
Bài tập	56
CHƯƠNG 3. XỬ LÝ NGOẠI LỆ - EXCEPTION HANDLING	58
3.1. Giới thiệu ngoại lệ Exception	59
3.2. Xử lý ngoại lệ Exception.....	60
3.2.1. Khối try/catch	61
3.2.2. Khối finally	61
3.2.3. Ủy nhiệm ngoại lệ throw, throws	61
3.3. Ngoại lệ được kiểm tra và ngoại lệ không được kiểm tra.....	62
3.4. Định nghĩa một ngoại lệ mới	63
Case Study Quản Lý Sinh Viên	64
Bài tập	65
CHƯƠNG 4. ĐỌC FILE VÀ GHI FILE.....	67
4.1. Đọc ghi file nhị phân.....	68
4.2. Đọc ghi file văn bản	71
4.3. Đọc ghi đối tượng	73
4.4. Lớp File	75
Case Study Quản Lý Sinh Viên	77

Bài tập	80
CHƯƠNG 5. LẬP TRÌNH VỚI CƠ SỞ DỮ LIỆU.....	81
5.1. JDBC - Java Database Connectivity	82
5.1.1. JDBC Driver	82
5.1.2. Kết nối cơ sở dữ liệu bằng JDBC	83
5.2. Tạo ứng dụng JDBC đơn giản	85
5.2.1. Ứng dụng 1: Thao tác với CSDL.....	85
5.2.2. Ứng dụng 2: Thao tác với CSDL dùng store procedure.....	88
Case Study Quản Lý Sinh Viên	93
Bài tập	98
CHƯƠNG 6. THIẾT KẾ GIAO DIỆN NGƯỜI DÙNG GUI.....	101
6.1. Thành phần GUI	102
6.1.1. Frame	102
6.1.2. Một số thành phần cơ bản của GUI.....	103
6.1.3. Một số Layout thông dụng	104
6.2. Xử lý sự kiện.....	108
6.3. Graphics 2D	113
6.3.1. Font chữ.....	113
6.3.2. Color	115
6.3.3. Các hình vẽ cơ bản	118
Case Study Quản Lý Sinh Viên	124
Bài tập	126
CHƯƠNG 7. COLLECTION	128
7.1. Interface Collection.....	129
7.1.1. ArrayList class.....	130
7.1.2. HashSet class	131
7.1.3. HashMap class.....	132
7.2. Class Collections.....	132
7.2.1. Phương thức Sort: Sử dụng interface Comparable, Comparator.....	132
7.2.2. Một số phương thức của lớp Collections	135

Case Study Quản Lý Sinh Viên	136
Bài tập	140
CHƯƠNG 8. THIẾT KẾ HƯỚNG ĐÓI TƯỢNG	144
8.1. Lớp trừu tượng Abstract class	145
8.2. Giao tiếp Interface	146
8.3. Thiết kế ứng dụng	147
8.3.1. Sử dụng UML	147
8.3.2. Ứng dụng Quản lý hóa đơn.....	148
8.3.3. Ứng dụng Bài toán tổng hợp (has-a, is-a, abstract class, Interface)	155
8.4. Mô hình MVC	155
Case Study Quản Lý Sinh Viên	159
Bài tập	166
CHƯƠNG 9. ĐA LUỒNG MULTITHREAD.....	169
9.1. Thread, Multithread.....	170
9.2. Tạo và quản lý Thread.....	170
9.2.1. Main Thread.....	170
9.2.2. Lớp Thread.....	171
9.2.3. Tạo Thread.....	173
9.3. Độ ưu tiên của Thread	174
9.4. Đồng bộ các Thread	175
Case Study Quản Lý Sinh Viên	178
Bài tập	179

GIÁO TRÌNH HỌC PHẦN

Tên học phần: Lập trình Java

Mã học phần:

Vị trí, tính chất, ý nghĩa và vai trò của học phần:

- Vị trí: Đây là học phần chuyên ngành bắt buộc, hệ cao đẳng ngành công nghệ thông tin.
- Tính chất: Học phần này cung cấp cho sinh viên kiến thức về ngôn ngữ lập trình Java, tư duy về lập trình và những kỹ thuật cần thiết trong lập trình để sinh viên có thể xây dựng được ứng dụng phần mềm Java đơn giản.
- Ý nghĩa và vai trò của học phần: Thông qua các hoạt động học tập sinh viên có thể hoàn thiện dần tính tư duy hệ thống và thói quen tuân thủ các quy định trong môi trường làm việc chuyên nghiệp.

Mục tiêu của học phần:

- Về kiến thức:
 - + Tổng hợp được kiến thức cơ bản về ngôn ngữ lập trình Java
 - + Trình bày cách xây dựng một ứng dụng theo lập trình hướng đối tượng.
- Về kỹ năng:
 - + Thành thạo các công cụ IDE để phát triển Java
 - + Tạo được ứng dụng console và ứng dụng giao diện đồ họa GUI
 - + Phân tích thiết kế và xây dựng được ứng dụng Java tương tác với file hoặc cơ sở dữ liệu.
- Về năng lực tự chủ và trách nhiệm:
 - + Rèn luyện thói quen tư duy hệ thống, học tập tích cực, chủ động.
 - + Tuân thủ các yêu cầu về quy ước viết code.

CHƯƠNG 1. TỔNG QUAN VỀ JAVA

Học xong chương này, người học có thể:

- + Trình bày kiến thức cơ bản về ngôn ngữ Java như: lịch sử phát triển của Java, các đặc điểm của ngôn ngữ Java, khái niệm máy ảo Java.
- + Liệt kê các quy tắc đặt tên cơ bản của ngôn ngữ Java.
- + Sử dụng bộ công cụ Eclipse/Netbean để viết chương trình, biên dịch và thực thi một ứng dụng Java đơn giản.

1.1. Giới thiệu ngôn ngữ lập trình Java

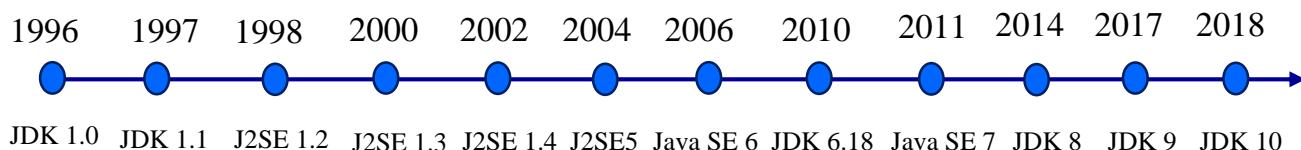
1.1.1. Lịch sử phát triển của Java

Ngôn ngữ lập trình Java được ra đời bởi tác giả là ông James Gosling và đồng nghiệp thuộc công ty Sun Microsystems vào đầu những năm 1990s. Ban đầu ngôn ngữ này có tên gọi là Oak, đến năm 1993 được đổi tên thành Java. Năm 2010, Sun Microsystems được Oracle mua lại.



Hình 1.1. James Gosling – Người tạo ra ngôn ngữ Java

Các phiên bản Java đã phát hành:



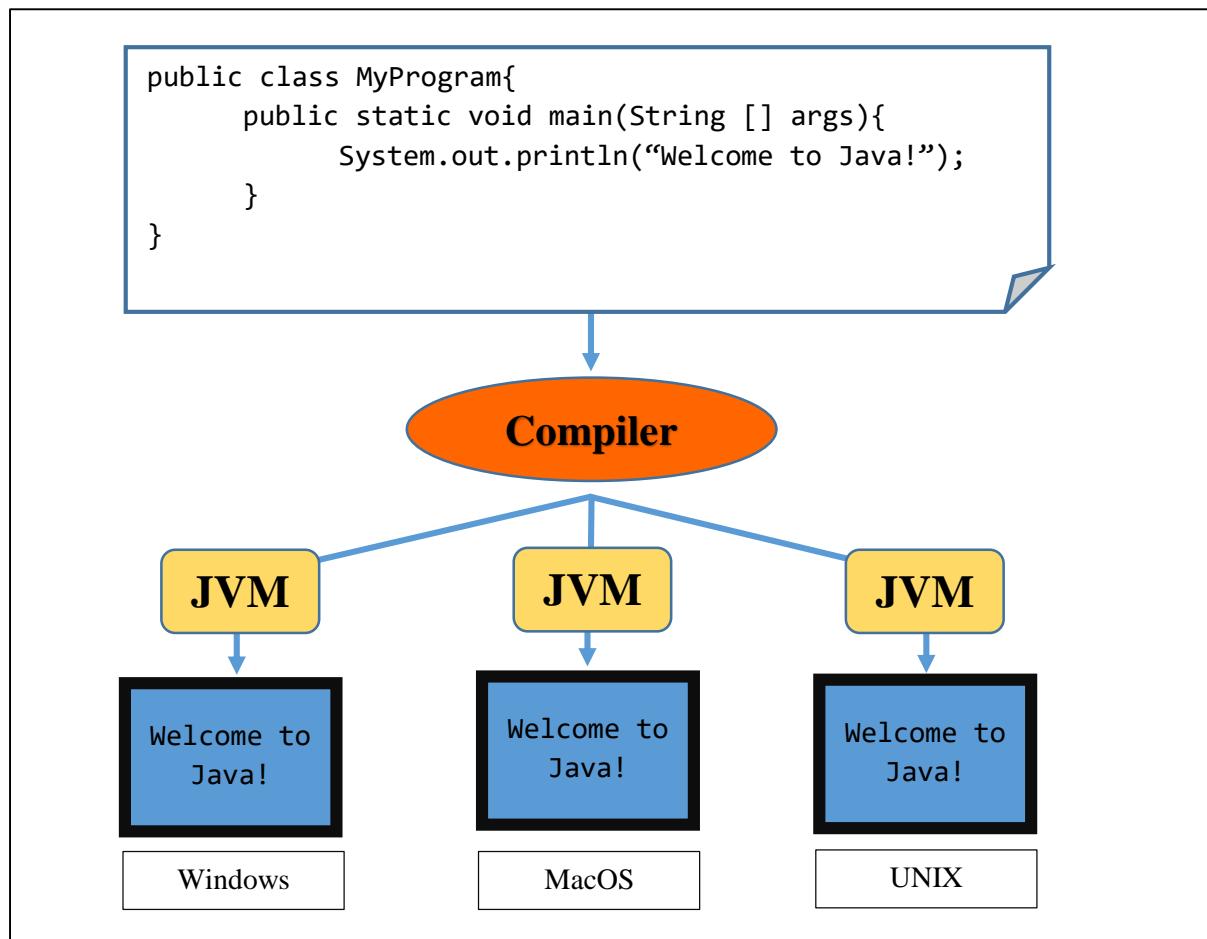
Hình 1.2. Logo Java

1.1.2. Đặc trưng của Java

Đơn giản: Java là ngôn ngữ dễ học, không có thao tác con trỏ, không có file header, không có đa kế thừa.

Hướng đối tượng: Java là ngôn ngữ hoàn toàn hướng đối tượng, tất cả dữ liệu và hàm đều nằm trong một lớp nhất định.

Độc lập nền tảng: Đối với ngôn ngữ khác, cần có các trình biên dịch khác nhau để biên dịch mã nguồn chương trình cho phù hợp với mỗi nền tảng (phần cứng và hệ điều hành) khác nhau. Ngôn ngữ Java được tạo ra với tiêu chí ‘Viết một lần, thực thi khắp nơi’ (Write Once, Run Anywhere – WORA) nhờ có máy ảo Java. Với Java, trình biên dịch **javac** sẽ biên dịch mã nguồn thành mã **bytecode** (độc lập nền tảng). Mã bytecode này được máy ảo Java (JVM - Java Virtual Machine) dịch thành mã máy thích hợp để thực thi chương trình trên các nền tảng khác nhau.



Hình 1.3. Java độc lập nền tảng nhờ có máy ảo Java

An toàn: Java quản lý thực thi chương trình với nhiều mức để kiểm soát tính an toàn: dữ liệu và các phương thức được đóng gói bên trong lớp; trình biên dịch kiểm tra đảm bảo mã lệnh tuân theo các nguyên tắc của Java; trình thông dịch kiểm soát đảm bảo các quy tắc an

toàn trước khi thực thi; kiểm soát việc nạp các lớp vào bộ nhớ để giám sát việc vi phạm giới hạn truy xuất trước khi nạp vào hệ thống.

Phân tán: Java hỗ trợ lập trình phân tán. Java RMI – Remote Method Invocation là một kỹ thuật cài đặt các đối tượng phân tán để giải quyết một số vấn đề trong việc truyền thông qua mạng giữa Client/Server.

Đa luồng: Chương trình Java hỗ trợ cơ chế đa luồng (MultiThreading) để thực thi nhiều công việc đồng thời.

Tự động thu gom rác: Máy ảo Java hỗ trợ cơ chế gom ‘rác’ tự động.

1.1.3. Các loại ứng dụng Java

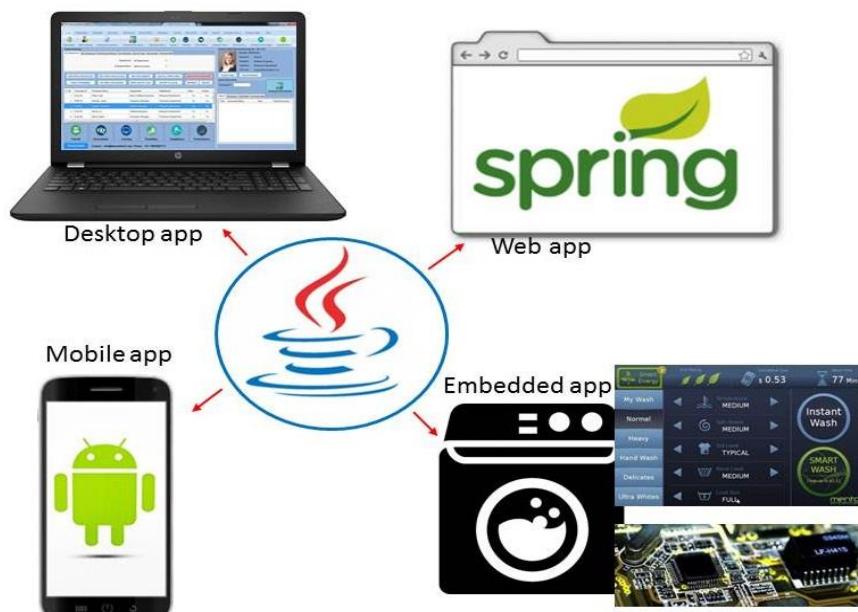
Ngôn ngữ Java có thể dùng để viết nhiều loại ứng dụng như ứng dụng Desktop, game, ứng dụng website, ứng dụng Mobile, ứng dụng nhúng Embedded.

J2SE - Java 2 Standard Edition: Phát triển các ứng dụng cơ bản

J2EE - Java 2 Enterprise Edition: Phát triển các ứng dụng nâng cao, như ứng dụng web

J2ME - Java 2 Micro Edition: Phát triển các ứng dụng dùng cho thiết bị nhỏ, như ứng dụng di động, ứng dụng nhúng

Android Studio và SDK (Software Development Kit): Phát triển ứng dụng di động

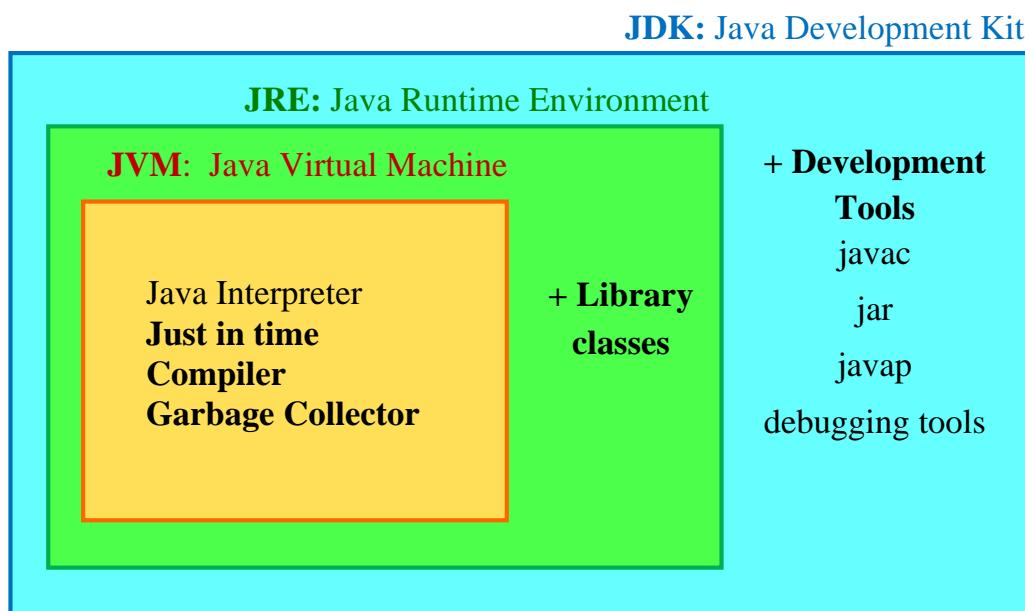


Hình 1.4. Các loại ứng dụng Java

1.2. Môi trường lập trình Java, máy ảo Java, JDK

1.2.1. Môi trường phát triển Java

Java Platform là tên cho một nhóm các chương trình cho phép phát triển và chạy chương trình viết bằng ngôn ngữ Java.



Hình 1.5. Java Platform

1.2.2. Máy ảo Java (JVM - Java Virtual Machine)

Máy ảo Java là môi trường để thực thi các file.class đã được biên dịch từ ngôn ngữ lập trình Java. Máy ảo Java là một phần mềm dựa trên cơ sở máy tính ảo. JVM có thể liên kết làm việc với phần cứng và hệ điều hành. Mỗi nền tảng/hệ điều hành khác nhau (Windows, IOS, Linux...) có một loại JVM khác nhau.

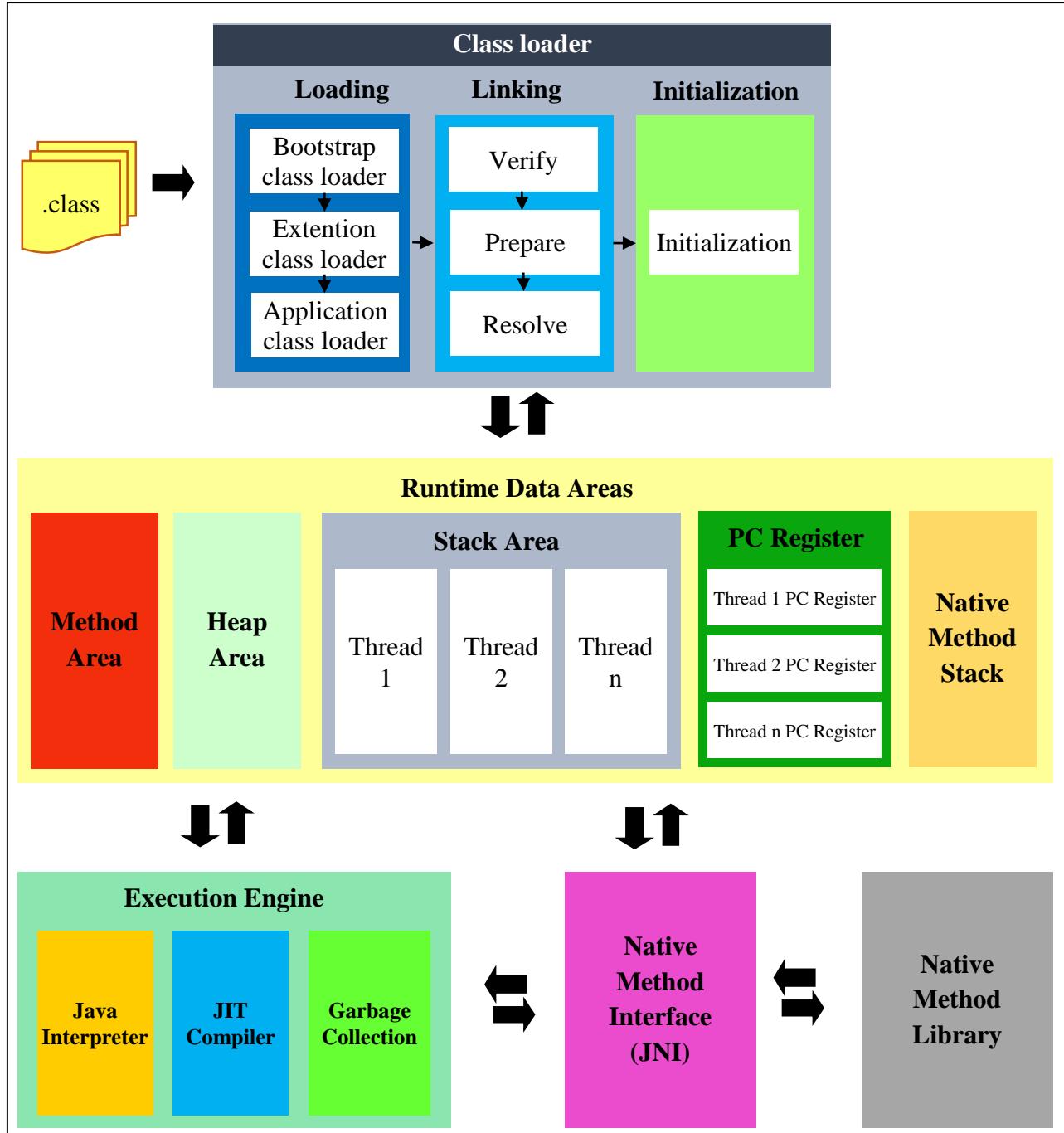
JVM thực hiện 3 công việc chính:

Class loader: Tải code (các class, resource), kiểm tra code

Runtime Data Areas: Cung cấp môi trường runtime

Execution Engine: Thực thi code

1.2.2.1. Cấu trúc của JVM



Hình 1.6. Cấu trúc máy ảo Java

1. Class Loader có 3 pha xử lý: Loading, Linking và Initialization.

Pha loading có 3 bộ loader tham gia vào việc loading:

- **Boot Strap Class Loader** - Chịu trách nhiệm load các class từ bootstrap classpath. Bộ loader này có mức ưu tiên cao nhất.
- **Extension Class Loader** - Load các class trong folder jre/lib
- **Application Class Loader** - load các class mức ứng dụng

Pha linking chia thành 3 bước:

- **Verify**: Bộ bytecode verifier sẽ kiểm tra đoạn bytecode được generate có hợp lệ hay không. Nếu không hợp lệ, verification error sẽ được xuất ra.
- **Prepare**: ở bước này, tất cả các biến static được cấp phát bộ nhớ và gán cho giá trị mặc định
- **Resolve**: tất cả tham chiếu bộ nhớ dạng ký hiệu (symbolic memory reference) được thay thế bằng tham chiếu dạng nguyên thủy (original reference)

Pha initialization là pha cuối cùng của bộ Class Loader. Tại pha này, tất cả biến static được gán giá trị (giá trị này đã được developer ghi trong file *.java) và các static block được thực thi.

2. Runtime Data Area: được chia nhỏ thành 5 mô-đun con

- i. **Method Area** - nơi lưu trữ dữ liệu mức class - toàn bộ các dữ liệu có trong một class sẽ lưu ở đây. Mỗi JVM chỉ có một Method Area và nó có thể được sử dụng bởi nhiều tiến trình.
- ii. **Heap Area** - lưu trữ object và những thứ liên quan như instance variable, arrays. Giống như Method Area, mỗi JVM chỉ có một Heap Area. Vì 2 vùng này được các tiến trình chia sẻ với nhau nên dữ liệu lưu ở đây **không đảm bảo thread-safe**.
- iii. **Stack Area** - Stack Area **đảm bảo thread-safe** bởi mỗi tiến trình sẽ được cấp phát một **runtime stack**. Tất cả biến cục bộ được tạo trong bộ nhớ stack. Mỗi khi có method call - lệnh gọi hàm, một Stack Frame được mở. Mỗi Stack Frame chứa 3 thực thể con:
 - **Local Variable Array** - Mảng các biến cục bộ
 - **Operand Stack** - ngăn xếp chứa các toán hạng

- **Frame Data** – chứa các ký hiệu liên quan tới method. Trong trường hợp xảy ra exception, thông tin trong khối catch cũng ở đây.

iv. **PC Registers** - PC là **Program Counter** - một thanh ghi lưu địa chỉ của lệnh đang thực thi. Mỗi thread sẽ sở hữu riêng một PC.

v. **Native Method stacks** - giữ các thông tin tự nhiên của method. Mỗi thread đều sở hữu một Native method stack.

3. Bộ thực thi - Execution Engine

Phần bytecode được gán qua **Runtime Data Area** sẽ được thực thi bởi **Execution Engine**. Tiếp đó, mô-đun này đọc và thực thi từng đoạn byte code.

Các mô-đun con của Execution Engine:

- i. **Interpreter**: Trình thông dịch. Trình thông dịch sẽ thông dịch bytecode nhanh nhưng nhược điểm là thực thi chậm. Bên cạnh đó, một nhược điểm nữa của trình thông dịch là method được gọi bao nhiêu lần thì cần bấy nhiêu lần thông dịch.
- ii. **JIT Compiler - Just In Time Compiler**: JIT Compiler sẽ trung hòa các nhược điểm của interpreter. Execution Engine dùng Interpreter để thông dịch code, và khi nó phát hiện ra code bị lặp lại, nó sẽ dùng JIT Compiler. JIT Compiler biên dịch toàn bộ bytecode (thay vì thông dịch từng dòng như interpreter) sau đó chuyển đổi thành native code. Chỗ native code này sẽ được sử dụng trực tiếp cho các lời gọi hàm lặp đi lặp lại, nhờ đó, hiệu năng được cải thiện đáng kể.
- iii. **Garbage Collector**: Xác định và loại bỏ các object đã tạo ra nhưng không được tham chiếu (unreferenced) khỏi bộ nhớ Heap, thu hồi dung lượng bộ nhớ để cấp phát cho đối tượng mới. Có thể kích hoạt thủ công bộ GC thông qua lệnh "*System.gc()*".

4. JNI và Native Method Libraries:

JNI - Java Native Interface - sẽ tương tác với Native Method Libraries và cung cấp Native Libraries cần thiết cho Execution Engine.

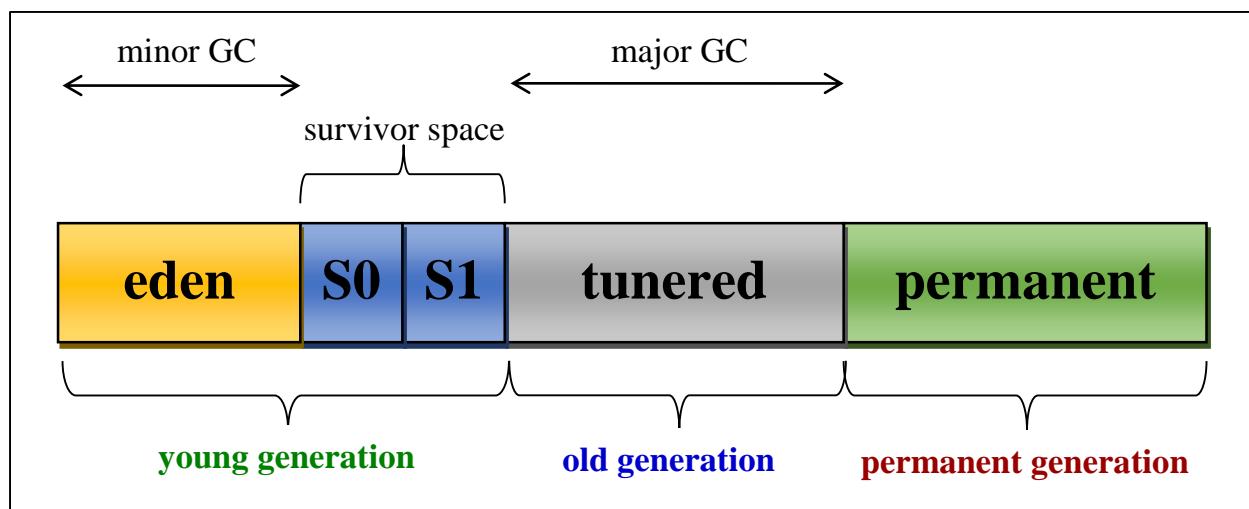
1.2.2. Quá trình dọn rác trong Java

Garbage Collection trong Java được xem như một quá trình tự động thực thi nhiệm vụ quản lý bộ nhớ.

Quá trình thu gom rác gồm 3 bước chính sau:

1. **Marking:** đánh dấu những Object còn sử dụng và những Object không còn sử dụng.
2. **Normal deleting:** Trình Garbage Collector sẽ xóa các Object không còn sử dụng.
3. **Deletion with Compacting:** Sau khi những Object không còn được sử dụng bị xóa, những Object còn được sử dụng sẽ được gom lại gần nhau. Điều đó làm tăng hiệu suất sử dụng bộ nhớ trống để cấp phát cho những Object mới.

Để thực hiện việc tự động giải phóng các Object khi chúng không được sử dụng thì bộ nhớ Heap được chia thành các phần nhỏ như sau: chia object theo "tuổi hoạt động"



Hình 1.7. Cơ chế tự động gom rác

Young Generation – thế hệ “trẻ”: Vùng Young generation được chia thành 2 vùng nhỏ hơn là Eden (khởi thủy) và survivor (sống sót). Vùng Survivor được chia thành 2 vùng nhỏ hơn là S0, S1. Ban đầu, các object mới được tạo sẽ nằm trong vùng Eden. Eden đầy thì Minor GC chuyển chúng sang vùng S0, S1. Khi vùng nhớ Young generation đầy thì garbage collector là Minor GC hoạt động.

Old generation - thế hệ “già”: vùng này chứa các object chuyển từ young generation (Minor GC liên tục theo dõi các Object ở S0, S1. Sau thời gian hoạt động đủ “lâu”, mỗi bộ garbage

collector sẽ định nghĩa bao nhiêu được coi là “lâu” mà Object vẫn còn được sử dụng thì chúng mới được chuyển sang vùng nhớ Old generation). Vùng **Old generation** được quản lý bởi garbage collector khác là Major GC.

Permanent generation – thế hệ “bất tử”: vùng này gồm metadata (các thư viện JavaSE, mô tả các class và các method của ứng dụng). Do đó, khi phải “dọn” các class, method không cần thiết, garbage collector sẽ tìm kiếm trong nhóm này.

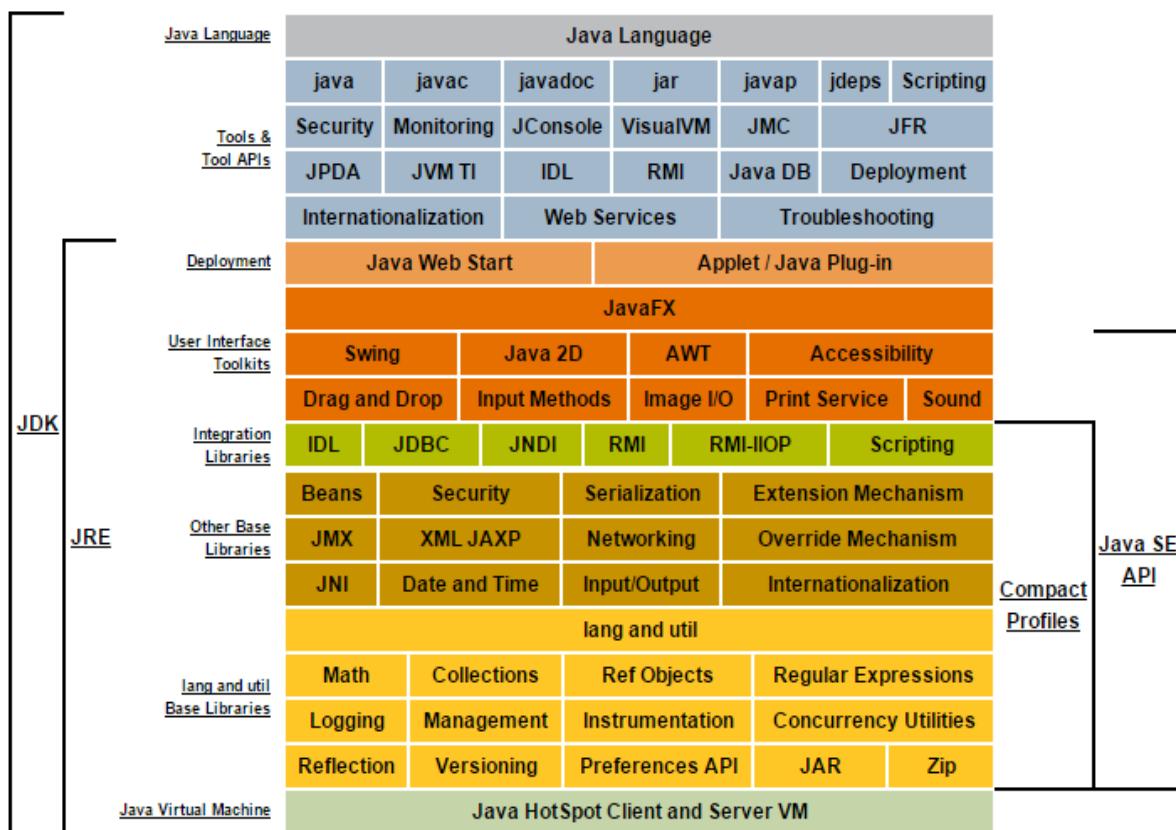
Lưu ý: không thể dự đoán garbage collector sẽ chạy ở thời điểm nào. Kể cả khi gọi nó một cách tường minh với System.gc() hay Runtime.gc(), vẫn không thể chắc chắn được garbage collector có chạy hay không. Về việc tùy chỉnh garbage collector, tốt nhất là điều chỉnh các setting flag của JVM (chính là các tham số tương ứng với từng bộ garbage collector được liệt kê ở trên), hoặc quy định kích thước khi cấp phát và kích thước tối đa của vùng nhớ heap mà chương trình sử dụng, hoặc điều chỉnh kích thước của từng nhóm “thế hệ”.

1.2.3. JDK

JDK (Java Development Kit): bộ công cụ phát triển Java chứa cả JRE và JVM, đây là phần lõi của môi trường Java và cung cấp tất cả các công cụ, thực thi chương trình, biên dịch file. Trên mỗi hệ điều hành khác nhau có bộ cài đặt JDK tương ứng.

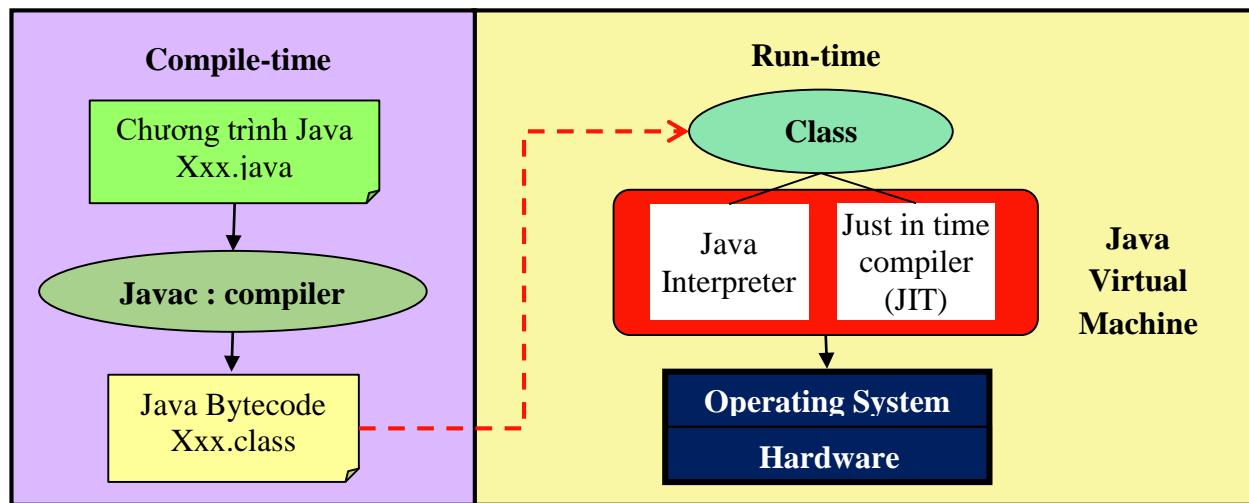
JRE (Java Runtime Environment): Môi trường chạy Java gồm các thư viện và các file được sử dụng khi runtime.

Lưu ý: JDK được dùng cho mục đích phát triển (lập trình, debug), nếu chỉ cần chạy chương trình Java thì ko nhất thiết phải cài JDK mà chỉ cần cài JRE



Hình 1.8. Cấu trúc JDK

Quá trình biên dịch và thông dịch chương trình Java



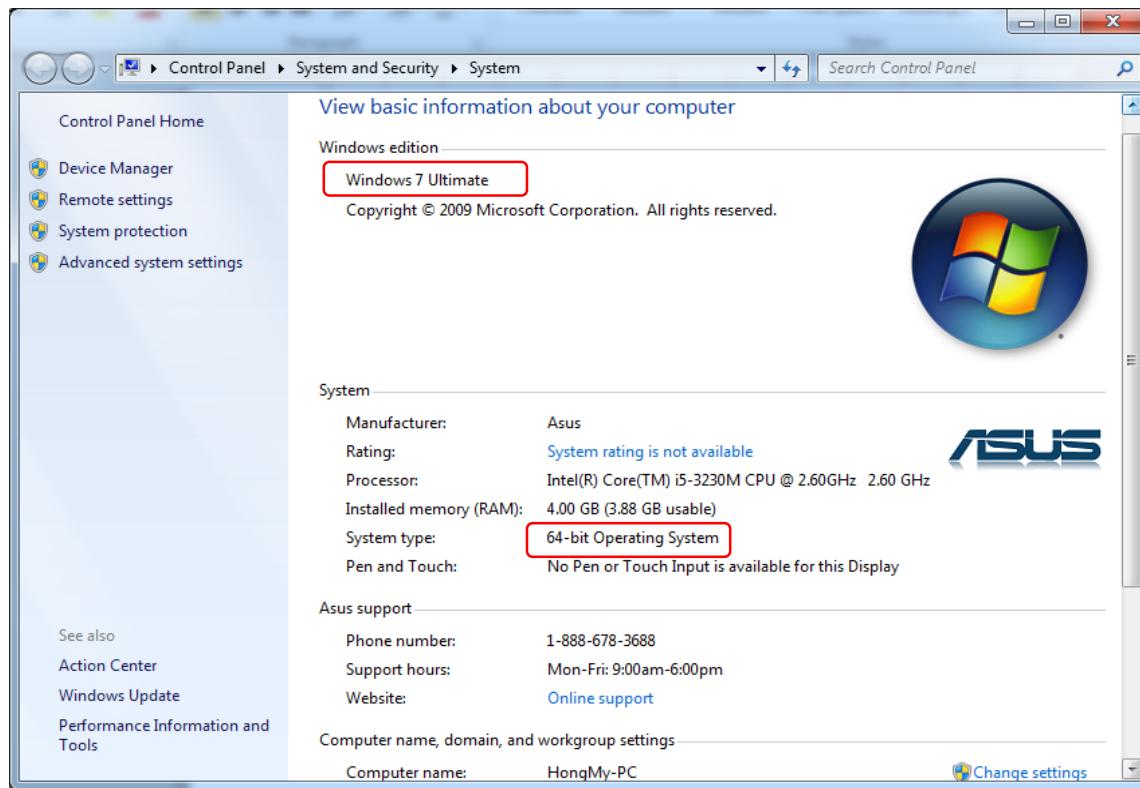
Hình 1.9. Quá trình biên dịch và thông dịch code Java

Cài đặt JDK:

Phiên bản JDK hiện tại: Java SE Development Kit 8

Link tải JDK: <http://www.oracle.com/technetwork/java/javase/downloads/>

Xem Properties của máy tính để biết loại hệ điều hành rồi chọn bản JDK phù hợp (Hệ điều hành nào – bao nhiêu bit). Chọn bản phù hợp, tải về và cài đặt.



