ĐẠI HỌC HUẾ
TRƯỜNG ĐẠI HỌC KHOA HỌC

NGUYỄN HOÀNG HÀ, NGUYỄN VĂN TRUNG
NGUYỄN DŨNG, NGUYỄN MẬU HÂN

GIÁO TRÌNH
JAVA CƠ BẢN

NHÀ XUẤT BẢN ĐẠI HỌC HUẾ
Huế, 2020

Biên mục trên xuất bản phẩm của Thư viện Quốc gia Việt Nam

ĐTTS ghi: Đại học Huế. Trường Đại học Khoa học. - Thư mục: tr. 211

006.740711 - dc23

DUF0253p - CIP

Mã số sách: GT/38-2019

LỜI NÓI ĐẦU

Giới thiệu về NNLT Java: ....

Giáo trình "Java cơ bản" sẽ hệ thống toàn bộ những nội dung kiến thức cơ bản liên quan đến kiến thức về ngôn ngữ lập trình Java. Giáo trình được sử dụng cho sinh viên chuyên ngành Công nghệ Thông tin, là môn cơ sở để lập trình di động, lập trình web JSP, ... Giáo trình được chia làm 3 chương. Cuối mỗi chương sẽ có phần tổng kết chương, giúp người học nắm được các nội dung và ý nghĩa của chương. Phần câu hỏi và bài tập ở cuối mỗi chương cũng giúp người học nâng cao kỹ năng thực hành trên cơ sở phần lý thuyết được trang bị.

Các chương của giáo trình được tóm tắt như sau:

Chương 1: *Lập trình với Java*,....

Chương 2: *Lập trình hướng đối tượng trong Java*, ...

Chương 3: *Lập trình ứng dụng csdl*. ...

Mặc dù đã rất cố gắng để hoàn thiện giáo trình này với mong muốn đến tay người đọc nhưng giáo trình không thể tránh khỏi những thiếu sót về cách diễn đạt, bố cục, nội dung và các lỗi cú pháp. Rất mong được bạn đọc góp ý.

Để hoàn tất giáo trình này chúng tôi xin cám ơn các Thầy Cô giáo của Khoa Công nghệ Thông tin, Trường Đại học Khoa học, Đại học Huế đã góp ý, chỉnh sửa để giáo trình sớm được ra mắt bạn đọc.

Huế, tháng 02 năm 2019

Nhóm tác giả

MỤC LỤC

DANH MỤC HÌNH VẼ

DANH MỤC BẢNG BIỂU

CHƯƠNG 1

TỔNG QUAN VỀ NGÔN NGỮ LẬP TRÌNH JAVA

1.1. Giới thiệu về ngôn ngữ lập trình Java

Java là ngôn ngữ lập trình bậc cao, hướng đối tượng (tựa C/C++) do hãng Sun Microsystem phát triển và phát hành vào năm 1995. Mục tiêu chung của ngôn ngữ lập trình Java là cho phép các nhà phát triển ứng dụng "Viết một lần chạy mọi nơi" (WORA – Write once run anywhere), nghĩa là mã Java đã được biên dịch có thể chạy trên tất cả nền tảng hỗ trợ Java mà không cần biên dịch lại.

Java được James Gosling và các cộng sự của Công ty Sun Microsystem (Sau được công ty Oracle mua lại) khởi xướng và phát triển vào đầu những năm 90. Ban đầu ngôn ngữ này có tên là Oak theo tên một cây sồi bên ngoài văn phòng của Gosling. Sau đó dự án có tên là Green và cuối cùng là Java. Java là tên gọi của một hòn đảo ở Indonexia. Đây là nơi nhóm nghiên cứu đã chọn để đặt tên cho ngôn ngữ lập trình của mình trong một chuyến đi tham quan và làm việc trên hòn đảo này. Hòn đảo Java này là nơi rất nổi tiếng với nhiều khu vườn trồng café, đó chính là lý do chúng ta thường thấy biểu tượng ly café trong nhiều sản phẩm phần mềm, công cụ lập trình Java của Sun.

Một số đặc trưng của ngôn ngữ Java là:

- Hướng đối tượng: Trong Java, mọi thứ đều là Đối tượng. Do đó chương trình viết bằng Java có thể dễ dàng được mở rộng vì nó dựa trên mô hình đối tượng.
- Độc lập nền tảng: Không giống như nhiều ngôn ngữ lập trình khác bao gồm C và C++, khi Java được biên dịch, nó không được biên dịch trực tiếp thành mã máy cụ thể, thay vào đó là mã bytecode độc lập với nền tảng. Mã bytecode này được Máy ảo Java (JVM – Java Virtual Machine) thông dịch và chạy trên bất

kỳ nền tảng nào mà máy ảo này đang được chạy như: Window, Linux, Sun Solaris, Mac/OS,... Với cách thiết kế này mà Java đã đáp ứng được phương châm mà họ đã đưa ra: "Viết một lần, Chạy mọi nơi".

- Đơn giản: Java được thiết kế để dễ học. Cú pháp của Java dựa trên C/C++, trong đó đã loại bỏ nhiều tính năng phức tạp như: Con trỏ, quá tải toán tử,... Đặc biệt trong Java có tính năng Bộ thu gom rác tự động (Garbage Collection), do đó chúng ta không cần phải loại bỏ các đối tượng khi không dùng nữa như C++.
- Bảo mật: Java bảo mật bởi vì:
  - Không có con trỏ tường minh.
  - Chương trình chạy bên trong máy ảo.
  - Classloader: Nó bổ sung tính bảo mật bằng cách tách gói cho các lớp của hệ thống tệp cục bộ từ các gói được nhập từ các nguồn mạng.
  - Bytecode Vertifier: Kiểm tra các đoạn mã để tìm ra các phần mã truy cập trái phép tới các đối tượng.
  - Security Manager: Nó xác định những tài nguyên mà một lớp có thể truy cập được như đọc và ghi đĩa cục bộ.
  - Những tính năng bảo mật này được cung cấp bởi Ngôn ngữ Java. Ngoài ra, một vài tính năng bảo mật khác được cung cấp thông qua nhà phát triển như SSL, JAAS, cryptography, ...
- Mạnh mẽ: Java quản lý bộ nhớ rất tốt, có bộ thu gom rác tự động; không sử dụng con trỏ để tránh các vấn đề về bảo mật; Có cơ chế xử lý ngoại và kiểm tra kiểu dữ liệu tốt.
- Kiến trúc trung lập: Java là kiến trúc trung lập vì không có các tính năng phụ thuộc vào môi trường phát triển và thực thi; ví dụ, kích thước của các kiểu dữ liệu cơ bản là cố định. Như trong ngôn ngữ lập trình C, kiểu dữ liệu int chiếm 2byte trong bộ nhớ cho kiến trúc 32bit và 4byte bộ nhớ cho kiến trúc 64bit. Tuy nhiên, nó chiếm 4byte bộ nhớ cho cả kiến trúc 32 và 64bit trong

Java.

- Linh động: Vì Java biển dịch mã thành bytecode, do đó chương trình có thể chạy trên bất kỳ nền tảng nào có máy ảo Java mà không cần phải viết lại.
- Đa luồng: Với tính năng đa luồng của Java, có thể viết các chương trình có thể thực hiện đồng thời nhiều tác vụ. Tính năng thiết kế này cho phép các nhà phát triển xây dựng các ứng dụng tương tác có thể chạy trơn tru.
- Thông dịch: Mã bytecode của Java được dịch một cách nhanh chóng sang tập lệnh của máy thực thi chương trình và không được lưu trữ ở bất cứ đâu.
- Hiệu suất cao: Với việc sử dụng các trình biên dịch, thông dịch đúng lúc, Java cho phép tạo ra các ứng dụng có hiệu năng cao. Java nhanh hơn các ngôn ngữ lập trình thông dịch truyền thống khác vì mã bytecode của Java "gần" với mã mãy. Nó vẫn chậm hơn một chút so với ngôn ngữ biên dịch như C++. Java là một ngôn ngữ thông dịch, đó là lý do tại sao nó chậm hơn các ngôn ngữ biên dịch như C/C ++.
- Phân tán: Java được thiết kế cho môi trường phân tán của internet. RMI và EJB là hai bộ thư viện được sử dụng để tạo các ứng dụng phân tán. Tính năng này của Java làm cho chúng ta có thể gọi các phương thức từ bất kỳ máy nào trên internet.
- Động: Java là một ngôn ngữ động. Nó hỗ trợ tải động các lớp, tức là các lớp được tải theo yêu cầu. Nó cũng hỗ trợ gọi các chức năng từ các ngôn ngữ khác như C/C++. Java hỗ trợ biên dịch động và quản lý bộ nhớ tự động.

Một số loại ứng dụng được viết bởi Java:

- Ứng dụng Desktop
- Ứng dụng Web
- Ứng dụng di động
- Hệ thống nhúng

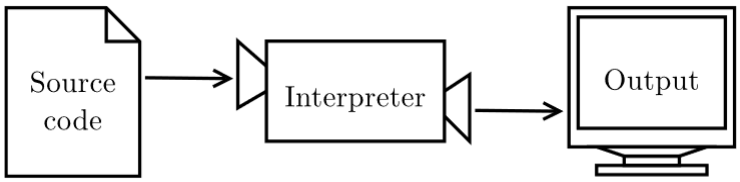
- Thẻ thông minh
- Hệ thống Robotic
- Game, ...

1.2. Mô hình biên dịch và thông dịch của Java

Các chương trình trong các ngôn ngữ lập trình bậc cao trước khi có thể chạy, chúng phải được dịch sang ngôn ngữ cấp thấp, còn được gọi là ngôn ngữ máy. Việc chuyển đổi này đôi khi là một nhược điểm nhỏ của các ngôn ngữ bậc cao. Tuy nhiên các ngôn ngữ bậc cao có hai ưu điểm:

- Dễ lập trình: Các chương trình được viết ra tốn ít thời gian lập trình, mã lệnh ngắn gọn, dễ đọc và có tính chính xác cao.
- Có thể chạy được trên nhiều nền tảng khác nhau mà chỉ cần một ít sửa đổi nhỏ hoặc không cần sửa đổi.

Có hai loại chương trình dịch để dịch ngôn ngữ bậc cao sang ngôn ngữ bậc thấp: trình thông dịch (interpreter) và trình biên dịch (compiler). Trình thông dịch đọc một chương trình bậc cao và thực hiện nó. Nó thực hiện chương trình theo từng lệnh một mà nó đã đọc, thay vì đọc toàn bộ các dòng và thực hiện tính toán. Hình 1.1 cho thấy cấu trúc của một trình thông dịch.



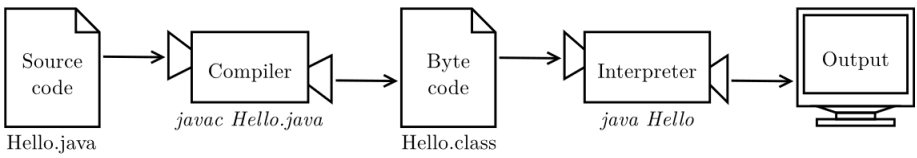
Hình 1.1. *Cấu trúc một chương trình thông dịch*

Ngược lại, một trình biên dịch đọc toàn bộ chương trình và dịch nó hoàn toàn trước khi chương trình bắt đầu chạy. Trong ngữ cảnh này, chương trình bậc cao được gọi là mã nguồn và chương trình đã dịch được gọi là mã đối tượng (object code) hoặc tệp thực thi. Khi một chương trình được biên dịch, chúng ta có thể chạy nó nhiều lần mà

không cần dịch lại. Kết quả là, các chương trình được biên dịch thường chạy nhanh hơn các chương trình thông dịch.

Java là ngôn ngữ vừa biên dịch và thông dịch. Thay vì dịch các chương trình trực tiếp sang mã máy, trình biên dịch Java tạo ra mã bytecode. Tương tự như mã máy, mã bytecode dễ dàng được thông dịch thành mã máy một cách nhanh chóng và dễ dàng. Và nhờ vào mã bytecode này, mà một chương trình có thể được biên dịch trên một máy và sau đó có thể được thông dịch và chạy trên máy khác. Trình thông dịch này được gọi là Máy ảo Java (JVM – Java Virtual Machine).

Hình 1.2 cho thấy các bước của quy trình này. Mặc dù có vẻ phức tạp, các bước này được tự động hóa cho chúng ta trong hầu hết các môi trường phát triển tích hợp (IDE – Intergrated Development Enviroment). Thông thường, chúng ta chỉ phải nhấn một nút hoặc gõ một lệnh duy nhất để biên dịch chương trình. Mặt khác, điều quan trọng là chúng ta cần phải biết những bước nào đang diễn ra, để nếu có sự cố xảy ra, chúng ta có thể tìm ra nó là gì và khắc phục nó.



Hình 1.2. *Mô hình biên dịch và thông dịch của chương trình Java*

1.3. Cấu trúc chương trình Java

Thông thường, chương trình đầu tiên chúng ta viết khi học một ngôn ngữ lập trình mới được gọi là chương trình `Hello World`. Tất cả những gì nó làm là hiển thị dòng chữ `Hello, World` trên màn hình. Trong Java, nó trông như thế này:

```
1 public class Hello {
2    public static void main(String[] args) {
3    //Xuất kết quả ra màn hình
4    System.out.println("Hello, World");
5    }
```

```
6 }
```

Khi chương trình này chạy, nó cho kết quả:

```
Hello, World
```

Các chương trình Java được tạo thành từ các định nghĩa lớp và phương thức, và các phương thức được tạo thành từ các câu lệnh. Một câu lệnh là một dòng mã thực hiện một hoạt động cơ bản. Trong chương trình `Hello World`, dòng sau là một câu lệnh nhằm hiển thị thông báo trên màn hình:

```
System.out.println("Hello, World");
```

`System.out.println` hiển thị kết quả trên màn hình và xuống hàng; tên `println` là viết tắt của dòng chữ `"print line"`. `"print"` có nghĩa là hiển thị trên màn hình hoặc gửi đến máy in. Trong giáo trình này, khi chúng tôi nói hiển thị, nghĩa là chúng tôi muốn xuất kết quả ra màn hình. Giống như hầu hết các câu lệnh, câu lệnh in kết thúc bằng dấu chấm phẩy (;).

Java là phân biệt chữ hoa chữ thường, có nghĩa là chữ hoa và chữ thường không giống nhau. Trong ví dụ này, `System` phải bắt đầu bằng một chữ cái viết hoa; trường hợp chúng ta viết `system` hoặc `SYSTEM` thì chương trình sẽ không hoạt động được.

Một phương thức là một chuỗi các câu lệnh được đặt tên. Chương trình này định nghĩa một phương thức có tên là main:

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

Tên và định dạng của phương thức main rất đặc biệt: khi chương trình chạy, nó bắt đầu ở câu lệnh đầu tiên trong main và kết thúc khi nó kết thúc câu lệnh cuối cùng.

Một lớp là một tập hợp các phương thức và thuộc tính. Chương trình này định nghĩa một lớp có tên `Hello`. Chúng ta có thể đặt cho một lớp bất kỳ tên nào (nhưng phải phù hợp với quy tắc đặt tên, sẽ được đề cập ở bên dưới), nhưng thông thường bắt đầu bằng chữ in hoa. Tên của lớp phải khớp với tên của tệp mà nó nằm trong, vì vậy lớp này phải nằm trong một tệp có tên `Hello.java`.

Java sử dụng cặp dấu ngoặc nhọn () để nhóm các câu lệnh lại với nhau. Trong `Hello.java`, cặp dấu ngoặc ngoài cùng chứa định nghĩa lớp `Hello` và cặp dấu ngoặc bên trong định nghĩa phương thức `main`.

Dòng bắt đầu bằng hai dấu gạch chéo (//) là một ghi chú, đó là một đoạn giải thích ngắn ý nghĩa của mã lệnh. Khi trình biên dịch nhìn thấy dấu //, nó bỏ qua mọi thứ từ đó cho đến hết dòng. Ghi chú không có tác dụng trong việc thực hiện chương trình, nhưng chúng giúp các lập trình viên khác (và bản thân của bạn trong tương lai) dễ hiểu hơn những gì bạn muốn làm.

1.4. Bộ từ khóa

Từ khóa là những từ có một ý nghĩa nhất định trong các ngôn ngữ lập trình. Ngôn ngữ lập trình Java có khoảng 50 từ khóa khác nhau và được viết bằng chữ thường. Chúng ta cần nhớ những từ khóa này vì chúng sẽ được dùng rất nhiều trong quá trình lập trình sau này.

- abstract: Khai báo lớp, phương thức, interface trừu tượng không có thể hiện (instance) cụ thể
- assert: Kiểm tra điều kiện đúng hay sai (thường dùng trong Unit Test)
- boolean: Kiểu dữ liệu logic với 2 giá trị: true, false.
- break: Thoát ra khỏi vòng lặp hoặc lệnh switch-case.
- byte: Kiểu byte với các giá trị nguyên chiếm 8bit (1byte).
- case: Trường hợp được tuyển chọn theo switch (chỉ được dùng khi đi kèm switch)
- catch: Được sử dụng để bắt ngoại lệ, được sử dụng cùng với try để xử lý các ngoại lệ xảy ra trong chương trình
- char: Kiểu ký tự Unicode, mỗi ký tự chiếm 16bit (2byte).
- class: Được sử dụng để định nghĩa lớp
- const: Chưa được sử dụng vì vậy chúng ta không thể dùng nó trong ngôn ngữ Java
- continue: Dừng chu trình lặp hiện tại và bắt đầu chu trình tiếp theo
- default: Mặc định được thực thi khi không có case nào trả về giá trị true (dùng trong switch-case)