Hình 1.10. Xem cấu hình máy tính để chọn phiên bản JDK phù hợp

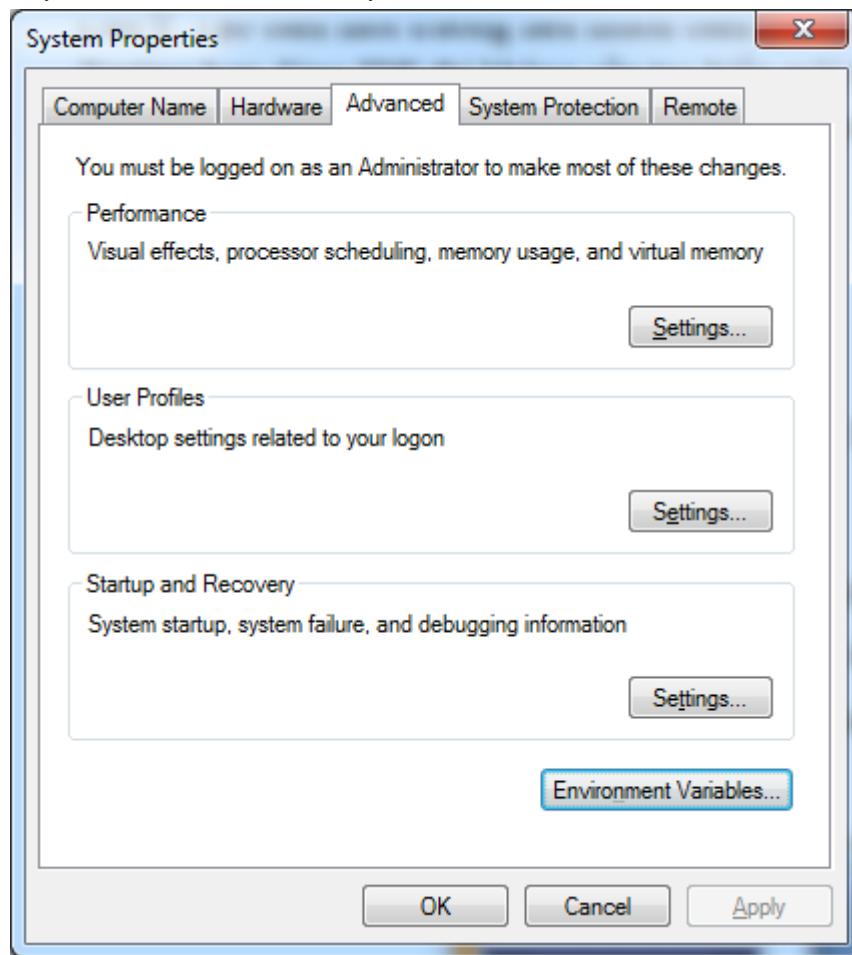
Java SE Development Kit 8u171		
You must accept the Oracle Binary Code License Agreement for Java SE to download this software.		
Thank you for accepting the Oracle Binary Code License Agreement for Java SE; you may now download this software.		
Product / File Description	File Size	Download
Linux ARM 32 Hard Float ABI	77.97 MB	jdk-8u171-linux-arm32-vfp-hflt.tar.gz
Linux ARM 64 Hard Float ABI	74.89 MB	jdk-8u171-linux-arm64-vfp-hflt.tar.gz
Linux x86	170.05 MB	jdk-8u171-linux-i586.rpm
Linux x86	184.88 MB	jdk-8u171-linux-i586.tar.gz
Linux x64	167.14 MB	jdk-8u171-linux-x64.rpm
Linux x64	182.05 MB	jdk-8u171-linux-x64.tar.gz
Mac OS X x64	247.84 MB	jdk-8u171-macosx-x64.dmg
Solaris SPARC 64-bit (SVR4 package)	139.83 MB	jdk-8u171-solaris-sparcv9.tar.Z
Solaris SPARC 64-bit	99.19 MB	jdk-8u171-solaris-sparcv9.tar.gz
Solaris x64 (SVR4 package)	140.6 MB	jdk-8u171-solaris-x64.tar.Z
Solaris x64	97.05 MB	jdk-8u171-solaris-x64.tar.gz
Windows x86	199.1 MB	jdk-8u171-windows-i586.exe
Windows x64	207.27 MB	jdk-8u171-windows-x64.exe

Hình 1.11. Tải phiên bản JDK phù hợp với hệ điều hành

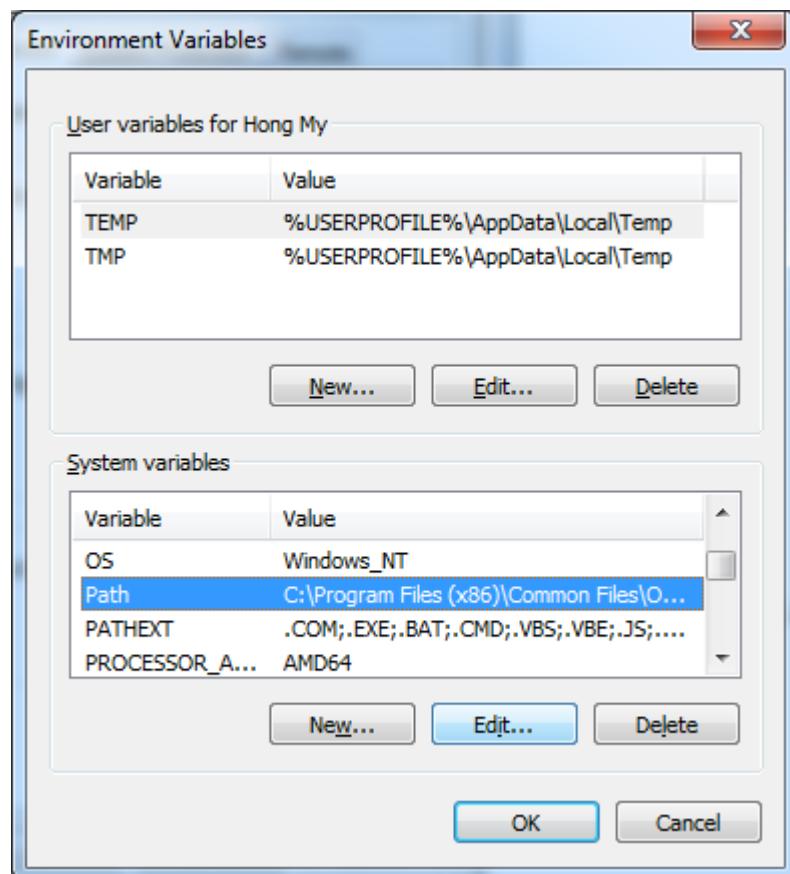
Lưu ý: Tạo biến môi trường nếu muốn biên dịch và chạy chương trình Java bằng console. Trường hợp dùng IDE thì không cần tạo biến môi trường.

Cách tạo biến môi trường (Environment variable) để chạy chương trình Java bằng command-line:

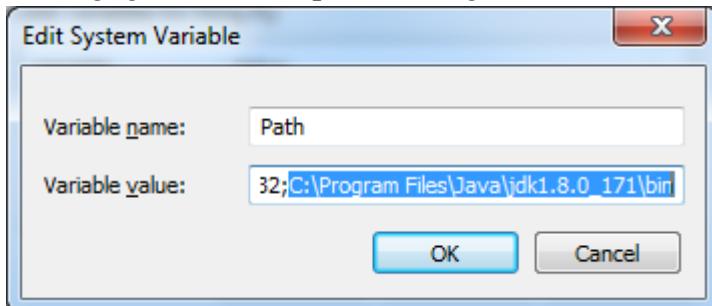
Computer -> right click -> Properties -> Advanced system settings: Hộp thoại System Properties -> Chọn thẻ Advanced -> Chọn Environment variables...



Tìm đến mục Path, trong System variables -> Chọn Edit



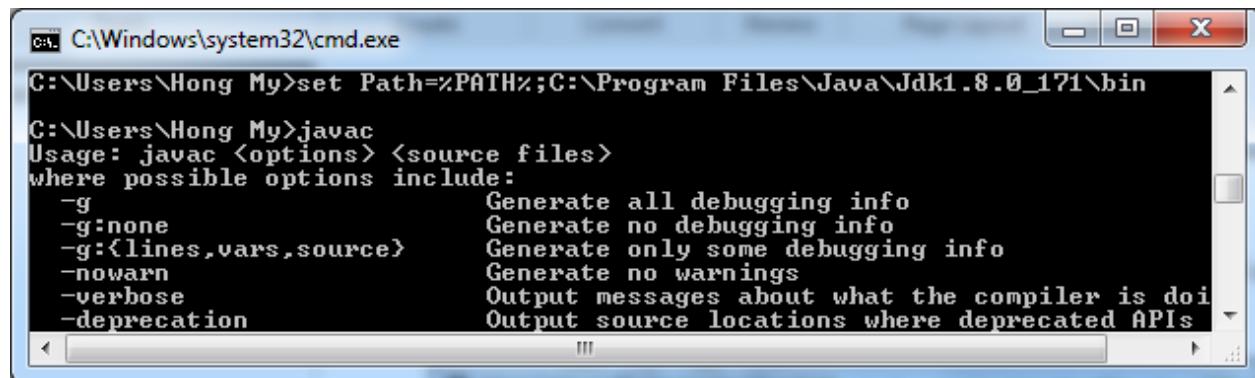
Đưa con trỏ vào cuối dòng, gõ dấu ;. Rồi paste đường dẫn tới thư mục bin đã cài đặt java.



Nhấn nút OK liên tiếp để đóng các cửa sổ đã mở ra.

Hoặc mở command-line: gõ set Path=%PATH%;C:\Program Files\Java\jdk1.8.0_171\bin

Mở command-line, gõ vào javac, nhấn enter. Nếu như cửa sổ chạy ra có dạng như hình là OK.

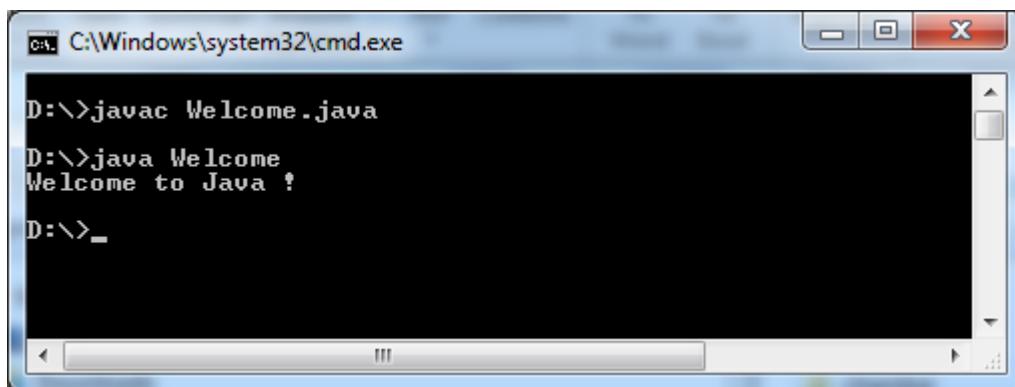


Trong ô đĩa D: tạo file Welcome.java có nội dung như sau:

```
1 public class Welcome {  
2     public static void main(String[] args) {  
3         // In ra màn hình dòng chữ "Welcome to Java !"  
4         System.out.println("Welcome to Java !");  
5     }  
6 }
```

Mở command-line tại ô đĩa D: dùng lệnh ‘javac Welcome.java’ để dịch -> chương trình sẽ dịch và tạo ra file D:\Welcome.class

Dùng lệnh ‘java Welcome’ để chạy chương trình. -> in ra màn hình câu chào:

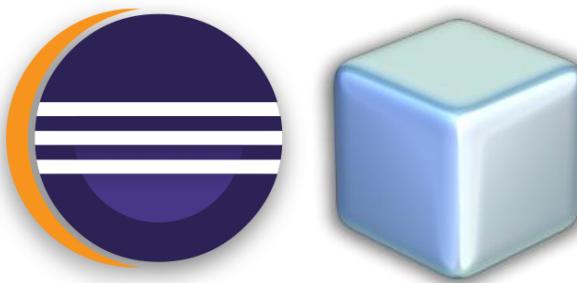


1.3. IDE để lập trình Java

Môi trường phát triển tích hợp - IDE (Integrated Development Environment) hỗ trợ phát triển và triển khai phần mềm dễ dàng hơn.

IDE bao gồm nhiều công cụ khác nhau như chương trình viết mã lệnh (code editor), chương trình sửa lỗi (debugger), chương trình mô phỏng ứng dụng khi chạy thực tế (simulator)...

Một số IDE hỗ trợ lập trình Java: Eclipse, Netbean, JCreator, Borland Jbuilder...



Hình 1.12. IDE Elipse và IDE Netbeans

Eclipse:

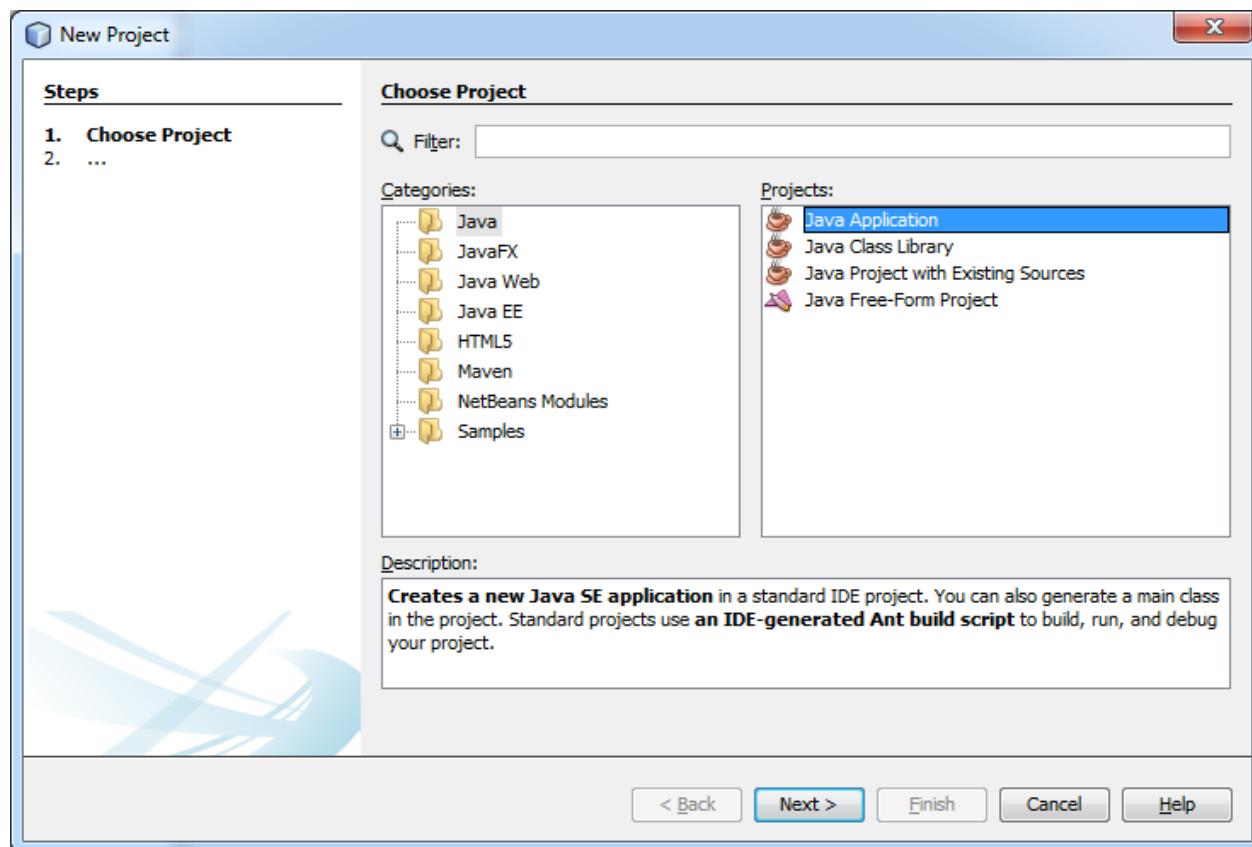
- + Link tải: <https://www.eclipse.org/downloads/>
- + Một số editing template thường dùng: Window -> Reference -> Java -> Editor -> Templates
Vd: gõ syso rồi nhấn Ctrl + Space: System.out.println();
- + Các phím tắt hay dùng: Help -> Key Assitst
- + Tiếng Việt (Unicode): Click phải chuột lên Project -> Properties -> Resources -> Text file encoding: UTF-8
Hoặc chỉnh cả workspace: Window -> References -> General -> Workspace -> Text file encoding: UTF-8 -> Apply

Netbean:

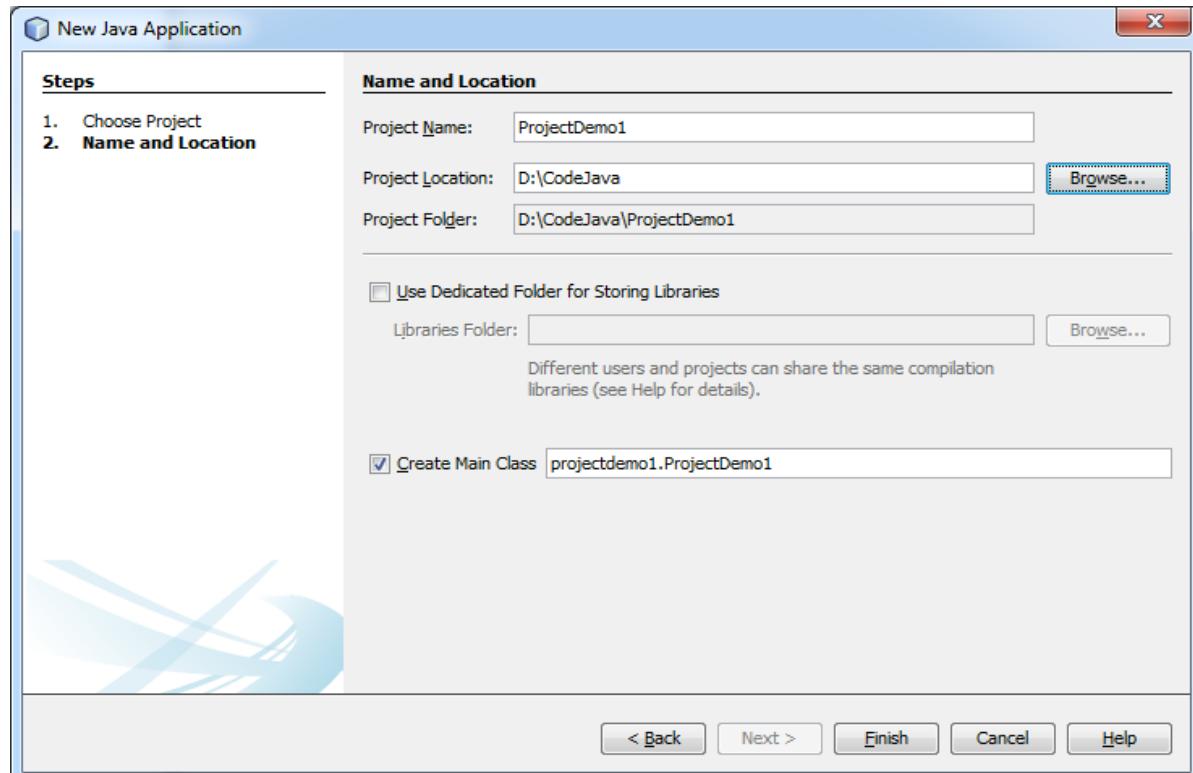
- + Link tải: <https://netbeans.org/downloads/>
- + Một số editing template thường dùng: gõ các ký tự đầu rồi nhấn Tab
Vd: gõ psvm + Tab: public static void main(String[] args){ }
- + Các phím tắt hay dùng: Tools -> Option -> Keymap
- Để thống nhất trong toàn bộ giáo trình sẽ sử dụng Netbean để viết chương trình.

Sử dụng Netbean:

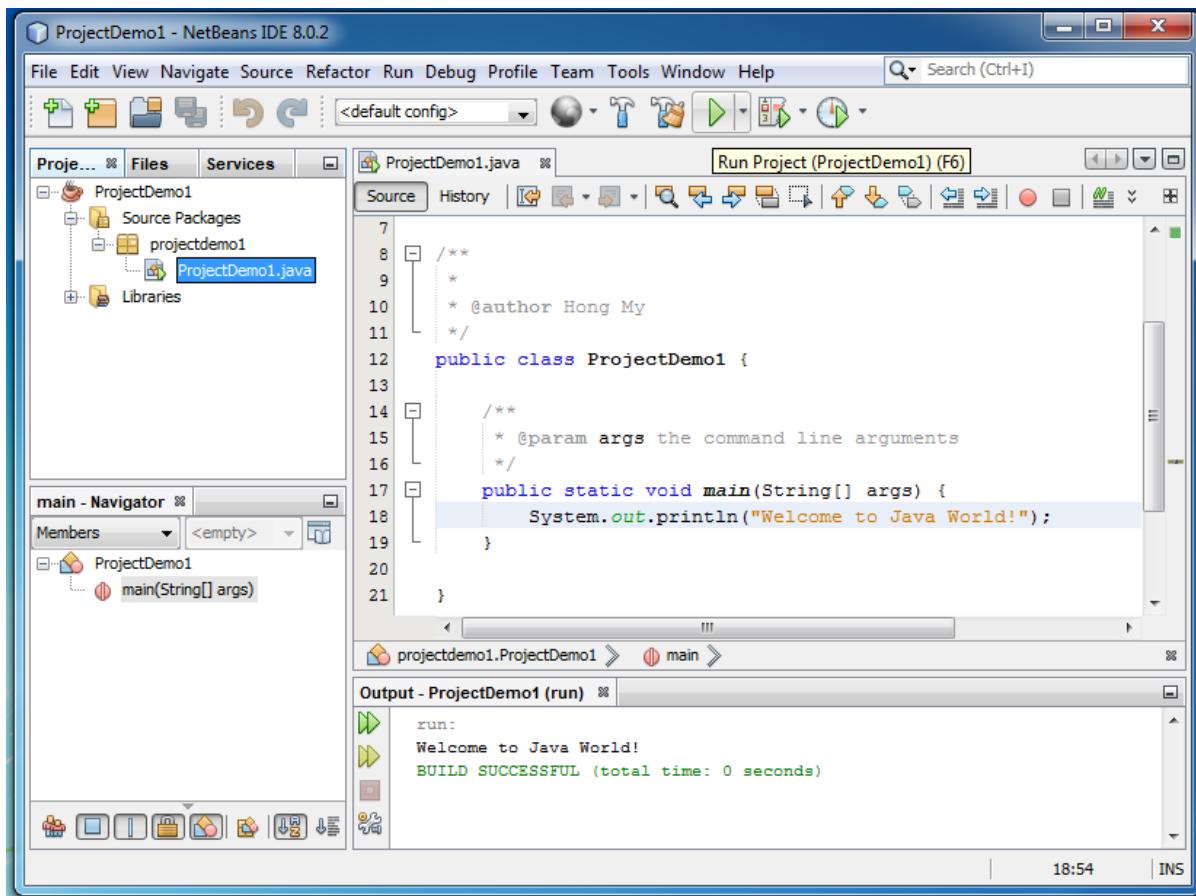
Mở Netbean -> File -> New Project -> Chọn Category Java -> Java Application



Điền tên project và chọn nơi lưu ứng dụng: Lớp Main được tạo tự động có tên giống với tên Project, có thê đổi tên lớp này.



Soạn thảo code và thực thi ứng dụng (nhấn nút Run hoặc ‘Shift + F6’)



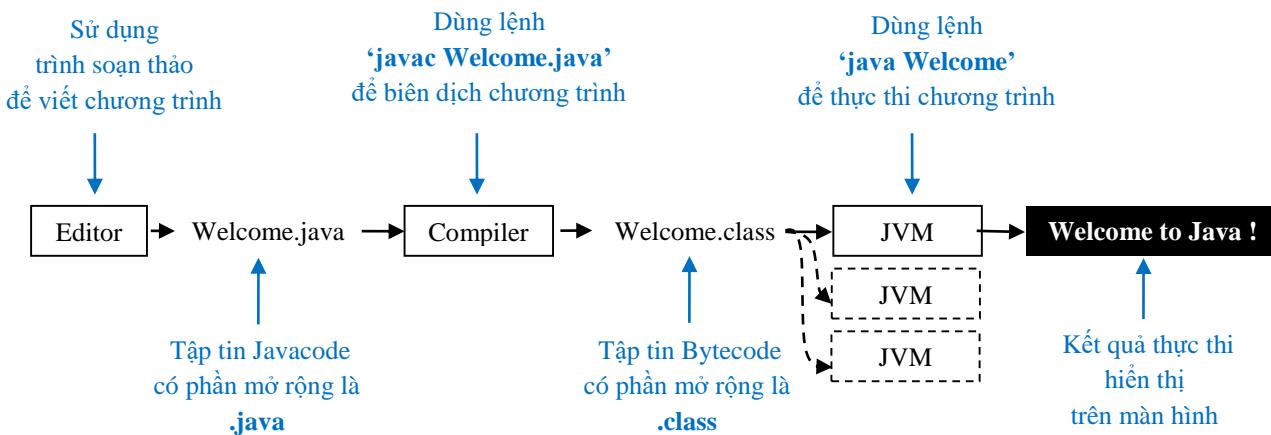
1.4. Cấu trúc chương trình Java đơn giản

```

1 public class Welcome {
2     public static void main(String[] args) {
3         // In ra màn hình dòng chữ "Welcome to Java !"
4         System.out.println("Welcome to Java !");
5     }
6 }
  
```

Welcome to Java !

Cơ chế biên dịch và thực thi



Hình 1.13. Cơ chế biên dịch và thực thi code Java

1.5. Các quy tắc cơ bản của ngôn ngữ Java

1.5.1. Cú pháp comment trong Java

Comment code trong Java là những đoạn code mà compiler sẽ bỏ qua lúc biên dịch. Các đoạn comment này dùng để giải thích về ý nghĩa, công dụng của các biến, phương thức, lớp ... để cho chương trình dễ hiểu và dễ bảo trì hơn. Comment cũng được dùng khi muốn bỏ qua biên dịch một đoạn code nào đó.

Trong Java, có 3 cách comment:

```
1 /*  
2  * Comment nhiều dòng  
3  * Dòng 1  
4  * Dòng 2  
5  * ...  
6  * Dòng n  
7  */  
8  
9 // Comment 1 dòng hoặc cuối dòng  
10  
11 /**  
12  * Comment đặc biệt dùng để tạo Java code documentation ở định dạng HTML  
13  * (Java Document)  
14 */
```

1.5.2. Quy tắc đặt tên trong Java

Quy tắc chung khi đặt tên biến, tên hằng, tên package, tên class, tên interface:

- Sử dụng ký tự alphabet, số, \$, dấu gạch dưới (_). Không được chứa khoảng trắng, các ký tự toán học. Tên không bắt đầu bằng số; phải được bắt đầu bằng chữ cái hoặc dấu gạch dưới _
- Không được trùng với các từ khóa của Java
- Không nên đặt tên dài quá 20 ký tự, và cũng không quá ngắn, trừ khi đặt tên tạm (ví dụ như: a, x, i, j, ...)
- Đặt tên có ý nghĩa và thể hiện được mục đích của lớp, biến, phương thức. Nên đặt tên lớp, tên thuộc tính, tên phương thức bằng tiếng Anh.
- Tên được đặt theo **quy tắc con lắc đà** (*Camel Case*)

Tên Project viết theo quy tắc con lắc đà. Ví dụ: QuanLySinhVien, QuanLyBanHang

Tên package viết thường toàn bộ. Ví dụ: quanlysinhvien.model, com.tencongty.tendoan

Tên lớp viết hoa ký tự đầu; tên biến và tên phương thức viết thường ký tự đầu tiên.

Tên hằng số được viết in hoa. Ví dụ: PI = 3.14; TY_GIA = 23000; HOUR_OF_DAY = 24; COMPANY = “TDC”

Lưu ý: Trong Java có phân biệt chữ hoa chữ thường.

Ví dụ:

Tên lớp	Tên thuộc tính	Tên phương thức
SinhVien	hoTen	tinhDiemTrungBinh()
HinhChuNhat	diemToan	inThongTin()
XeMay	diemVan	tinhDienTich()

Tùy khóa Java:

abstract	assert	boolean	break	byte
case	catch	char	class	const
continue	default	do	double	else
enum	extends	final	Finally	float
for	goto	if	implements	import

instanceof	int	interface	long	native
new	package	private	protected	public
return	short	static	strictfp	super
switch	synchronized	this	throw	throws
transient	try	void	volatile	while

Một số ký tự Escape character phổ biến trong Java:

- \t Chèn vào một tab vào chuỗi
- \b Xóa lùi 1 ký tự (backspace)
- \n Chèn một dòng mới
- \r Thay thế các ký tự trước \r bằng số các ký tự phía sau \r
- ' Chèn dấu nháy đơn
- \" Chèn dấu nháy kép
- \\" Chèn dấu \

1.6. Biên dịch và thực thi chương trình Java

Trình biên dịch chuyển mã nguồn thành tập các lệnh (bytecode) không phụ thuộc vào phần cứng cụ thể. Khi chạy chương trình, trình thông dịch trên mỗi máy chuyển bytecode thành chương trình thực thi nhờ có máy ảo Java. Máy ảo Java tạo ra một môi trường để thực thi các lệnh, nạp các file .class, quản lý bộ nhớ, gom rác.

Các lệnh biên dịch, thông dịch:

- Trình biên dịch, 'javac'
 - javac [options] SourceCodeName.java
- Trình thông dịch, 'java'
 - java [options] ClassName
- Công cụ sinh tài liệu, 'javadoc'
 - javadoc [options] SourceCodeName.java

Coding convention – Quy ước viết code

Để thống nhất cách viết code và dễ sửa lỗi cũng như bảo trì nâng cấp hệ thống, lập trình viên cần tuân thủ quy ước viết code. Hai bộ Quy ước viết mã lập trình Java: **Google Java Style Guide, Sun Java Style Guide**

Các tài liệu về lớp và phương thức trong thư viện Java được trình bày trong API document:
<http://docs.oracle.com/javase/7/docs/api/index.html>

Ví dụ:

```
1 package tdc.demo;
2
3 import java.util.Scanner;
4
5 public class ToanHoc {
6
7     public static int UCLN(int a, int b) { ←
8         while (a != b) {
9             if (a > b) {
10                 a = a - b;
11             } else {
12                 b = b - a;
13             }
14         }
15         return (a);
16     }
17
18     public static void main(String[] args) {
19         Scanner input = new Scanner(System.in);
20         System.out.print("Nhập a: ");
21         int a = input.nextInt();
22         System.out.print("Nhập b: ");
23         int b = input.nextInt();
24         System.out.println("UCLN của " + a + " và " + b + " là: " + UCLN(a, b));
25         System.out.println("BCNN của " + a + " và " + b + " là: "
26                         + ((a * b) / UCLN(a, b)));
27     }
28 }
```

Case Study Quản Lý Sinh Viên

Giáo trình dùng Case Study Quản Lý Sinh Viên xuyên suốt các chương để minh họa việc áp dụng kiến thức vào chương trình ứng dụng thực tiễn.

Yêu cầu: Xây dựng chương trình ứng dụng quản lý sinh viên cho một trường học, để quản lý các thông tin: sinh viên, môn học, khoa, lớp cố định, lớp học phần, đăng ký môn học. Với các chức năng cập nhật dữ liệu: Thêm, Xóa, Sửa, xem danh sách các thông tin, tương tác dữ liệu với file, với Cơ sở dữ liệu.

Bài tập

Bài 1. Viết chương trình hiển thị câu “A journey of a thousand miles begins with a single step”

```
A journey of thousand miles begins with a single step.
```

Bài 2. Viết chương trình in tên của mình, có dạng như sau

TTTTTTT	DDDD	CCCC
T	D	D C
T	D	D C
T	D	D C
T	DDDD	CCCC

Bài 3. Viết chương trình cho phép nhập vào tên và xuất ra lời chào

```
Enter your name: Hong My  
Welcome Hong My to Java World!
```

Bài 4. Viết chương trình cho phép nhập hai cạnh của hình chữ nhật, tính diện tích và chu vi của hình chữ nhật.

```
Enter width: 25  
Enter length: 4  
Area of Rectangle(25, 4) is 100  
Perimeter of Rectangle(25, 4) is 58
```

Bài 5. Viết chương trình nhập vào một số giây, xử lý in ra màn hình dãy số dưới dạng ‘giờ:phút:giây’

```
Enter amount of seconds: 6543  
Result: 01:49:03
```

Bài 6. Viết chương trình chuyển đổi giữa độ C (Celsius) và độ F (Fahrenheit). Biết rằng:

$$C = \frac{5.0}{9.0} * (F - 32)$$

$$F = \frac{9.0}{5.0} * C + 32$$

```
Change temperature F <-> C  
1. C2F  
2. F2C  
Choose: 1  
Enter Celsius: 40  
Result: 40 Celsius = 104 Fahrenheit
```

Bài 7. Tính chỉ số BMI và in ra kết quả (Body Mass Index)
$$\text{BMI} = (\text{weightInPounds} * 703) / (\text{heightInInches} * \text{heightInInches})$$

$$\text{BMI} = \text{weightInKilograms} / (\text{heightInMeters} * \text{heightInMeters})$$

BMI VALUES:

Underweight: < 18.5

Normal: [18.5 - 24.9]

Overweight: [25 - 29.9]

Obese: >=30

Bài 8. Bổ sung code: tách phương thức nhap1() để nhập số a và số b

Thêm code kiểm tra dữ liệu nhập không phải là số nguyên nhap2(): có kiểm tra và báo lỗi và cho phép nhập lại cho tới khi nhập đúng

```

public static int nhap1() {
    Scanner input = new Scanner(System.in);

    int n = 0;
    n = input.nextInt();
    return n;
}

public static int nhap2() {
    Scanner input = new Scanner(System.in);
    boolean check = false;
    int n = 0;
    while (!check) {
        try {
            n = input.nextInt();
            check = true;
        } catch (Exception e) {
            System.out.println("Bạn phải nhập số nguyên! Hãy nhập lại...");
            input.nextLine();
        }
    }
}

```

```
    }
    return n;
}

public static void main(String[] args) {
    Scanner input = new Scanner(System.in);
    System.out.print("Nhập a: ");
    int a = input.nextInt(); nhap1();
    System.out.print("Nhập b: ");
    int b = input.nextInt(); nhap1();
    System.out.println("UCLN của " + a + " và " + b + " là: " + UCLN(a, b));
    System.out.println("BCNN cua " + a + " và " + b + " là: "
        + ((a * b) / UCLN(a, b)));
}
}
```

CHƯƠNG 2. NỀN TẢNG JAVA CƠ BẢN

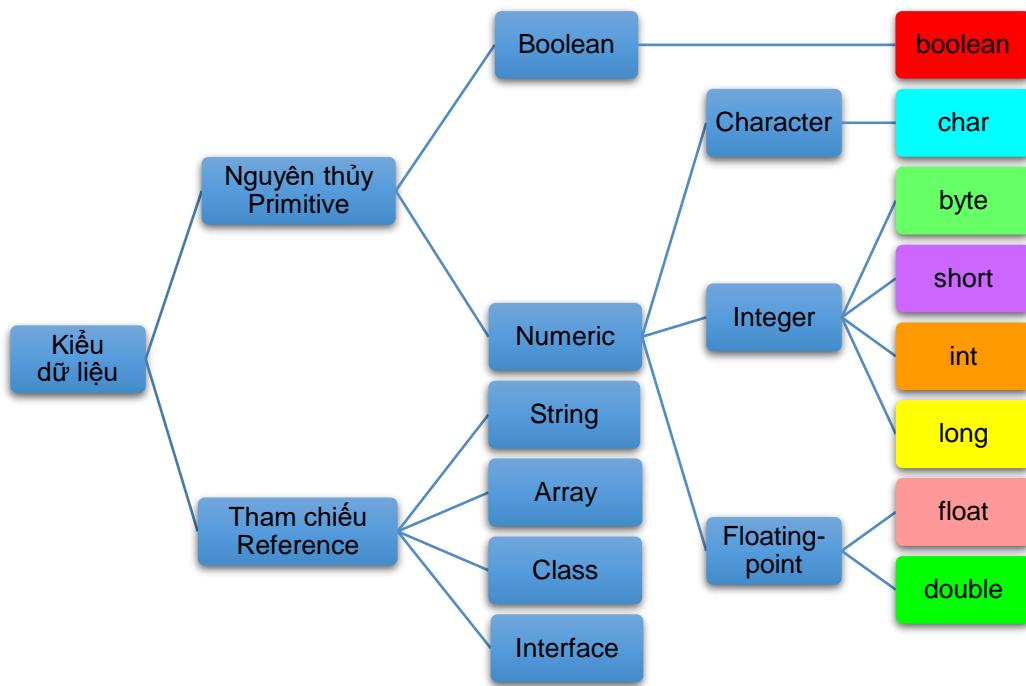
Học xong chương này, người học có thể:

- + Trình bày các cú pháp cơ bản trong Java như các kiểu dữ liệu, biến và hằng, các toán tử, chuỗi, mảng, cấu trúc điều khiển, cấu trúc lặp;
- + Trình bày các đặc điểm của lập trình hướng đối tượng;
- + Tạo lớp và đối tượng trong Java;
- + Phân biệt được ghi đè phương thức (override) và nạp chồng phương thức (overload);
- + Sử dụng Unit Test để test chương trình.

2.1. Cú pháp Java cơ bản

2.1.1. Kiểu dữ liệu và biến

Kiểu dữ liệu trong Java:



Hình 2.1. Các kiểu dữ liệu trong Java

Kiểu dữ liệu	Giá trị mặc định	Kích thước	Miền giá trị
boolean	false	1 bit	true, false (1, 0)
char	'\u0000'	2 byte	'\u0000' (0) đến '\uffff' (65,535)
byte	0	1 byte	-128 đến 127 (-2^7 đến 2^7 - 1)
short	0	2 byte	- 32,768 đến 32,767 (-2^15 đến 2^15 - 1)
int	0	4 byte	-2,147,483,648 đến 2,147,483,647 (-2^31 đến 2^31 - 1)
long	0L	8 byte	-9,223,372,036,854,775,808 đến 9,223,372,036,854,775,807 (-2^63 đến 2^63 - 1)
float	0.0f	4 byte	-+ 3.40282347E+38F (3.40282347 x 10^38)
double	0.0	8 byte	-+ 3.179769313486231570E+308F

Wrapper class (lớp bao bọc)

Các lớp Wrapper sẽ giúp chúng ta chuyển đổi qua lại giữa một kiểu dữ liệu nguyên thủy sang kiểu dữ liệu đối tượng và ngược lại.

Trong Java, ứng với mỗi kiểu dữ liệu cơ sở sẽ có một kiểu dữ liệu Wrapper class. Nó ‘bọc’ kiểu dữ liệu nguyên thủy vào trong một lớp đối tượng. Vì vậy, Wrapper class là kiểu dữ liệu vừa có thể lưu trữ giá trị đơn và vừa có thêm các phương thức khác.

```

1 int i = 79; // biến i có kiểu dữ liệu int nguyên thủy
2 Integer n = Integer.valueOf(i); // biến n có kiểu dữ liệu đối tượng Integer
3 Integer m = i; // biến m có kiểu dữ liệu đối tượng Integer
4 Integer p = 300; // biến p có kiểu dữ liệu đối tượng Integer
5 int j = p.intValue(); // j là biến có kiểu dữ liệu int nguyên thủy
6 int k = p; // k là biến có kiểu dữ liệu int nguyên thủy

```

Danh sách các Wrapper class ứng với mỗi kiểu dữ liệu nguyên thủy:

Primitive data	boolean	char	byte	short	int	long	float	double
Wrapper class	Boolean	Character	Byte	Short	Integer	Long	Float	Double

Ví dụ Chuyển chuỗi sang kiểu dữ liệu nguyên thủy:

Lưu ý dùng try ... catch để kiểm soát lỗi dữ liệu sai kiểu:

```

1 try {
2     String s1 = "1";
3     int num1 = Integer.parseInt(s1);
4     System.out.println(num1);
5
6     int x = Integer.parseInt("10");
7     float f = Float.parseFloat("4.5");
8     boolean b = Boolean.parseBoolean("true");
9 } catch (Exception ex) {
10     System.out.println("Dữ liệu không hợp lệ!");
11 }

```

Ép kiểu (Type Casting) trong Java

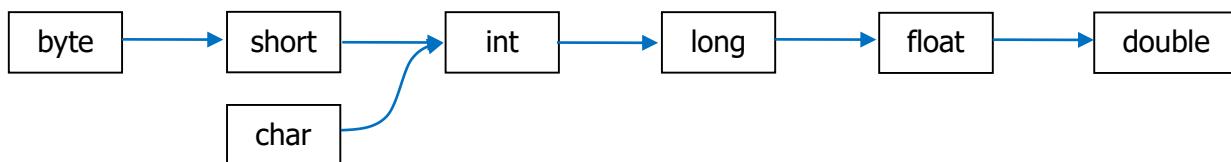
Ép kiểu trong Java là quá trình chuyển đổi kiểu dữ liệu này sang kiểu dữ liệu khác.

Khi thực hiện phép chia số nguyên 2 cho 4, nếu không sử dụng đến ép kiểu thì kết quả của phép toán này sẽ trả về 0, như vậy thì yêu cầu của bài toán đã không còn đúng nữa. Cần trả về 0.5

Có 2 loại ép kiểu trong Java:

- + Implicit Casting (Ép kiểu rộng/ Ép kiểu không tường minh): Chuyển từ kiểu có vùng lưu trữ nhỏ lên kiểu có vùng lưu trữ lớn hoặc bằng. Kiểu chuyển này không làm mất dữ liệu.
- + Explicit Casting (Ép kiểu hẹp/ Ép kiểu tường minh): Chuyển từ kiểu có vùng lưu trữ lớn về kiểu có vùng lưu trữ nhỏ. Kiểu chuyển này có thể làm mất dữ liệu.

Đối với kiểu dữ liệu nguyên thủy: Ép kiểu rộng theo hướng mũi tên; Ép kiểu hẹp là theo hướng ngược lại.



Hình 2.2. Ép kiểu đối với dữ liệu nguyên thủy

2.1.2. Toán tử

Loại	Toán tử	Mô tả
Toán tử số học Arithmetic operator	+	Addition (Phép cộng)
	-	Subtraction (Phép trừ)
	*	Multiplication (Phép nhân)
	/	Division (Phép chia)
	%	Modulus (Phép chia lấy số dư)
Toán tử gán Assignment operator	=	Phép gán
	+=	
	-=	
	*=	Cú pháp viết tắt toán tử số học và phép gán
	/=	
	%=	
Toán tử so sánh Relational operators	++	Increment (Tăng 1)
	--	Decrement (Giảm 1)
	==	So sánh bằng. Nếu bằng nhau thì kết quả trả về true, ngược lại trả về false.
	!=	So sánh không bằng
	< , >	So sánh nhỏ hơn, lớn hơn
Toán tử logic Logical operators	<=, >=	So sánh nhỏ hơn hoặc bằng, lớn hơn hoặc bằng
	&&	Phép toán luận lý AND trên 2 giá trị. Kết quả trả về true khi cả hai đều đúng.
		Phép toán luận lý OR trên 2 giá trị. Kết quả trả về false khi cả hai đều sai.
	!	Phép phủ định (NOT)
Toán tử với Bit Bitwise operators	~	NOT (Phủ định trên bit)
	&,	AND, OR (Phép và , phép hoặc trên bit)
	^	XOR (Exclusive OR : phép XOR trên bit)
	<<, >>	Shift left, shift right (Dịch chuyển sang trái 1 bit, sang phải 1 bit)

Độ ưu tiên của các toán tử :

Các toán tử một ngôi !, ++ và -- có độ ưu tiên cao nhất

Tiếp theo là các toán tử hai ngôi *, / và %

Sau đó là các toán tử +, -

Cuối cùng là các phép toán so sánh <, >, <=, >=

Ví dụ: $3 + 4 > 5 + 1$ //cho kết quả là true.

Có thể dùng các cặp ngoặc () để định rõ thứ tự ưu tiên trong biểu thức.

Ví dụ:

$2 * 3 + 4$ //cho kết quả là 10

$2 * (3 + 4)$ //cho kết quả là 14

2.1.3. Cấu trúc điều khiển

2.1.3.1. Cấu trúc điều khiển if-else

Cú pháp:

```
if (biểu thức điều kiện){  
    <các lệnh khi biểu thức điều kiện ĐÚNG>  
}  
[else{  
    <các lệnh khi biểu thức điều kiện SAI>  
}] //có thể có hoặc không
```

```
1     double score = 5.5;  
2     String result = "";  
3     if (score < 4.0)  
4         result = "Failed";  
5     else  
6         result = "Passed";
```

Cú pháp viết tắt sử dụng dấu ? và dấu :

```
(biểu thức điều kiện) ? <lệnh khi BTĐK ĐÚNG> : <lệnh khi BTĐK SAI>
```

```
String result = (score < 4.0) ? "Failed" : "Passed";
```

Lệnh if lồng nhau:

```

1  double score = 5.5;
2  char grade = 'F';
3  if (score >= 8.5)
4      grade = 'A';
5  else if (score >= 7.0)
6      grade = 'B';
7  else if (score >= 5.5)
8      grade = 'C';
9  else if (score >= 4.0)
10     grade = 'D';
11 else
12     grade = 'F';

```

2.1.3.2. Cấu trúc rẽ nhánh switch-case

Cấu trúc rẽ nhánh có nhiều lựa chọn, có thể sử dụng nhiều lệnh if-else lồng nhau hoặc dùng lệnh switch-case nếu các lựa chọn là hằng số (hoặc là String từ JavaSE7 trở lên)

Cú pháp:

```

switch (biểu_thức) {
    case hằng_1:
        tập_lệnh_1;
        break;
    case hằng_2:
        tập_lệnh_2;
        break;
    ...
    default:
        tập_lệnh_mặc định;
}

```

```

1  char grade = 'A';
2  switch (grade) {
3      case 'A':
4          System.out.println("Excellent!");
5          break;
6      case 'B':
7          System.out.println("Great!");
8      case 'C':
9      case 'D':
10         System.out.println("Well done!");

```

```
11     break;
12 case 'F':
13     System.out.println("Sorry, you failed.");
14     break;
15 default:
16     System.out.println("Error! Invalid grade.");
17 }
```

2.1.4. Vòng lặp

2.1.4.1. Vòng lặp while

```
int count = 1;
while (count <= 10) {
    System.out.print(count + ", ");
    count++;
}
```

2.1.4.2. Vòng lặp do while

```
int count = 1;
do {
    System.out.print(count + ", ");
    count++;
} while (count <= 10);
```

2.1.4.3. Vòng lặp for

```
for (int count = 1; count <= 10; count++) {
    System.out.print(count + ", ");
}
```

Lưu ý:

Lệnh break được dùng trong lệnh switch - case hoặc vòng lặp để ngắt khi cần thiết, chương trình sẽ thực thi các lệnh nằm tiếp sau cấu trúc đó. Lệnh break thường được dùng để ngắt (không cần kiểm tra các case còn lại) trong cấu trúc switch-case hoặc để kết thúc sớm vòng lặp (không cần đợi đến lượt kiểm tra điều kiện lặp).

Lệnh continue trong một vòng lặp có tác dụng bỏ qua lần lặp hiện tại của vòng lặp đó.

Ví dụ

```
1 for (int i = 1; i < 10; i++) {
2     if (i == 5) {
```

```

3         break;
4     }
5     System.out.println(i);
6 }
7 for (int i = 1; i < 10; i++) {
8     if (i == 5) {
9         continue;
10    }
11    System.out.println(i);
12 }

```

Ví dụ: Tính giai thừa của 5 là 120: $5 \times 4 \times 3 \times 2 \times 1$

Sử dụng vòng lặp	Sử dụng đệ quy
<pre> private static long factorial(int n){ long f = 1; for (int i = 1; i <= n; i++) f = f * i; return f; } </pre>	<pre> private static long factorial(int n){ if (n == 1) return 1; else return n * factorial(n - 1); } </pre>

Ví dụ: Dùng đệ quy để liệt kê cây thư mục

```

1 private static void listDirectories(File dir, String indent) {
2     File[] dirs = dir.listFiles();
3     for (File f : dirs) {
4         if (f.isDirectory()) {
5             System.out.println(indent + f.getName());
6             listDirectories(f, indent + " ");
7         }
8     }
9 }

```

2.1.5. Chuỗi (String) trong Java

Một số phương thức thông dụng của chuỗi String

Phương thức	Mô tả
char charAt(int index)	Trả về một ký tự tại vị trí có chỉ số được chỉ định.

int compareTo(String anotherString)	So sánh hai chuỗi theo từ điển. (Phân biệt chữ hoa chữ thường)
String concat(String str)	Nối chuỗi được chỉ định đến cuối của chuỗi này.
boolean equals(Object anObject)	So sánh với một đối tượng
int indexOf(int ch)	Lấy chỉ số trong chuỗi mà xuất hiện ký tự này đầu tiên
int indexOf(String str)	Lấy chỉ số trong chuỗi mà xuất hiện chuỗi con này đầu tiên
int lastIndexOf(int ch)	Lấy chỉ số trong chuỗi mà xuất hiện ký tự này cuối cùng
int length()	Trả về độ dài chuỗi.
String replace(char oldChar, char newChar)	Trả về một chuỗi mới từ thay thế tất cả các lần xuất hiện của ký tự oldChar trong chuỗi này với ký tự newChar.
String[] split(String regex)	Tách chuỗi này thành các chuỗi con, tại các chỗ khớp với biểu thức chính quy cho trước.
String substring(int beginIndex)	Trả về một chuỗi ký tự mới là một dãy con của dãy này. Từ chỉ số cho trước tới cuối
String substring(int beginIndex, int endIndex)	Trả về một chuỗi ký tự mới là một dãy con của dãy này. Từ chỉ số bắt đầu cho tới chỉ số cuối.
char[] toCharArray()	Chuyển chuỗi này thành mảng ký tự.
String toLowerCase()	Chuyển tất cả các ký tự của chuỗi này sang chữ thường, sử dụng miền địa phương mặc định (default locale)
String toUpperCase()	Chuyển tất cả các ký tự của chuỗi này sang chữ hoa, sử dụng miền địa phương mặc định (default locale)
String trim()	Trả về một String mới, sau khi loại bỏ các ký tự trắng (whitespace) bên trái và bên phải.

Ví dụ:

```

1  String s1 = "ABC";
2  String txt = s1.concat("DEF12345");
3  txt.length();
4  txt.toLowerCase();
5  String s2 = txt.substring(6); // 12345
6  String s3 = s2 + 90; // "124590"

```

2.1.6. Mảng một chiều, Mảng hai chiều

Mảng là tập hợp nhiều phần tử có cùng kiểu dữ liệu và các phần tử trong mảng được truy xuất thông qua chỉ số của nó trong mảng. Chỉ số mảng bắt đầu từ 0.

Mảng một chiều

```

1 int number[] = { 1, 2, 3 };
2 String[] cars = { "BMW", "Ford", "Mazda" };
3 System.out.println(cars[0]);
4 cars[0] = "Chevrolet";
5 cars[2] = "Toyota";
6 for (String i : cars) {
7     System.out.println(i);
8 }
9 }
```

Mảng hai chiều

```

1 int[][] myNumbers = { { 1, 2, 3, 4 }, { 5, 6, 7 } };
2 int x = myNumbers[1][1];
3 System.out.println(x); // Outputs 6
4 for (int i = 0; i < myNumbers.length; ++i) {
5     for (int j = 0; j < myNumbers[i].length; ++j) {
6         System.out.println(myNumbers[i][j]);
7     }
8 }
```

2.1.7. Một số thư viện thường dùng

Thư viện xử lý số ngẫu nhiên - Lớp Random

```
Random rand = new Random();
```

```
int n = rand.nextInt(50); // [0 - 49]
```

Thư viện toán học trong Java – lớp Math.

Một số hàm thông dụng của lớp Math: min(), max(), sqrt(), abs(), pow(), round(), floor(), ceil(), random()

Math.random() : trả về một số tự trong [0,1) bao gồm số 0, không bao gồm số 1

JOptionPane: showMessageDialog(), showInputDialog(), showConfirmDialog()

Thư viện xử lý ngày tháng trong Java - Lớp LocalDate, lớp Calendar, SimpleDateFormat, DateFormater:

```
1 LocalDateTime currentDateTime = LocalDateTime.now();
2 System.out.println("Before formatting: " + currentDateTime);
3
4 DateTimeFormatter myFormat = DateTimeFormatter
5     .ofPattern("dd-MM-yyyy HH:mm:ss");
6
7 String formattedDate = currentDateTime.format(myFormat);
8 System.out.println("After formatting: " + formattedDate);
9
10 Calendar cal = Calendar.getInstance();
11 // Calendar cal = new GregorianCalendar(2019,1,28,13,24,56);
12 Date date = cal.getTime();
13 System.out.println("Thời gian hiện tại của hệ thống là: " + date);
14 System.out.println("Ngày: " + cal.get(Calendar.DAY_OF_MONTH));
15 System.out.println("Tháng: " + cal.get(Calendar.MONTH) + 1); // 0-11
16 System.out.println("Năm: " + cal.get(Calendar.YEAR));
17
18 System.out.println("Giờ hiện tại là " + cal.get(Calendar.HOUR)); // 4
19 System.out.println("Giờ hiện tại là " + cal.get(Calendar.HOUR_OF_DAY)); // 16
20 System.out.println("Phút hiện tại là " + cal.get(Calendar.MINUTE));
21 System.out.println("Giây hiện tại là " + cal.get(Calendar.SECOND));
22
23 cal.set(Calendar.DAY_OF_MONTH, 9); // thay đổi thành ngày 9
24 cal.set(Calendar.MONTH, Calendar.DECEMBER); // tháng 12
25 cal.set(Calendar.YEAR, 2018); // năm 2018
26 System.out.println("Thời gian hiện tại sau khi thay đổi: " + cal.getTime());
27
28 cal.set(2016, Calendar.APRIL, 30, 20, 23, 8); // có thể thay đổi tất cả 1 lần
29
30 System.out.println("Thời gian hiện tại sau khi thay đổi 2: "
31     + cal.getTime());
```

2.1.8. Định dạng Formating

Định dạng chuỗi in: printf và format

Bảng ký hiệu được sử dụng khi định dạng chuỗi in:

Ký hiệu	Flag	Ý nghĩa
S		Một chuỗi String
D		Một số nguyên digit
F		Một số thực float
N		Một ký tự dòng mới phù hợp với nền tảng chạy ứng dụng. Nên sử dụng %n, hơn là \n.
	05	Chỉ định chiều rộng là 5 ký tự, bao gồm cả số 0 nếu cần thiết.
	+	In ra ký tự +.
	,	Nhóm các số bằng dấu ‘,’ ví dụ 12,345,678
	–	Căn trái.
	.3	Ba số sau dấu thập phân.
	10.3	Rộng 10 ký tự, căn phải, với 3 số sau dấu thập phân.
tB		In ra tên tháng trong biến date & time, ví dụ: April.
td, te		In ra ngày trong biến date & time, 2 số của ngày trong tháng. ‘td’ sẽ in ra cả số 0 chặng hạn như ngày 07, ‘te’ thì không.
ty, tY		In ra năm trong biến date & time, ty = 2-số cuối của năm, tY = 4-số của năm.
Tl		In ra giờ trong biến date & time, theo định dạng 12-giờ.
tM		In ra phút trong biến date & time, bao gồm 2 số, cả số 0 nếu cần thiết.
Tp		In ra date & time dưới dạng am/pm (chữ thường).
Tm		In ra tháng của biến date & time, gồm 2 số, cả số 0 nếu cần thiết.
tD		In ra ngày của biến date & time như định dạng %tm%td%ty

\$s: chuỗi

%d: số nguyên

%f: số thực. Mặc định có 6 số lẻ

%.3f: số thực có 3 số lẻ

```
System.out.println(String.format("%-8s%7.2f%3d", "book", 17.5, 8));
```

Hoặc

```
System.out.printf("%-8s%7.2f%3d", "book", 17.5, 8);
```



Ví dụ:

```
1 public class TestFormat {
2     public static void main(String[] args) {
3         long n = 123456;
4         System.out.format("%d%n", n); // "123456"
5         System.out.format("%08d%n", n); // "00123456"
6         System.out.format("%+8d%n", n); // "+123456"
7         System.out.format(",%8d%n", n); // " 123,456"
8         System.out.format("%+,8d%n%n", n); // "+123,456"
9
10        double pi = Math.PI;
11
12        System.out.format("%f%n", pi); // "3.141593"
13        System.out.format("%.3f%n", pi); // "3.142"
14        System.out.format("%10.3f%n", pi); // " 3.142"
15        System.out.format("%-10.3f%n", pi); // "3.142"
16        System.out.format(Locale.FRANCE, "%-10.4f%n%n", pi); // "3,1416"
17
18        Calendar c = Calendar.getInstance();
19        System.out.format("%tB %te, %tY%n", c, c, c); // "Sep 03, 2018"
20
21        System.out.format("%tl:%tM %tp%n", c, c, c); // "7:55 am"
22
23        System.out.format("%tD%n", c); // "09/03/18"
24    }
25 }
```

2.2. Lớp và đối tượng trong Java

2.2.1. Lớp - Class

Class được xem như một khuôn mẫu của đối tượng. Mỗi Class có các thuộc tính và các phương thức cần thiết.

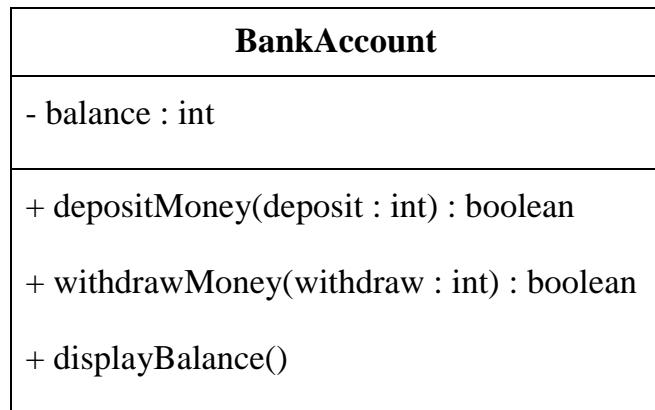
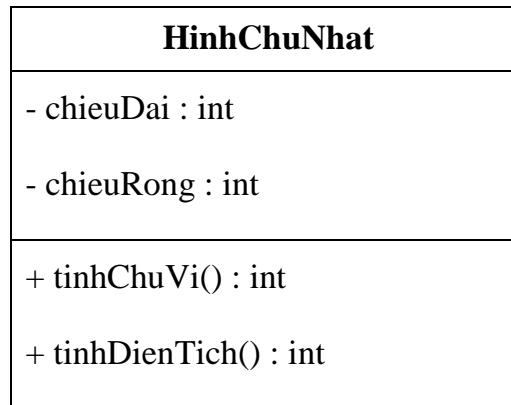
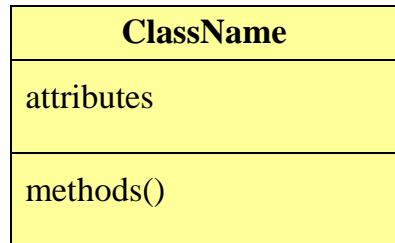
```
public class TenLop{  
    //các thuộc tính  
    //các phương thức  
}
```

Ngoài ra, lớp còn có Constructor – hàm khởi tạo. Nó là phương thức đặc biệt của lớp. Constructor không có giá trị trả về và có thể có tham số hoặc không tham số. Constructor phải có tên trùng với tên lớp. Constructor được gọi tự động khi dùng từ khóa new để khởi tạo đối tượng. Một lớp mặc định có một constructor không tham số (default constructor).

Tạo lớp Hình chữ nhật có các thuộc tính là chiều dài và chiều rộng, có phương thức tính chu vi, phương thức tính diện tích.

```
1 package javacoban;  
2  
3 public class HinhChuNhat {  
4     private int chieuDai;  
5     private int chieuRong;  
6  
7     public HinhChuNhat() {  
8     }  
9  
10    public HinhChuNhat(int chieuDai, int chieuRong) {  
11        this.chieuDai = chieuDai;  
12        this.chieuRong = chieuRong;  
13    }  
14  
15    public int tinhDienTich() {  
16        return chieuDai * chieuRong;  
17    }  
18  
19    public int tinhChuVi() {  
20        return (chieuDai + chieuRong) * 2;  
21    }  
22}
```

Khi phân tích bài toán, thường dùng sơ đồ lớp Class diagram – một loại của UML



2.2.2. Đối tượng - Object

Đối tượng được tạo ra từ các lớp nên được gọi là các thể hiện của lớp (class instance).

Ví dụ tạo đối tượng (thể hiện của lớp HinhChuNhat):

```
HinhChuNhat hcn1 = new HinhChuNhat(6, 11);
```

```
HinhChuNhat hcn2 = new HinhChuNhat(30, 24);
```

2.2.3. Từ khóa static

Từ khóa static trong Java được sử dụng để giúp chương trình tiết kiệm bộ nhớ và . Có thể dùng từ khóa static với các biến, các phương thức, các khôi, các lớp lồng nhau.

1. Biến static có thể được sử dụng để tham chiếu thuộc tính chung của tất cả đối tượng.
2. Biến static sẽ lấy bộ nhớ chỉ một lần, nếu bất cứ đối tượng nào thay đổi giá trị của biến static, nó sẽ vẫn ghi nhớ giá trị sau cùng.
3. Các biến static hoặc phương thức static (phương thức tĩnh) được gọi mà không cần tạo một instance của một lớp.
4. Phương thức static không thể sử dụng biến non-static hoặc gọi trực tiếp phương thức non-static.
5. Một phương thức static có thể được kế thừa, nhưng không được ghi đè (override)

Ví dụ:

```

1 public class Student {
2     private int rollno;
3     private String name;
4     private static String college = "TDC"; // chỉ dùng bộ nhớ một lần để lưu all
5     private static int count = 0;
6
7     public Student(int r, String n) {
8         rollno = r;
9         name = n;
10        count++;
11        System.out.println("count = " + count);
12    }
13    public void display() {
14        System.out.println(rollno + " - " + name + " - " + college);
15    }
16
17    public static void main(String args[]) {
18        Student s1 = new Student(111, "Minh");
19        Student s2 = new Student(222, "Phuc");
20
21        s1.display();

```

```
22     s2.display();  
23 }  
24 }
```

```
count = 1  
count = 2  
111 - Minh - TDC  
222 - Phuc - TDC
```

2.2.4. Từ khóa final

Từ khóa final cho biết lần gán giá trị đầu tiên cũng là lần gán giá trị cuối cùng.

Lưu ý:

- + Không thể thay đổi giá trị của biến final.
- + Một biến final mà không được khởi tạo tại thời điểm khai báo được gọi là biến final trống, và nó chỉ có thể được khởi tạo trong Constructor.
- + Tất cả các phương thức của lớp final đều phải là final, nhưng không yêu cầu đối với thuộc tính. Một phương thức final không thể được overload và override.

2.2.5. Package

Dùng package để nhóm các lớp vào với nhau thành các đơn vị nhỏ hơn -> bố trí gọn gàng hơn và dễ quản lý. Một package có chứa nhiều class hay interface bên trong, đồng thời cũng có thể chứa package khác bên trong.

Khi đóng gói, package nhóm các lớp thành thư viện dùng chung.

Dòng khai báo package phải là dòng đầu tiên trong tập tin khai báo lớp. Sử dụng gói thư viện package: **import <tên_package>.<tên_class>;**

Ví dụ:

```
import java.util.Math;
```

```
import java.io.*; //dấu * là import tất cả các class của package java.io
```

Lưu ý:

- Việc import tất cả các lớp trong gói sẽ làm tốn bộ nhớ. Thông thường chỉ nên import những lớp cần dùng trong chương trình.

- Nếu các package khác nhau mà có các lớp trùng tên nhau thì khi sử dụng phải import đầy đủ tên package và tên class.

Một số gói thư viện thường dùng: java.lang, java.util, java.io, java.awt, java.swing,...

2.2.6. Phạm vi truy xuất

Có 4 kiểu khai báo phạm vi truy xuất: Public, private, protected, default

Phạm vi truy xuất				
Access Modifier	Bên trong class	Bên trong package	Bên ngoài package bởi class con	Bên ngoài package
private	Y			
default	Y	Y		
protected	Y	Y	Y	
public	Y	Y	Y	Y

Lưu ý: Để an toàn cho dữ liệu của các đối tượng: tránh khai báo phạm vi truy xuất là public, mà thường chọn private để ngăn cản quyền truy xuất từ bên ngoài class đó.

2.3. Đặc điểm hướng đối tượng trong Java

2.3.1. Tính đóng gói (Encapsulation)

Tính đóng gói thể hiện qua việc dùng lớp (class). Một lớp là mô tả về một tập hợp các đối tượng có cùng các thuộc tính, phương thức. Một lớp bao đóng hoàn toàn khi tất cả dữ liệu của lớp là private và sử dụng phương thức getter và setter để lấy và thay đổi dữ liệu. Tính đóng gói giúp cho các đối tượng che dấu các dữ liệu cục bộ và chi tiết cài đặt, chỉ công bố (public) ra ngoài những thứ cần thiết để làm việc với đối tượng khác. Ngoài ra, tính đóng gói còn thể hiện ở việc các lớp liên quan đến nhau có thể được gom chung lại thành package.

2.3.2. Tính kế thừa (Inheritance)

Tính kế thừa cho phép xây dựng một lớp mới dựa trên một lớp đã có. Lớp đã có gọi là lớp Cha (superclass, base class), lớp mới gọi là lớp Con (subclass, derived class). Một lớp con

có thể kế thừa tất cả các thuộc tính và phương thức của lớp cha. Lớp con có thể thêm thuộc tính và phương thức mới của riêng nó và có thể override phương thức của lớp cha.

Trong Java, có 3 dạng kế thừa chính: kế thừa từ Class, kế thừa từ Abstract class và kế thừa từ Interface.

Mỗi lớp trong Java mặc định là con lớp Object, vì vậy các lớp trong Java có các phương thức của lớp Object.

Ví dụ kế thừa:

Superclass	Subclasses
Student	GraduateStudent, UndergraduateStudent
Shape	Circle, Triangle, Rectangle, Sphere, Cube
Loan	CarLoan, HomeImprovementLoan, MortgageLoan
Employee	SalariedEmployee, CommissionEmployee, HourlyEmployee
BankAccount	CheckingAccount, SavingsAccount

```
public class Xe{  
    protected String nhaSanXuat;  
    protected int namSanXuat;  
    //các phương thức khác...  
}
```

```
public class XeMay extends Xe{  
    //từ khóa super  
}
```

2.3.3. Tính trừu tượng (Abstraction)

Tính trừu tượng là sự ẩn đi của những chi tiết bên trong. Có hai cách thực hiện tính trừu tượng như bên dưới: Lớp trừu tượng (Abstract Class), Giao diện (Interface). Phương thức trừu tượng trong Java được khai báo là Abstract và không có code triển khai cụ thể.

Tính trừu tượng có mối liên hệ mật thiết với tính đa hình và kế thừa.

Lưu ý không nhầm lẫn tính trừu tượng trong Java với việc trừu tượng hóa đối tượng thực để tạo Class: việc trừu tượng hóa này giúp bỏ qua nhiều đặc điểm của đối tượng thực, chỉ lấy những thuộc tính và phương thức cần thiết để lập trình giải quyết bài toán cụ thể.

Overloading và Overriding

Overloading method – Nạp chồng phương thức

Trong một lớp nhiều phương thức có cùng tên nhưng khác tham số (khác kiểu dữ liệu, khác số lượng tham số) gọi là nạp chồng phương thức (overloading method).

```
public class XeMay {
    // khai báo thuộc tính ...
    public float tinhGiaBan() {
        return 2 * chiPhiSX;
    }

    public float tinhGiaBan(float hueHong) {
        return (2 * chiPhiSX + hueHong);
    }
}
```

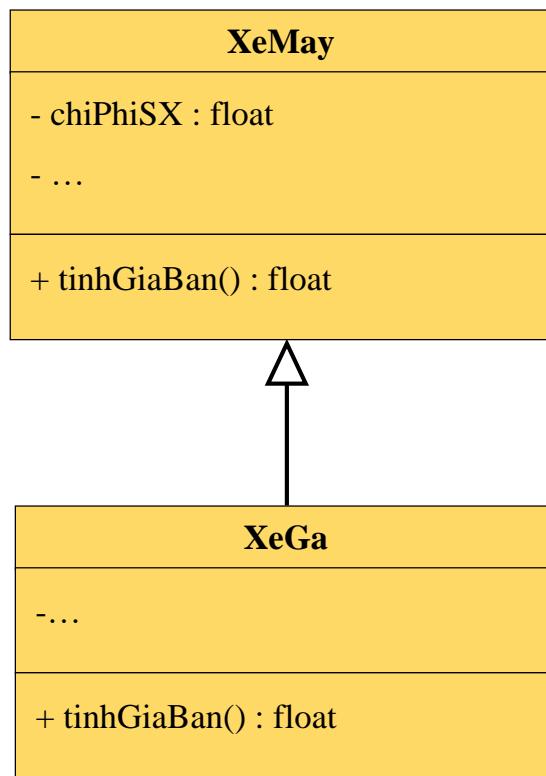
XeMay
- chiPhiSX : float - ...
+ tinhGiaBan() : float + tinhGiaBan(float hueHong) : float

Overriding method – Ghi đè phương thức

Trong kế thừa, các lớp con thừa hưởng các thuộc tính và phương thức public và protected của lớp cha. Lớp con cũng có thể ghi đè phương thức của lớp cha. Khi đó phương thức ở lớp con giống với phương thức ở lớp cha về tên phương thức, kiểu trả về, kiểu và số lượng tham số.

Ví dụ: Overriding phương thức tinhGiaBan()

```
public class XeMay {  
    // khai báo thuộc tính ...  
    public float tinhGiaBan() {  
        return 2 * chiPhiSX;  
    }  
  
    public class XeGa extends XeMay{  
        public float tinhGiaBan() {  
            return 3 * chiPhiSX;  
        }  
        //các thuộc tính và phương thức khác  
    }  
}
```



Ép kiểu – casting:

Ép kiểu là việc thực hiện gán đổi tượng này sang một kiểu dữ liệu khác.

```
Object obj = new Student(); //ép kiểu không tường minh (ngầm), upcasting
```

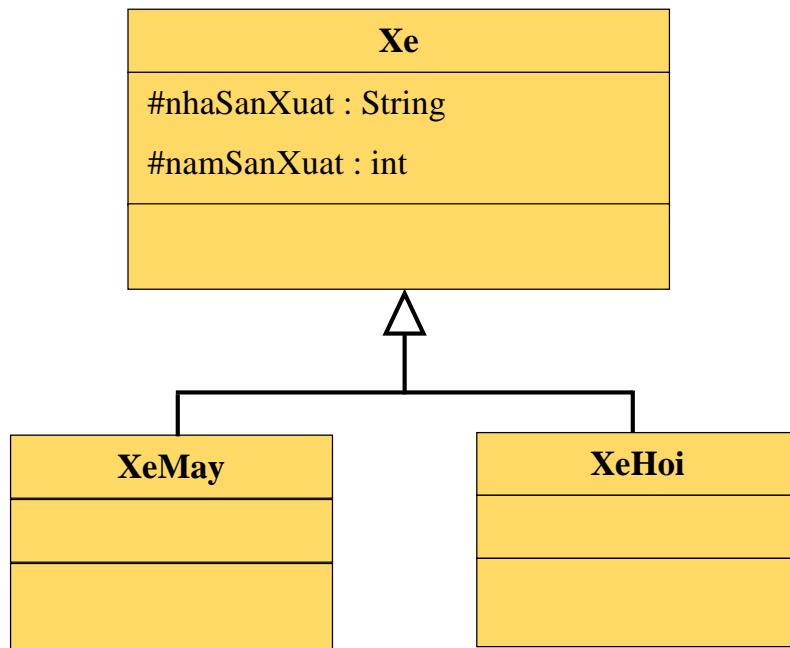
```
Student st1 = (Student) obj; //ép kiểu tường minh, downcasting
```

2.3.4. Tính đa hình (Polymorphism)

Tính đa hình cho phép một phương thức có cách thực hiện khác nhau trên nhiều loại đối tượng khác nhau (overload) hoặc một phương thức có cách thực hiện khác nhau tùy ngữ cảnh khác nhau (override).

Tính đa hình cho phép cài đặt các lớp dẫn xuất (lớp con) khác nhau từ một lớp nguồn (lớp cha). Tính đa hình trong Java là khi một đối tượng có thể có nhiều kiểu khác nhau.

Tính đa hình hỗ trợ tính đóng gói.



```

public class Xe{
    protected String nhaSanXuat;
    protected int namSanXuat;
    //các thuộc tính và phương thức khác...
}

public class XeMay extends Xe{
}

public class XeHoi extends Xe{
}
  
```

2.4. JUnit Test

Viết Unit Test trong Java với JUnit Framework:

Lớp test phải thừa kế từ lớp junit.framework.TestCase.

Mỗi unit test là một phương thức public và không có tham số.

Junit cung cấp các Assertion để kiểm tra kết quả mong đợi.

JUnit cho thấy kết quả test một cách trực quan: đúng như mong đợi (Pass) là màu xanh; không đúng như mong đợi (Fail) là màu đỏ.

Unit Test case: là phần test để đảm bảo rằng một phương thức được thực hiện như mong đợi. Mỗi phương thức có thể có nhiều test case. Mỗi unit test chỉ nên kiểm tra phần cụ thể của một chức năng.

- **Setup()**: là phương thức được chạy trước khi chạy các test case, thường dùng để chuẩn bị dữ liệu để chạy test.
- **Teardown()**: là phương thức được chạy sau khi các test case chạy xong, thường dùng để xóa dữ liệu, giải phóng bộ nhớ.
- **AssertXXX()**: Mỗi test case sẽ có một hoặc nhiều câu lệnh Assert, để kiểm tra tính đúng đắn của phương thức.