- do: Dùng trong vòng lặp do... while
- double: Kiểu số thực với các giá trị biểu diễn theo dạng dấu phẩy động 64bit (8byte)
- else: Rẽ nhánh theo điều kiện ngược lại của if.
- enum: Định nghĩa kiểu dữ liệu enum
- extends: Được sử dụng để định nghĩa lớp con kế thừa các thuộc tính và phương thức từ lớp cha.
- final: Chỉ ra các biến, phương thức không được thay đổi sau khi đã được định nghĩa. Các phương thức final không thể được kế thừa và override
- finally: Thực hiện một khối lệnh đến cùng bất chấp các ngoại lệ có thể xảy ra. Được sử dụng trong try-catch
- float: Kiểu số thực với các giá trị biểu diễn theo dạng dấu phẩy động 32bit.
- for: Sử dụng cho vòng lặp for với bước lặp được xác định trước
- goto: Chưa được sử dụng
- if: Lệnh chọn theo điều kiện logic
- implements: Xây dựng một lớp mới cài đặt những phương thức từ interface xác định trước.
- import: Yêu cầu một hay một số lớp ở các gói chỉ định cần nhập vào để sử dụng trong ứng dụng hiện thời.
- instanceof: Kiểm tra xem một đối tượng nào đó có phải là một thể hiện của một lớp được định nghĩa trước hay không
- int: Kiểu số nguyên với các giá trị chiếm 32bit (4byte).
- interface: Được sử dụng để định nghĩa interface
- long: Kiểu số nguyên lớn với các giá trị chiếm 64bit (8byte).
- native: Giúp lập trình viên có thể sử dụng code được viết bằng các ngôn ngữ khác

- new: Khởi tạo đối tượng
- package: Xác định một gói sẽ chứa một số lớp ở trong file mã nguồn.
- private: Khai báo biến dữ liệu, phương thức riêng trong từng lớp và chỉ cho phép truy cập trong lớp đó.
- protected: Khai báo biến dữ liệu, phương thức chỉ được truy cập ở lớp cha và các lớp con của lớp đó.
- public: Khai báo lớp, biến dữ liệu, phương thức công khai có thể truy cập ở mọi nơi trong hệ thống.
- return: Kết thúc phương thức và trả về giá trị cho phương thức
- short: Kiểu số nguyên ngắn với các giá trị chiếm 16 bit (2 byte).
- static: Định nghĩa biến, phương thức của một lớp có thể được truy cập trực tiếp từ lớp mà không thông qua khởi tạo đôi tượng của lớp
- super: Biến chỉ tới đối tượng ở lớp cha
- switch: Sử dụng trong câu lệnh điều khiển switch case
- synchronized: Chỉ ra là ở mỗi thời điểm chỉ có một đối tượng hoặc một lớp có thể truy nhập đến biến dữ liệu, hoặc phương thức loại đó, thường được sử dụng trong lập trình đa luồng (multithreading)
- this: Biến chỉ tới đối tượng hiện thời.
- throw: Tạo một đối tượng exception để chỉ định một trường hợp ngoại lệ xảy ra
- throws: Chỉ định cho qua ngoại lệ khi exception xảy ra
- transient: Chỉ định rằng nếu một đối tượng được serialized, giá trị của biến sẽ không cần được lưu trữ
- try: Thử thực hiện cho đến khi gặp một ngoại lệ.
- void: Chỉ định một phương thức không trả về giá trị

- volatile: Báo cho chương trình dịch biết là biến khai báo volatile có thể thay đổi tùy ý trong các luồng (thread).
- while: Được sử dụng trong lệnh điều khiển while

1.5. Quy tắc đặt tên

Quy tắc đặt tên được áp dụng cho tên gói, tên lớp, tên phương thức, tên thuộc tính, tên biến, tên hằng, ... Quy tắc này được mô tả như sau:

- Phân biệt hoa thường
- Chỉ chứa các ký tự là chữ cái, số,
- Không chứa các ký tự đặc biệt, không chứa khoảng trắng, ngoài dấu "_" và "$".
- Không được bắt đầu bằng số
- Không được trùng với từ khóa

1.6. Ghi chú

Các ghi chú trong Java được sử dụng để cung cấp thêm thông tin hoặc giải thích về biến, phương thức, lớp, hoặc bất kỳ câu lệnh nào. Nó cũng có thể được sử dụng để tạm thời vô hiệu hóa một đoạn mã nào đó trong Java. Các ghi chú không được thực thi bởi các trình biên dịch và trình thông dịch.

Trong Java, để ghi chú, chúng ta có 3 cách:

1.6.1. Ghi chú trên một dòng

Cú pháp:

```
//Nội dung ghi chú
```

Ví dụ:

```
1 int i = 10; //Đây là một biến kiểu int
```

1.6.2. Ghi chú trên nhiều dòng

Cú pháp:

```
/*
Nội dung ghi chú thứ nhất
Nội dung ghi chú thứ hai
...
*/
```

Ví dụ:

```
1 /*
2 - Khai báo biến
3 - In giá trị của nó ra màn hình
4 */
5 int i = 10;
6 System.out.println(i);
```

1.6.3. Ghi chú tài liệu

Ghi chú dạng tài liệu là ghi chú được sử dụng để tạo ra tài liệu cho các API. Được dùng chủ yếu để giải thích ý nghĩa, tham số đầu vào, kết quả đầu ra của các API mà tác giả đã xây dựng. Sau đó, chúng ta có thể sử dụng công cụ javadoc để kết xuất tài liệu này ra dạng tài liệu HTML.

Cú pháp:

```
/**
Đây là ghi chú dạng tài liệu
*/
```

1.7. Biến

Biến là vùng nhớ dùng để lưu trữ dữ liệu khi chương trình chạy. Dữ liệu lưu trong một biến được gọi là giá trị của biến đó. Chúng ta có thể truy cập, gán hay thay đổi giá trị của các biến, khi biến được gán một giá trị mới, giá trị cũ sẽ bị ghi đè. Mỗi biến gắn liền với một kiểu dữ liệu và một định danh duy nhất gọi là tên biến. Trong java, biến có

thể được khai báo ở bất kỳ nơi đâu trong chương trình.

Cách khai báo:

<kiểu_dữ_liệu> <tên_biến>;

<kiểu_dữ_liệu> <tên_biến> = <giá_trị>;

Gán giá trị cho biến:

<tên_biến> = <giá_trị>;

Chẳng hạn:

```
1 int x, y = 10;
2 x = 20;
```

Trong Java có 3 loại kiểu biến bao gồm:

- Biến local
- Biến instance
- Biến static

1.7.1. Biến local (biến địa phương)

Biến local được khai báo trong các phương thức, hàm tạo hoặc trong các khối lệnh . Đặc điểm của biến local:

- Biến local sẽ bị phá hủy khi kết thúc các phương thức, hàm tạo và khối lệnh.
- Không được sử dụng từ khóa chỉ mức độ truy cập (access modifier) khi khai báo biến local.
- Các biến local được lưu trên vùng nhớ stack của bộ nhớ.
- Cần khởi tạo giá trị mặc định cho biến local trước khi có thể sử dụng.

Ví dụ: Biến sum bên dưới là một biến local

```
1 int cong2So(int n1, int n2){
2    int sum = 0;
3    sum = n1 + n2;
```

```
4    return sum;
5 }
```

1.7.2. Biến instance (biến toàn cục)

Biến instance được khai báo trong một lớp, bên ngoài các phương thức, hàm tạo và các khối lệnh (). Đặc điểm của loại biến này:

- Biến instance được lưu trong bộ nhớ heap.
- Biến instance được tạo khi một đối tượng được tạo bằng việc sử dụng từ khóa `"new"` và sẽ bị phá hủy khi đối tượng bị phá hủy.
- Biến instance có thể được sử dụng bởi các phương thức, hàm tạo, khối lệnh, ... Nhưng nó phải được sử dụng thông qua một đối tượng cụ thể.
- Chúng ta được phép sử dụng từ khóa chỉ mức độ truy cập khi khai báo biến instance, mặc định là "private".
- Biến instance có giá trị mặc định phụ thuộc vào kiểu dữ liệu của nó. Ví dụ nếu là kiểu `int, short, byte` thì giá trị mặc định là 0, kiểu `double` thì là `0.0d`, ... Vì vậy, chúng ta sẽ không cần khởi tạo giá trị cho biến instance trước khi sử dụng.
- Bên trong lớp mà chúng ta khai báo biến instance, chúng ta có thể gọi nó trực tiếp bằng tên khi sử dụng ở khắp nới bên trong lớp đó.

Cú pháp:

```
1 class <Tên_Lớp>
2 {
3    <mức_độ_truy_cập> <kiểu_dữ_liệu> <biến_instanse_1>;
4    <mức_độ_truy_cập> <kiểu_dữ_liệu> <biến_instanse_2>;
5 }
```

1.7.3. Biến static

Biến static được khai báo trong một lớp với từ khóa `"static"`, phía bên ngoài các phương thức, hàm tạo và khối lệnh. Đặc điểm của loại biến này:

- Sẽ chỉ có duy nhất một bản sao của các biến static được tạo ra, dù bạn tạo bao nhiêu đối tượng từ lớp tương ứng.
- Biến static được lưu trữ trong bộ nhớ static riêng.
- Biến static được tạo khi chương trình bắt đầu chạy và chỉ bị phá hủy khi chương trình dừng.
- Giá trị mặc định của biến static phụ thuộc vào kiểu dữ liệu chúng ta khai báo tương tự biến instance.
- Biến static được truy cập thông qua tên của class chứa nó, với cú pháp: TenClass.tenBien
- Trong lớp, các phương thức sử dụng biến static bằng cách gọi tên của nó khi phương thức đó cũng được khai báo với từ khóa `"static"`.