Các phương thức assertXXX() của lớp junit.framework.Assert:

- **assertEquals()**: Test sẽ được chấp nhận nếu các giá trị bằng nhau.
- **assertFalse()**: Test sẽ được chấp nhận nếu biểu thức sai.
- **assertTrue()**: Test sẽ được chấp nhận nếu biểu thức đúng.
- **assertNull()**: Test sẽ được chấp nhận nếu tham chiếu là null.
- **assertNotNull()**: Test sẽ được chấp nhận nếu tham chiếu đối tượng khác null.
- **assertSame()**: So sánh địa chỉ vùng nhớ của 2 tham chiếu đối tượng bằng cách sử dụng toán tử `==`. Test sẽ được chấp nhận nếu cả 2 đều tham chiếu đến cùng một đối tượng.
- **assertNotSame()**: So sánh địa chỉ vùng nhớ của 2 tham chiếu đối tượng bằng cách sử dụng toán tử `==`. Test sẽ được chấp nhận nếu cả 2 tham chiếu đến các đối tượng khác nhau.

```
assertEquals(num1, num2, delta); //So sánh số thập phân, có sai số delta
assertEquals(employeeA, employeeB); // So sánh đối tượng
assertEquals("Employees are not equals", employeeA, employeeB); //Có thông báo mô tả
tại sao test thất bại.
```

Ví dụ:

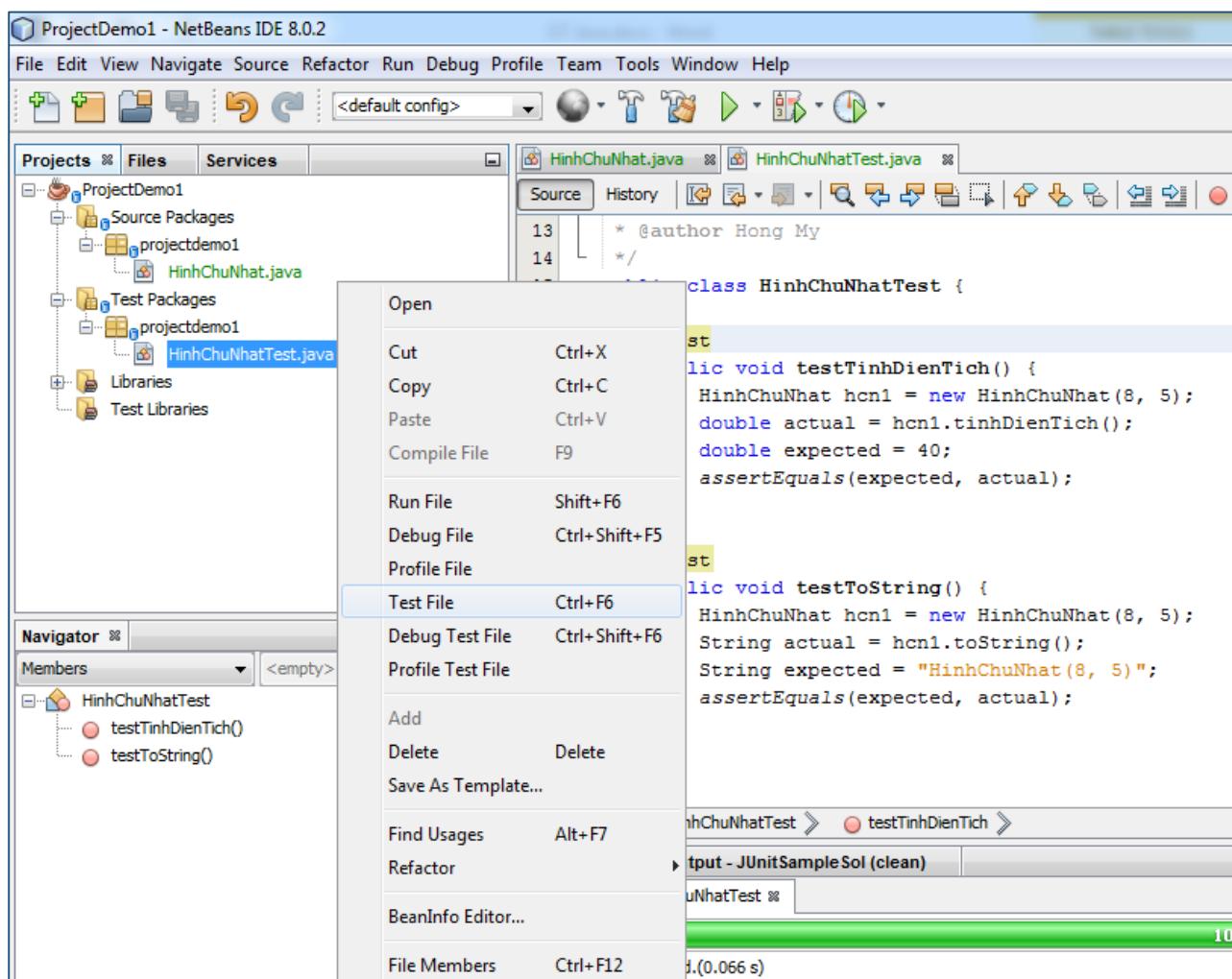
```
1 public class HinhChuNhat {
2
3     private double chieuDai;
4     private double chieuRong;
5
6     public HinhChuNhat(double chieuDai, double chieuRong) {
7         if(chieuDai < 0 || chieuRong < 0){
8             throw new IllegalArgumentException("Tham số không được là số âm!");
9         }
10        this.chieuDai = chieuDai;
11        this.chieuRong = chieuRong;
12    }
13
14
15    public String toString() {
16        return "HinhChuNhat(" + chieuDai + ", " + chieuRong + ')';
17    }
18 }
```

Trong Netbeans để test: cần có thư viện junit4.jar

Tại lớp muốn tạo TestCase -> chọn menu Tools -> Create/Update Tests -> Viết nội dung cần test.

```
1 public class HinhChuNhatTest {
2
3     @Test (expected = IllegalArgumentException.class)
4     public void testConstructor_Abnormal () {
5         HinhChuNhat hcn1 = new HinhChuNhat(-8, 5);
```

```
6      HinhChuNhat hcn1 = new HinhChuNhat(8, -5);
7  }
8  @Test
9  public void testTinhDienTich() {
10     HinhChuNhat hcn1 = new HinhChuNhat(8, 5);
11     double actual = hcn1.tinhDienTich();
12     double expected = 40;
13     assertEquals(expected, actual, 0.0001);
14 }
15
16 @Test
17 public void testToString() {
18     HinhChuNhat hcn1 = new HinhChuNhat(8, 5);
19     String actual = hcn1.toString();
20     String expected = "HinhChuNhat(8, 5)";
21     assertEquals(expected, actual);
22 }
23 }
```



Hơn nữa, cần kiểm tra độ bao phủ (coverage) của Unit Test để có thể đánh giá được chất lượng của Unit Testcase. Để đảm bảo các đoạn code được kiểm thử và các trường hợp có thể xảy ra đều được kiểm thử.

Case Study Quản Lý Sinh Viên

Tạo các Class cơ bản cần có cho chương trình ứng dụng:

- Khoa(maKhoa, tenKhoa)
- LopCoDinh(maLop, tenLop, khoaHoc, maKhoa)
- SinhVien(maSV, lop, hoTen, ngaySinh, soDienThoai, diaChi)
- MonHoc(maMon, tenMon, soTinChi)
- LopHocPhan(maLopHocPhan, maMon, namHoc, hocKy, giaoVien)
- DiemHocPhan(maSV, maLopHocPhan, diemQuaTrinh, diemThi)

Yêu cầu tạo các lớp với các thuộc tính tương ứng và có thêm phương thức in thông tin và các phương thức cần thiết khác. Viết TestCase cho mỗi lớp.

Lớp Khoa:

```
1 public class Khoa {  
2     private String maKhoa;  
3     private String tenKhoa;  
4  
5     public Khoa(String maKhoa, String tenKhoa) {  
6         this.maKhoa = maKhoa;  
7         this.tenKhoa = tenKhoa;  
8     }  
9  
10    public String getMaKhoa() {  
11        return maKhoa;  
12    }  
13  
14    public void setMaKhoa(String maKhoa) {  
15        this.maKhoa = maKhoa;  
16    }  
17  
18    public String getTenKhoa() {  
19        return tenKhoa;  
20    }  
21  
22    public void setTenKhoa(String tenKhoa) {  
23        this.tenKhoa = tenKhoa;  
24    }  
25}
```

```
26     public String showInfo() {  
27         return String.format("Khoa: %6s - %-30s", maKhoa, tenKhoa);  
28     }  
29 }
```

Bài tập

Bài 1. Viết chương trình in bảng cửu chương:

- a. Chương trình nhận vào một số k ($2 \leq k \leq 9$), xử lý in một bảng cửu chương k
- b. Chương trình in bảng cửu chương từ 2 đến 9 nằm ngang.

Bài 2. Viết chương trình nhận vào một chuỗi và cho xuất ra chuỗi viết tắt là các ký tự đầu tiên của mỗi từ. Vd: Cao Dang Cong Nghe Thu Duc -> CDCNTD

Bài 3. Viết chương trình in ra tổng $1+3+5+\dots+n$ nếu n là số lẻ, $2+4+6+\dots+n$ nếu n là số chẵn.

Bài 4. Viết chương trình in ra bảng Binary, Octal, và Hexadecimal từ 1 đến 256.

Bài 5. Viết chương trình lấy số nguyên ngẫu nhiên giữa hai số. ví dụ: 15 và 30

Bài 6. Viết chương trình chuyển một số thập phân sang dạng số LA MÃ và ngược lại.

Ví dụ: MDCLXVI = 1666; M:1000; D:500; C:100; L:50; X:10; V:5; I:1

Bài 7. Viết ứng dụng tung xí ngầu 6 mặt, thống kê số lần xuất hiện của mỗi mặt trong n lần tung.

Bài 8. Viết ứng dụng cho bé tập tính toán cộng các số trong phạm vi 10, và xuất thông báo ngẫu nhiên cho đáp án đúng và sai.

Khi trả lời đúng:

- Very good!
- Excellent!
- Nice work!
- Keep up the good work!

Khi trả lời sai:

- No. Please try again.
- Wrong. Try once more.
- Don't give up!
- Wrong. Keep trying.

Bài 9. Tạo class phân số có tử số và mẫu số. Thực hiện các phương thức sau:

- a. Tối giản phân số
- b. Cộng một phân số khác

- c. Trừ một phân số khác
- d. Nhân một phân số khác
- e. Chia một phân số khác

Bài 10. Tạo lớp điểm có hoành độ và tung độ, có phương thức tính khoảng cách tới gốc tọa độ, phương thức tính khoảng cách tới một điểm khác.

Bài 11. Cho HìnhChuNhật(chieuRong, chieuCao). Kiểm tra một điểm có thuộc hình chữ nhật không. Tính giao nhau của hình chữ nhật này với hình chữ nhật khác.

Bài 12. * Viết chương trình tính toán các số nguyên lớn (nhiều hơn 20 chữ số) và viết Test Case. (Xử lý chuỗi, ASCII, Testcase)

- a. Tổng hai số nguyên lớn
- b. Hiệu hai số nguyên lớn

CHƯƠNG 3. XỬ LÝ NGOẠI LỆ - EXCEPTION HANDLING

Học xong chương này, người học có thể:

- + Trình bày ưu điểm của việc xử lý ngoại lệ
- + Mô tả được mô hình xử lý ngoại lệ
- + Mô tả cách sử dụng các khối ‘try’, ‘catch’ và ‘finally’
- + Giải thích cách sử dụng các từ khoá ‘throw’ và ‘throws’
- + Phân biệt được hai loại ngoại lệ checked và unchecked
- + Tạo ngoại lệ mới

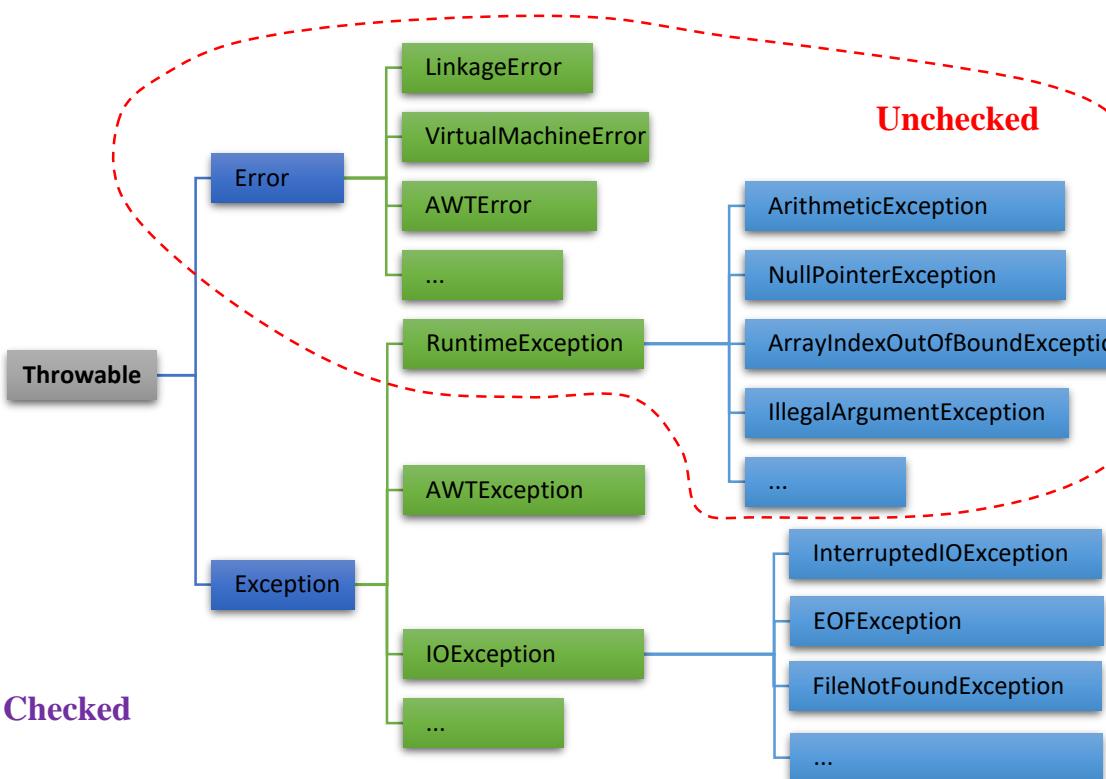
3.1. Giới thiệu ngoại lệ Exception

Exception là sự kiện xảy ra trong quá trình thực thi chương trình, làm gián đoạn luồng làm việc của chương trình. Chương trình đang chạy sẽ lập tức ngừng lại và thông báo lỗi.

Một ngoại lệ có thể xảy ra do nhiều nguyên nhân khác nhau, như là: Người dùng nhập sai dữ liệu quy định; Một file nào đó cần mở mà không có; Kết nối mạng bị mất khi cần truyền dữ liệu; Bộ nhớ bị thiếu; ... Ngoại lệ có thể là do lỗi từ người dùng, lỗi từ lập trình viên, lỗi do hệ thống vật lý.

Một chương trình tốt cần phải xử lý các ngoại lệ có thể xảy ra. Bởi vì chương trình sẽ bị ngắt khi một ngoại lệ xảy ra. Khi đó, tất cả các nguồn tài nguyên mà hệ thống đã cấp cho chương trình sẽ không được giải phóng, dẫn tới lãng phí tài nguyên.

Ví dụ: Khi thao tác vào ra (I/O) trong một tập tin. Nếu việc chuyển đổi kiểu dữ liệu không thực hiện đúng, một ngoại lệ sẽ xảy ra và chương trình bị ngắt mà không đóng tập tin lại. Lúc đó tập tin có thể bị hư hại và các nguồn tài nguyên được cấp phát cho tập tin không được trả lại cho hệ thống.



RuntimeException là những ngoại lệ xảy ra khi chạy chương trình

Ưu điểm của việc xử lý ngoại lệ:

- Làm chương trình dễ đọc và an toàn hơn
- Dễ dàng chuyển điều khiển đến nơi có khả năng xử lý ngoại lệ
- Tách xử lý ngoại lệ khỏi đoạn mã thông thường
- Không bỏ sót ngoại lệ (tung ngoại lệ tự động)
- Gom nhóm và phân loại các ngoại lệ
- Làm chương trình dễ đọc và an toàn hơn

3.2. Xử lý ngoại lệ Exception

Mô hình xử lý ngoại lệ giám sát việc thực thi mã để phát hiện ngoại lệ. Trong mô hình này, khi một ngoại lệ xảy ra, ngoại lệ sẽ bị chặn và chương trình chuyển đến một khối xử lý ngoại lệ.

- Sử dụng khối try ... catch để xử lý.
- Sử dụng khối try ... catch ... finally.
- Sử dụng Nested try (lồng một khối try trong một try khác).
- Sử dụng từ khóa throw và throws.

Cấu trúc mô hình xử lý ngoại lệ try catch finally:

```
try {  
    // đoạn mã có khả năng gây ra ngoại lệ  
}  
catch(Exception e1) {  
    // xử lý ngoại lệ cho loại ngoại lệ e1  
}  
catch(Exception eN) {  
    // xử lý ngoại lệ cho loại ngoại lệ e2  
}  
finally {  
    // khối lệnh này luôn được thực hiện cho dù ngoại lệ có xảy ra hay không.  
}
```

3.2.1. Khối try/catch

```

1   try {
2       int a = 8;
3       int b = 2;
4       int c = a / b;
5
6       int[] arr = { 5, 10 };
7       arr[3] = 15;
8
9       System.out.println(c);
10 } catch (ArithmetricException ae) {
11     System.out.println("Lỗi chia cho 0: " + ae);
12 } catch (Exception e) {
13     System.out.println("Lỗi khác: " + e);
14 }
```

3.2.2. Khối finally

Các câu lệnh trong khối finally đảm bảo luôn được thực hiện cho dù ngoại lệ có xảy ra hay không. Sử dụng khối finally để viết code gọn gàng và giải phóng bộ nhớ.

```

1   try {
2       int result = 45 / 0;
3       System.out.println(result);
4   } catch (ArithmetricException e) {
5       System.out.println(e);
6   } finally {
7       System.out.println("Khối finally luôn được thực thi");
8   }
9   System.out.println("Phần code khác");
10 }
```

3.2.3. Ủy nhiệm ngoại lệ throw, throws

```

1   public int div(int a, int b) throws ArithmetricException {
2       if (b == 0) {
3           throw new ArithmetricException("Lỗi chia cho không");
4       } else {
5           return a / b;
6       }
7 }
```

7 }

3.3. Ngoại lệ được kiểm tra và ngoại lệ không được kiểm tra

Trong Java có 2 loại Exception là Checked Exception và Unchecked Exception.

Checked Exception là các Exception xảy ra tại thời điểm Compile time (*là thời điểm chương trình đang được biên dịch*). Những Exception này thường liên quan đến lỗi cú pháp (*syntax*) và bắt buộc chúng ta phải bắt (*catch*) nó.

Unchecked Exception: là các Exception xảy ra tại thời điểm Runtime (*là thời điểm chương trình đang chạy*). Những Exception này thường liên quan đến lỗi logic và không bắt buộc chúng ta phải bắt (*catch*) nó.

Ví dụ Checked Exception:

```
1 package tdc.demo;  
2  
3 public class CheckedException {  
4     public static void main(String[] args) {  
5         System.out.println(abc);  
6     }  
7 }
```

Dòng code `System.out.println(abc);` sẽ bị lỗi. Lý do là chuỗi đó phải nằm trong cặp dấu ngoặc " ". Đây chính là Checked Exception, nếu chạy chương trình thì sẽ có thông báo lỗi như sau:

```
Exception in thread "main" java.lang.Error: Unresolved compilation problem:  
  abc cannot be resolved to a variable  
  
  at tdc.demo.MyException.main(MyException.java:5)
```

Ví dụ Unchecked Exception:

```
1 package tdc.demo;  
2  
3 public class UncheckedException {  
4     public static void main(String[] args) {  
5         int a = 5, b = 0;  
6         System.out.println(a/b);  
7     }  
8 }
```

Chương trình này sẽ không báo lỗi gì trong đoạn code của chúng ta nhưng khi biên dịch thì sẽ có thông báo lỗi ‘/ by zero’ (lỗi chia cho 0) trong màn hình Console. Đây chính là Unchecked Exception. Thông báo lỗi như sau:

```
Exception in thread "main" java.lang.ArithmetricException: / by zero
at tdc.demo.MyException.main(MyException.java:6)
```

Điểm khác nhau cơ bản giữa Checked Exception và Unchecked Exception chính là thời điểm xác định được Exception có thể xảy ra. Đối với Checked Exception, việc kiểm tra được thực hiện ngay tại thời điểm Compile time (*bien dịch*), một số IDE sẽ giúp hiển thị lỗi cú pháp. Còn đối với Unchecked Exception, việc xác định có Exception xảy ra hay không chỉ có thể thực hiện tại thời điểm Runtime (chạy chương trình), Exception này thường liên quan đến lỗi logic và các IDE sẽ không xác định được các ngoại lệ này.

Một số Checked Exception thường gặp: ClassNotFoundException, IOException, FileNotFoundException, EOFException.

Một số Unchecked Exception thường gặp: ArithmeticException, NullPointerException, IllegalArgumentException, IndexOutOfBoundsException, InputMismatchException.

3.4. Định nghĩa một ngoại lệ mới

Trong một số trường hợp, các ngoại lệ của Java không đủ để kiểm soát tất cả các lỗi. Cần phải có các lớp ngoại lệ do người dùng định nghĩa.

Ngoại lệ tự định nghĩa là class kế thừa từ class Exception hoặc lớp con của class Exception. Lớp ngoại lệ mới này có tất cả các phương thức của lớp Throwable.

```
1 public class MyException extends Exception {
2     public MyException(String msg) {
3         super(msg);
4     }
5
6     public MyException(String msg, Throwable cause) {
7         super(msg, cause);
8     }
9 }
```

```
1 public class FileExample {  
2     public void copyFile(String fName1, String fName2) throws MyException {  
3         if (fName1.equals(fName2))  
4             throw new MyException("Same name error");  
5         // các lệnh thực hiện copy file  
6         System.out.println("Copy completed");  
7     }  
8 }
```

Case Study Quản Lý Sinh Viên

1. Bổ sung code, xử lý và thông báo khi nhập các trường số là số âm, các trường String không được null.

```
1 public MonHoc(String maMonHoc, String tenMonHoc, int soTinchi) {  
2     if (maMonHoc == null || tenMonHoc == null) {  
3         throw new IllegalArgumentException("MaMonHoc va TenMonHoc khong la null");  
4     }  
5     if (soTinchi < 0) {  
6         throw new IllegalArgumentException("SoTinChi khong duoc la so am");  
7     }  
8     this.maMonHoc = maMonHoc;  
9     this.tenMonHoc = tenMonHoc;  
10    this.soTinchi = soTinchi;  
11 }
```

2. Tạo ngoại lệ mới để báo lỗi khi nhập điểm không hợp lệ. Điều kiện điểm phải nằm trong khoảng từ 0 đến 10. Điều chỉnh code trong lớp DangKyHocPhan để kiểm lỗi nhập điểm.

```
1 public class InvalidMarkException extends Exception {  
2     public InvalidMarkException(String msg) {  
3         super(msg);  
4     }  
5     public InvalidMarkException(String msg, Throwable cause) {  
6         super(msg, cause);  
7     }  
8 }
```

```
1 public void setDiemQuaTrinh(float diemQuaTrinh) throws InvalidMarkException {  
2     if (diemQuaTrinh < 0 || diemQuaTrinh > 10)  
3         throw new InvalidMarkException("Diem phai tu 0 den 10");  
4     this.diemQuaTrinh = diemQuaTrinh;
```

```
5 }
```

3. Viết TestCase để test các trường hợp bắt lỗi dữ liệu.

```
1 @Test(expected = IllegalArgumentException.class)
2 public void testConstructorException() {
3     MonHoc monHoc1 = new MonHoc("CNC1900111", null, -5);
4 }
```

Bài tập

Bài 1. Thêm phần xử lý ngoại lệ cho các bài tập ở các chương trước.

Bài 2. Viết chương trình thực hiện phép chia, có xử lý lỗi ngoại lệ với trường hợp số bị chia bằng 0.

Bài 3. Viết chương trình cho nhập vào 2 số nguyên (dùng Scanner hoặc JOptionPane), xuất kết quả phép chia 2 số này. Yêu cầu kiểm tra việc nhập số (tử và mẫu không được nhập chữ, mẫu không được bằng 0). Chương trình sẽ hiển thị thông báo khi người dùng nhập sai dạng dữ liệu toán hạng và cho nhập lại đến khi nhập đúng.

(NumberFormatException)

Bài 4. Một lớp HinhTamGiac với 3 cạnh. Trong một hình tam giác, tổng 2 cạnh luôn lớn hơn cạnh còn lại. Hãy xây dựng lớp IllegalTriangleException chỉnh sửa lại Constructor HinhTamGiac tung (throws) ra ngoại lệ IllegalTriangleException khi các cạnh truyền vào vi phạm luật trên.

```
public Triangle(int a, int b, int c) throws IllegalTriangleException {
    // Implement it
}
```

Bài 5. Viết chương trình khởi tạo mảng 50 số nguyên ngẫu nhiên, người dùng nhập vào một chỉ số và chương trình hiển thị giá trị của phần tử tại chỉ số đó. Trường hợp người dùng nhập vào chỉ số nằm ngoài mảng, chương trình hiển thị thông báo: “Out of Bounds”

CHƯƠNG 4. ĐỌC FILE VÀ GHI FILE

Học xong chương này, người học có thể:

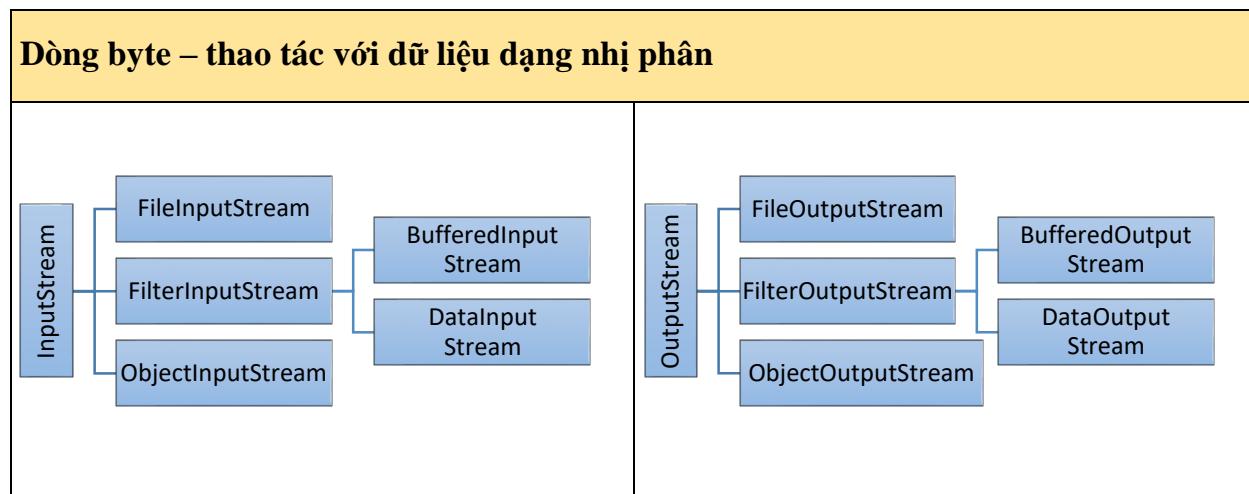
- + Viết chương trình đọc ghi file nhị phân
- + Viết chương trình thao tác với file văn bản
- + Viết chương trình đọc ghi kiểu đối tượng
- + Sử dụng lớp File trong Java để thao tác với tập tin và thư mục.

4.1. Đọc ghi file nhị phân

Đọc và ghi file trong java là các hoạt động nhập/xuất dữ liệu (đọc dữ liệu từ bàn phím, đọc dữ liệu từ file, ghi dữ liệu lên màn hình, ghi ra file, ghi ra đĩa, ghi ra máy in...) đều được gọi là dòng (stream).

Dòng nhập chuẩn: System.in để lấy dữ liệu từ bàn phím

Dòng xuất chuẩn: System.out để hiển thị dữ liệu ra màn hình



Sử dụng luồng byte trong các trường hợp như nhập xuất kiểu dữ liệu nguyên thủy (như kiểu int, float, double, boolean), nhập xuất kiểu dữ liệu kiểu đối tượng (object)

Một số phương thức xử lý dữ liệu nhị phân của **DataInputStream**, **DataOutputStream**

DataInputStream	DataOutputStream
boolean readBoolean()	void writeBoolean (boolean val)
byte readByte()	void writeByte (int val)
char readChar()	void writeChar (int val)
double readDouble()	void writeDouble (double val)
float readFloat()	void writeFloat (float val)
int readInt()	void writeInt(int val)
long readLong()	void writeLong (long val)
short readShort()	void writeShort (short val)

Ví dụ: Đọc ghi dữ liệu nhị phân / dữ liệu nguyên thủy

```
1 package file_accessing;
2
3 import java.io.DataOutputStream;
4 import java.io.FileNotFoundException;
5 import java.io.FileOutputStream;
6 import java.io.IOException;
7 import java.io.DataInputStream;
8 import java.io.FileInputStream;
9
10 public class ReadWriteBinaryData {
11     public void writeBinaryData() {
12         try {
13             // B1: Tạo luồng và liên kết file dữ liệu
14             FileOutputStream fos = new FileOutputStream("data.bin");
15             DataOutputStream dos = new DataOutputStream(fos);
16
17             // B2: Ghi dữ liệu
18             dos.writeInt(100);
19             dos.writeDouble(9.5);
20
21             // B3: Đóng luồng
22             fos.close();
23             dos.close();
24             System.out.println("Done!");
25         } catch (IOException ex) {
26             ex.printStackTrace();
27         }
28     }
29
30     public void readBinaryData() {
31         try {
32             // B1: Tạo luồng và liên kết file dữ liệu
33             FileInputStream fis = new FileInputStream("data.bin");
34             DataInputStream dis = new DataInputStream(fis);
35
36             // B2: Đọc dữ liệu
37             int n = dis.readInt();
38             double m = dis.readDouble();
```

```
39
40     // B3: Đóng luồng
41     fis.close();
42     dis.close();
43
44     // Hiển thị nội dung đọc từ file
45     System.out.println("Số nguyên: " + n);
46     System.out.println("Số thực: " + m);
47 } catch (IOException ex) {
48     ex.printStackTrace();
49 }
50 }
51 }
```

Ví dụ sao chép file bằng cách đọc và ghi dữ liệu nhị phân (binary data)

```
1 package file_accessing;
2
3 import java.io.FileInputStream;
4 import java.io.FileOutputStream;
5 import java.io.IOException;
6
7 public class CopyFile {
8     public static void main(String[] args) throws IOException {
9         FileInputStream inputStream = null;
10        FileOutputStream outputStream = null;
11
12        try {
13            inputStream = new FileInputStream("inStream.txt");
14            outputStream = new FileOutputStream("outStream.txt");
15
16            int c;
17            while ((c = inputStream.read()) != -1) {
18                outputStream.write(c);
19            }
20        } finally {
21            if (inputStream != null) {
22                inputStream.close();
23            }
24            if (outputStream != null) {
25                outputStream.close();
26            }
27        }
28    }
29}
```

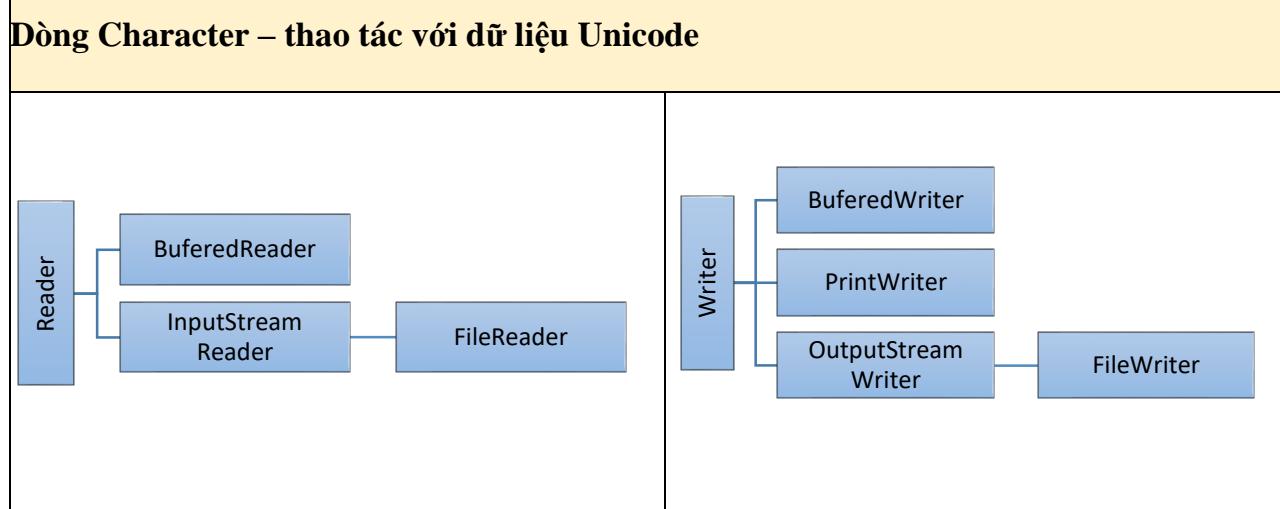
```

28 }
29 }
```

4.2. Đọc ghi file văn bản

Xử lý nhập xuất dữ liệu bằng dòng character (character stream)

Dòng character được dùng để thao tác với file chứa văn bản Unicode.



Ghi dữ liệu với FileWriter

```

1 package file_accessing;
2
3 import java.io.File;
4 import java.io.FileWriter;
5 import java.io.IOException;
6
7 public class FileWriterExample {
8     public static void main(String[] args) {
9         try {
10             // B1: Tạo stream và liên kết file dữ liệu
11             File f = new File("data3.txt");
12             FileWriter fw = new FileWriter(f);
13
14             // B2: Ghi dữ liệu
15             fw.write("Thực hành ghi dữ liệu bằng dòng character"); // nhieu hon
16
17             // B3: Đóng dòng
```

```
18     fw.close();
19 } catch (IOException ex) {
20     System.out.println("Loi ghi file: " + ex);
21 }
22 }
23 }
```

Đọc dữ liệu với FileReader

```
1 package file_accessing;
2
3 import java.io.BufferedReader;
4 import java.io.File;
5 import java.io.FileReader;
6
7 public class FileReaderExample {
8     public static void main(String[] args) {
9         try {
10     //B1: Tạo dòng đọc và liên kết file dữ liệu
11         File f = new File("data3.txt");
12         FileReader fr = new FileReader(f);
13
14     //B2: Đọc dữ liệu
15         BufferedReader br = new BufferedReader(fr);
16         String line;
17         while ((line = br.readLine()) != null){
18             System.out.println(line);
19         }
20
21     //B3: Đóng
22         fr.close();
23         br.close();
24     } catch (Exception ex) {
25         System.out.println("Loi doc file: "+ex);
26     }
27 }
28 }
```

Sao chép file bằng cách sử dụng luồng Character

```

1 import java.io.FileReader;
2 import java.io.FileWriter;
3 import java.io.IOException;
4
5 public class CopyFileCharacter {
6     public static void main(String[] args) throws IOException {
7         FileReader in = null;
8         FileWriter out = null;
9
10        try {
11            in = new FileReader("input.txt");
12            out = new FileWriter("output.txt");
13
14            int c;
15            while ((c = in.read()) != -1) {
16                out.write(c);
17            }
18        } finally {
19            if (in != null) {
20                in.close();
21            }
22            if (out != null) {
23                out.close();
24            }
25        }
26    }
27 }
```

4.3. Đọc ghi đối tượng

Đọc và ghi dữ liệu kiểu object

Tạo lớp Student và lớp này phải là Serializable

```

1 import java.io.Serializable;
2
3 public class Student implements Serializable{
4     private int id;
5     private String name;
6     private double grade;
7     private int yearOfBirth;
8
9     public Student(int id, String name, int yearOfBirth, double grade) {
```

```
10     this.id = id;
11     this.name = name;
12     this.yearOfBirth = yearOfBirth;
13     this.grade = grade;
14 }
15
16 @Override
17 public String toString() {
18     return id + "-" + name + "-" + yearOfBirth + "-" + grade;
19 }
20 }
```

```
1 import java.io.FileInputStream;
2 import java.io.FileOutputStream;
3 import java.io.IOException;
4 import java.io.ObjectInputStream;
5 import java.io.ObjectOutputStream;
6
7 public class ReadWriteObject {
8     public void writeListOfStudent() {
9         try {
10             // B1: Tạo dòng và liên kết file dữ liệu
11             FileOutputStream fos = new FileOutputStream("StudentList.bin");
12             ObjectOutputStream oos = new ObjectOutputStream(fos);
13             // Khai báo mảng
14             Student listOfStudent[] = { new Student(1111, "Lan", 1999, 5.5),
15                 new Student(1112, "Huong", 200, 9),
16                 new Student(1113, "Minh", 1998, 7.5) };
17             // B2: Ghi mảng đối tượng vào file
18             oos.writeObject(listOfStudent);
19
20             // B3: Đóng dòng
21             fos.close();
22             oos.close();
23         } catch (IOException ex) {
24             System.out.println("Loi ghi file: " + ex);
25         }
26     }
27 }
```

```

28  public void readListOfStudent() {
29      try {
30          // B1: Tạo dòng và liên kết file dữ liệu
31          FileInputStream fis = new FileInputStream("StudentList.bin");
32          ObjectInputStream ois = new ObjectInputStream(fis);
33
34          // B2: Đọc dữ liệu
35          Student sArr[] = (Student[]) ois.readObject();
36          for (Student st : sArr) {
37              System.out.println(st.toString());
38          }
39
40          // B3: Đóng dòng
41          fis.close();
42          ois.close();
43      } catch (Exception ex) {
44          System.out.println("Loi doc file: " + ex);
45      }
46  }
47
48  public static void main(String[] args) {
49      ReadWriteObject rwo = new ReadWriteObject();
50      rwo.writeListOfStudent();
51      rwo.readListOfStudent();
52  }
53 }
```

Lưu ý một số ngoại lệ thường gặp khi thao tác file: FileNotFoundException, EOFException, NotSerializableException

4.4. Lớp File

Lớp File dùng để biết được các thông tin chi tiết về tập tin cũng như thư mục (tên, ngày tạo, kích thước, ...)

Constructor:

```
new File("filename.txt"); // đường dẫn tương đối  
new File("E:\\Data\\JavaProgramming\\filename.txt"); // đường dẫn tuyệt đối  
File dir = new File("C:/Test/Demo"); //đường dẫn tuyệt đối tới thư mục
```

Một số phương thức của lớp File

boolean delete()	Xóa file
boolean exists()	Kiểm tra file tồn tại
String getName()	Lấy tên file
String getAbsolutePath()	Lấy đường dẫn tuyệt đối của file
long length()	Lấy kích thước file tính bằng bytes
String[] list()	Liệt kê tên thư mục con và tên file trong thư mục
File[] listFiles()	Lấy các file trong thư mục
boolean mkdir()	Tạo thư mục

Case Study Quản Lý Sinh Viên

Đọc và ghi dữ liệu danh sách môn học vào file

Dùng cách đọc ghi văn bản

```
1 public class ReadWriteFileHelper {  
2     public void writeDanhSachMonHoc(ArrayList<MonHoc> dsmh, String fileName) {  
3         try {  
4             FileWriter fileWriter = new FileWriter(fileName);  
5             BufferedWriter bufferedWriter = new BufferedWriter(fileWriter);  
6  
7             MonHoc monHoc = null;  
8             for (int i = 0; i < dsmh.size(); i++) {  
9                 monHoc = dsmh.get(i);  
10                bufferedWriter.write(monHoc.toString());  
11                if (i < dsmh.size() - 1)  
12                    bufferedWriter.write("\n");  
13            }  
14  
15            bufferedWriter.close();  
16        } catch (IOException ex) {  
17            ex.printStackTrace();  
18        }  
19    }  
20  
21    public void appendMonHoc(MonHoc monHoc, String fileName) {  
22        try {  
23            FileWriter fileWriter = new FileWriter(fileName, true); //ghi tiếp vào file  
24            BufferedWriter bufferedWriter = new BufferedWriter(fileWriter);  
25  
26            bufferedWriter.newLine();  
27            bufferedWriter.write(monHoc.toString());  
28  
29            bufferedWriter.close();  
30        } catch (IOException ex) {  
31            ex.printStackTrace();  
32        }  
33    }  
34}
```

```
33     }
34
35     public ArrayList<MonHoc> readDanhSachMonHoc(String fileName) {
36         ArrayList<MonHoc> dsmh = new ArrayList<MonHoc>();
37         try {
38             FileReader fileReader = new FileReader(fileName);
39             BufferedReader bufferedReader = new BufferedReader(fileReader);
40             MonHoc monHoc = null;
41             String line = null;
42             while ((line = bufferedReader.readLine()) != null) {
43                 String[] monHocInfo = line.split("-");
44                 monHoc = new MonHoc(monHocInfo[0], monHocInfo[1],
45                                     Integer.parseInt(monHocInfo[2]));
46                 dsmh.add(monHoc);
47             }
48
49             bufferedReader.close();
50         } catch (IOException ex) {
51             ex.printStackTrace();
52         }
53         return dsmh;
54     }
55 }
```

Dùng cách đọc ghi đối tượng:

```
1  public class ReadWriteObjectToFileHelper {
2      public void writeDanhSachMonHoc(ArrayList<MonHoc> dsmh, String fileName) {
3          try {
4              FileOutputStream fos = new FileOutputStream(fileName);
5              ObjectOutputStream oos = new ObjectOutputStream(fos);
6
7              oos.writeObject(dsmh);
8
9              fos.close();
10             oos.close();
11         } catch (IOException ex) {
12             ex.printStackTrace();
13         }
14     }
15 }
```

```
13      }
14  }
15
16  public ArrayList<MonHoc> readDanhSachMonHoc(String fileName) {
17      ArrayList<MonHoc> dsmh = new ArrayList<MonHoc>();
18      try {
19          FileInputStream fis = new FileInputStream(fileName);
20          ObjectInputStream ois = new ObjectInputStream(fis);
21
22          dsmh = (ArrayList<MonHoc>) ois.readObject();
23
24      } catch (Exception ex) {
25          ex.printStackTrace();
26      }
27      return dsmh;
28  }
29 }
```

Bài tập

Bài 1. Viết chương trình đọc nội dung của một file txt và hiển thị nội dung đó lên màn hình.

Bài 2. Cho file văn bản data.txt chứa một dãy các số thực với số lượng chưa biết trước, mỗi số cách nhau một khoảng trắng. Hãy viết chương trình đọc file trên và in ra màn hình giá trị tổng và trung bình cộng của dãy số trong file đó.

Bài 3. Viết chương trình tạo ra một file text có tên NumText.txt nếu tập tin đó chưa tồn tại. Tạo và ghi 10 số nguyên ngẫu nhiên vào file trên, mỗi số cách nhau một khoảng trắng. Đọc file và in ra màn hình dãy số nguyên đã được sắp xếp.

Bài 4. Chương trình ReplaceText.java cho phép đọc văn bản từ file sourceFile, thay đổi các chuỗi oldString thành chuỗi mới newString và lưu văn bản đó và file mới destFile. Hãy cài đặt chương trình trên.

Bài 5. Viết chương trình đọc nội dung từ file text ‘NumText.txt’, file này có chứa 10 số nguyên ngẫu nhiên, các số cách nhau một khoảng trắng. Sau đó ghi 10 số nguyên này vào tập tin nhị phân có tên ‘NumBin.dat’ (tạo mới nếu tập tin chưa tồn tại). Hiển thị kích thước của file text và file nhị phân. Đọc và in ra màn hình các số nguyên từ file nhị phân trên.

Bài 6. Viết chương trình tạo 10 đối tượng SinhVien (SinhVien gồm các thông tin: mã số, họ tên, năm sinh, lớp) và lưu các đối tượng này vào tập tin dssinhvien.dat. Viết chương trình đọc tập tin dssinhvien.dat và hiển thị ra màn hình các thông tin: sinh viên có Số tuổi lớn nhất; số tuổi trung bình của các sinh viên trong tập tin.

Bài 7. Tạo lớp NhanVien gồm các thông tin mã nhân viên, họ tên, tuổi và lương. Viết phương thức cho phép ghi thêm nhân viên vào file ‘nhanvien.txt’. Sử dụng FileWriter để ghi thông tin (ghi đối tượng). Viết phương thức để đọc nội dung (đọc đối tượng) file nhanvien.txt và in danh sách nhân viên ra màn hình.

Bài 8. Tạo lớp QuanLyNhanVien gồm danh sách các nhân viên. Viết phương thức ghi toàn bộ mảng nhân viên vào file ‘nhanvien.bin’, sử dụng ObjectOutputStream để ghi dữ liệu. Viết phương thức để đọc file ‘nhanvien.bin’ và in danh sách nhân viên ra màn hình.

CHƯƠNG 5. LẬP TRÌNH VỚI CƠ SỞ DỮ LIỆU

Học xong chương này, người học có thể:

- + Trình bày cách kết nối và truy xuất cơ sở dữ liệu.
- + Liệt kê được ưu nhược điểm của cách dùng câu lệnh trực tiếp và dùng store procedure.
- + Lập trình kết nối và truy xuất cơ sở dữ liệu: thêm, xóa, sửa, hiển thị dữ liệu.

5.1. JDBC - Java Database Connectivity

5.1.1. JDBC Driver

Một số hệ quản trị cơ sở dữ liệu (DBMS-Database Management System) thường dùng: SQL Sever, MySQL, Oracle, MS Access.

Để truy cập các DBMS từ chương trình viết bằng Java cần có các JDBC driver tương ứng. JDBC Driver là một thành phần phần mềm cho phép ứng dụng Java tương tác với cơ sở dữ liệu.

Cú pháp tạo kết nối CSDL: `DriverManager.getConnection(dbUrl, username, password);`

Một số Driver:

DBMS	Định dạng URL (chuỗi kết nối đến CSDL)
MySQL com.mysql.jdbc.Driver	<code>jdbc:mysql://hostname/dbName</code>
ORACLE oracle.jdbc.driver.OracleDriver	<code>jdbc:oracle:thin:@hostname:port:dbName</code>
IMB-DB2 com.ibm.db2.jdbc.net.DB2Driver	<code>jdbc:db2:hostname:port/dbName</code>
Cầu nối ODBC sun.jdbc.odbc.JdbcOdbcDriver	<code>jdbc:odbc:DataSourceName</code>

Các kiểu chuỗi kết nối CSDL đối với SQL server -

`com.microsoft.jdbc.sqlserver.SQLServerDriver`

- Kết nối với trường hợp mặc định của máy chủ SQL đang chạy trên cùng một máy với máy khách JDBC, sử dụng chứng thực Windows:

`jdbc:sqlserver://localhost;integratedSecurity=true;`

- Kết nối tới sqlexpress trên máy chủ lưu trữ dbHost, sử dụng xác thực SQL Server:

`jdbc:sqlserver://dbHost\sqlexpress;user=sa;password=secret`

- Kết nối với một cơ sở dữ liệu có tên testdb trên localhost bằng xác thực Windows:

`jdbc: sqlserver: // localhost: 1433; databaseName = testdb; integratedSecurity = true;`

Link tải driver:

<https://docs.microsoft.com/en-us/sql/connect/jdbc/>

<https://www.oracle.com/technetwork/database/features/jdbc>

<https://dev.mysql.com/doc/connector-j>

5.1.2. Kết nối cơ sở dữ liệu bằng JDBC

Các thành phần của JDBC API:

1. **DriverManager**: `java.sql.DriverManager`

Class DriverManager dùng để quản lý danh sách các **Driver** (database drivers).

2. **Driver**: `java.sql.Driver`

Là một Interface dùng để liên kết các connection với cơ sở dữ liệu.

3. **Connection**

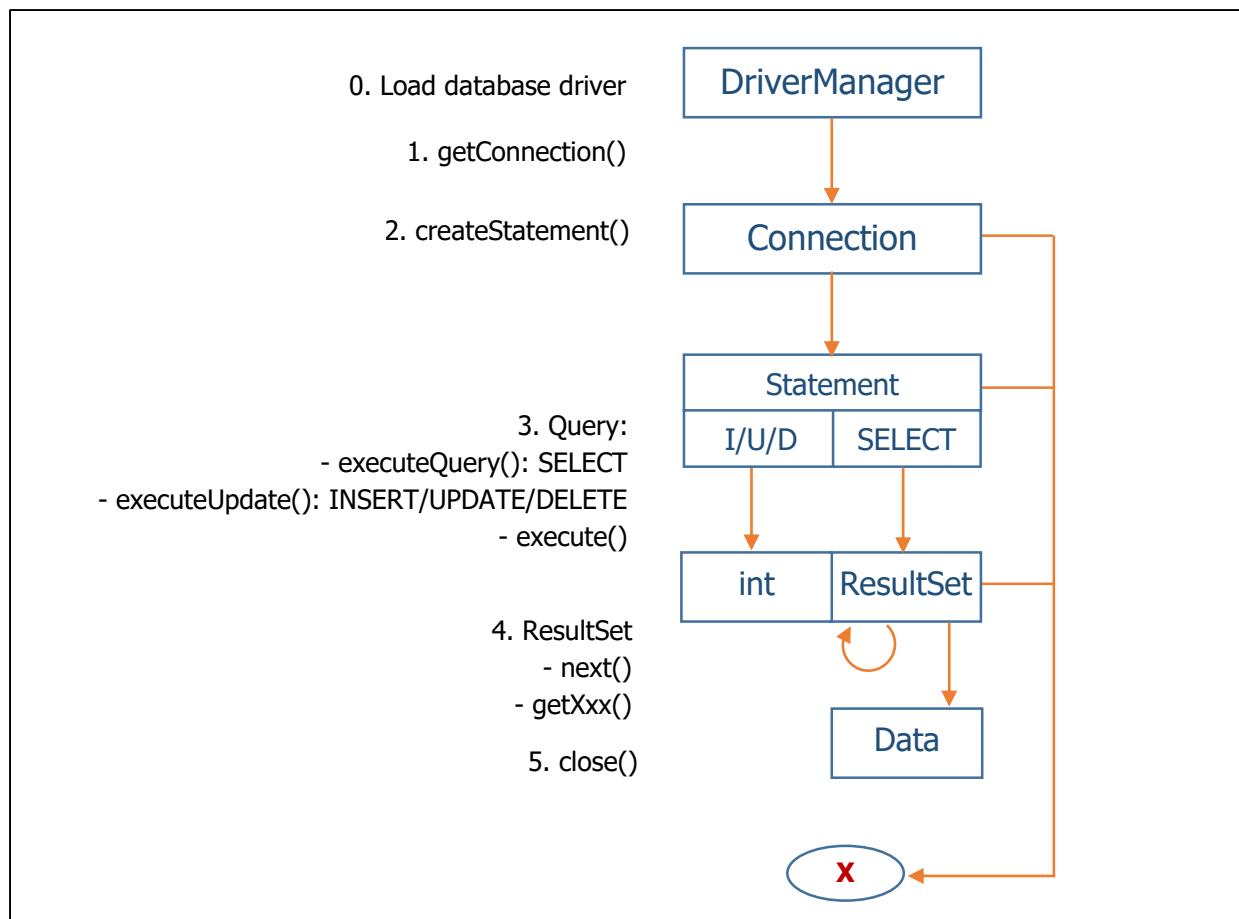
Là một Interface có các phương thức kết nối với database.

4. **Statement**

Là một Interface, gói gọn một câu lệnh SQL gửi tới cơ sở dữ liệu được phân tích, tổng hợp, lập kế hoạch và thực hiện.

5. **ResultSet**

ResultSet đại diện cho tập hợp các bản ghi lấy do thực hiện truy vấn.



Quá trình kết nối CSDL

getXxx(): getString(), getBigDecimal(), getBoolean(), getInt(), getByte(), getShort(),
getLong(), getFloat(), getDouble(), getDate(), getTime().

Các thao tác tạo ứng dụng tương tác với CSDL:

+ Tải driver

Ví dụ: <http://dev.mysql.com/downloads/>: mysql-connector-java-8.0.{xx}.jar

+ Import driver vào thư viện của project

Chuột phải **Libraries** -> Chọn **Add JAR/Folder...** -> Chỉ định thư mục chứa JDBC

Driver và chọn driver là tập tin .jar > chọn **Open**

+ Đăng ký Driver class: //java 6 tự tìm Driver

```
Class.forName("oracle.jdbc.driver.OracleDriver");
```

+ Tạo kết nối: Create connection

```
Connection con = DriverManager.getConnection(
    "jdbc:oracle:thin:@localhost:1521:demodatabase","system","password");
```

+ Tạo câu lệnh truy vấn: Create statement

```
Statement stmt=con.createStatement();
```

+ Thực thi câu lệnh: Execute queries

```
ResultSet rs = stmt.executeQuery("SELECT * FROM book");
while(rs.next()){
    System.out.println(rs.getInt(1)+" "+rs.getString(2));
}
```

+ Đóng kết nối: Close connection

```
con.close();
```

5.2. Tạo ứng dụng JDBC đơn giản

5.2.1. Ứng dụng 1: Thao tác với CSDL

Tạo lớp DataBaseUtil để thuận tiện cho việc thao tác với CSDL

Ví dụ: Bảng Student có các cột: student_id, name, email

```
1 public class DataBaseUtil {
2     public static Connection c;
3     private static String db_url = "jdbc:mysql://localhost:3306/studentdb";
4     private static String username = "root";
5     private static String password = "#on9My";
6
7     public static Connection getConnection() throws Exception {
8         if (c == null) {
9             Class.forName("com.mysql.jdbc.Driver");// Từ Java 6 sẽ tự tìm Driver
10            c = DriverManager.getConnection(db_url, username, password);
11        }
12        return c;
13    }
14
15    // Send data To Database
```

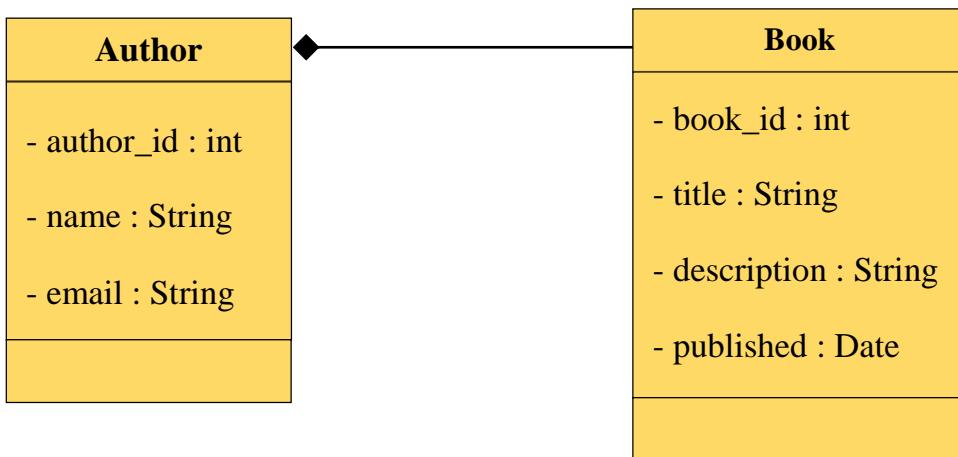
```
16     public static int setData(String sql) throws Exception {  
17         return DataBaseUtil.getConnection().createStatement().executeUpdate(sql);  
18     }  
19  
20     // Get data From Database  
21     public static ResultSet getData(String sql) throws Exception {  
22         ResultSet rs = DataBaseUtil.getConnection().createStatement()  
23             .executeQuery(sql);  
24         return rs;  
25     }  
26 }
```

Thao tác với dữ liệu từ CSDL

```
1 public class QueryDataDemo {  
2     public static void main(String[] args) throws Exception {  
3         // Lấy ra đối tượng Connection kết nối vào DB.  
4         Connection connection = DataBaseUtil.getConnection();  
5  
6         String sql = "Select student_id, name, email from Student";  
7  
8         // Thực thi câu lệnh SQL trả về đối tượng ResultSet.  
9         ResultSet rs = DataBaseUtil.getData(sql);  
10  
11        // Duyệt trên kết quả trả về.  
12        while (rs.next()) {// Di chuyển con trỏ tới mẫu tin kế tiếp.  
13            int stuId = rs.getInt(1);  
14            String stuName = rs.getString(2);  
15            String stuEmail = rs.getString("email");  
16            System.out.println("-----");  
17            System.out.println("StuId:" + stuId);  
18            System.out.println("StuName:" + stuName);  
19            System.out.println("StuEmail:" + stuEmail);  
20        }  
21  
22        String sql2 = "Delete from student where student_id = 111";  
23        DataBaseUtil.setData(sql2);  
24  
25        // Lấy lại danh sách sinh viên.  
26        rs = DataBaseUtil.getData(sql);  
27        // Duyệt trên kết quả trả về.  
28        while (rs.next()) {// Di chuyển con trỏ tới mẫu tin kế tiếp.
```

```
29     int stuId = rs.getInt(1);
30     String stuName = rs.getString(2);
31     String stuEmail = rs.getString("email");
32     System.out.println("-----");
33     System.out.println("StuId:" + stuId);
34     System.out.println("StuName:" + stuName);
35     System.out.println("StuEmail:" + stuEmail);
36 }
37 // Đóng kết nối
38 connection.close();
39 }
40 }
```

5.2.2. Ứng dụng 2: Thao tác với CSDL dùng store procedure



Tạo procedure trong MySQL

```

CREATE PROCEDURE `booksdb`.`create_author` (IN name VARCHAR(45), email
VARCHAR(45))
BEGIN
    DECLARE newAuthorID INT;

    INSERT INTO author (name, email) VALUES (name, email);

    SET newAuthorID = (SELECT author_id FROM author a WHERE a.name = name);

    INSERT INTO book (title, description, published, author_id, price,
rating)
        VALUES (CONCAT('Life Story of ', name),
                CONCAT('Personal Stories of ', name),
                date('2016-12-30'), newAuthorID, 90000, 0);
END
  
```

```

1 public class DataBaseUtil2 {
2     public static Connection c;
3     private static String db_url = "jdbc:mysql://localhost:3306/booksdb";
4     private static String username = "root";
5     private static String password = "#on9My";
6
7     public static Connection getConnection() throws Exception {
8         if (c == null) {
9             Class.forName("com.mysql.jdbc.Driver");// Từ Java 6 sẽ tự tìm Driver
10            c = DriverManager.getConnection(db_url, username, password);
11        }
12        return c;
13    }
  
```

```
14 }
```

```

1 public class StoreProcedureCallDemo1 {
2     public static void main(String[] args) {
3         try {
4             Connection conn = DataBaseUtil2.getConnection();
5             CallableStatement statement = conn
6                 .prepareCall("{call create_author(?, ?)}");
7
8             statement.setString(1, "James Gosling");
9             statement.setString(2, "james@abc.com");
10
11            statement.execute();
12            statement.close();
13            conn.close();
14            System.out.println("Stored procedure called successfully!");
15        } catch (SQLException ex) {
16            ex.printStackTrace();
17        }
18    }
19 }
```

Tạo Stored Procedure từ code Java

```

1 public class StoreProcedureCreatingDemo {
2     public static void main(String[] args) {
3         try {
4             Connection conn = DataBaseUtil2.getConnection();
5             Statement statement = conn.createStatement();
6
7             String queryDrop = "DROP PROCEDURE IF EXISTS delete_book";
8
9             String queryCreate = "CREATE PROCEDURE delete_book (IN bookID INT) ";
10            queryCreate += "BEGIN ";
11            queryCreate += "DELETE FROM book WHERE book_id = bookID; ";
12            queryCreate += "END";
13
14            // xóa procedure nếu đã có
15            statement.execute(queryDrop);
16            // rồi tạo stored procedure mới
```

```
17     statement.execute(queryCreate);
18     statement.close();
19     conn.close();
20     System.out.println("Stored procedure created successfully!");
21 } catch (SQLException ex) {
22     ex.printStackTrace();
23 }
24 }
25 }
```

Gọi Stored Procedure có tham số OUT và INOUT

```
CREATE PROCEDURE `summary_report`(
    IN title VARCHAR(45),
    OUT totalBooks INT,
    OUT totalValue DOUBLE,
    INOUT highPrice DOUBLE
)
BEGIN
    DECLARE maxPrice DOUBLE;

    SELECT COUNT(*) AS bookCount, SUM(price) as total
        FROM book b JOIN author a ON b.author_id = a.author_id
        AND b.title LIKE CONCAT('%', title, '%')
    INTO totalBooks, totalValue;

    SELECT MAX(price) FROM book WHERE price INTO maxPrice;

    IF (maxPrice > highPrice) THEN
        SET highPrice = maxPrice;
    END IF;
END
```

```
1 public class StoreProcedureCallDemo2 {
2     public static void main(String[] args) {
3         try {
4             Connection conn = DataBaseUtil2.getConnection();
5             CallableStatement statement = conn
6                 .prepareCall("{call summary_report(?, ?, ?, ?)}");
7
8             statement.registerOutParameter(2, Types.INTEGER);
9             statement.registerOutParameter(3, Types.DOUBLE);
10            statement.registerOutParameter(4, Types.DOUBLE);
11
12            statement.setString(1, "Java");
13            statement.setDouble(4, 50);
14        }
```

```

15     statement.execute();
16
17     Integer totalBook = (Integer) statement.getObject(2, Integer.class);
18     Double totalValue = statement.getDouble(3);
19     Double highPrice = statement.getDouble("highPrice");
20
21     System.out.println("Total books: " + totalBook);
22     System.out.println("Total value: " + totalValue);
23     System.out.println("High price: " + highPrice);
24
25     statement.close();
26     conn.close();
27 } catch (SQLException ex) {
28     ex.printStackTrace();
29 }
30 }
31 }
```

Gọi Stored Procedure trả về một Result Set

```

CREATE PROCEDURE `get_books` (IN rate INT)
BEGIN
    SELECT * FROM book WHERE rating >= rate;
END
```

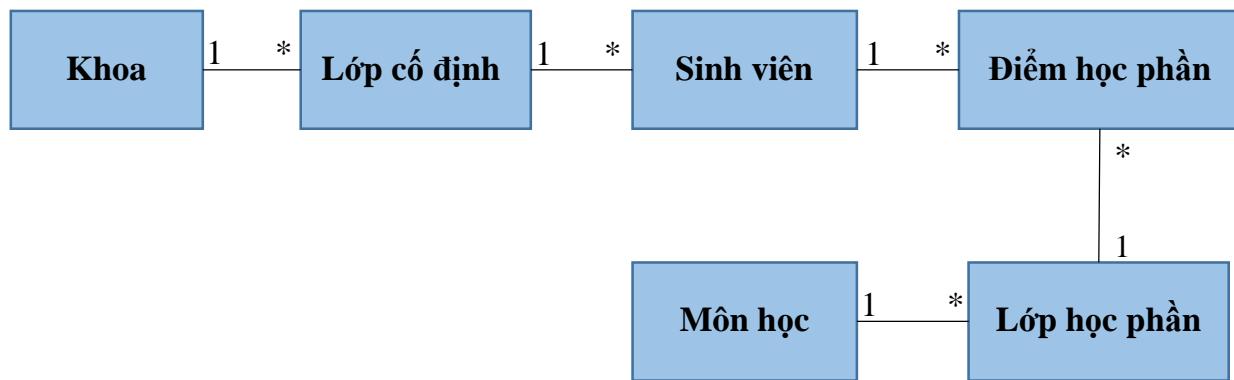
```

1 public class StoreProcedureCallDemo3 {
2     public static void main(String[] args) {
3         try {
4             Connection conn = DataBaseUtil2.getConnection();
5             CallableStatement statement = conn.prepareCall("{call get_books(?)}");
6             statement.setInt(1, 5);
7             boolean hadResults = statement.execute();
8
9             // in dòng tiêu đề
10            System.out.println("| Title | Description | Rating |");
11            System.out.println("=====");
12
13            while (hadResults) {
14                ResultSet resultSet = statement.getResultSet();
15
16                // Xử lý dữ liệu kết quả truy vấn được
```

```
17     while (resultSet.next()) {  
18         String title = resultSet.getString("title");  
19         String description = resultSet.getString("description");  
20         int rating = resultSet.getInt("rating");  
21  
22         System.out.println(" | " + title + " | " + description + " | "  
23             + rating + " | ");  
24     }  
25     hadResults = statement.getMoreResults();  
26 }  
27 statement.close();  
28 } catch (SQLException ex) {  
29     ex.printStackTrace();  
30 }  
31 }  
32 }
```

Case Study Quản Lý Sinh Viên

Tạo CSDL Quản lý sinh viên, viết code thao thác với các Table.



Thao tác với CSDL MySQL, thực hiện thêm, xóa, sửa, lấy tất cả môn học

```

1  public class DataBaseUtil {
2      public static Connection c;
3      private static String db_url = "jdbc:mysql://localhost:3306/quanlysinhvien";
4      private static String username = "root";
5      private static String password = "#on9My";
6
7      public static Connection getConnection() throws Exception {
8          if (c == null) {
9              Class.forName("com.mysql.jdbc.Driver");// Từ Java 6 sẽ tự tìm Driver
10             c = DriverManager.getConnection(db_url, username, password);
11         }
12         return c;
13     }
14
15     // Send data To Database
16     public static int setData(String sql) throws Exception {
17         return DataBaseUtil.getConnection().createStatement().executeUpdate(sql);
18     }
19
20     // Get data From Database
21     public static ResultSet getData(String sql) throws Exception {
22         ResultSet rs = DataBaseUtil.getConnection().createStatement()
23             .executeQuery(sql);
24         return rs;
  
```

```
25    }
26
27    public static void createSP_DeleteMonHoc() {
28        try {
29            Connection conn = DataBaseUtil.getConnection();
30            Statement statement = conn.createStatement();
31
32            String queryDrop = "DROP PROCEDURE IF EXISTS DeleteMonHoc";
33            String queryCreate = "CREATE PROCEDURE DeleteMonHoc "
34                + "(IN maMH VARCHAR(10)) ";
35            queryCreate += "BEGIN ";
36            queryCreate += "DELETE FROM monhoc WHERE mamonhoc = maMH; ";
37            queryCreate += "END";
38
39            // xóa procedure nếu đã có
40            statement.execute(queryDrop);
41            // rồi tạo stored procedure mới
42            statement.execute(queryCreate);
43
44            statement.close();
45            conn.close();
46        } catch (Exception ex) {
47            ex.printStackTrace();
48        }
49    }
50
51    public static void createSP_UpdateMonHoc() {
52        try {
53            Connection conn = DataBaseUtil.getConnection();
54            Statement statement = conn.createStatement();
55
56            String queryDrop = "DROP PROCEDURE IF EXISTS UpdateMonHoc";
57            String queryCreate = "CREATE PROCEDURE UpdateMonHoc "
58                + "(IN maMH VARCHAR(10), tenMH VARCHAR(60), soTC INT) ";
59            queryCreate += "BEGIN ";
60            queryCreate += "UPDATE monhoc SET tenmonhoc = tenMH, sotinchi = soTC"
61                + "WHERE mamonhoc = maMH; ";
62            queryCreate += "END";
63            // xóa procedure nếu đã có
64            statement.execute(queryDrop);
65            // rồi tạo stored procedure mới
66            statement.execute(queryCreate);
```

```

67
68     statement.close();
69     conn.close();
70 } catch (Exception ex) {
71     ex.printStackTrace();
72 }
73 }
74
75 public static void createSP_GetAllMonHoc() {
76     try {
77         Connection conn = DataBaseUtil.getConnection();
78         Statement statement = conn.createStatement();
79
80         String queryDrop = "DROP PROCEDURE IF EXISTS GetAllMonHoc";
81
82         String queryCreate = "CREATE PROCEDURE GetAllMonHoc () ";
83         queryCreate += "BEGIN ";
84         queryCreate += "SELECT * FROM monhoc WHERE 1; ";
85         queryCreate += "END";
86
87         // xóa procedure nếu đã có
88         statement.execute(queryDrop);
89         // rồi tạo stored procedure mới
90         statement.execute(queryCreate);
91
92         statement.close();
93         conn.close();
94     } catch (Exception ex) {
95         ex.printStackTrace();
96     }
97 }
98 }
```

```

1 public class MonHocDAO {
2     public static boolean themMonHoc(MonHoc monHoc) {
3         // sử dụng query
4         int i = -1;
5         try {
6             Connection conn = DataBaseUtil.getConnection();
7             String sql = "INSERT INTO monhoc VALUES('" + monHoc.getMaMonHoc() + "', '"
```

```
8             + monHoc.getTenMonHoc() + "','" + monHoc.getSoTinchi() + ")";
9         i = DataBaseUtil.setData(sql);
10        //conn.close();
11    } catch (Exception e) {
12        e.printStackTrace();
13    }
14
15    return (i != -1);
16 }
17
18 public static void xoaMonHoc(String maMonHoc) {
19     // sử dụng Store Procedure
20     try {
21         Connection conn = DataBaseUtil.getConnection();
22         CallableStatement statement = conn.prepareCall("{call DeleteMonHoc(?)}");
23         statement.setString(1, maMonHoc);
24         statement.execute();
25         //conn.close();
26     } catch (Exception e) {
27         e.printStackTrace();
28     }
29 }
30
31 public static void suaMonHoc(MonHoc monHoc) {
32     // sử dụng Store Procedure
33     try {
34         Connection conn = DataBaseUtil.getConnection();
35         CallableStatement statement = conn
36             .prepareCall("{call UpdateMonHoc(?, ?, ?)}");
37         statement.setString(1, monHoc.getMaMonHoc());
38         statement.setString(2, monHoc.getTenMonHoc());
39         statement.setInt(3, monHoc.getSoTinchi());
40         statement.execute();
41         //conn.close();
42     } catch (Exception e) {
43         e.printStackTrace();
44     }
45 }
46
47 public static ArrayList<MonHoc> getDanhSachMonHoc() {
48     // //sử dụng Store Procedure
49     ArrayList<MonHoc> result = new ArrayList<MonHoc>();
```

```
50     try {
51         Connection conn = DataBaseUtil.getConnection();
52         CallableStatement statement = conn.prepareCall("{call GetAllMonHoc()}");
53         boolean hadResults = statement.execute();
54
55         while (hadResults) {
56             ResultSet resultSet = statement.getResultSet();
57
58             // Xử lý dữ liệu kết quả truy vấn được
59             while (resultSet.next()) {
60                 String maMonHoc = resultSet.getString("mamonhoc");
61                 String tenMonHoc = resultSet.getString("tenmonhoc");
62                 int sotinchi = resultSet.getInt("sotinchi");
63
64                 result.add(new MonHoc(maMonHoc, tenMonHoc, sotinchi));
65             }
66             hadResults = statement.getMoreResults();
67         }
68         statement.close();
69     } catch (Exception ex) {
70         ex.printStackTrace();
71     }
72     return result;
73 }
74 }
```

Bài tập

Bài 1. Viết ứng dụng Quản lý thông tin sinh viên và lớp học: Sinh viên (mã số, họ tên, email, điện thoại) và lớp học (mã lớp, tên lớp, giáo viên chủ nhiệm), dữ liệu lưu trong CSDL. Sử dụng store procedure để thao tác với CSDL.

Bài 2. Thêm chức năng đăng nhập hệ thống cho ứng dụng Quản lý thông tin sinh viên. Username và password được lưu trong CSDL.

Bài 3. Viết chương trình Quản lý sách: tương tác với CSDL và in ra thông tin bảng dữ liệu theo mẫu. Table book:

id	title	author	price	qty
1001	Java for dummies	Tan Ah Teck	11.11	11
1002	More Java for dummies	Tan Ah Teck	22.22	22
1003	More Java for more dummies	Mohammad Ali	33.33	33
1004	A Cup of Java	Kumar	44.44	44
1005	A Teaspoon of Java	Kevin Jones	55.55	55

```
create database if not exists ebookshop;

use ebookshop;

drop table if exists books;
create table books (
    id int,
    title varchar(50),
    author varchar(50),
    price float,
    qty int,
    primary key (id));

insert into books values (1001, 'Java for dummies', 'Tan Ah Teck', 11.11, 11);
insert into books values (1002, 'More Java for dummies', 'Tan Ah Teck', 22.22, 22);
insert into books values (1003, 'More Java for more dummies', 'Mohammad Ali', 33.33, 33);
insert into books values (1004, 'A Cup of Java', 'Kumar', 44.44, 44);
insert into books values (1005, 'A Teaspoon of Java', 'Kevin Jones', 55.55, 55);

select * from books;
```

Bài 4. Bổ sung yêu cầu cho ứng dụng bài Quản lý sách với một số điều chỉnh như sau:

- + Tăng giá 50% cho sách “A Cup of Java”
- + Đặt số lượng là 0 cho sách “A Teaspoon of Java”
- + Xóa tất cả sách có id > 6000; và thêm các dòng: (6001, 'Java ABC', 'Mr Author', 15.55, 55) và (8002, 'Java XYZ', 'Mr Author', 25.55, 55);

CHƯƠNG 6. THIẾT KẾ GIAO DIỆN NGƯỜI DÙNG GUI

Học xong chương này, người học có thể:

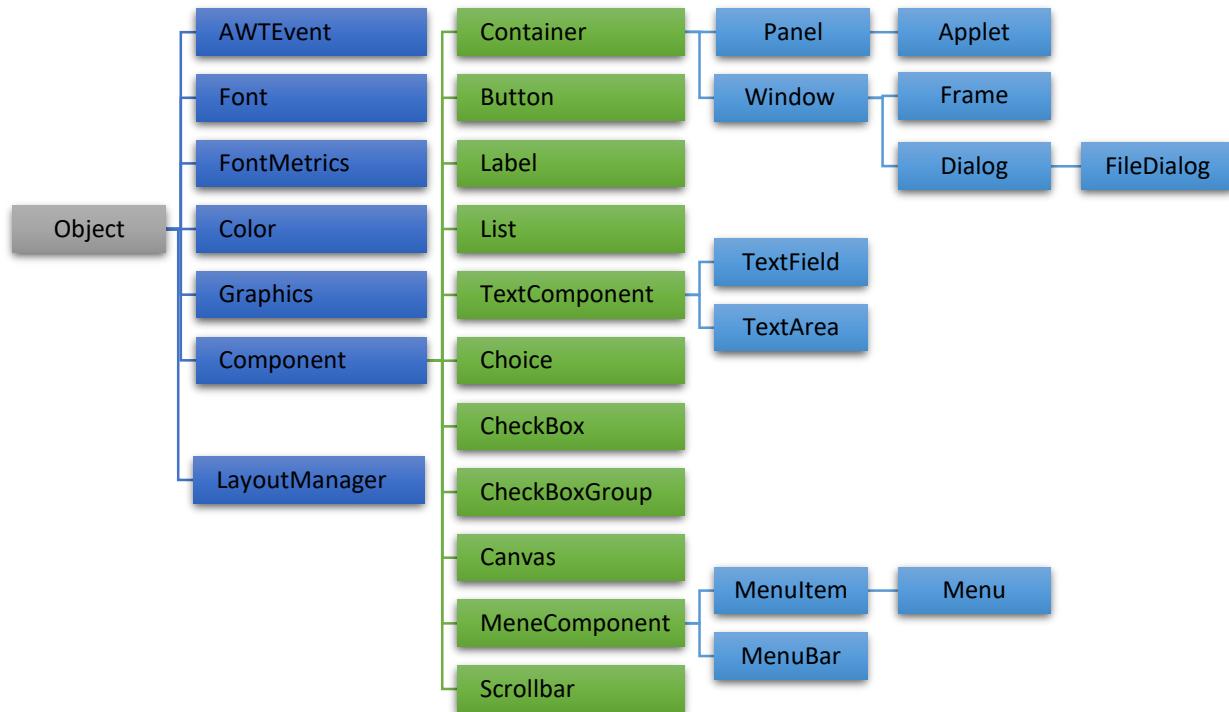
- + Thiết kế giao diện JFrame và các thành phần trên Frame;
- + Sử dụng một số Layout thông dụng
- + Dùng Graphics 2D để vẽ các đối tượng cơ bản: Rectangle, Ellipse, Line

6.1. Thành phần GUI

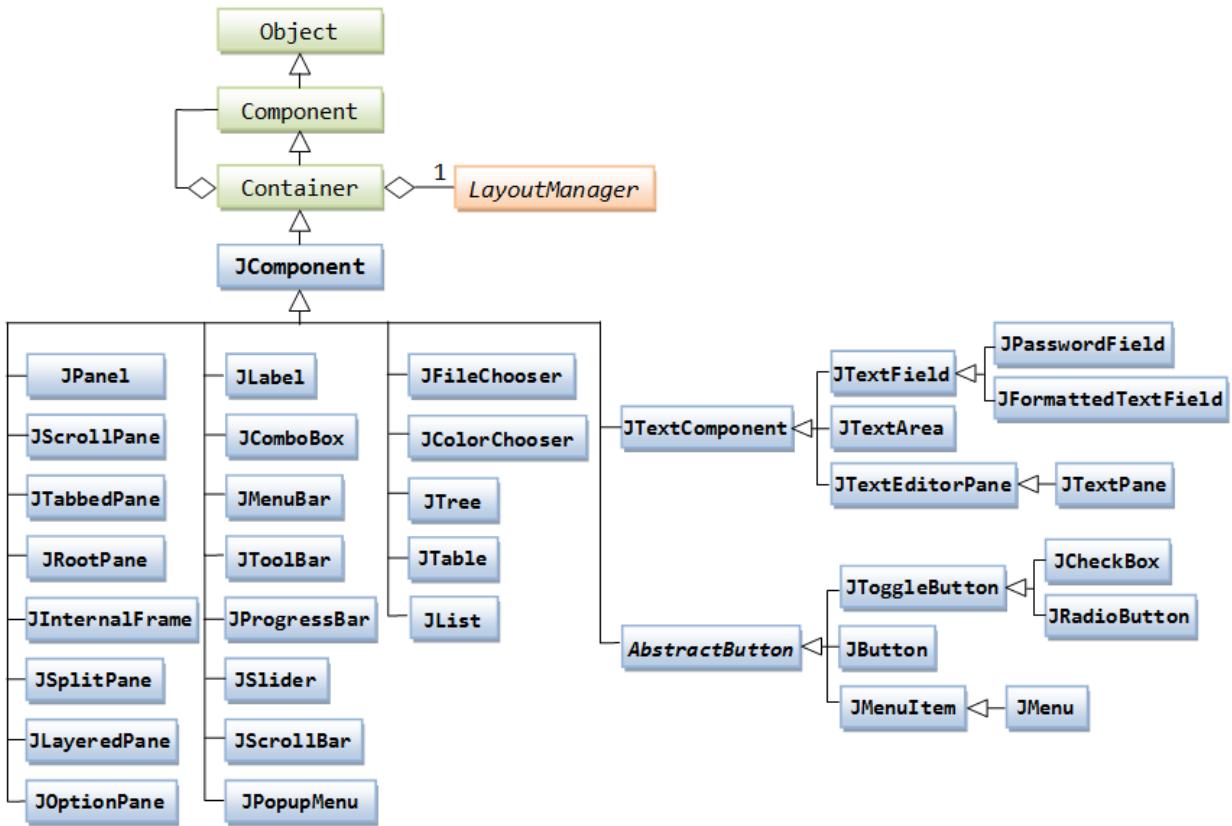
6.1.1. Frame

Thư viện các lớp hỗ trợ thiết kế xây dựng giao diện đồ họa người dùng GUI (Graphics User Interface): java.awt.*; java.awt.event.*; javax.swing.*;

AWT (Abstract Window Toolkit)



Swing:



Các bước cơ bản để tạo cửa sổ Frame:

```

JFrame frame = new JFrame();
frame.setSize(300, 400);
frame.setTitle("Demo an empty frame");
frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
frame.setVisible(true);
  
```

6.1.2. Một số thành phần cơ bản của GUI

JLabel: Hiển thị văn bản hoặc biểu tượng hình ảnh

JTextField: Trường nhập dữ liệu từ bàn phím, cũng có thể hiển thị thông tin

JButton: Nút nhấn dùng kích hoạt sự kiện

JCheckBox: Hộp kiểm tra cho phép chọn hoặc không chọn

JComboBox: Hộp danh mục thả xuống để chọn các thành phần trong danh sách

JPanel: Một Container trong đó có các thành phần GUI, có thể bố trí theo Layout

JFrame: Khung GUI để trình bày các thành phần GUI

Quản lý bố cục- Layout

Để quản lý sắp xếp các thành phần GUI

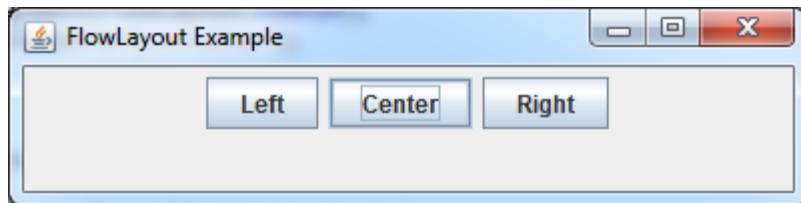
6.1.3. Một số Layout thông dụng

6.1.3.1. FlowLayout: được sử dụng để sắp xếp các đối tượng từ trái sang phải. FlowLayout là bố cục mặc định (default layout) của **panel** hoặc **applet**.