Ví dụ:

```
class <Tên_Lớp>
{
   <mức_độ_truy_cập> static <kiểu_dữ_liệu> <biến_static_1>;
   <mức_độ_truy_cập> static <kiểu_dữ_liệu> <biến_static_2>;
}
```

1.8. Kiểu dữ liệu

Kiểu dữ liệu trong Java chia làm 2 nhóm:

- Kiểu dữ liệu cơ bản (hay còn gọi là nguyên thủy): `byte, short, int, long, float, double, boolean và char`
- Kiểu dữ liệu tham chiếu: `String, Array, Class`

Một kiểu dữ liệu cơ bản có kích thước cố định và không có phương thức bổ sung. Kiểu dữ liệu dạng số chia làm 2 nhóm:

- Kiểu số nguyên:`byte, short, int và long`
- Kiểu số thực: `float và double`.

Có tám kiểu dữ liệu nguyên thủy trong Java:

Kiểu	Kích thước	Vùng
byte	1 byte	-128 ⇒ 127
short	2 bytes	-32,768 ⇒ 32,767
int	4 bytes	-2,147,483,648 ⇒ 2,147,483,647
long	8 bytes	-9,223,372,036,854,775,808 ⇒ 9,223,372,036,854,775,8
float	4 bytes	Độ chính xác đơn. Lưu 6 to 7 chữ số thập phân
double	8 bytes	Độ chính xác kép. Lưu 15 chữ số thập phân
boolean	1 bit	True/False
char	2 bytes	Lưu 1 ký tự đơn trong bảng mã Unicode

1.9. Hằng

Hằng là một giá trị bất biến trong chương trình, nghĩa là chúng ta không thể dùng phép gán để gán lại giá trị cho hằng sau khi đã định nghĩa. Một số đặc điểm của hằng trong Java:

- Tên hằng được đặt theo qui ước giống như tên biến.
- Hằng số nguyên: trường hợp giá trị hằng ở dạng long ta thêm vào cuối chuỗi số chữ "l" hay "L". Chẳng hạn: 1L
- Hằng số thực: trường hợp giá trị hằng có kiểu float ta thêm tiếp vị ngữ "f" hay "F", còn kiểu số double thì ta thêm tiếp vị ngữ "d" hay "D". Chẳng hạn: 2.3f
- Hằng Boolean: java có 2 hằng boolean là `true, false`.
- Hằng ký tự: là một ký tự đơn nằm giữa nằm giữa 2 dấu ngoặc đơn (chẳng hạn: 'a'). Một số hằng ký tự đặc biệt:

  `\b: Xóa lùi (BackSpace)`
  `\t: Tab`

```
\n: Xuống hàng
\r: Dấu enter
\": Nháy kép
\': Nháy đơn
\\: Số ngược
\f: Đẩy trang
\uxxxx: Ký tự unicode
```

- Hằng chuỗi: là tập hợp các ký tự được đặt giữa hai dấu nháy kép "". Một hằng chuỗi không có ký tự nào là một hằng chuỗi rỗng. Chẳng hạn: "Hello, world!".

Cú pháp:

```
final <kiểu_dữ_liệu> <tên_hằng> = <giá_trị_hằng>;
```

Ví dụ:

```
final double PI = 3.1415926535897;
```

1.10. Các phép toán cơ bản

1.10.1. Phép toán số học

Java hỗ trợ năm phép toán số học sau:
(bảng)

Các phép toán này chỉ áp dụng được cho các biến kiểu cơ bản và không áp dụng được cho các kiểu tham chiếu. Lưu ý với phép chia được thực hiện cho hai giá trị kiểu nguyên sẽ cho kết quả là thương nguyên.

Chẳng hạn: biểu thức `10/3` cho kết quả bằng `1`, còn `3/5` cho kết quả bằng `0`.

1.10.2. Phép toán tăng giảm

(bảng)

Các phép toán ++ và – có đặt trước hoặc sau biến và sự khác nhau ở chỗ:

- Nếu đặt trước biến a thì tăng/giảm giá trị biến a rồi mới sử dụng giá trị của a
- Nếu đặt sau biến a thì sử dụng giá trị của biến a rồi mới tiến hành tăng/giảm giá trị của a.

Chẳng hạn:

```
int a = 5;
int x = ++a; // thì x = 6, a = 6
//còn
int x = a++; // thì x = 5, a = 6
```

1.10.3. Toán tử trên bit

Toán tử	Ý nghĩa	Dạng thức
&	AND	a & b
|	OR	a | b
$<<$	Dịch trái	a $<<$ b
$>>$	Dịch phải	a $>>$ b
$>>>$	Dịch phải và điền 0 vào bit trống	a $>>>$ b

Các toán tử xử lý bit là các toán tử chỉ thực hiện trên các toán hạng có kiểu dữ liệu là số nguyên. Bảng sau cho kết quả của các phép toán:

a	b	a	a&b	a|b	a$\hat{}$b
0	0	1	0	0	0
0	1	1	0	1	1
1	0	0	0	1	1
1	1	0	1	1	0

1.10.4. Phép toán quan hệ & logic

Toán tử	Ý nghĩa	Dạng thức
==	So sánh bằng	a == b
!=	So sánh khác	a != b
>	So sánh lớn hơn	a > b
<	So sánh nhỏ hơn	a < b
>=	So sánh lớn hơn hay bằng	a >= b
<=	So sánh nhỏ hơn hay bằng	a <= b
||	Phép hoặc	a || b
&&	Phép giao	a && b
!	Phủ định	!a

Chú ý: Kết quả của phép toán quan hệ và logic là True/False.

Ý nghĩa của phép toán logic được cho ở bảng sau:

a	b	!a	a&&b	a||b
true	true	false	true	true
true	false	false	false	true
false	true	true	false	true
false	false	true	false	false

1.10.5. Phép toán điều kiện

Cú pháp:

`<điều_kiện> ? <biểu_thức_1> : < biểu_thức_2>`

Nếu điều kiện đúng thì có giá trị, hay thực hiện `<biểu_thức_1>`, còn ngược lại là `<biểu_thức_2>`. Trong đó:

- `<điều_kiện>`: là một biểu thức logic
- `<biểu_thức_1>, <biểu_thức_2>`: có thể là hai giá trị, hai biểu thức hoặc hai hành động.

Chẳng hạn: `z = ((x < y) ? (x) : (y));` sẽ gán giá trị nhỏ nhất của `x và y cho z`.

1.10.6. Phép gán

Phép gán là cách gắn một giá trị cho một biến hoặc thay đổi giá trị của một biến. Lệnh gán trong Java có công thức: `biến = biểu_thức;`

Trong đó, dấu bằng (=) được gọi là dấu gán hay toán tử gán, biểu thức ở vế phải dấu gán được tính rồi lấy kết quả gán cho biến nằm ở vế trái.

Biểu thức tại vế phải có thể là một giá trị trực tiếp, một biến, hoặc một biểu thức phức tạp. Ngoài ra chúng ta có thể kết hợp phép gán với các phép toán khác theo cách gộp như ví dụ sau:

- a += b; tương đương với a = a + b;
- a -= b; tương đương với a = a - b;
- a *= b; tương đương với a = a * b;
- a /= b; tương đương với a = a / b;
- a %= b; tương đương với a = a % b;

1.10.7. Chuyển đổi kiểu

Trong một biểu thức, các toán hạng có thể có các kiểu dữ liệu khác nhau, để việc tính toán được chính xác, đôi lúc cần phải chuyển đổi kiểu dữ liệu cho phù hợp. Có 2 cách chuyển đổi kiểu dữ liệu:

- Chuyển kiểu tự động:
  Khi 2 toán hạng trong một phép toán có kiểu dữ liệu khác nhau thì toán hạng có kiểu dữ liệu bé hơn sẽ tự động chuyển thành kiểu dữ liệu lớn hơn của toán hạng kia trước khi thực hiện phép toán. Đối với lệnh gán thì sự chuyển kiểu căn cứ vào kiểu dữ liệu của biến ở vế trái.
- Chuyển kiểu ép buộc:
  Thực hiện theo cú pháp sau: `(kiểu) biểu_thức`. Nếu chuyển từ kiểu dữ liệu nhỏ sang lớn thì sẽ không làm mất mát thông tin. Ngược lại nó sẽ mất mát một phần thông tin.

Chẳng hạn:

```
float a = 10.3f;
int b = (int)a;
```

Lúc này b sẽ có giá trị là 10.