FlowLayout(): tạo ra một bố cục mà các đối tượng được sắp xếp đúng thứ tự trước sau và khoảng trống mặc định của các đối tượng là 5 đơn vị theo chiều ngang và chiều dọc. Bố cục được căn giữa trong JFrame.

FlowLayout(int align): bố cục có canh lề có thể là: FlowLayout.LEADING (canh lề trái, FlowLayout.CENTER (canh giữa), FlowLayout.TRAILING (canh lề phải)).

FlowLayout(int align, int hgap, int vgap): bố cục có canh lề và khoảng trống giữa các đối tượng.

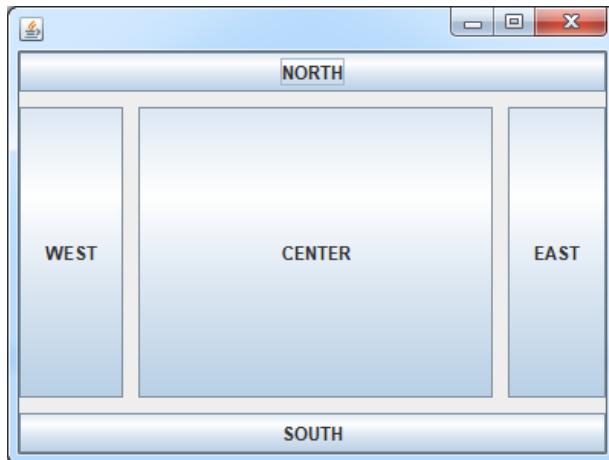


```
1 import java.awt.Container;
2 import java.awt.FlowLayout;
3 import java.awt.event.ActionEvent;
4 import java.awt.event.ActionListener;
5
6 import javax.swing.JButton;
7 import javax.swing.JFrame;
8
9 public class FlowLayoutExample {
10     public FlowLayoutExample() {
11         JFrame jFrame = new JFrame("FlowLayout Example");
```

```
12     FlowLayout layout = new FlowLayout();
13     Container container = jFrame.getContentPane();
14     JButton btnLeft = new JButton("Left");
15     JButton btnCenter = new JButton("Center");
16     JButton btnRight = new JButton("Right");
17     container.add(btnLeft);
18     container.add(btnCenter);
19     container.add(btnRight);
20     jFrame.setLayout(layout);
21
22     btnLeft.addActionListener(new ActionListener() {
23         @Override
24         public void actionPerformed(ActionEvent e) {
25             layout.setAlignment(FlowLayout.LEFT);
26             layout.layoutContainer(container);
27         }
28     });
29
30     btnCenter.addActionListener(new ActionListener() {
31         @Override
32         public void actionPerformed(ActionEvent e) {
33             layout.setAlignment(FlowLayout.CENTER);
34             layout.layoutContainer(container);
35         }
36     });
37
38     btnRight.addActionListener(new ActionListener() {
39         @Override
40         public void actionPerformed(ActionEvent e) {
41             layout.setAlignment(FlowLayout.RIGHT);
42             layout.layoutContainer(container);
43         }
44     });
45
46     jFrame.setSize(400, 100);
47     jFrame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
48     jFrame.setVisible(true);
49 }
50
51 public static void main(String[] args) {
```

```
52     new FlowLayoutExample();  
53 }  
54 }
```

6.1.3.2. BorderLayout



BorderLayout sắp xếp các thành phần với bối cảnh 5 vùng: đông, tây, nam, bắc và trung tâm.

BorderLayout là layout mặc định của Frame. Mỗi khu vực chỉ có thể chứa một thành phần và mỗi thành phần trong mỗi khu vực được nhận diện bởi các hằng số tương ứng là:

```
public static final int NORTH
```

```
public static final int SOUTH
```

```
public static final int EAST
```

```
public static final int WEST
```

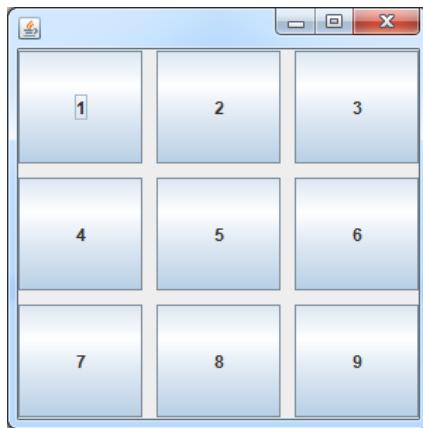
```
public static final int CENTER
```

```
1 Public void BorderLayoutDemo() {  
2     JFrame f = new JFrame();  
3  
4     BorderLayout blayout = new BorderLayout();  
5     blayout.setHgap(10);  
6     blayout.setVgap(10);  
7     f.setLayout(blayout);  
8  
9     JButton b1 = new JButton("NORTH");  
10    JButton b2 = new JButton("SOUTH");  
11    JButton b3 = new JButton("EAST");
```

```

12     JButton b4 = new JButton("WEST");
13     JButton b5 = new JButton("CENTER");
14
15     f.add(b1, BorderLayout.NORTH);
16     f.add(b2, BorderLayout.SOUTH);
17     f.add(b3, BorderLayout.EAST);
18     f.add(b4, BorderLayout.WEST);
19     f.add(b5, BorderLayout.CENTER);
20
21     f.setSize(400, 300);
22     f.setVisible(true);
23 }
```

6.1.3.3. GridLayout



GridLayout chia container thành một lưới gồm các dòng và các cột xác định. Các thành phần được thêm theo thứ tự từ trái sang phải và từ trên xuống dưới.

Một số constructor và phương thức của lớp GridLayout:

GridLayout(): Tạo Layout mặc định là mỗi cột một thành phần, trong một hàng đơn.

GridLayout(int rows, int columns): Layout với số dòng và cột đã cho, không có khoảng cách giữa các thành phần.

GridLayout(int rows, int columns, int hgap, int vgap): Layout với số dòng và cột đã cho, có khoảng cách giữa các thành phần theo chiều dọc và ngang đã xác định.

void addLayoutComponent(String name, Component comp): Thêm thành phần ‘comp’ với tên ‘name’ vào layout.

void setColumns(int cols): Thiết lập số cột trong layout

void setRows(int rows): Thiết lập số dòng trong layout

void setHgap(int hgap): Thiết lập khoảng cách theo chiều ngang giữa các thành phần

void setVgap(int vgap): Thiết lập khoảng cách theo chiều dọc giữa các thành phần

```
1 public void GridLayoutDemo() {  
2     JFrame f = new JFrame();  
3  
4     GridLayout glayout = new GridLayout(3, 3);  
5     glayout.setHgap(10);  
6     glayout.setVgap(10);  
7     f.setLayout(glayout);  
8  
9     JButton b1 = new JButton("1");  
10    JButton b2 = new JButton("2");  
11    JButton b3 = new JButton("3");  
12    JButton b4 = new JButton("4");  
13    JButton b5 = new JButton("5");  
14    JButton b6 = new JButton("6");  
15    JButton b7 = new JButton("7");  
16    JButton b8 = new JButton("8");  
17    JButton b9 = new JButton("9");  
18  
19    f.add(b1);  
20    f.add(b2);  
21    f.add(b3);  
22    f.add(b4);  
23    f.add(b5);  
24    f.add(b6);  
25    f.add(b7);  
26    f.add(b8);  
27    f.add(b9);  
28    f.setSize(300, 300);  
29    f.setVisible(true);  
30 }
```

6.2. Xử lý sự kiện

Một sự kiện xảy ra khi có sự tác động trong GUI, như khi click vào button, click vào combobox,... Các thành phần đồ họa (components) gây ra các sự kiện này được gọi là

nguồn sự kiện (event source). Các sự kiện được xử lý bởi một trình lắng nghe sự kiện (event listener). Các thành phần khác nhau của GUI sẽ có các event listener khác nhau.

Ba cách gắn trình lắng nghe sự kiện:

- Sử dụng Anonymous Inner Class: là dùng lớp vô danh, thường dùng cho một component.
- Sử dụng Inner Class Listener có tên: phù hợp với các chương trình nhỏ. Nhiều component có thể dùng chung Inner Class này.
- Sử dụng Top-level Listeners (this): thường được sử dụng khi có một event source

Một số sự kiện trong giao diện GUI:

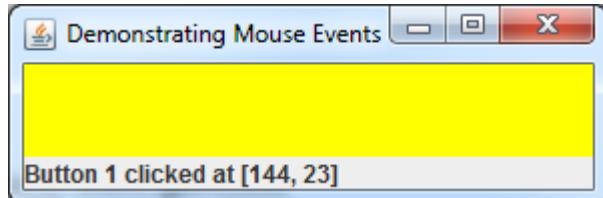
Event Object	Phương thức thêm và xóa trình lắng nghe sự kiện	Interface Method	Event Source
ActionEvent	addActionListener removeActionListener	Interface ActionListener actionPerformed(ActionEvent)	JButton, JCheckBox, JComboBox, JTextField, JRadioButton
ItemEvent	addItemListener removeItemListener	Interface ItemListener itemStateChanged(ItemEvent)	JCheckBox, JComboBox, JRadioButton
MouseEvent	addMouseListener removeMouseListener	Interface MouseListener mousePressed(MouseEvent) mouseReleased(MouseEvent) mouseEntered(MouseEvent) mouseExited(MouseEvent) mouseClicked(MouseEvent)	Bất kỳ thành phần của GUI
KeyEvent	addKeyListener	Interface KeyListener keyTyped(KeyEvent) keyPressed(KeyEvent) keyReleased(KeyEvent)	JTextField

Ví dụ:

```
button1.addActionListener(new ActionListener() {
    @Override
```

```
public void actionPerformed(ActionEvent arg0) {  
    textField1.setText("Hello World!");  
}  
});
```

Mouse click:



```
import java.awt.Color;  
import java.awt.BorderLayout;  
import java.awt.event.MouseListener;  
import java.awt.event.MouseMotionListener;  
import java.awt.event.MouseEvent;  
import javax.swing.JFrame;  
import javax.swing.JLabel;  
import javax.swing.JPanel;  
  
public class MouseEventDemo extends JFrame {  
    private JPanel mousePanel;  
    private JLabel statusBar;  
  
    public MouseEventDemo() {  
        super("Demonstrating Mouse Events");  
        mousePanel = new JPanel();  
        mousePanel.setBackground(Color.WHITE);  
        add(mousePanel, BorderLayout.CENTER);  
        statusBar = new JLabel("Mouse outside JPanel");  
        add(statusBar, BorderLayout.SOUTH);  
        // create and register listener for mouse and mouse motion events  
        MouseHandler handler = new MouseHandler();  
        mousePanel.addMouseListener(handler);  
        mousePanel.addMouseMotionListener(handler);  
    }  
  
    private class MouseHandler implements MouseListener, MouseMotionListener {
```

```
public void mouseClicked(MouseEvent event) {
    String buttonClick = "";
    if (event.getButton() == MouseEvent.NOBUTTON) {
        buttonClick = "No button clicked";
    } else if (event.getButton() == MouseEvent.BUTTON1) {
        buttonClick = "Button 1 clicked";
    } else if (event.getButton() == MouseEvent.BUTTON2) {
        buttonClick = "Button 2 clicked";
    } else if (event.getButton() == MouseEvent.BUTTON3) {
        buttonClick = "Button 3 clicked";
    }
    statusBar.setText(String.format(buttonClick + " at [%d, %d]",
                                    event.getX(), event.getY()));
}

public void mousePressed(MouseEvent event) {
    statusBar.setText(String.format("Pressed at [%d, %d]", event.getX(),
                                    event.getY()));
}

public void mouseReleased(MouseEvent event) {
    statusBar.setText(String.format("Released at [%d, %d]", event.getX(),
                                    event.getY()));
}

public void mouseEntered(MouseEvent event) {
    statusBar.setText(String.format("Mouse entered at [%d, %d]", event.getX(),
                                    event.getY()));
    mousePanel.setBackground(Color.YELLOW);
}

public void mouseExited(MouseEvent event) {
    statusBar.setText("Mouse outside JPanel");
    mousePanel.setBackground(Color.WHITE);
}

public void mouseDragged(MouseEvent event) {
```

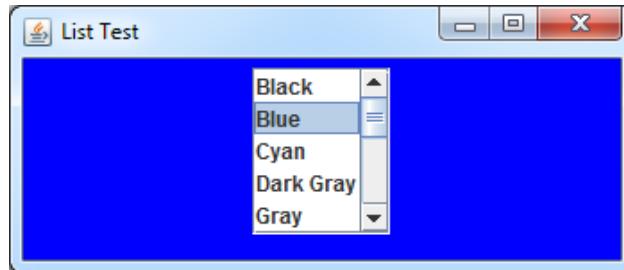
```
        statusBar.setText(String.format("Dragged at [%d, %d]", event.getX(),
                                         event.getY()));
    }

    public void mouseMoved(MouseEvent event) {
        statusBar.setText(String.format("Moved at [%d, %d]", event.getX(),
                                         event.getY()));
    }

}

public static void main(String[] args) {
    MouseEventDemo mouseTrackerFrame = new MouseEventDemo();
    mouseTrackerFrame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    mouseTrackerFrame.setSize(300, 100);
    mouseTrackerFrame.setVisible(true);
}
```

JList



```
import java.awt.FlowLayout;
import java.awt.Color;
import javax.swing.JFrame;
import javax.swing.JList;
import javax.swing.JScrollPane;
import javax.swing.event.ListSelectionListener;
import javax.swing.event.ListSelectionEvent;
import javax.swing.ListSelectionModel;

public class ListFrame extends JFrame {

    private static final String[] colorNames = { "Black", "Blue", "Cyan",
                                                "Dark Gray", "Gray", "Green", "Light Gray", "Magenta", "Orange", "Pink",
```

```

        "Red", "White", "Yellow" };

private static final Color[] colors = { Color.BLACK, Color.BLUE, Color.CYAN,
    Color.DARK_GRAY, Color.GRAY, Color.GREEN, Color.LIGHT_GRAY, Color.MAGENTA,
    Color.ORANGE, Color.PINK, Color.RED, Color.WHITE, Color.YELLOW };

private JList colorJList;

public ListFrame() {
    super("List Test");
    setLayout(new FlowLayout());
    colorJList = new JList(colorNames);
    colorJList.setVisibleRowCount(5);
    colorJList.setSelectionMode(ListSelectionModel.SINGLE_SELECTION);
    add(new JScrollPane(colorJList));
    colorJList.addListSelectionListener(new ListSelectionListener() {
        public void valueChanged(ListSelectionEvent event) {
            getContentPane().setBackground(colors[colorJList.getSelectedIndex()]);
        }
    });
}

public static void main(String[] args) {
    ListFrame listFrame = new ListFrame();
    listFrame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    listFrame.setSize(350, 150);
    listFrame.setVisible(true);
}
}

```

6.3. Graphics 2D

6.3.1. Font chữ

Font(String name, int style, int size)

Font("Tahoma", Font.PLAIN, 14)

Style: Font.BOLD, Font.ITALIC, Font.PLAIN

Font("Garamond", Font.BOLD | Font.ITALIC, 12) //bold italic

Thiết lập font cho component:

```
JLabel textLabel = new JLabel("Let it go!");  
Font f = new Font("Tahoma", Font.PLAIN, 16);  
textLabel.setFont(f);  
// textLabel.setFont(new Font("Tahoma", Font.PLAIN, 16));
```

```
JPanel panel1 = new JPanel();  
panel1.setFont(new Font("Tahoma", Font.PLAIN, 16));  
JButton b1 = new JButton("Agree");  
b1.setFont(null); //same parent font  
panel1.add(b1);  
JButton b2 = new JButton("Cancel");  
b2.setFont(null);  
panel1.add(b2);
```

Lấy danh sách Font có sẵn trên máy:

```
GraphicsEnvironment g;  
g = GraphicsEnvironment.getLocalGraphicsEnvironment();  
String[] fonts;  
fonts = g.getAvailableFontFamilyNames();  
JComboBox fontComboBox = new JComboBox(fonts);  
String name = (String) fontComboBox.getSelectedItem();  
Font f = new Font(name, Font.PLAIN, 12);
```

6.3.2. Color

Sử dụng lớp Color để đổi màu cho các đối tượng trong giao diện GUI

Color (red, green, blue), các màu có giá trị từ 0 đến 255

```
Color c = new Color(255, 255, 0);
```

```
Color c = Color.RED;//BLACK, GRAY, MAGENTA, ORANGE, PINK...
```

```
Color c = new Color(255, 0, 0, 128); //tham số thứ tư là alpha: color transparency -  
độ trong suốt của màu.
```

Một số màu hệ thống - System Color: info, infoText, menu, menuText, textHighlight, textHighlight, window, windowBorder, windowText

```
button1.setBackground(SystemColor.windowBorder);
```

```
JLabel errorMessage = new JLabel("Be yourself!");
```

```
errorMessage.setForeground(Color.RED);
```

Ví dụ về màu chữ, font, size



```

1 import javax.swing.*;
2 import java.awt.event.*;
3 import java.awt.*;
4
5 public class Fonts extends JFrame implements ActionListener {
6
7     private JLabel sampleText;
8     private JComboBox fontComboBox;
9     private JComboBox sizeComboBox;
10    private JCheckBox boldCheck, italicCheck;

```

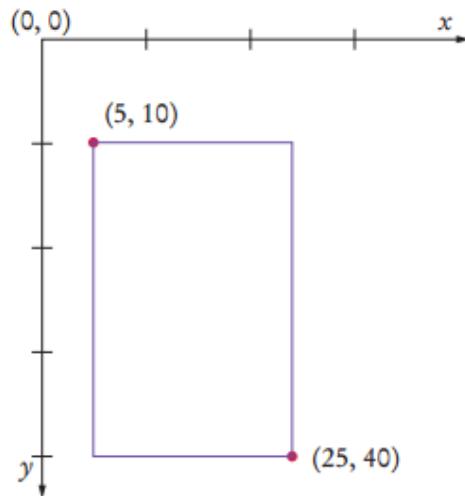
```
11  private String[] fonts;
12
13  private JButton btnChooseColor;
14  private Color color;
15
16  public Fonts() {
17      this.setSize(600, 150);
18      this.setTitle("Demo Using Fonts ");
19      this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
20
21      sampleText = new JLabel("Practice makes perfect!");
22      this.add(sampleText, BorderLayout.CENTER);
23
24      GraphicsEnvironment g;
25      g = GraphicsEnvironment.getLocalGraphicsEnvironment();
26      fonts = g.getAvailableFontFamilyNames();
27
28      JPanel controlPanel = new JPanel();
29
30      fontComboBox = new JComboBox<String>(fonts);
31      fontComboBox.addActionListener(this);
32      controlPanel.add(new JLabel("Family: "));
33      controlPanel.add(fontComboBox);
34
35      Integer[] sizes = { 7, 8, 9, 10, 11, 12, 14, 18, 20, 22, 24, 36 };
36      sizeComboBox = new JComboBox<Integer>(sizes);
37      sizeComboBox.setSelectedIndex(5);
38      sizeComboBox.addActionListener(this);
39      controlPanel.add(new JLabel("Size: "));
40      controlPanel.add(sizeComboBox);
41
42      boldCheck = new JCheckBox("Bold");
43      boldCheck.addActionListener(this);
44      controlPanel.add(boldCheck);
45
46      italicCheck = new JCheckBox("Italic");
47      italicCheck.addActionListener(this);
48      controlPanel.add(italicCheck);
49
50      btnChooseColor = new JButton("Choose Color");
51      btnChooseColor.addActionListener(new ActionListener() {
52          @Override
```

```
53     public void actionPerformed(ActionEvent e) {
54         color = JColorChooser.showDialog(null, "Choose a Color",
55             sampleText.getForeground());
56         updateText();
57     }
58 });
59 controlPanel.add(btnChooseColor);
60
61 updateText();
62 this.add(controlPanel, BorderLayout.SOUTH);
63 this.setVisible(true);
64 }
65
66 public void updateText() {
67     String name = (String) fontComboBox.getSelectedItem();
68     Integer size = (Integer) sizeComboBox.getSelectedItem();
69     int style;
70     if (boldCheck.isSelected() && italicCheck.isSelected())
71         style = Font.BOLD | Font.ITALIC;
72     else if (boldCheck.isSelected())
73         style = Font.BOLD;
74     else if (italicCheck.isSelected())
75         style = Font.ITALIC;
76     else
77         style = Font.PLAIN;
78     Font f = new Font(name, style, size.intValue());
79     sampleText.setFont(f);
80
81     if (color != null)
82         sampleText.setForeground(color);
83 }
84
85 @Override
86 public void actionPerformed(ActionEvent e) {
87     updateText();
88 }
89
90 public static void main(String[] args) {
91     new Fonts();
92 }
```

93 }

6.3.3. Các hình vẽ cơ bản

Hệ tọa độ 2D trong GUI:



Rectangle rect1 = new Rectangle(5, 10, 20, 30);

Ví dụ:

```
1 public class RectangleComponent extends JPanel {
2     public void paintComponent(Graphics g) {
3         Graphics2D g2 = (Graphics2D) g;
4
5         Rectangle box = new Rectangle(50, 30, 100, 120);
6         g2.setStroke(new BasicStroke(4)); // độ dày đường viền
7         g2.setColor(Color.BLUE);
8         g2.draw(box);
9         g2.setColor(Color.MAGENTA);
10        g2.fill(box);
11    }
12 }
```

```
1     public static void main(String[] args) {
```

```
2     JFrame frame = new JFrame();
3
4     frame.setSize(300, 400);
5     frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
6
7     RectangleComponent component = new RectangleComponent();
8     frame.add(component);
9
10    frame.setVisible(true);
11 }
```

Để khử răng cưa của hình:

```
g2.setRenderingHint(RenderingHints.KEY_ANTIALIASING,
RenderingHints.VALUE_ANTIALIAS_ON);
```

Một số hình thông dụng:

Arc2D.Float(float x, float y, float w, float h, float start, float extent, int type)

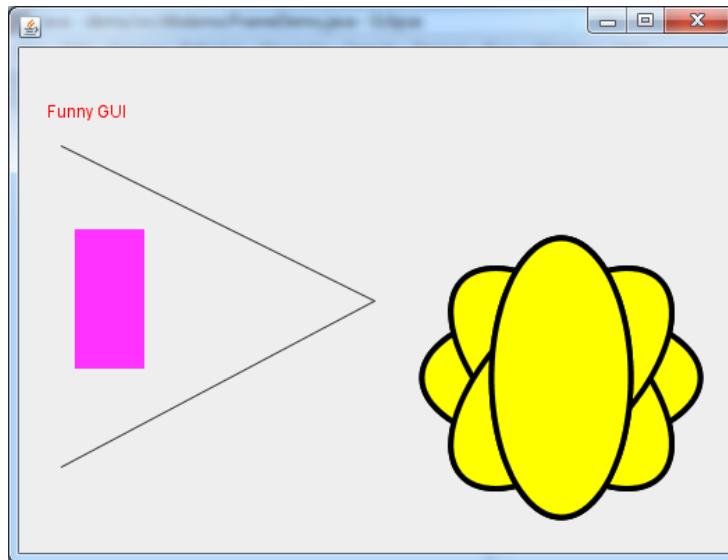
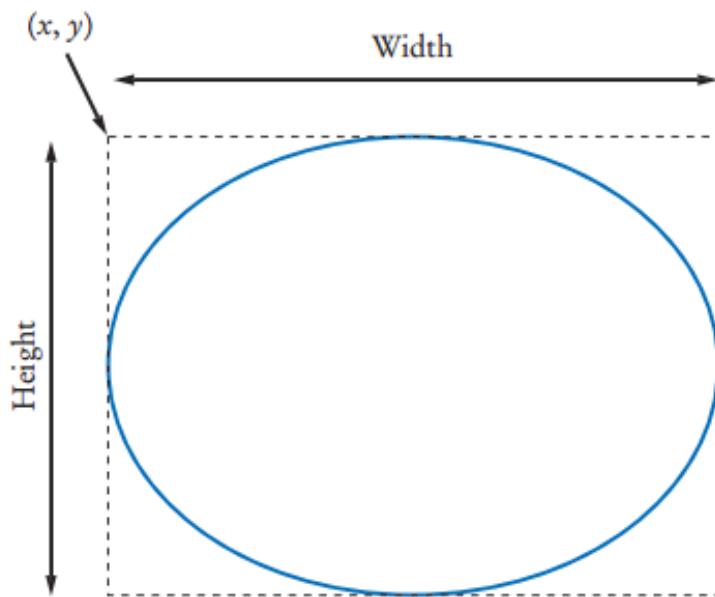
Ellipse2D.Float(float x, float y, float w, float h)

Line2D.Float(float x1, float y1, float x2, float y2)

Line2D.Float(Point2D p1, Point2D p2)

Rectangle2D.Float(float x, float y, float w, float h)

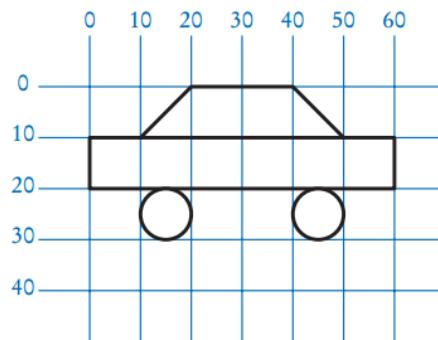
RoundRectangle2D.Float(float x, float y, float w, float h, float arcw, float arch)



```
1 public class DemoGUIComponents extends JComponent {  
2     public void paintComponent(Graphics g) {  
3         Graphics2D g2 = (Graphics2D) g;  
4         g2.setRenderingHint(RenderingHints.KEY_ANTIALIASING,  
5             RenderingHints.VALUE_ANTIALIAS_ON); // Khử răng cưa  
6  
7         int x1 = 30, y1 = 70;  
8         int cx = getSize().width / 2; // center X;  
9         int cy = getSize().height / 2; // center Y;  
10  
11         Point2D.Double from = new Point2D.Double(x1, y1);
```

```
12     Point2D.Double to = new Point2D.Double(cx, cy);
13     Line2D.Double segment = new Line2D.Double(from, to);
14
15     g2.draw(segment);
16     g2.drawLine(cx, cy, x1, 300);
17
18     g2.setColor(Color.RED);
19     g2.drawString("Funny GUI", 20, 50); // basepoint
20
21     Color myPink = new Color(255, 50, 255);
22     g2.setColor(myPink);
23     Rectangle rect1 = new Rectangle(40, 130, 50, 100);
24     g2.fill(rect1);
25
26     g2.translate(400, 10);
27     g2.rotate(Math.toRadians(45));
28     g2.rotate(Math.toRadians(45), 100, 150); // tọa độ điểm xoay quanh
29
30     int x = 50;
31     int y = 75;
32     int width = 200;
33     int height = 100;
34     Shape e1 = new Ellipse2D.Float(x, y, width, height);
35     for (int angle = 0; angle <= 360; angle += 45) {
36         g2.rotate(Math.toRadians(angle), x + width / 2, y + height / 2);
37         g2.setPaint(Color.YELLOW);
38         g2.fill(e1);
39         g2.setStroke(new BasicStroke(4));
40         g2.setPaint(Color.BLACK);
41         g2.draw(e1);
42     }
43 }
```

Ví dụ vẽ xe



```
1  /**
2   * A car shape that can be positioned anywhere on the screen.
3   */
4  public class MyCar {
5      private int xLeft;
6      private int yTop;
7
8      /**
9       * Constructs a car with a given top left corner.
10      *
11      * @param x
12      *          the xcoordinate of the topleft corner
13      * @param y
14      *          the ycoordinate of the topleft corner
15      */
16     public MyCar(int x, int y) {
17         xLeft = x;
18         yTop = y;
19     }
20
21     /**
22      * Draws the car.
23      *
24      * @param g2
25      *          the graphics context
26      */
27     public void draw(Graphics2D g2) {
28         Rectangle body = new Rectangle(xLeft, yTop + 10, 60, 10);
29         Ellipse2D.Double frontTire = new Ellipse2D.Double(xLeft + 10, yTop + 20,
30                  10, 10);
31         Ellipse2D.Double rearTire = new Ellipse2D.Double(xLeft + 40, yTop + 20, 10,
```

```

32         10);
33
34     // The bottom of the front windshield
35     Point2D.Double r1 = new Point2D.Double(xLeft + 10, yTop + 10);
36     // The front of the roof
37     Point2D.Double r2 = new Point2D.Double(xLeft + 20, yTop);
38     // The rear of the roof
39     Point2D.Double r3 = new Point2D.Double(xLeft + 40, yTop);
40     // The bottom of the rear windshield
41     Point2D.Double r4 = new Point2D.Double(xLeft + 50, yTop + 10);
42
43     Line2D.Double frontWindshield = new Line2D.Double(r1, r2);
44     Line2D.Double roofTop = new Line2D.Double(r2, r3);
45     Line2D.Double rearWindshield = new Line2D.Double(r3, r4);
46
47     g2.draw(body);
48     g2.draw(frontTire);
49     g2.draw(rearTire);
50     g2.draw(frontWindshield);
51     g2.draw(roofTop);
52     g2.draw(rearWindshield);
53 }
54 }
```

```

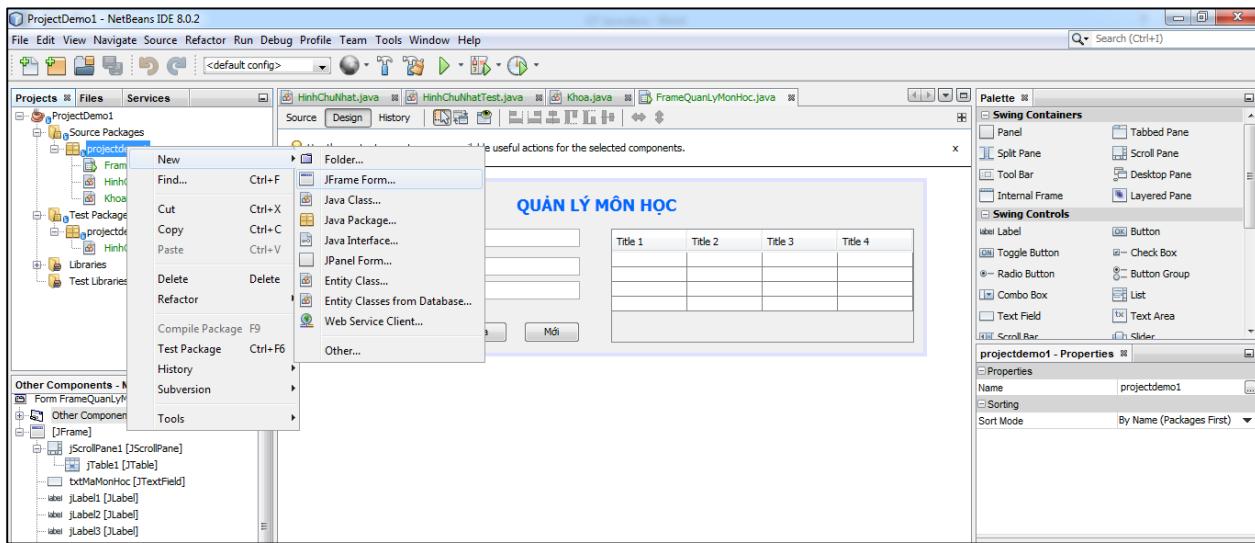
1 public class CarComponent extends JComponent {
2     public void paintComponent(Graphics g) {
3         Graphics2D g2 = (Graphics2D) g;
4
5         MyCar car1 = new MyCar(0, 0);
6
7         int x = getWidth() - 60;
8         int y = getHeight() - 30;
9         MyCar car2 = new MyCar(x, y);
10        car1.draw(g2);
11        car2.draw(g2);
12    }
13 }
```

Case Study Quản Lý Sinh Viên

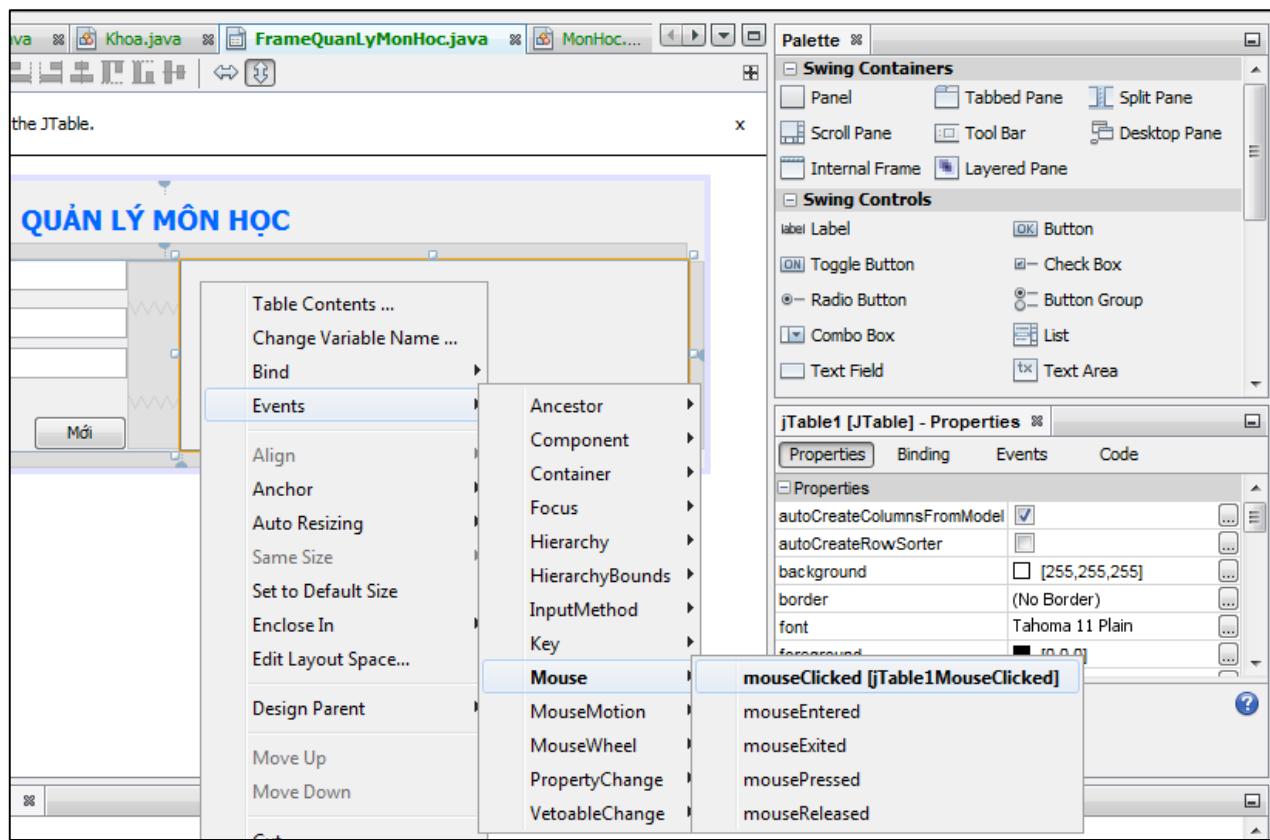
Tạo Giao diện đồ họa cho ứng dụng Quản lý sinh viên: Quản lý môn học, Quản lý Khoa, Quản lý lớp cố định, quản lý lớp học phần, Quản lý đăng ký môn học

Ví dụ: Form Quản lý Môn học

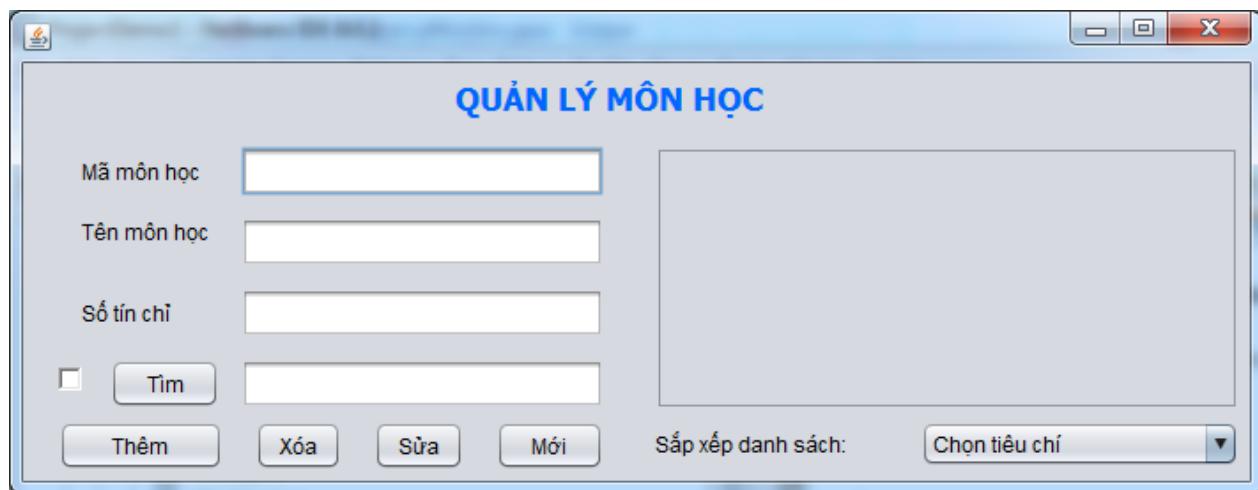
Chọn New -> JFrame Form -> Kéo thả các thành phần để thiết kế Form.