1.11. Các cấu trúc điều khiển

1.11.1. Cấu trúc if

Cú pháp:

```
if (<điều_kiện_exp>) {
   <khối_lệnh_1>;
} [else {
   <khối_lệnh_2>;
}]
```

Sơ đồ thực hiện:

Ý nghĩa: Nếu biểu thức điều kiện exp là True thì chương trình sẽ thực hiện `khối_lệnh_1`, ngược lại thì thực hiện `khối_lệnh_2` (nếu có phần else). Các câu lệnh sau if hoặc else có thể là một khối lệnh. Nếu các câu lệnh if lồng nhau trong phạm vi một khối thì else sẽ đi với if gần nó nhất.

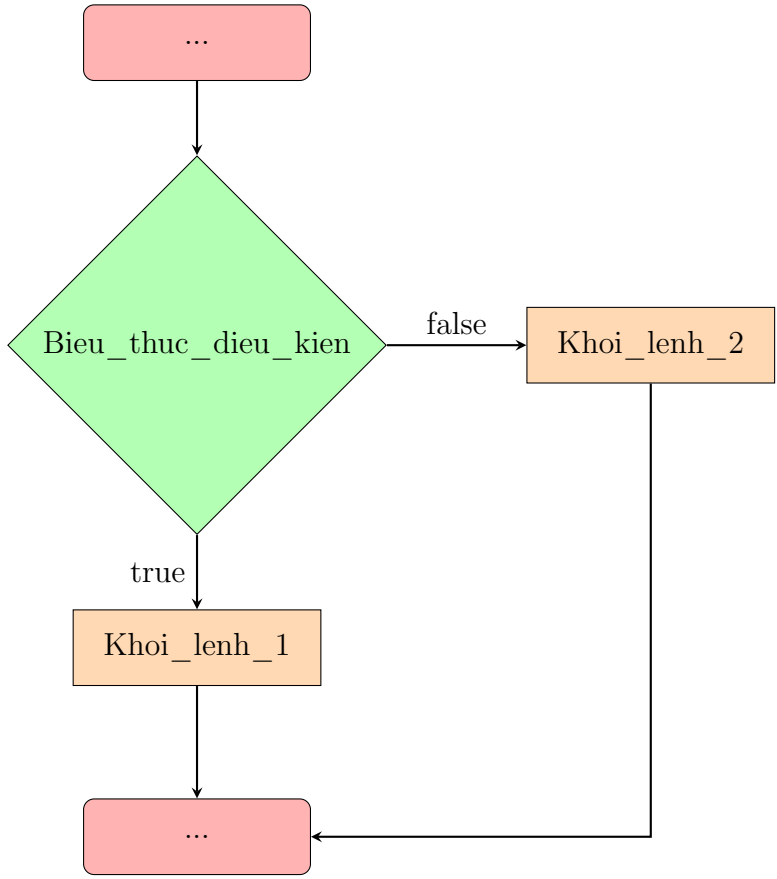
Ví dụ 1:

```
int a = 20, b = 10, c = 0;
c = a;
System.out.print("So lon nhat la: ");
if (a < b) {
   c = b;
}
System.out.print(c);
```

Kết quả:

```
So lon nhat la 20
```

Ví dụ 2:



Hình 1.3. *Sơ đồ thực hiện cấu trúc if*

```
int number = 13;
if (number % 2 == 0) {
   System.out.println("So " + number + " la so chan");
} else {
   System.out.println("So " + number + " la so le.");
}
```

Kết quả:

```
So 13 la so le
```

1.11.2. Cấu trúc switch

Cú pháp:

```
switch (<biểu_thức>) {
   case <giá_trị_1>:
      <khối_lệnh_1>;
      break;
      ...
   case <giá_trị_n>:
      <khối_lệnh_n>;
      break;
   default:
      <khối_lệnh_default>;
}
```

Ý nghĩa: Nếu `<biểu_thức>` có giá trị bằng `giá_trị_k` nào đó thì chương trình sẽ chuyển điều khiển đến `case giá_trị_k` và thực hiện `<khối_lệnh_k>` cho đến khi gặp lệnh `break` mới thoát khỏi cấu trúc `switch`. Ngược lại nếu giá trị của `<biểu_thức>` không có giá trị trong các giá trị được liệt kê sau `case` thì chương trình sẽ thực hiện `<khối_lệnh_default>` (nếu có).

Trong đó:

`<biểu_thức>, <giá_trị_1>, ..., <giá_trị_n>`: phải có giá trị nguyên

Ví dụ:

```
1  int day = 4;
2  switch (day) {
3     case 1:
4        System.out.println("Thu hai");
5        break;
6     case 2:
7        System.out.println("Thu ba");
8        break;
9     case 3:
10       System.out.println("Thu tu");
11       break;
```

```
12    case 4:
13       System.out.println("Thu nam");
14       break;
15    case 5:
16       System.out.println("Thu sau");
17       break;
18    case 6:
19       System.out.println("Thu bay");
20       break;
21    case 7:
22       System.out.println("Chu nhat");
23       break;
24 }
```

1.11.3. Cấu trúc for

Cú pháp:

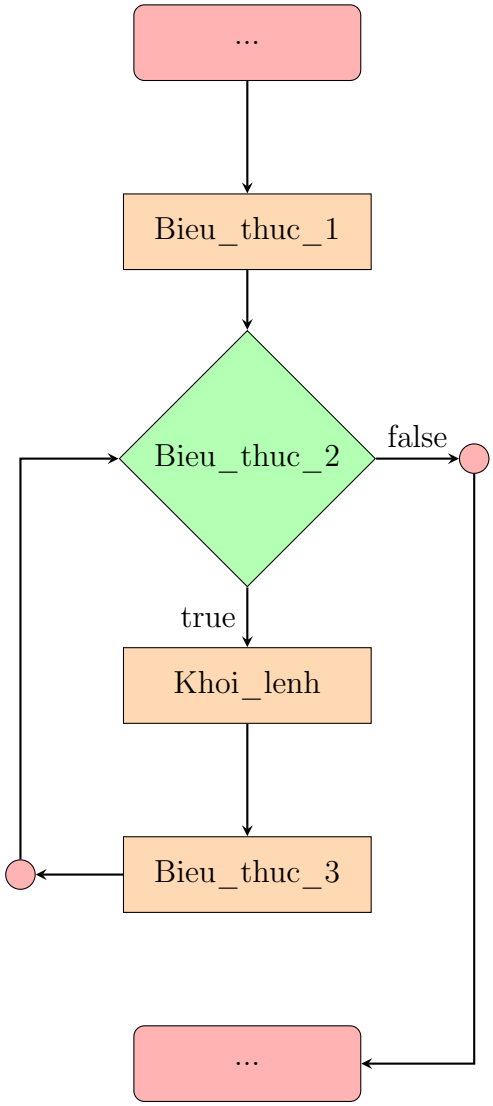
```
for (<biểu_thức_1>;<biểu_thức_2>;<biểu_thức_3>)
   <khối_lệnh>;
```

Sơ đồ thực hiện (hình 1.4):

Ý nghĩa:

- Đầu tiên `biểu_thức_1` được thực hiện. Đây thường là phép khởi gán để khởi động một hay nhiều biến. `Biểu_thức_1` chỉ được thực hiện 1 lần duy nhất khi bắt đầu thực hiện cấu trúc.
- Tiếp theo, `biểu_thức_2` được thực hiện. Đây là phần điều kiện thực hiện `khối_lệnh` của cấu trúc for. Nếu `biểu_thức_2` sai thì chương trình sẽ thoát khỏi cấu trúc for. Nếu `biểu_thức_2` đúng thì chương trình sẽ thực hiện `khối_lệnh`.
- Sau khi thực hiện `khối_lệnh` thi các biến tăng/giảm ở `biểu_thức_3` mới được thực hiện và cấu trúc sẽ lặp lại việc kiểm tra `biểu_thức_2`.

Ví dụ:



Hình 1.4. *Sơ đồ thực hiện cấu trúc for*

```
for (int i = 0; i <= 10; i = i + 2) {
  System.out.println(i);
}
```

Kết quả:

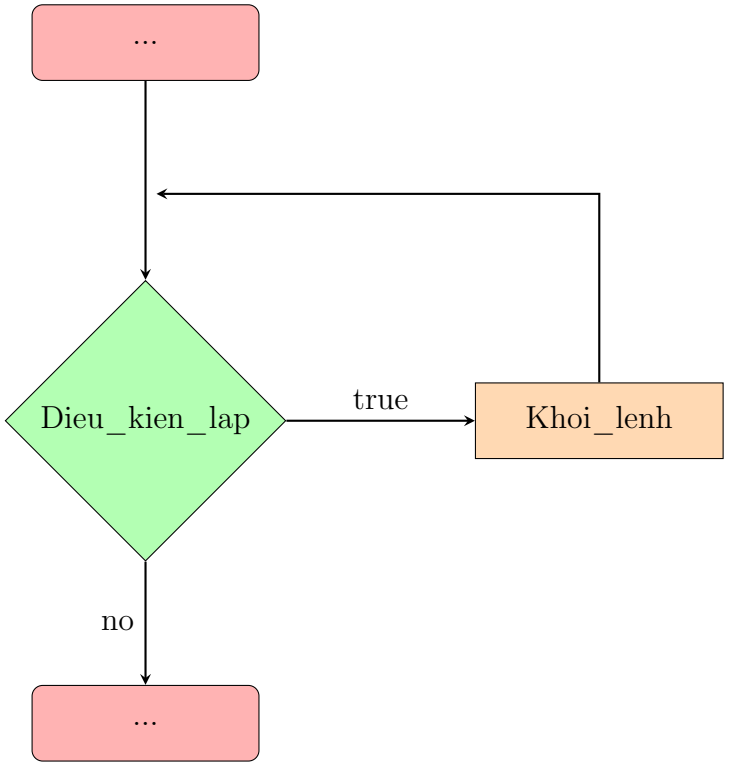
```
0
2
4
6
8
10
```

1.11.4. Cấu trúc while

Cú pháp:

```
while (<điều_kiện_lặp>)
    <khối_lệnh>;
```

Sơ đồ thực hiện (hình 1.5):



Hình 1.5. *Sơ đồ thực hiện cấu trúc while*

Ý nghĩa: Trong khi `điều_kiện_lặp` vẫn còn đúng thì thực hiện `khối_lệnh`.

Ví dụ:

```
int i = 0;
while (i < 5) {
   System.out.println(i);
   i++;
}
```

Kết quả:

```
0
1
2
3
4
```

1.11.5. Cấu trúc do...while

Cú pháp: `do`
`<khối_lệnh>;`
`while (<điều_kiện_lặp>);`

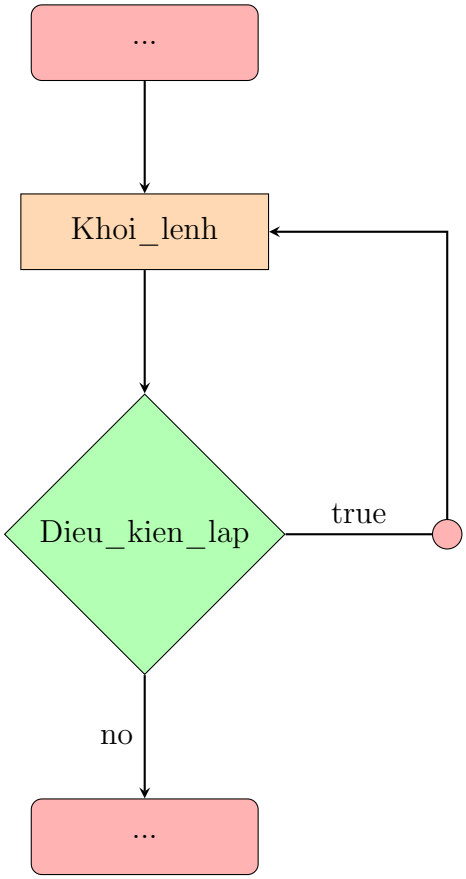
Sơ đồ thực hiện (hình 1.6):

Ý nghĩa: Cấu trúc `do...while` sẽ thực hiện `khối_lệnh` cho đến khi nào `điều_kiện_lặp` là sai.

Ví dụ:

```
int i = 0;
do {
   System.out.println(i);
   i++;
} while (i < 5);
```

Kết quả:



Hình 1.6. *Sơ đồ thực hiện cấu trúc do... while*

```
0
1
2
3
4
```

1.11.6. Cấu trúc lệnh nhảy

Lệnh break: trong cấu trúc switch chúng ta dùng câu lệnh break để thoát thỏi cấu trúc switch trong cùng chứa nó. Tương tự như vậy, trong cấu trúc lặp, câu lệnh break dùng để thoát khỏi cấu trúc lặp trong cùng chứa nó.

Lệnh continue: dùng để ngắt vòng lặp hiện tại và tiếp tục vòng lặp tiếp theo.

1.12. Các lớp bao kiểu dữ liệu cơ bản

Đôi khi, ta muốn đối xử với một giá trị kiểu cơ bản như là một đối tượng. Trong trường hợp như vậy, ta có các lớp bọc ngoài mỗi kiểu cơ bản (wrapper class). Các lớp bọc ngoài này có tên gần trùng với tên kiểu cơ bản tương ứng: `Boolean, Character, Byte, Short, Integer, Long, Float, Double`. Mỗi đối tượng thuộc các lớp trên bao bọc một giá trị kiểu cơ bản tương ứng, kèm theo các phương thức để thao tác với giá trị đó. Ví dụ:

```
1 //Tạo một đối tượng bọc ngoài 1 giá trị
2 int i = 10;
3 Integer iWrap = new Integer(i);
4 //Lấy giá trị bên trong đối tượng
5 int iUnwrap = iWrap.intValue();
6 //Biến chuỗi số thành số
7 Int i = Integer.parseInt("10");
8 //Biến số thành chuỗi số
9 String iString = iWrap.toString();
```

Các lớp bọc ngoài khác cũng có cách sử dụng và phương thức tương tự như `Integer`. Lúc này việc sử dụng kiểu dữ liệu cơ bản và đối tượng của lớp bọc kiểu cơ bản sẽ có một vài trường hợp sử dụng như sau:

- Đối số của phương thức: dù một phương thức khai báo tham số kiểu cơ bản hay kiểu lớp bọc ngoài thì nó vẫn chấp nhận đối số ở cả dạng cơ bản cũng như kiểu lớp bọc ngoài.
- Giá trị trả về: dù một phương thức khai báo kiểu trả về kiểu cơ bản hay bọc ngoài thì lệnh return trong phương thức dùng giá trị ở cả dạng cơ bản cũng như bọc ngoài đều được.
- Biểu thức boolean: ở những vị trí yêu cầu một biểu thức boolean, ta có thể dùng biểu thức cho giá trị boolean (chẳng hạn $2 < a$), hoặc một biến boolean, hoặc một tham chiếu kiểu Boolean đều được. Phép toán số học: ta có thể dùng tham chiếu kiểu bọc ngoài làm toán hạng của các phép toán số học, kể cả phép ++.

- Phép gán: ta có thể dùng một tham chiếu kiểu bọc ngoài để gán trị cho một biến kiểu cơ bản và ngược lại. Ví dụ: `Double d = 10.0;`

1.13. Mảng

1.13.1. Khái niệm

Mảng là tập hợp nhiều phần tử có cùng tên, cùng kiểu dữ liệu và mỗi phần tử trong mảng được truy xuất thông qua chỉ số của nó trong mảng.

1.13.2. Khai báo mảng

Java cũng như các ngôn ngữ khác có 2 loại mảng: Mảng 1 chiều và mảng nhiều chiều.

Cú pháp khai báo mảng 1 chiều:

`<kiểu dữ liệu> <tên mảng>[];` hoặc

`<kiểu dữ liệu>[] <tên mảng>;`

Ví dụ:

`int arrInt[];` hoặc

`int[] arrInt;`

Cú pháp khai báo mảng nhiều chiều:

`<Kiểu dữ liệu>[][]...[] <Tên mảng>;` hoặc

`<Kiểu dữ liệu> <Tên mảng> [][]...[];`

Ví dụ:

```
1 int a[][];
2 int[][] a;
```

1.13.3. Cấp phát bộ nhớ cho mảng

Không giống như trong C/C++ kích thước của mảng được xác định khi khai báo. Chẳng hạn như: `int arrInt[100];`. Để cấp phát bộ nhớ cho mảng trong Java ta cần dùng từ khóa `new`. Chẳng hạn để cấp phát vùng nhớ cho mảng trong Java ta làm như sau:

```
int arrInt = new int[100];
int arrInt2Dim[][]= new int[2][3];
```

1.13.4. Khởi tạo mảng

Chúng ta có thể khởi tạo giá trị ban đầu cho các phần tử của mảng khi nó được khai báo.

Ví dụ:

```
int arrInt[] = {1, 2, 3};
char arrChar[] = {'a', 'b','c'};
String arrString[] = {"ABC", "EFG", "GHI"};
int a2Dim[][]={{3,4},{2,8}};
```

1.13.5. Truy cập mảng

Chỉ số mảng trong Java bắt đầu tư 0. Vì vậy phần tử đầu tiên có chỉ số là 0, và phần tử thứ n có chỉ số là n-1. Các phần tử của mảng được truy xuất thông qua chỉ số của nó đặt giữa cặp dấu ngoặc vuông ([]).

Ví dụ:

```
int arrInt[] = {1, 2, 3};
int x = arrInt[0]; // x sẽ có giá trị là 1.
int y = arrInt[1]; // y sẽ có giá trị là 2.
int z = arrInt[2]; // z sẽ có giá trị là 3.
```

1.13.6. Duyệt mảng

Để duyệt mảng, ngoài việc sử dụng các cấu trúc lặp cổ điển như `for, while hay do...while` thì trong Java còn hỗ trợ cấu trúc `for-each` cho phép duyệt mảng rất tiện lợi.