Thêm sự kiện khi click chọn dòng trong Jtable: Click phải chuột trên Table chọn Events -> Mouse -> mouseClicked -> Viết code xử lý

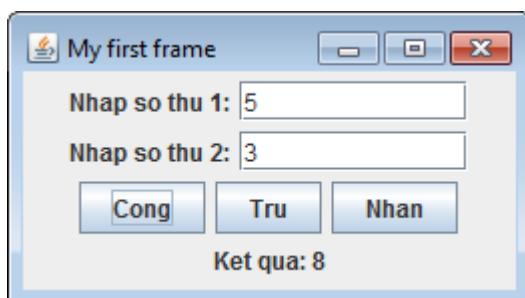


Frame Quản lý Môn học:



Bài tập

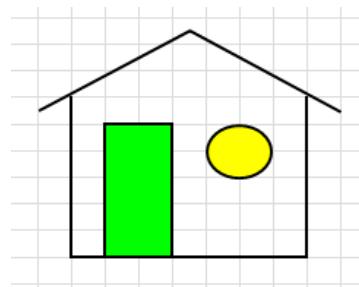
Bài 1. Tạo Frame tính toán sau



Bài 2. Tạo Frame máy tính như sau



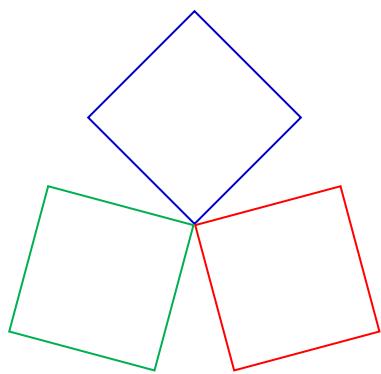
Bài 3. Vẽ nhà



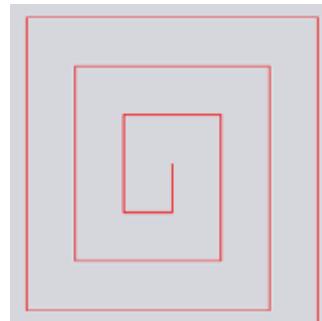
Bài 4. Vẽ cầu vồng



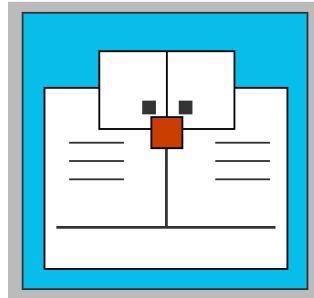
Bài 5. Vẽ các hình thoi hoặc hình vuông như hình



Bài 6. Vẽ xoắn ốc hình vuông



Bài 7. Vẽ Doreamon vuông, thay đổi sang dạng tròn



CHƯƠNG 7. COLLECTION

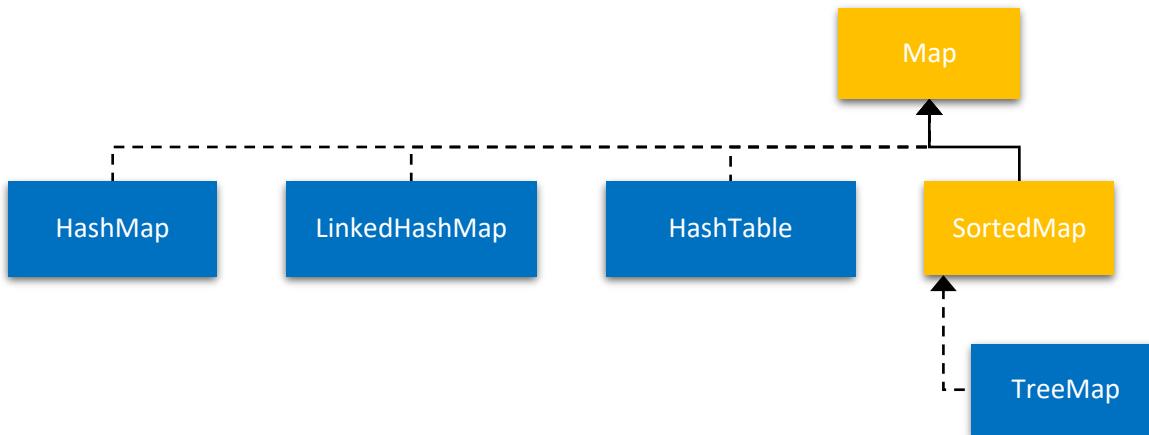
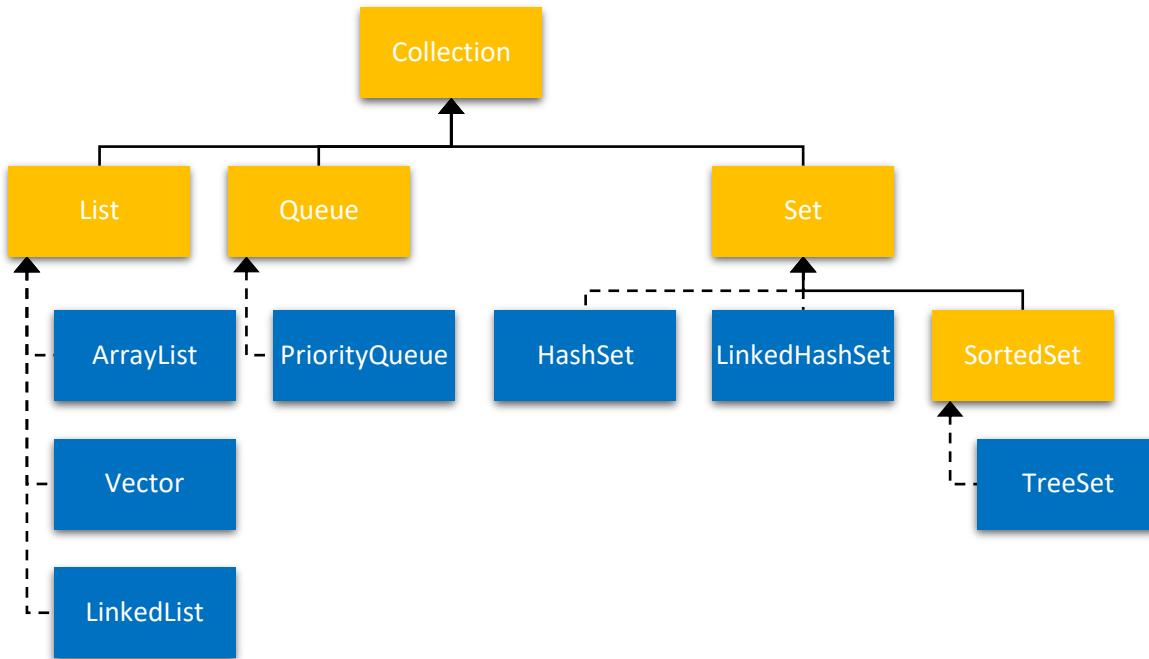
Học xong chương này, người học có thể:

- + Sử dụng các lớp trong hệ thống Collection Java: List, Set, Map
- + Hiện thực interface Comparable và interface Comparator để so sánh đối tượng
- + Sử dụng các phương thức của lớp Collections: sort(), search(), insert(), remove()

7.1. Interface Collection

Lợi ích của Collection Framework: Giảm thời gian lập trình, tăng hiệu năng chương trình, có khả năng tái sử dụng.

Collection<E>, List<E>, Stack<E>, Set<E>, Map<K, V>



Một số phương thức của Collection

boolean isEmpty()

int size()

boolean contain(Object obj)

boolean add(Object obj)

boolean remove(Object obj)

Iterator<E> iterator()

Một số phương thức của Iterator: boolean hasNext(), E next(), void remove()

Duyệt collection bằng foreach:

```
Collection<String> words = new ArrayList<String>();
// ... gán giá trị cho ArrayList
for (String word : words) {
    // ... process word
}
```

Duyệt collection bằng iterator:

```
Collection<String> words = new ArrayList<String>();
// ... gán giá trị cho ArrayList
Iterator<String> iter = words.iterator();
while (iter.hasNext()) {
    String word = iter.next();
    // ... process word
}
```

7.1.1. ArrayList class

Một số phương thức của lớp ArrayList

Phương thức	Mô tả
boolean add(Object o)	thêm phần tử được chỉ định vào cuối một danh sách.
void add(int index, Object element)	chèn các phần tử được chỉ định tại các chỉ số vị trí quy định trong một danh sách.
boolean addAll(Collection c)	thêm tất cả các phần tử trong collection được chỉ định vào cuối của danh sách này
boolean addAll(int index, Collection c)	chèn tất cả các phần tử trong collection được chỉ định vào danh sách này, bắt đầu từ vị trí đã chỉ định.
void retainAll(Collection c)	xóa những phần tử không thuộc collection c và không thuộc list hiện tại khỏi list hiện tại.

void removeAll(Collection c)	xóa những phần tử thuộc collection c và thuộc list hiện tại khỏi list hiện tại.
int indexOf(Object o)	trả về chỉ mục trong danh sách với sự xuất hiện đầu tiên của phần tử được chỉ định, hoặc -1 nếu danh sách không chứa phần tử này.
int lastIndexOf(Object o)	trả về chỉ mục trong danh sách với sự xuất hiện cuối cùng của phần tử được chỉ định, hoặc -1 nếu danh sách không chứa phần tử này.
Object[] toArray()	trả về một mảng chứa tất cả các phần tử trong danh sách này theo đúng thứ tự.

7.1.2. HashSet class

HashSet chứa các đối tượng duy nhất, không trùng nhau.

Một số phương thức:

boolean add(E e): Thêm phần tử nếu chưa có trong HashSet

void clear(): Xóa tất cả các phần tử

boolean contains(Object o): Kiểm tra có chứa phần tử o không

boolean remove(Object o): Xóa phần tử o ra khỏi HashSet

int size(): Trả về kích thước của HashSet

Ví dụ:

```
HashSet<String> hset = new HashSet<String>();

hset.add("Apple");
hset.add("Mango");
hset.add("Grapes");
hset.add("Orange");
hset.add("Banana");
//Addition of duplicate elements
hset.add("Apple");
hset.add("Mango");
//Addition of null values
hset.add(null);
```

```
hset.add(null);  
  
//Displaying HashSet elements  
System.out.println(hset);
```

```
[null, Apple, Grapes, Mango, Orange, Banana]
```

7.1.3. HashMap class

Map<K, V>

Một số phương thức:

boolean isEmpty() : kiểm tra xem map có rỗng không

int size() : lấy kích thước map

V get(K key) : lấy value khi biết key

V put(K key, V value) : thêm cặp key – value vào map

V remove(K key) : xóa phần tử khi biết key

boolean containKey(Object key) : kiểm tra xem có chứa khóa key không

Set<K> keySet() : lấy tập hợp khóa key

7.2. Class Collections

7.2.1. Phương thức Sort: Sử dụng interface Comparable, Comparator

Java cung cấp 2 Interface **Comparable** và **Comparator** để so sánh và sắp xếp vị trí 2 đối tượng của một Collection. Có thể thực hiện bất kỳ kiểu sắp xếp nào, chỉ cần thực hiện logic so sánh hai đối tượng.

Sử dụng Comparable:

```
1 public class Student implements Comparable<Student> {  
2     private int id;  
3     private String name;  
4     private int age;  
5 }
```

```

6  // ... constructor, getter, setter, methods
7
8  public int compareTo(Student st) {
9      if (age == st.age)
10         return 0;
11     else if (age > st.age)
12         return 1;
13     else
14         return -1;
15 }
16 }
```

```

1  public static void main(String args[]) {
2      ArrayList<Student> listOfDayStudent = new ArrayList<Student>();
3      listOfDayStudent.add(new Student(111, "An", 19));
4      listOfDayStudent.add(new Student(118, "Binh", 18));
5      listOfDayStudent.add(new Student(114, "Phuc", 22));
6
7      Collections.sort(listOfDayStudent);
8      for (Student st : listOfDayStudent) {
9          System.out.println(st.getId() + " " + st.getName() + " " + st.getAge());
10     }
11 }
```

```

118 Binh 18
111 An 19
114 Phuc 22
```

Sử dụng Comparator:

```

1  public class Student {
2      private int id;
3      private String name;
4      private int age;
5
6      // ... constructor, getter, setter, methods
7 }
```

```
1 public class AgeComparator implements Comparator {  
2     public int compare(Object o1, Object o2) {  
3         Student s1 = (Student) o1;  
4         Student s2 = (Student) o2;  
5  
6         if (s1.getAge() == s2.getAge())  
7             return 0;  
8         else if (s1.getAge() > s2.getAge())  
9             return 1;  
10        else  
11            return -1;  
12    }  
13 }
```

```
1 public static void main(String args[]) {  
2     ArrayList<Student> listOfStudent = new ArrayList<Student>();  
3     listOfStudent.add(new Student(111, "An", 19));  
4     listOfStudent.add(new Student(118, "Binh", 18));  
5     listOfStudent.add(new Student(114, "Phuc", 22));  
6  
7     System.out.println("Sorting by Age: ");  
8     Collections.sort(listOfStudent, new AgeComparator());  
9     Iterator itr2 = listOfStudent.iterator();  
10    while (itr2.hasNext()) {  
11        Student st = (Student) itr2.next();  
12        System.out.println(st.getId() + " " + st.getName() + " " + st.getAge());  
13    }  
14  
15    System.out.println("\nSorting by Name...");  
16    // use anonymous class  
17    Collections.sort(listOfStudent, new Comparator<Student>() {  
18        public int compare(Student st1, Student st2) {  
19            return st1.getName().compareTo(st2.getName());  
20        }  
21    });  
22  
23    for (Student st : listOfStudent) {  
24        System.out.println(st.getId() + " " + st.getName() + " " + st.getAge());  
25    }  
26}
```

```

25      }
26  }

```

Sorting by Age:

```

118 Binh 18
111 An 19
114 Phuc 22

```

Sorting by Name...

```

111 An 19
118 Binh 18
114 Phuc 22

```

So sánh giữa Comparable và Comparator:

Comparable	Comparator
Chỉ có một trình tự sắp xếp	Có thể có nhiều trình tự sắp xếp
Ảnh hưởng đến class gốc (Phải implement Comparable)	Không ảnh hưởng tới class gốc
Dùng phương thức compareTo() để so sánh với một đối tượng khác	Dùng phương thức compare() để so sánh hai đối tượng
Thuộc gói java.lang (không cần import)	Thuộc gói java.util (phải import)
Sắp xếp sử dụng Collections.sort(List)	Sắp xếp sử dụng Collections.sort(List,Comparator)

7.2.2. Một số phương thức của lớp Collections

min() : lấy phần tử nhỏ nhất

max() : lấy phần tử lớn nhất

search() : tìm kiếm vị trí của phần tử

reverse() : đảo thứ tự trong Collection

frequency() để đếm số lần xuất hiện của phần tử trong Collection

shuffle() để xáo trộn các phần tử trong Collection

Case Study Quản Lý Sinh Viên

Tạo lớp quản lý môn học với các chức năng sau: (làm tương tự cho các lớp quản lý khác)

1. Thêm môn học
 2. Xóa môn học khi biết mã
 3. Chính sửa thông tin môn học
 4. Sắp xếp môn học theo tên
 5. Sắp xếp môn học theo số tín chỉ tăng dần rồi theo tên thứ tự A-Z
 6. Tìm kiếm các môn học có số tín chỉ bằng với số tín chỉ cho trước.
 7. Tìm kiếm các môn học có tên gần đúng tên cho trước.
 8. Hiển thị danh sách môn học
- * Bổ sung các chức năng từ 4 - 8 lên giao diện đồ họa

```
1 import java.util.ArrayList;
2 import java.util.Collections;
3 import java.util.Comparator;
4
5 public class ModelQuanLyMonHoc {
6     private ArrayList<MonHoc> dsMonHoc;
7
8     public ModelQuanLyMonHoc() {
9         dsMonHoc = MonHocDAO.getDanhSachMonHoc();
10    }
11
12    public ArrayList<MonHoc> getDsMonHoc() {
13        return dsMonHoc = MonHocDAO.getDanhSachMonHoc();
14    }
15
16    public void themMonHoc(MonHoc monHoc) throws Exception {
17        MonHocDAO.themMonHoc(monHoc);
18    }
19
20    public void xoaMonHoc(String maMonHoc) throws Exception {
21        MonHocDAO.xoaMonHoc(maMonHoc);
22    }
23
24    public void suaMonHoc(MonHoc monHoc) throws Exception {
25        MonHocDAO.suaMonHoc(monHoc);
```

```
26     }
27
28     // sắp xếp và tìm kiếm có thể dùng câu query truy vấn CSDL
29
30     public ArrayList<MonHoc> sapXepTheoTen() {
31         ArrayList<MonHoc> result = dsMonHoc;
32         Collections.sort(result, new Comparator<MonHoc>() {
33             @Override
34             public int compare(MonHoc mh1, MonHoc mh2) {
35                 return mh1.getTenMonHoc().compareTo(mh2.getTenMonHoc());
36             }
37         });
38         return result;
39     }
40
41     public ArrayList<MonHoc> sapXepTheoSoTinChi() {
42         ArrayList<MonHoc> result = dsMonHoc;
43         Collections.sort(result, new Comparator<MonHoc>() {
44             @Override
45             public int compare(MonHoc mh1, MonHoc mh2) {
46                 return mh1.getSoTinchi() - mh2.getSoTinchi();
47             }
48         });
49         return result;
50     }
51
52     public ArrayList<MonHoc> sapXepTheoSoTinChiRoiTheoTen() {
53         ArrayList<MonHoc> result = dsMonHoc;
54         Collections.sort(result, new Comparator<MonHoc>() {
55             @Override
56             public int compare(MonHoc mh1, MonHoc mh2) {
57                 if (mh1.getSoTinchi() == mh2.getSoTinchi())
58                     return mh1.getTenMonHoc().compareTo(mh2.getTenMonHoc());
59                 return mh1.getSoTinchi() - mh2.getSoTinchi();
60             }
61         });
62         return result;
63     }
64
65     public ArrayList<MonHoc> getDsMonHocTheoMa(String maMonHoc) {
```

```
66     ArrayList<MonHoc> result = new ArrayList<MonHoc>();
67     for (MonHoc monHoc : dsMonHoc) {
68         if (monHoc.getMaMonHoc().contains(maMonHoc))
69             result.add(monHoc);
70     }
71     return result;
72 }
73
74 public ArrayList<MonHoc> getDsMonHocTheoTen(String tenMonHoc) {
75     ArrayList<MonHoc> result = new ArrayList<MonHoc>();
76     for (MonHoc monHoc : dsMonHoc) {
77         if (monHoc.getTenMonHoc().contains(tenMonHoc))
78             result.add(monHoc);
79     }
80     return result;
81 }
82
83 public ArrayList<MonHoc> getDSMonHocTheoTinChi(int soTinChi) {
84     ArrayList<MonHoc> result = new ArrayList<MonHoc>();
85     for (MonHoc monHoc : dsMonHoc) {
86         if (monHoc.getSoTinchis() == soTinChi)
87             result.add(monHoc);
88     }
89     return result;
90 }
91
92 public ArrayList<MonHoc> getDsMonHocCoTuKhoa(String tuKhoa) {
93     ArrayList<MonHoc> result = new ArrayList<MonHoc>();
94     for (MonHoc monHoc : dsMonHoc) {
95         if (monHoc.getMaMonHoc().contains(tuKhoa)
96             || monHoc.getTenMonHoc().contains(tuKhoa)
97             || (monHoc.getSoTinchis() + "").contains(tuKhoa))
98             result.add(monHoc);
99     }
100    return result;
101 }
102
103 public String showDsMonHoc() {
104     String result = "";
105     for (MonHoc monHoc : dsMonHoc) {
106         result += String.format("%-10s%-30s%2d%n", monHoc.getMaMonHoc(),
107             monHoc.getTenMonHoc(), monHoc.getSoTinchis());
```

```
108      }
109      return result;
110     }
111 }
```

Bài tập

Bài 1. Viết chương trình Quản lý Nhân viên bán hàng (mã nhân viên, họ tên, doanh thu).

Sắp xếp giảm dần theo doanh thu.

Bài 2. Tạo danh sách các Circle có bán kính là số ngẫu nhiên (từ 10 đến 50)

a. In ra các diện tích của các Circle.

b. Tính diện tích trung bình của các Circle

c. Sắp xếp danh sách các Circle theo diện tích tăng dần

d. Sắp xếp danh sách các Circle theo diện tích giảm dần

e. Tìm hình tròn có diện tích lớn nhất

f. Tìm hình tròn có diện tích nhỏ nhất

Bài 3. Viết chương trình nhận vào một chuỗi và xuất ra vị trí của các ký tự trong chuỗi đó.

Ví dụ: Hello World

{H=[0], e=[1], l=[2, 3, 8], W=[5], o=[4, 6], r=[7], d=[9]}

Bài 4. Một quân bài trong bộ bài gồm hai thuộc tính loại bài (cơ, rô, chuồn, bích) và thứ tự quân bài (2, 3, 4, 5, 6, 7, 8, 9, 10, J, Q, K, A).

Viết các lớp sau:

- Lớp mô tả quân bài

- Lớp bộ bài có danh sách 52 quân bài không trùng nhau, có phương thức xào bài (dùng shuffle()), in danh sách các quân bài.

Bài 5. Tạo lớp ThiSinh gồm các thông tin: soBaoDanh, hoTen, diemMon1, diemMon2, diemMon3; và có phương thức tính tổng điểm. Tạo lớp QuanLyThiSinh có danh sách các thí sinh; có phương thức thêm, xóa, sửa Thí sinh; có phương thức xuất thông tin của các thí sinh trúng tuyển, thí sinh trúng tuyển là thí sinh có tổng điểm lớn hơn hoặc bằng điểm chuẩn.

Bài 6. Thư viện cần quản lý các thông tin về đầu sách gồm: tên sách, tác giả, số lượng, năm xuất bản. Thư viện cần hiển thị thông tin của các quyển sách có năm xuất bản bằng với năm truyền vào.

Bài 7. Một trường cao đẳng cần quản lý phòng học gồm: phòng học lý thuyết, phòng máy tính và phòng thực hành. Mỗi phòng học đều có mã phòng, dãy nhà, diện tích, số bóng đèn, sức chứa bao nhiêu sv.

Phòng học lý thuyết thì cần quan tâm xem có máy chiếu không.

Phòng máy tính thì cần biết là có bao nhiêu máy tính.

Phòng thực hành thì thêm thông tin chuyên ngành, có thiết bị phòng cháy chữa cháy không
Ngoài ra, cần phải xem xét phòng học có đạt chuẩn không.

- Một phòng học đạt chuẩn nếu: đủ ánh sáng (trung bình 10m² - 1 bóng đèn), và:

- + Phòng lý thuyết: nếu có máy chiếu.
- + Phòng máy tính: nếu trung bình 1.5m² đặt một máy.
- + Phòng thực hành: có thiết bị phòng cháy chữa cháy

Thực hiện cài đặt cho mỗi loại phòng cụ thể trên.

Viết lớp quản lý danh sách phòng học. Dùng một List (ArrayList, LinkedList, Vector) để lưu trữ danh sách phòng học.

- + Tạo constructor khởi tạo danh sách.
- + Viết phương thức thêm một phòng học vào danh sách (thêm thành công nếu không bị trùng mã phòng).
- + Viết phương thức tìm kiếm một phòng học nào đó khi biết mã phòng.
- + Viết phương thức in toàn bộ danh sách các phòng học.
- + Viết các phương thức để in danh sách các phòng học đạt chuẩn.
- + Viết phương thức để sắp xếp danh sách tăng dần theo thuộc tính dãy nhà.
- + Viết phương thức để sắp xếp danh sách giảm dần theo diện tích.
- + Viết phương thức để sắp xếp danh sách tăng dần theo mã phòng rồi giảm dần theo sức chứa.
- + Viết phương thức để cập nhật số máy tính cho một phòng máy tính nào đó khi biết mã phòng.
- + Viết phương thức để xóa một phòng học nào đó khi biết mã phòng. Lưu ý khi test chương trình, khi xóa cần phải xác minh rằng có chắc chắn xóa không?
- + Viết phương thức để in ra tổng số phòng học.
- + Viết các phương thức để in danh sách các phòng máy có 60 máy.

CHƯƠNG 8. THIẾT KẾ HƯỚNG ĐỐI TƯỢNG

Học xong chương này, người học có thể:

- + Sử dụng lớp trừu tượng Abstract class, giao diện Interface;
- + Xác định yêu cầu, phân tích yêu cầu ứng dụng;
- + Sử dụng các công cụ hỗ trợ vẽ sơ đồ lớp cho các yêu cầu phần mềm;
- + Sử dụng sơ đồ lớp cài đặt code cho ứng dụng phần mềm
- + Cài đặt code theo mô hình MVC – Model View Controller

8.1. Lớp trừu tượng Abstract class

Lớp trừu tượng là một dạng lớp đặc biệt, trong đó có phương thức chỉ được khai báo mà không được cài đặt chi tiết. Việc cài đặt chi tiết các phương thức chỉ thực hiện được ở các lớp con cụ thể.

Không thể tạo đối tượng từ lớp trừu tượng. Lớp trừu tượng có thể có hoặc không có phương thức trừu tượng. Nhưng một lớp có chứa ít nhất một phương thức trừu tượng thì lớp đó phải là lớp trừu tượng. Bắt buộc các lớp con kế thừa trực tiếp một lớp trừu tượng phải hiện thực các phương thức trừu tượng hoặc tiếp tục là abstract class.

Ví dụ:

```

1 public abstract class MyShape {
2     protected int x, y;
3
4     protected MyShape(int x, int y) {
5         this.x = x;
6         this.y = y;
7     }
8
9     public abstract double calculateArea();
10
11    public void move(int x2, int y2) {
12        x = x2;
13        y = y2;
14    }
15
16    public String toString() {
17        return "This is a shape";
18    }
19 }
```

```

1 public class MyRectangle extends MyShape {
2     private int width, height;
3 }
```

```
4  public MyRectangle(int x, int y, int width, int height) {  
5      super(x, y);  
6      this.width = width;  
7      this.height = height;  
8  }  
9  
10 @Override  
11 public double calculateArea() {  
12     return width * height;  
13 }  
14  
15 public String toString() {  
16     return "This is a Rectangle";  
17 }  
18 }
```

8.2. Giao tiếp Interface

Trong interface, các biến là final và các phương thức là abstract. Khi một class implements một interface thì phải override tất cả các phương thức của interface đó. Một class có thể hiện thực nhiều interface.

Thuộc tính hay phương thức của giao tiếp luôn có phạm vi truy xuất là public. Đối với thuộc tính, thì luôn phải là hằng (final) và tĩnh (static).

Một số interface của Java API: Comparable, Serializable, Runnable, GUI event-listener.

Demo Interface:

```
1 public interface Drawable {  
2     public void draw(Graphics2D g2d);  
3 }
```

```
1 public class MyRectangle extends MyShape implements Drawable {  
2     ...  
3  
4     @Override  
5     public void draw(Graphics2D g2d) {  
6         Rectangle box = new Rectangle(x, y, width, height);
```

```

7     g2d.draw(box);
8     g2d.drawLine(x, y, x + width, y + height);
9     g2d.drawLine(x, y + height, x + width, y);
10    }
11 }

```

8.3. Thiết kế ứng dụng

8.3.1. Sử dụng UML

Ngôn ngữ mô hình hóa thống nhất (UML - Unified Modeling Language)

UML có nhiều loại: Sơ đồ lớp (Class Diagram), Sơ đồ đối tượng (Object Diagram), Sơ đồ tình huống sử dụng (Use Cases Diagram), Sơ đồ trình tự (Sequence Diagram), Sơ đồ cộng tác (Collaboration Diagram), Sơ đồ hoạt động (Activity Diagram).

Phần này giới hạn dùng Sơ đồ lớp Class Diagram để thiết kế chương trình.

Object-Oriented Analysis and Design (OOAD): Phân tích thiết kế hướng đối tượng sử dụng UML

Relationship thể hiện mối quan hệ giữa các Class với nhau. Một số ký hiệu mối quan hệ thường dùng:

Mối quan hệ	Ký hiệu
Ké thừa (is-a) extends (Generalization)	————→
Hiện thực interface implements	-----→
Tập hợp, có (has-a)	◇————
Thành phần (Composition)	◆————
Phụ thuộc (Dependency)	----->

Quan hệ Composition:

```
public class A {
```

```
private B b;  
}
```

Quan hệ Dependency:

```
public class A {  
    public void doSomething(B b){  
        //...  
    }  
}
```

8.3.2. Ứng dụng Quản lý hóa đơn

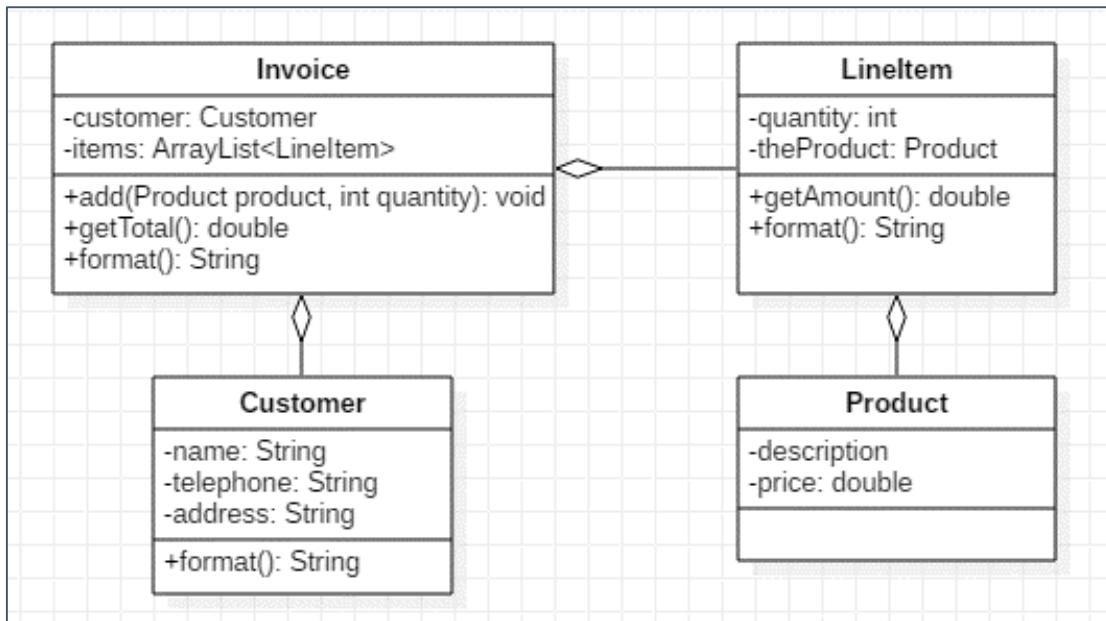
- Xác định lớp và phương thức dựa trên yêu cầu

I N V O I C E				
Hong My 0987654321 123 Vo Van Ngan, Thu Duc, Tp.HCM				

Description Price Qty Amount				
Notebook 200 pages		30,000	4	120,000
Ruler		5,000	1	5,000
Ballpen		2,500	3	7,500

Total: 132,500 VND				

- Vẽ sơ đồ lớp UML thể hiện mối quan hệ giữa các lớp



3. Hiện thực các lớp từ sơ đồ lớp UML

```

1 /**
2  * Describes a product with a description and a price.
3 */
4 public class Product {
5     private String description;
6     private double price;
7
8     /**
9      * Constructs a product from a description and a price.
10     *
11     * @param aDescription
12     *         the product description
13     * @param aPrice
14     *         the product price
15     */
16     public Product(String aDescription, double aPrice) {
17         description = aDescription;
18         price = aPrice;
19     }
20
21     /**
  
```

```
22     * Gets the product description.  
23     *  
24     * @return the description  
25     */  
26     public String getDescription() {  
27         return description;  
28     }  
29  
30     /**  
31     * Gets the product price.  
32     *  
33     * @return the unit price  
34     */  
35     public double getPrice() {  
36         return price;  
37     }  
38 }
```

```
1  /**  
2   * Describes a quantity of an article to purchase.  
3   */  
4   public class LineItem {  
5       private int quantity;  
6       private Product theProduct;  
7  
8       /**  
9        * Constructs an item from the product and quantity.  
10       *  
11       * @param aProduct  
12       *          the product  
13       * @param aQuantity  
14       *          the item quantity  
15       */  
16       public LineItem(Product aProduct, int aQuantity) {  
17           theProduct = aProduct;  
18           quantity = aQuantity;  
19       }  
20 }
```

```

21  /**
22   * Computes the total cost of this line item.
23   *
24   * @return the total price
25  */
26  public double getAmount() {
27      return theProduct.getPrice() * quantity;
28  }
29
30 /**
31  * Formats this item.
32  *
33  * @return a formatted string of this line item
34  */
35  public String format() {
36      return String.format("%-30s%,12.0f%5d%,12.0f",
37          theProduct.getDescription(),
38          theProduct.getPrice(), quantity, getAmount());
39  }

```

```

1 /**
2  * Describes a billing customer.
3 */
4 public class Customer {
5     private String name;
6     private String telephone;
7     private String address;
8
9 /**
10  * Constructs a billing customer.
11  *
12  * @param cName
13  *          the recipient name
14  * @param cTelephone
15  *          the telephone
16  * @param cAddress
17  *          the address

```

```
18  /*
19   public Customer(String cName, String cTelephone, String cAddress) {
20     name = cName;
21     telephone = cTelephone;
22     address = cAddress;
23   }
24
25 /**
26 * Formats the customer information.
27 *
28 * @return the customer information as a string with three lines
29 */
30 public String format() {
31   return name + "\n" + telephone + "\n" + address;
32 }
33 }
```

```
1 import java.util.ArrayList;
2
3 /**
4 * Describes an invoice for a set of purchased products.
5 */
6 public class Invoice {
7   private Customer customer;
8   private ArrayList<LineItem> items;
9
10 /**
11 * Constructs an invoice.
12 *
13 * @param aCustomer
14 *          the billing customer
15 */
16 public Invoice(Customer aCustomer) {
17   items = new ArrayList<LineItem>();
18   customer = aCustomer;
19 }
20
21 /**
```

```
22     * Adds a charge for a product to this invoice.
23     *
24     * @param aProduct
25     *          the product that the customer ordered
26     * @param quantity
27     *          the quantity of the product
28     */
29     public void add(Product aProduct, int quantity) {
30         LineItem anItem = new LineItem(aProduct, quantity);
31         items.add(anItem);
32     }
33
34     /**
35      * Formats the invoice.
36      *
37      * @return the formatted invoice
38      */
39     public String format() {
40         String r = "\n\t\t\tI N V O I C E\n\n"
41             + customer.format()
42             + "\n\n-----"
43             + String.format("\n%-30s%12s%5s%12s\n", "Description", "Price", "Qty",
44                     "Amount");
45
46         for (LineItem item : items) {
47             r = r + item.format() + "\n";
48         }
49
50         r = r + "-----";
51         r = r + String.format("\nTotal: %,.2f VND", getTotal());
52         return r;
53     }
54
55     /**
56      * Computes the total money of the bill.
57      *
58      * @return the total
```

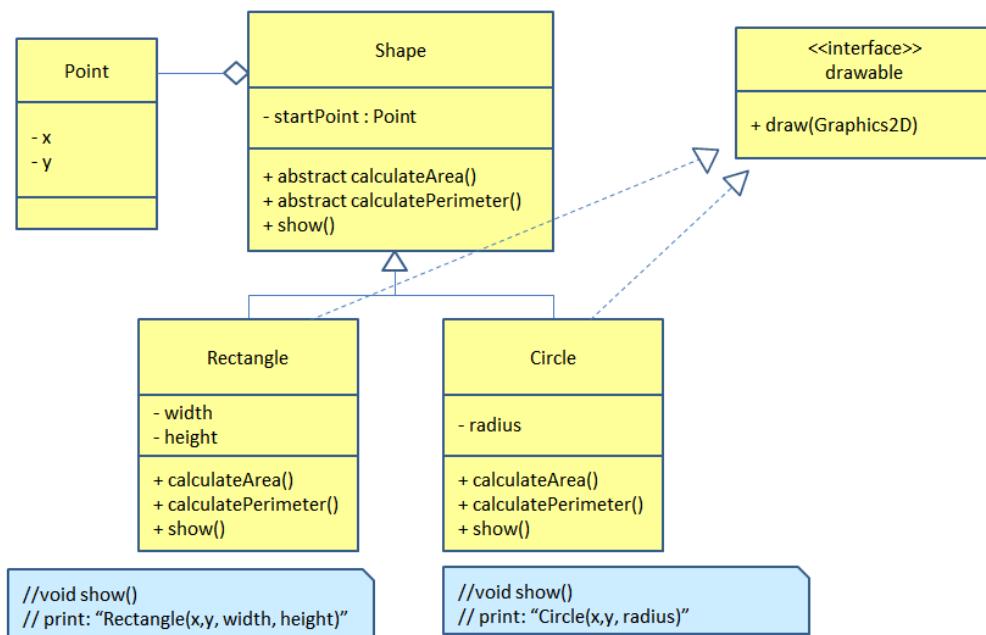
```
59     */
60     private double getTotal() {
61         double total = 0;
62         for (LineItem item : items) {
63             total = total + item.getAmount();
64         }
65         return total;
66     }
67 }
```

```
1 /**
2  * This program demonstrates the invoice classes by printing a sample invoice.
3 */
4 public class InvoicePrinter {
5     public static void main(String[] args) {
6         Customer customer1 = new Customer("Hong My", "0987654321",
7             "123 Vo Van Ngan, Thu Duc, Tp.HCM");
8         Invoice invoice1 = new Invoice(customer1);
9         invoice1.add(new Product("Notebook 200 pages", 30000), 4);
10        invoice1.add(new Product("Ruler", 5000), 1);
11        invoice1.add(new Product("Ballpen", 2500), 3);
12        System.out.println(invoice1.format());
13    }
14 }
```

Yêu cầu bổ sung:

- + Hóa đơn có thêm thuộc tính mã hóa đơn.
- + Thêm số thứ tự trước mỗi món hàng khi in hóa đơn.
- + Tạo lớp quản lý hóa đơn có các phương thức: thêm hóa đơn, xóa hóa đơn khi biết mã hóa đơn, tìm các hóa đơn có tổng tiền cao nhất, tìm các hóa đơn khi biết tên khách hàng, sắp xếp hóa đơn theo thứ tự tổng tiền tăng dần.

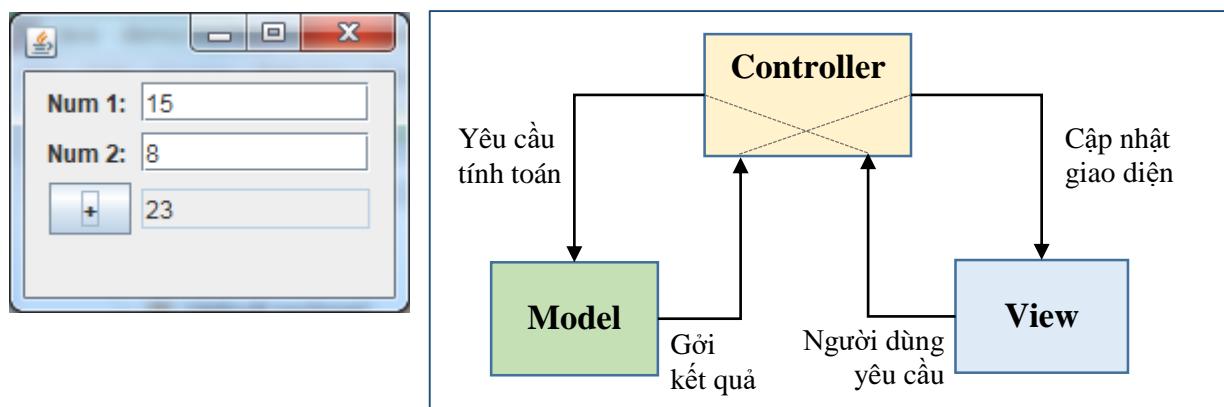
8.3.3. Ứng dụng Bài toán tổng hợp (has-a, is-a, abstract class, Interface)



1. Từ sơ đồ lớp UML, viết mô tả yêu cầu bài toán.
2. Hiện thực các lớp từ sơ đồ lớp UML
3. Tạo lớp quản lý hình có các phương thức: thêm hình, hiển thị danh sách các hình, đếm số lượng hình chữ nhật, lấy danh sách các hình chữ nhật có diện tích lớn nhất, sắp xếp các hình theo diện tích tăng dần.

8.4. Mô hình MVC

Mô hình MVC – Model View Controller. Demo ứng dụng tính toán đơn giản.



Model

```
1 public class CalculatorModel {  
2  
3     public int add(int n1, int n2) {  
4         return n1 + n2;  
5     }  
6 }
```

View

```
1 import java.awt.event.ActionListener;  
2  
3 import javax.swing.JButton;  
4 import javax.swing.JFrame;  
5 import javax.swing.JLabel;  
6 import javax.swing.JPanel;  
7 import javax.swing.JTextField;  
8  
9 public class CalculatorView extends JFrame {  
10    private JLabel lbl_num1, lbl_num2;  
11    private JTextField txt_num1, txt_num2, txt_result;  
12    private JButton btn_plus;  
13  
14    public CalculatorView() {  
15        setSize(200, 150);  
16        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);  
17  
18        lbl_num1 = new JLabel("Num 1: ");  
19        lbl_num2 = new JLabel("Num 2: ");  
20        txt_num1 = new JTextField(10);  
21        txt_num2 = new JTextField(10);  
22        btn_plus = new JButton("+");  
23        txt_result = new JTextField(10);  
24        txt_result.setEditable(false);  
25  
26        JPanel calPanel = new JPanel();  
27        calPanel.add(lbl_num1);  
28        calPanel.add(txt_num1);  
29        calPanel.add(lbl_num2);  
30        calPanel.add(txt_num2);  
31        calPanel.add(btn_plus);
```

```

32     calPanel.add(txt_result);
33
34     add(calPanel);
35     setVisible(true);
36 }
37 public JTextField getTxt_num1() {
38     return txt_num1;
39 }
40
41 public JTextField getTxt_num2() {
42     return txt_num2;
43 }
44
45 public JTextField getTxt_result() {
46     return txt_result;
47 }
48
49 public JButton getBtn_plus() {
50     return btn_plus;
51 }
52 }
```

Controller

```

1 import java.awt.event.ActionEvent;
2 import java.awt.event.ActionListener;
3
4 public class CalculatorController {
5     private CalculatorModel model;
6     private CalculatorView view;
7
8     public CalculatorController(CalculatorModel model, CalculatorView view) {
9         this.model = model;
10        this.view = view;
11
12        view.getBtn_plus().addActionListener(new ActionListener() {
13
14             @Override
```

```
15     public void actionPerformed(ActionEvent arg0) {  
16         int num1 = Integer.parseInt(view.getTxt_num1().getText());  
17         int num2 = Integer.parseInt(view.getTxt_num1().getText());  
18         int result = model.add(num1, num2);  
19         view.getTxt_result().setText(result + "");  
20     }  
21 };  
22 }  
23 }  
24 // làm tương tự cho các nút -, *, /
```

Case Study Quản Lý Sinh Viên

Code chương trình quản lý sinh viên tương tác CSDL. Yêu cầu xây dựng theo mô hình MVC. Chức năng thêm, xóa, sửa, tìm kiếm sinh viên, môn học, lớp cố định, lớp học phần; Đăng ký học phần sinh viên; Chức năng quản lý điểm; Xem bảng điểm cá nhân, Bảng điểm lớp học phần, Danh sách sinh viên của lớp cố định.

Ví dụ: Phần Quản lý Môn Học

```

1 import java.awt.Checkbox;
2 import java.awt.event.ActionEvent;
3 import java.awt.event.ActionListener;
4 import java.awt.event.ItemEvent;
5 import java.awt.event.ItemListener;
6 import java.awt.event.MouseAdapter;
7 import java.util.ArrayList;
8 import javax.swing.JOptionPane;
9 import javax.swing.table.DefaultTableModel;
10
11 public class ControllerQuanLyMonHoc {
12     FrameQuanLyMonHoc frameQLMH;
13     ModelQuanLyMonHoc modelQLMH;
14
15     public ControllerQuanLyMonHoc(FrameQuanLyMonHoc frameQLMH,
16                                     ModelQuanLyMonHoc modelQLMH) {
17         this.frameQLMH = frameQLMH;
18         this.modelQLMH = modelQLMH;
19         showTableMonHoc(modelQLMH.getDsMonHoc());
20
21         frameQLMH.getBtnXoa().setEnabled(false);
22         frameQLMH.getBtnSua().setEnabled(false);
23         frameQLMH.getBtnTim().setEnabled(false);
24         frameQLMH.getTxtKeyword().setEnabled(false);
25
26         frameQLMH.getBtnThem().addActionListener(themListener());
27         frameQLMH.getBtnXoa().addActionListener(xoaListener());
28         frameQLMH.getBtnSua().addActionListener(suaListener());

```

```
29     frameQLMH.getBtnMoi().addActionListener(moiListener());
30     frameQLMH.getTableDanhSachMonHoc().addMouseListener(tableListener());
31     frameQLMH.getCbbSapxep().addItemListener(sapXepListener());
32     frameQLMH.getCbxTim().addItemListener(timAction());
33     frameQLMH.getBtnTim().addActionListener(timListener());
34
35     frameQLMH.setVisible(true);
36 }
37
38 public void showTableMonHoc(ArrayList<MonHoc> dsMonHoc) {
39     DefaultTableModel tableModel = (DefaultTableModel) frameQLMH
40         .getTableDanhSachMonHoc().getModel();
41     tableModel.setRowCount(0);
42     tableModel.setColumnCount(0);
43     tableModel.addColumn("Mã môn học");
44     tableModel.addColumn("Tên môn học");
45     tableModel.addColumn("Số tín chỉ");
46
47     for (MonHoc monHoc : dsMonHoc) {
48         tableModel.addRow(new Object[] { monHoc.getMaMonHoc(),
49             monHoc.getTenMonHoc(), monHoc.getSoTinChi() });
50     }
51     frameQLMH.getTableDanhSachMonHoc().setModel(tableModel);
52 }
53
54 private ActionListener themListener() {
55     return new ActionListener() {
56
57         @Override
58         public void actionPerformed(ActionEvent arg0) {
59             MonHoc monHoc = new MonHoc();
60             monHoc.setMaMonHoc(frameQLMH.getTxtMaMonHoc().getText());
61             monHoc.setTenMonHoc(frameQLMH.getTxtTenMonHoc().getText());
62             int soTinChi = Integer.parseInt(frameQLMH.getTxtSoTinChi().getText());
63             monHoc.setSoTinchi(soTinChi);
64             try {
65                 modelQLMH.themMonHoc(monHoc);
66             } catch (Exception e) {
```

```
67         JOptionPane.showMessageDialog(frameQLMH, e.toString());
68     }
69     showTableMonHoc(modelQLMH.getDsMonHoc());
70   }
71 }
73
74 private ActionListener xoaListener() {
75   return new ActionListener() {
76     @Override
77     public void actionPerformed(ActionEvent arg0) {
78       String maMonHoc = frameQLMH.getTxtMaMonHoc().getText();
79       if (!(maMonHoc.equals("")))) {
80         int confirm = JOptionPane.showConfirmDialog(frameQLMH,
81               "Bạn có muốn xóa Môn Học có mã '" + maMonHoc + "' ?");
82         if (confirm == 0) {
83           try {
84             modelQLMH.xoaMonHoc(maMonHoc);
85           } catch (Exception e) {
86             JOptionPane.showMessageDialog(frameQLMH, e.toString());
87           }
88         }
89       }
90       showTableMonHoc(modelQLMH.getDsMonHoc());
91       frameQLMH.getTxtMaMonHoc().setEnabled(false);
92       frameQLMH.getTxtTenMonHoc().setEnabled(false);
93       frameQLMH.getTxtSoTinChi().setEnabled(false);
94       frameQLMH.getBtnThem().setEnabled(false);
95       frameQLMH.getBtnXoa().setEnabled(false);
96       frameQLMH.getBtnSua().setEnabled(false);
97       frameQLMH.getBtnMoi().setEnabled(true);
98     }
99   };
100 }
101
102 private ActionListener suaListener() {
103   return new ActionListener() {
```

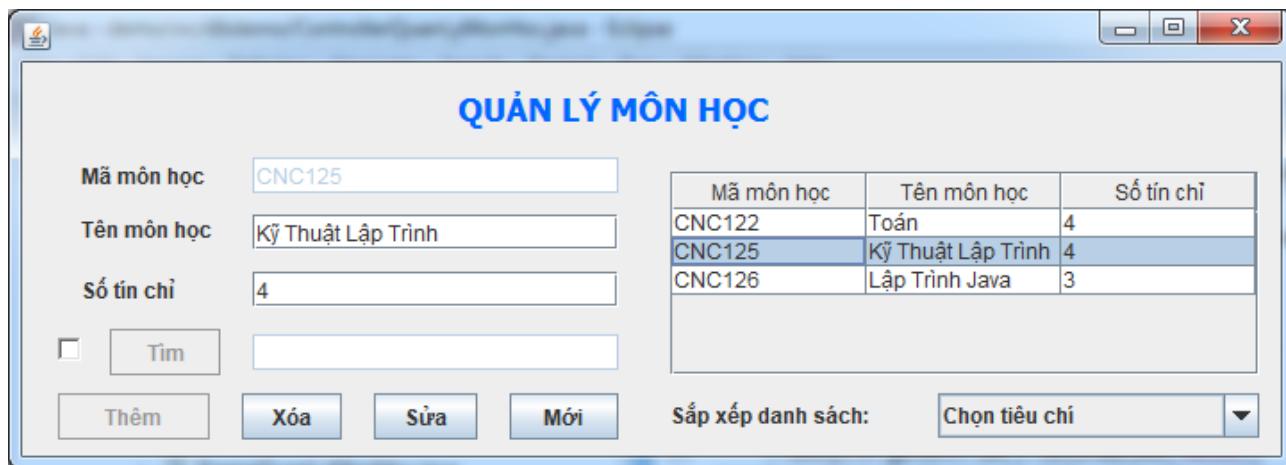
```
104     @Override
105     public void actionPerformed(ActionEvent arg0) {
106         MonHoc monHoc = new MonHoc();
107         monHoc.setMaMonHoc(frameQLMH.getTxtMaMonHoc().getText());
108         monHoc.setTenMonHoc(frameQLMH.getTxtTenMonHoc().getText());
109         int soTinChi = Integer.parseInt(frameQLMH.getTxtSoTinChi().getText());
110         monHoc.setSoTinchi(soTinChi);
111         try {
112             modelQLMH.suaMonHoc(monHoc);
113         } catch (Exception e) {
114             JOptionPane.showMessageDialog(frameQLMH, e.toString());
115         }
116         showTableMonHoc(modelQLMH.getDsMonHoc());
117     }
118 };
119 }
120
121     public ActionListener moiListener() {
122         return new ActionListener() {
123             @Override
124             public void actionPerformed(ActionEvent arg0) {
125                 frameQLMH.getTxtMaMonHoc().setText("");
126                 frameQLMH.getTxtTenMonHoc().setText("");
127                 frameQLMH.getTxtSoTinChi().setText("");
128
129                 frameQLMH.getTxtMaMonHoc().setEnabled(true);
130                 frameQLMH.getTxtTenMonHoc().setEnabled(true);
131                 frameQLMH.getTxtSoTinChi().setEnabled(true);
132                 frameQLMH.getBtnThem().setEnabled(true);
133                 frameQLMH.getBtnXoa().setEnabled(true);
134                 frameQLMH.getBtnSua().setEnabled(true);
135             }
136         };
137     }
138
139     public MouseAdapter tableListener() {
140         return new java.awt.event.MouseAdapter() {
141             public void mouseClicked(java.awt.event.MouseEvent evt) {
```

```
142         int index = frameQLMH.getTableDanhSachMonHoc().getSelectedRow();
143         String ma = frameQLMH.getTableDanhSachMonHoc().getValueAt(index, 0)
144         String ten = frameQLMH.getTableDanhSachMonHoc().getValueAt(index, 1)
145         String tinChi = frameQLMH.getTableDanhSachMonHoc().getValueAt(index, 2)
146         frameQLMH.getTxtMaMonHoc().setText(ma);
147         frameQLMH.getTxtTenMonHoc().setText(ten);
148         frameQLMH.getTxtSoTinChi().setText(tinChi);
149         frameQLMH.getTxtMaMonHoc().setEnabled(false);
150         frameQLMH.getTxtTenMonHoc().setEnabled(true);
151         frameQLMH.getTxtSoTinChi().setEnabled(true);
152         frameQLMH.getBtnThem().setEnabled(false);
153         frameQLMH.getBtnXoa().setEnabled(true);
154         frameQLMH.getBtnSua().setEnabled(true);
155         frameQLMH.getBtnMoi().setEnabled(true);
156     }
157 };
158 }
159
160 public ItemListener sapXepListener() {
161     return new ItemListener() {
162         @Override
163         public void itemStateChanged(ItemEvent arg0) {
164             int sx = frameQLMH.getCbbSapxep().getSelectedIndex();
165             switch (sx) {
166                 case 1:
167                     showTableMonHoc(modelQLMH.sapXepTheoTen());
168                     break;
169                 case 2:
170                     showTableMonHoc(modelQLMH.sapXepTheoSoTinChi());
171                     break;
172                 case 3:
173                     showTableMonHoc(modelQLMH.sapXepTheoSoTinChiRoiTheoTen());
174                     break;
175             default:
176                 showTableMonHoc(modelQLMH.getDsMonHoc());
177             break;
178         }
179     }
180 }
```

```
179      }
180    };
181  }
182
183  public ItemListener timAction() {
184    return new ItemListener() {
185      @Override
186      public void itemStateChanged(ItemEvent ie) {
187        if (ie.getStateChange() == 1) {
188          frameQLMH.getBtnTim().setEnabled(true);
189          frameQLMH.getTxtKeyword().setEnabled(true);
190          frameQLMH.getBtnThem().setEnabled(false);
191          frameQLMH.getBtnXoa().setEnabled(false);
192          frameQLMH.getBtnSua().setEnabled(false);
193          frameQLMH.getBtnMoi().setEnabled(false);
194          frameQLMH.getCbbSapxep().setEnabled(false);
195        } else {
196          frameQLMH.getBtnTim().setEnabled(false);
197          frameQLMH.getTxtKeyword().setEnabled(false);
198          frameQLMH.getBtnThem().setEnabled(true);
199          frameQLMH.getBtnMoi().setEnabled(true);
200          frameQLMH.getCbbSapxep().setEnabled(true);
201          showTableMonHoc(modelQLMH.getDsMonHoc());
202        }
203      }
204    };
205  }
206
207  public ActionListener timListener() {
208    return new ActionListener() {
209      @Override
210      public void actionPerformed(ActionEvent arg0) {
211        if (!frameQLMH.getTxtKeyword().getText().equals("")) {
212          showTableMonHoc(modelQLMH.getDsMonHocCoTuKhoa(frameQLMH
213              .getTxtKeyword().getText()));
214        } else if (!frameQLMH.getTxtSoTinChi().getText().equals("")) {
215          showTableMonHoc(modelQLMH.getDSMonHocTheoTinChi(Integer
216              .parseInt(frameQLMH.getTxtSoTinChi().getText())));
217      }
218    };
219  }
220}
```

```

217     } else if (!frameQLMH.getTxtTenMonHoc().getText().equals("")) {
218         showTableMonHoc(modelQLMH.getDsMonHocTheoTen(frameQLMH
219                         .getTxtTenMonHoc().getText()));
220     } else if (!frameQLMH.getTxtMaMonHoc().getText().equals("")) {
221         showTableMonHoc(modelQLMH.getDsMonHocTheoMa(frameQLMH
222                         .getTxtMaMonHoc().getText()));
223     }
224 }
225 };
226 }
227
228 public static void main(String[] args) {
229     new ControllerQuanLyMonHoc(new FrameQuanLyMonHoc(),
230         new ModelQuanLyMonHoc());
231 }
232 }
```



Bài tập

Bài 1. Tạo lớp Xe gồm các thông tin: nhãn hiệu, giá, năm sản xuất. Lớp Oto kế thừa lớp Xe và có thêm thông tin: số chỗ và trọng tải. Xây dựng lớp QuanLyXe có các phương thức: thêm xe, in danh sách các xe, xuất thông tin các Oto có trọng tải lớn hơn 1000 tấn, xuất thông tin các Oto của nhãn hiệu ‘KIA’.

Bài 2. Quản lý cho thuê bãi đỗ xe trong ngày

Xe hơi gồm các thông tin: biển số, màu xe, số chỗ ngồi, hãng xe. Có phương thức dongPhi được tính bằng số chỗ ngồi * 5000, khuyến mãi giảm 20% cho xe lớn hơn 30 chỗ.

Xe buýt gồm các thông tin: biển số, màu xe, số chỗ ngồi, tuyến xe. Có phương thức dongPhi được tính bằng số chỗ ngồi * 1000.

Yêu cầu xây dựng ứng dụng có tính kế thừa. Xây dựng lớp QuanLyBaiXe, nhập vào 5 xe bất kỳ. Có phương thức tính tổng tiền phí thu được.

Bài 3. Một hiệu sách nhỏ cần quản lý tài liệu

Sách: mã sách, tên nhà xuất bản, tên sách, giá, tên tác giả, số trang

Tạp chí: mã tạp chí, tên nhà xuất bản, tên tạp chí, giá, số phát hành, tháng phát hành

Báo: mã báo, tên nhà xuất bản, tên báo, giá, ngày phát hành

Các tài liệu có chương trình khuyến mại:

Tạp chí giảm giá 10% cho các số phát hành là chẵn

Báo giảm giá 10% cho các báo phát hành ngày 15 hàng tháng

Sách giảm giá 5% cho các sách của tác giả ‘ABC’

Thiết và xây dựng chương trình ứng dụng có tính kế thừa.

Tạo lớp quản lý tài liệu, nhập 7 tài liệu bất kỳ

Tìm tài liệu theo loại

Tính tổng tiền theo loại

Tìm tài liệu theo tên gần đúng

Bài 4. Công ty vận tải quản lý thông tin của các chuyến xe. Thông tin của 2 loại chuyến xe:

- Chuyến xe nội thành: Mã số chuyến, họ tên tài xế, số xe, số tuyến, số km đi được, doanh thu. Có phương thức tính thưởng = $10\% * \text{doanh thu}$.
- Chuyến xe ngoại thành: Mã số chuyến, họ tên tài xế, số xe, nơi đến, số ngày đi được, doanh thu. Có phương thức tính thưởng = $5\% * \text{doanh thu}$.

Viết chương trình thực hiện các yêu cầu sau:

- Xây dựng các lớp có quan hệ thừa kế.
- Tạo lớp quản lý chuyến xe có các phương thức: Tìm xe các xe có tổng doanh thu cao nhất; Tính tổng doanh thu khi biết loại; Tổng tiền thưởng khi biết loại.

Bài 5. Xây dựng lớp NhanVien gồm các thông tin: maSo, hoTen. Nhân viên chính thức có thêm hệ số lương, nhân viên hợp đồng mức lương cơ bản và số ngày công, nhân viên thời vụ có số giờ làm. Các loại nhân viên có cách tính lương khác nhau. Nhân viên chính thức có cách tính lương: $\text{Lương} = \text{heSoLuong} * 1.150.000$. Nhân viên hợp đồng: lương cơ bản * số ngày công /30. Nhân viên thời vụ: số giờ * 50.000. Cả nhân viên chính thức và nhân viên hợp đồng khi tính lương nếu có tăng ca thì cộng thêm tiền tăng ca = giờ tăng ca * 60000. Viết Test thêm 5 đối tượng nhân viên vào mảng, đếm số lượng nhân viên hợp đồng, sau đó xuất thông tin của các nhân viên có lương cao nhất.

Bài 6. Viết chương trình quản lý nhân viên giao diện GUI thao tác với CSDL, có các chức năng: thêm, xóa, sửa, xem danh sách nhân viên.

Bài 7. Viết ứng dụng cho bé học từ vựng tiếng Anh, mô hình các thẻ từ vựng. Ứng dụng cho phép load danh sách từ vựng.

CHƯƠNG 9. ĐA LUỒNG MULTITHREAD

Học xong chương này, người học có thể:

- + Trình bày khái niệm Thread, MultiThread
- + Sử dụng Thread class và Runnable interface để tạo Thread
- + Quản lý các Thread trong chương trình MultiThread
- + Xử lý đồng bộ hóa các Thread

9.1. Thread, Multithread

Luồng (Thread) là đơn vị nhỏ nhất trong chương trình có thể thực hiện được một công việc riêng biệt. Một luồng gồm có 4 thành phần chính đó là: định danh, một bộ đếm chương trình, một tập thanh ghi và ngăn xếp.

Chương trình đa luồng (Mutithread) là chương trình có nhiều hơn một hoạt động chạy đồng thời. Ví dụ: Một ứng dụng Game có thể vừa có nhạc nền và vừa chơi, Game cờ vua online hai người chơi cùng lúc, Game có từ 2 đối tượng hoạt động đồng thời, ... Tuy nhiên các chương trình được thực thi đồng thời nhưng thời gian để CPU chuyển đổi qua lại giữa các luồng vẫn có độ trễ nhất định nhưng rất nhỏ (tính bằng đơn vị nano giây).

Khái niệm đa nhiệm (MultiTasking) là nhiều chương trình chạy cùng lúc trên máy tính. (Ví dụ: MS Word, Google Chrome, Window Media Player, ...)

9.2. Tạo và quản lý Thread

9.2.1. Main Thread

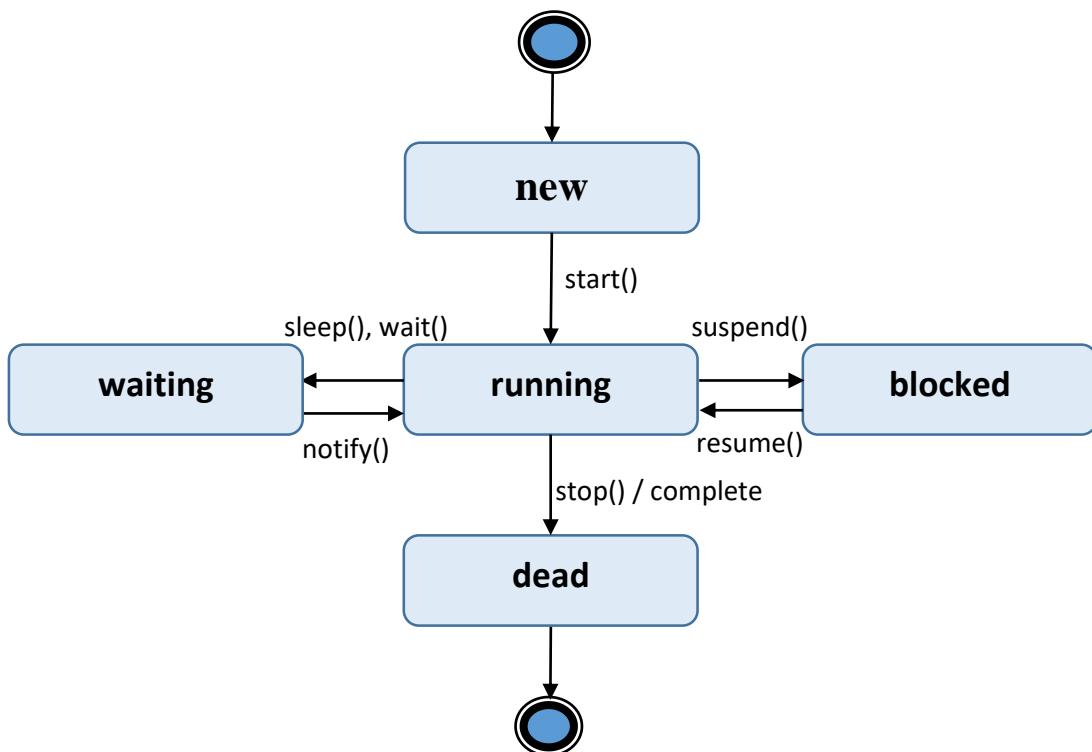
Khi chương trình java thực thi thì một thread bắt đầu chạy, thread này gọi là main thread.

```
1 public class MainThreadDemo {  
2     public static void main(String args[]) {  
3         Thread t = Thread.currentThread();  
4         System.out.println("Current thread: " + t);  
5         t.setName("My Thread");  
6         System.out.println("After name change: " + t);  
7         try {  
8             for (int n = 3; n > 0; n--) {  
9                 System.out.println(n);  
10                Thread.sleep(1000);  
11            }  
12        } catch (InterruptedException e) {  
13            System.out.println("Main thread interrupted");  
14        }  
15        System.out.println("Main thread finished");  
16    }  
17 }
```

```
Current thread: Thread[main,5,main]  
After name change: Thread[My Thread,5,main]  
3  
2  
1  
Main thread finished
```

9.2.2. Lớp Thread

Các trạng thái của Thread



New: một thread mới bắt đầu.

Running: trạng thái này chỉ thread đang chạy

Waiting: thread ở trạng thái chờ một thread khác hoàn thành để tiếp tục chạy

Dead: thread vào trạng thái này khi thi hành xong tác vụ

Sự khác nhau giữa wait() và sleep()

wait()	sleep()
Phương thức wait() giải phóng khóa	Phương thức sleep() không giải phóng khóa.
Là phương thức của lớp Object	Là phương thức của lớp Thread
Là phương thức non-static	Là phương thức static
Nên được đánh thức bởi các phương thức notify() hoặc notifyAll()	Sleep hoàn thành sau một khoảng thời gian nhất định.

Phương thức khởi tạo Thread:

Thread()

Thread(String name)

Thread(Runnable r)

Thread(Runnable r, String name)

Một số phương thức của Thread:

public void start(): Bắt đầu thực hiện thread, gọi phương thức run() trên thread.

public void run(): thực hiện tác vụ cho một thread.

public int setPriority(int priority): Thay đổi độ ưu tiên; getPriority(): lấy độ ưu tiên.

public void setName(String name): đổi tên của thread; getName() lấy tên.

public void sleep(long miliseconds): thread tạm ngừng thực thi trong khoảng thời gian tính bằng mili giây.

public void join(): Đợi cho một thread chết.

public void join(long miliseconds): block tới khi thread khác kết thúc hoặc sau khoảng thời gian là số mili giây.

public Thread currentThread(): Trả về tham chiếu của thread đang được thi hành.

public int getId(): Trả về id của thread.

public boolean isAlive(): Kiểm tra nếu thread còn sống.

public void yield(): Làm cho các đối tượng thread đang thực thi dừng tạm thời và cho phép các thread khác để thực thi.

public boolean isDaemon(): Kiểm tra nếu thread là một luồng ngầm.

public void setDaemon(boolean b): Đánh dấu thread là luồng ngầm hoặc luồng người dùng.

public void interrupt(): Ngắt thread.

public boolean isInterrupted(): Kiểm tra nếu thread đã bị ngắt.

9.2.3. Tạo Thread

9.2.3.1. Tạo Thread bằng cách extends Thread class

Khi một Thread được tạo ra, cần gọi phương thức start() để đặt Thread ở trạng thái sẵn sàng. Tiếp theo hệ thống sẽ gọi hàm run() để thực thi các câu lệnh trong phương thức này. Thread sẽ kết thúc khi thực hiện hết các lệnh trong run() hoặc khi phương thức stop() được gọi.

```

1 public class MyThread extends Thread {
2     public void run() {
3         System.out.println("Hello");
4     }
5
6     public static void main(String[] args) {
7         MyThread myThread = new MyThread();
8         myThread.start();
9
10        // Tạo lớp vô danh anonymous class
11        Thread thread = new Thread() {
12            public void run() {
13                System.out.println("Thread2 is Running");
14            }
15        };

```

```
16     thread.start();
17 }
18 }
```

9.2.3.2. Tạo Thread bằng cách implements Runnable interface

Runnable Interface chỉ có duy nhất một phương thức run()

```
1 public class MyRunnable implements Runnable {
2     public void run() {
3         System.out.println("Hi");
4         try {
5             Thread.sleep(5000);
6         } catch (InterruptedException e) {
7             e.printStackTrace();
8         }
9         System.out.println("Goodbye");
10    }
11
12    public static void main(String[] args) {
13        Thread thread = new Thread(new MyRunnable());
14        thread.start();
15
16        // Tạo lớp vô danh anonymous class
17        Runnable myRunnable = new Runnable() {
18            public void run() {
19                System.out.println("Runnable2 is running");
20            }
21        };
22        Thread thread2 = new Thread(myRunnable);
23        thread2.start();
24    }
25 }
```

9.3. Độ ưu tiên của Thread

Mỗi thread trong Java đều có độ ưu tiên (priority), nhằm giúp hệ thống quyết định thứ tự chạy. Độ ưu tiên có giá trị từ 1 đến 10 và được thiết lập bằng phương thức setPriority()

Ba hằng số độ ưu tiên trong lớp Thread:

Thread.MIN_PRIORITY // 1

Thread.NORM_PRIORITY // 5

Thread.MAX_PRIORITY //10

Lưu ý: Một Thread mới sẽ thừa kế độ ưu tiên từ Thread tạo ra nó.

9.4. Đồng bộ các Thread

Giao tiếp giữa các thread là một cơ chế trong đó thread bị tạm dừng và một thread khác được phép xen vào thực hiện. Nó được thực hiện bằng các phương thức sau của lớp Object:

public final void wait(): Chờ cho đến khi đối tượng được thông báo.

public final void wait(long timeout): Chờ một khoảng thời gian cụ thể.

public final void notify(): Đánh thức luồng đang chờ của đối tượng này.

public final void notifyAll(): Đánh thức tất cả các luồng đang chờ của đối tượng này.

Ví dụ: wait(), notify()

```

1  class Customer {
2      int amount = 10000;
3
4      synchronized void withdraw(int amount) {
5          System.out.println("Rút tiền...");
6
7          if (this.amount < amount) {
8              System.out.println("Tài khoản không đủ; đợi gửi tiền...");
9              try {
10                  wait();
11              } catch (Exception e) {
12              }
13          }
14          this.amount -= amount;
15          System.out.println("Hoàn thành rút tiền!");
16      }
17

```

```
18     synchronized void deposit(int amount) {  
19         System.out.println("Gửi tiền...");  
20         this.amount += amount;  
21         System.out.println("Hoàn thành gửi tiền!");  
22         notify();  
23     }  
24 }  
25  
26 public class TestThread {  
27     public static void main(String args[]) {  
28         final Customer c = new Customer();  
29         new Thread() {  
30             public void run() {  
31                 c.withdraw(15000);  
32             }  
33         }.start();  
34         new Thread() {  
35             public void run() {  
36                 c.deposit(10000);  
37             }  
38         }.start();  
39     }  
40 }
```

```
Rút tiền...  
Tài khoản không đủ; đợi gửi tiền...  
Gửi tiền...  
Hoàn thành gửi tiền!  
Hoàn thành rút tiền!
```

Khi có hai tiến trình trở lên cần truy xuất tới một tài nguyên chia sẻ, cần đảm bảo rằng tại một thời điểm tài nguyên đó chỉ được sử dụng bởi một thread, các thread khác phải đợi.

Quá trình này được gọi là đồng bộ hóa thread synchronization.

Ví dụ:

```
1 public class Synch {  
2     public static void main(String args[]) {  
3         CallMe target = new CallMe();  
4         Caller ob1 = new Caller(target, "Hello");
```

```
5     Caller ob2 = new Caller(target, "JAVA");
6     Caller ob3 = new Caller(target, "World");
7     // wait for threads to end
8     try {
9         ob1.t.join();
10    ob2.t.join();
11    ob3.t.join();
12 } catch (InterruptedException e) {
13     System.out.println("Interrupted");
14 }
15 }
16 }
17
18 class CallMe {
19     void call(String msg) {
20         System.out.print("[ " + msg);
21         try {
22             Thread.sleep(1000);
23         } catch (InterruptedException e) {
24             System.out.println("Interrupted");
25         }
26         System.out.println(" ]");
27     }
28 }
29
30 class Caller implements Runnable {
31     String msg;
32     CallMe target;
33     Thread t;
34
35     public Caller(CallMe targ, String s) {
36         target = targ;
37         msg = s;
38         t = new Thread(this);
39         t.start();
40     }
41 }
```

```
42     public void run() {  
43         target.call(msg);  
44     }  
45 }
```

```
[Hello[JAVA[World]  
]  
]
```

Hướng giải quyết

```
1 //Cách 1: Đồng bộ phương thức  
2 class CallMe {  
3     synchronized void call(String msg) { //synchronized method  
4         ...  
5     }
```

```
1 //Cách 2: Đồng bộ khối lệnh  
2 class Caller implements Runnable {  
3     ...  
4     public void run() {  
5         synchronized(target) { //synchronized block  
6             target.call(msg);  
7         }  
8     }  
9 }
```

```
[Hello]  
[JAVA]  
[World]
```

Case Study Quản Lý Sinh Viên

Mô phỏng quá trình in mỗi bảng điểm cá nhân sleep 5 giây. Xử lý đồng bộ các threads in bảng điểm.

Bài tập

Bài 1. Viết chương trình bộ đếm ngược. Cứ mỗi một giây sẽ in một con số giảm dần cho đến khi hết giờ.

Bài 2. Tạo lớp PracticeThread, thực hiện in ra n số tự nhiên đầu tiên với mỗi lần in cách nhau 1 giây. Tạo hai đối tượng từ lớp PracticeThread với thứ tự ưu tiên lần lượt là 1 và 10. Quan sát và nhận xét kết quả.

Bài 3. Viết chương trình để tạo thread thứ nhất in ra lần lượt n số tự nhiên đầu tiên, và trong khi đó thread thứ hai in ra bình phương của từng số đó. In ra dưới dạng như sau:

[1] -> (1)

[2] -> (4)

[3] -> (9)

[4] -> (16)

Bài 4. Dùng một chương trình ứng dụng đã có, điều chỉnh bổ sung để thêm âm thanh nền cho ứng dụng.

THUẬT NGỮ VIẾT TẮT

API Application Programming Interface

CSDL Cơ sở dữ liệu

GUI Graphic User Interface

JDBC Java DataBase Connectivity

JDK Java Development Kit

JRE Java Runtime Environment

JVM Java Virtual Machine

ODBC Open Database Connectivity

RDBMS Relational Database Management Systems

UML Unified Modeling Language

TÀI LIỆU THAM KHẢO

- [1] Cay Horstmann, Java concepts 7th edition, Wiley, 2014
- [2] Paul Deitel, Java How to Program 9th edition, Prentice Hall, 2012
- [3] FSOFT-GST Training Material, FPT Software, 2015.

Website:

- [1] <https://docs.oracle.com/javase>
- [2] <http://java.sun.com>
- [3] <https://www.codejava.net>