Cú pháp:

```
for (<kiểu_dữ_liệu> <tên_biến>: <tên_mảng>)
    <khối_lệnh>;
```

Ví dụ:

```
int arr[] = { 10, 20, 30, 40 };
for (int i : arr) {
   System.out.println(i);
}
```

Kết quả:

```
10
20
30
40
```

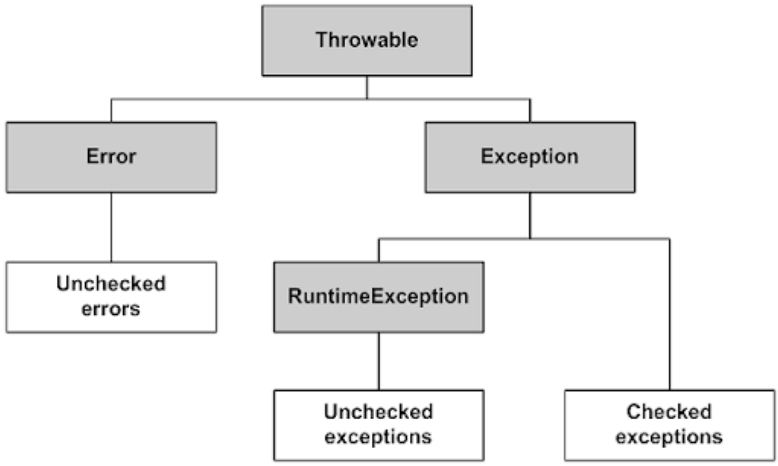
1.14. Xử lý ngoại lệ

Một ngoại lệ (Exception) trong Java là một vấn đề phát sinh trong quá trình thực thi chương trình. Khi xảy ra ngoại lệ, luồng xử lý (flow) bị gián đoạn, chương trình/ứng dụng dừng bất thường. Nó là một đối tượng được ném ra tại Runtime. Ngoại lệ trong Java có thể xảy ra vì nhiều lý do khác nhau:

- Nhập dữ liệu không hợp lệ.
- Không tìm thấy file cần mở.
- Kết nối mạng bị ngắt trong quá trình thực hiện giao tác.
- JVM hết bộ nhớ.
- Truy cập vượt ngoài chỉ số của mảng, ...

Ngoại lệ xảy ra có thể do người dùng, lập trình viên hoặc số khác do tài nguyên bị lỗi. Java Exeption được triển khai bằng cách sử dụng các lớp như `Throwable, Exception, RuntimeException` và các từ khóa như `throw, throws, try, catch và finally`.

Dựa vào tính chất các ngoại lệ, người ta chia ngoại lệ thành ba loại như hình 1.7:



Hình 1.7. *Cấu trúc của các ngoại lệ*

- Ngoại lệ được kiểm tra (Checked Exceptions): Là một ngoại lệ được kiểm tra và thông báo bởi trình biên dịch tại thời điểm biên dịch, chúng cũng có thể được gọi là ngoại lệ thời gian biên dịch (Compile-time Exceptions).
- Ngoại lệ không được kiểm tra (Unchecked Exceptions): Là một ngoại lệ không được kiểm tra trong quá trình biên dịch. Chúng cũng được gọi là ngoại lệ thời gian chạy (Runtime Exceptions)
- Lỗi (Error): Error là những vấn đề nghiêm trọng liên quan đến môi trường thực thi của ứng dụng hoặc hệ thống mà lập trình viên không thể kiểm soát. Nó thường làm chết chương trình. Ví dụ: OutOfMemoryError, VirtualMachineError, and StackOverflowError, ...

Xử lý ngoại lệ trong Java là một cơ chế mạnh mẽ để xử lý các lỗi runtime (Runtime exeption) để có thể duy trì luồng bình thường của ứng dụng. Khối lệnh try trong java được sử dụng để chứa một đoạn code có thế xảy ra một ngoại lệ. Nó phải được khai báo trong phương thức. Sau một khối lệnh try bạn phải khai báo khối lệnh catch hoặc finally hoặc cả hai.

1.14.1. Khối lệnh try-catch trong Java

Cú pháp:

```
try {
     // code có thể ném ra ngoại lệ
} catch (Exception_class_Name ref) {
     // code xử lý ngoại lệ
}
```

Khối catch trong Java được sử dụng để xử lý các Exception. Nó phải được sử dụng sau khối try. Chúng ta có thể sử dụng nhiều khối catch với một khối try duy nhất.

1.14.2. Khối lệnh try-finally trong Java

Cú pháp:

```
try {
     // code có thể ném ra ngoại lệ
} finally {
     // code trong khối này luôn được thực thi
}
```

Khối lệnh finally trong Java luôn được thực thi cho dù có ngoại lệ xảy ra hay không hoặc gặp lệnh return trong khối try. Khối lệnh finally trong Java được khai báo sau khối lệnh try hoặc sau khối lệnh catch.

1.15. Hướng dẫn Cài đặt JDK và Eclipse

1.15.1. Hướng dẫn Cài đặt JDK

- Bước 1. Tải JDK
  Truy cập vào địa chỉ: https://www.oracle.com/java/technologies/javase-downloads.html và chọn phiên bản JDK mà bạn muốn và nhấn chọn nút JDK Download (Giả sử chúng ta chọn Java SE 11 (LTS))
- Bước 2. Cài đặt JDK
  Sau khi tải JDK về, chúng ta tiến hành cài đặt bằng cách double-click vào chương trình mà chúng ta đã tải về và làm theo hướng dẫn trên màn hình cài đặt

- Bước 3. Thiết lập biến môi trường
  Giả sử JDK chúng ta cài đặt tại thư mục: C:/Java/JDK11, thì lúc này ta có thể thiết lập biến môi trường như sau:
  - Vào Control panel và chọn mục System
  - Nhấn Advanced System settings và chọn Environment Variables
  - Thêm thư mục bin trong thư mục JDK mà chúng ta đã cài đặt (C:/Java/JDK11/bin) vào biến PATH trong System Variables

  Lưu ý các đường dẫn thư mục trong PATH phải cách nhau bởi dấu ; và không phân biệt HOA thường.

- Bước 4. Kiểm tra cài đặt
  Để chắc chắn việc cài đặt ở trên là đúng đắn, chúng ta mở cửa sở Terminal của Window và gõ: `java - version`. Nếu kết quả trả về là phiên bản của JDK mà chúng ta đã cài thì việc cài đặt JDK đã hoàn tất.

1.15.2. Hướng dẫn Cài đặt Eclipse

- Bước 0. Để lập trình bằng ngôn ngữ Java thì chúng ta phải chắc chắn là đã cài JDK ở bước trước.

- Bước 1. Truy cập vào địa chỉ: https://www.eclipse.org/downloads/ và chọn nút "Download Packages" thay vì chọn nút "Download x86_64"

- Bước 2. Ở trang tiếp theo, chúng ta chọn "Windows x86_64" ở mục Eclipse IDE for Java Developers" và tiến hành Download

- Bước 3. Để chạy Eclipse, đơn giản là chúng ta tiến hành giải nén tập tin vừa mới download về và đặt ở thư mục phù hợp, chẳng hạn: C:
  Eclipse. Vì chương trình Eclipse mà chúng ta vừa tải chỉ là thư mục nén và không hề chạy chương trình để cài đặt lên máy tính, do đó chúng ta có thể đổi tên, di chuyển sang thư mục khác mà không ảnh hưởng gì.

1.15.3. Tạo chương trình đầu tiên, chương trình "Hello world"

- Khởi động Eclipse bằng cách nhấn vào file Eclipse.exe trong thư mục mà chúng ta vừa giải nén
- Chọn thư mục làm Workspace, nơi lưu trữ các chương trình của chúng ta
- Nếu xuất hiện màn hình Welcome thì chúng ta có thể đóng nó bằng cách nhấn nút Close ở thanh tiêu đề
- Để tạo dự án, chúng ta chọn menu "File" ⇒ "New" ⇒ "Java project" (Hoặc "File" ⇒ "New" ⇒ "Project" ⇒ "Java project").
- Cửa sổ "New Java Project" xuất hiện:
  - Trong "Project name", chúng ta gõ HelloWorld
  - Tích vào tùy chọn: Use default location
  - Trong JRE, chúng ta chọn "Use default JRE (currently 'JDK11.0.x')"
  - Trong "Project Layout" chúng ta nhấn vào tùy chọn "Use project folder as root for sources and class files"
  - Nhấn nút Finish để hoàn thành tạo mới dự án
  - Trong cửa sổ "Create module-info.java", chúng ta chọn "Don't"
- Trong khung "Package Explorer" nằm bên trái, nhấn chuột phải lên HelloWorld (hoặc trong menu File) chọn ⇒ New ⇒ Class. Cửa sổ "New Java Class" xuất hiện:
  - Trong "Source folder", vẫn để "HelloWorld".
  - Trong mục "Package" thì chúng ta tạm thời để rỗng
  - Trong mục "Name", chúng ta gõ Hello
  - Tích chọn "public static void main(String[] args)" để hệ thống sinh ra hàm main trong lớp Hello
  - Nhấn nút Finish để hoàn thành tạo lớp
  - File "Hello.java" trong khung soạn thảo sẽ trông như sau:

```
public class Hello {
   public static void main(String[] args) {
       System.out.println("Hello, world!");
```

```
    }
}
```

- Biên dịch và thực thi chương trình:
  - Để chạy chương trình, chúng ta nhấn chuột phải vào file Hello.java (Hoặc chọn menu Run) ⇒ Runas ⇒ Java Application
  - Trong khung Console sẽ xuất hiện Hello, World!
  - Như vậy việc cài đặt JDK và IDE Eclipse đã thành công.

1.16. Bài tập

Bài 1: Viết chương trình tìm ước số chung lớn nhất, bội số chung nhỏ nhất của hai số tự nhiên a và b.

Bài 2: Viết chương trình chuyển đổi một số tự nhiên ở hệ cơ số 10 thành số ở hệ cơ số b bất kì ($1 < b \leq 36$).

Bài 3: Hãy viết chương trình tính tổng các chữ số của một số nguyên bất kỳ. Ví dụ: Số 8545604 có tổng các chữ số là: $8+5+4+5+6+0+4=32$.

Bài 4: Viết chương trình phân tích một số nguyên thành các thừa số nguyên tố. Ví dụ: Số 28 được phân tích thành 2 x 2 x 7

Bài 5: Viết chương trình phân tích một số nguyên thành các thừa số nguyên tố. Ví dụ: Số 28 được phân tích thành 2 x 2 x 7

Bài 6: Viết chương trình liệt kê tất cả các số nguyên tố nhỏ hơn n cho trước.

Bài 7: Viết chương trình liệt kê n số nguyên tố đầu tiên.

Bài 8: Dãy số Fibonacci được định nghĩa như sau: $F_0 = 1, F_1 = 1; F_n = F_{n-1} + F_{n-2}$ với $n \geq 2$. Hãy viết chương trình tìm số Fibonacci thứ n.

Bài 9: Một số được gọi là số thuận nghịch độc nếu ta đọc từ trái sang phải hay từ phải sang trái số đó ta vẫn nhận được một số giống nhau. Hãy liệt kê tất cả các số thuận nghịch độc có sáu chữ số (Ví dụ số: 558855).

Bài 10: Viết chương trình liệt kê tất cả các xâu nhị phân độ dài n.

Bài 11: Viết chương trình liệt kê tất cả các tập con k phần tử của $1, 2, ..., n (k \leq n)$.

CHƯƠNG 3

LẬP TRÌNH ỨNG DỤNG CƠ SỞ DỮ LIỆU

Chương này tập trung vào sử dụng thư viện JDBC (Java DataBase Connectivity) để kết nối và truy xuất vào các sở dữ liệu. Nôi dung của chương bao gồm:

- Trình bày được khái niệm về JDBC; Các thành phần JDBC; Phân loại JDBC; Cơ chế hoạt động JDBC,...
- Thực hiện được việc tải và cài đặt JDBC driver cho project
- Trình bày được các phương pháp kết nối CSDL với các hệ QT CSDL SQL Server, Ocracle, MySQL,...
- Thực hiện được việc kết nối và truy xuất cơ sở dữ liệu
- Thực hiện được việc xử lý kết xuất kết quả truy xuất CSDL.
- Xây dựng được ứng dụng Quản lý CSDL

3.1. Giới thiệu JDBC

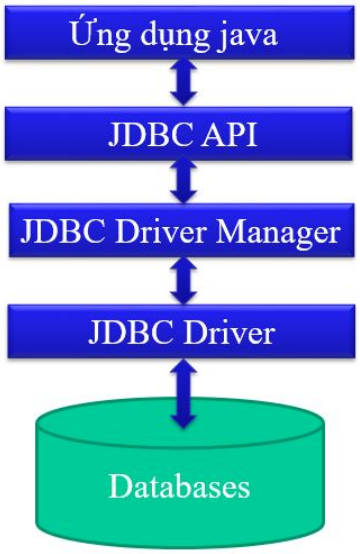
JDBC là một thư viện chuẩn cho phép các ứng dụng Java kết nối với nhiều cơ sở dữ liệu được cài đặt trên các hệ quản trị cơ sở dữ liệu như MS Access, SQL Server, Oracle, DB2,... JDBC hỗ trợ các chức năng như tạo một kết nối đến một cơ sở dữ liệu, tạo câu lệnh SQL (Structured Query Language), thực thi câu lệnh SQL, xem và thay đổi dữ liệu.

3.2. Các thành phần trong JDBC

JDBC bao gồm các thành phần như Hình 3.1, trong đó:

- JDBC API: là một API (Application Programming Interface) hoàn toàn dựa trên Java, giúp truy xuất cơ sở dữ liệu từ Java.
- JDBC Diver Manager: một lớp (class) giúp kết nối giữa ứng dụng Java đến các JDBC Driver.

- JDBC Driver: là thành phần cho phép ứng dụng Java tương tác với các cơ sở dữ liệu khác nhau.

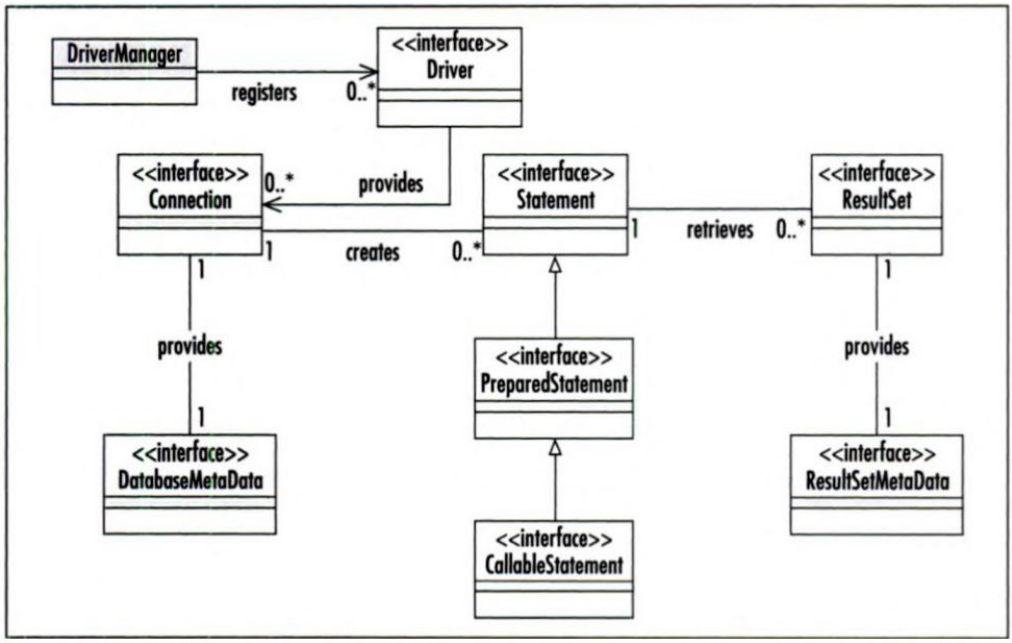


Hình 3.1. *Các thành phần trong JDBC*

3.3. Lược đồ lớp của JDBC

JDBC bao gồm các lớp (class) và giao diện (interface) nằm trong gói java.sql.* như thể hiện ở Hình 3.2, trong đó:

- **Driver:** Giao diện này chứa các hàm trừu tượng để xử lý các kết nối đến cơ sở dữ liệu. Trong lập trình, chúng ta chủ yếu sử dụng lớp DriverManager được cài đặt từ giao diện Driver để quản lý các kết nối đến cơ sở được liệu.
- **DriverManager:** Lớp này quản lý một danh sách trình điều khiển cơ sở dữ liệu (database drivers). Nó được dùng để nạp các Driver vào trong bộ nhớ và mở các kết nối tới một cơ sỡ dữ liệu nào đó.
- **Connection:** Đại diện cho một kết nối đến cơ sở dữ liệu. Được dùng để tạo ra các đối tượng Statement, PreparedStatement và CallableStatement.
- **Statement:** Đối tượng dùng để thực thi các câu lệnh SQL như



Hình 3.2. *Các lớp và giao diện của JDBC*

câu lệnh thêm dữ liệu (insert), câu lệnh thay đổi dữ liệu (update), câu lệnh xoá dữ liệu (delete), câu lệnh xem dữ liệu (select), ...

- **PreparedSatement:** Giống như Statement nhưng có thể truyền tham số vào câu lệnh SQL.
- **CallableStatement:** được sử dụng để thực thi một thủ tục (Stored Procedure) được lưu trữ trong hệ quản trị cơ sở dữ liệu.
- **ResultSet:** Đối tượng này sẽ quản lý dữ liệu sau khi chúng ta thực thi câu lệnh Select. Một đối tượng ResultSet duy trì một con trỏ trỏ tới hàng dữ liệu hiện tại của nó, sử dụng con trỏ này để lấy dữ liệu từ các bảng (Tables) trong cơ sở dữ liệu.
- **DatabaseMetaData:** cung cấp các phương thức để lấy siêu dữ liệu (metadata) của cơ sở dữ liệu như tên sản phẩm cơ sở dữ liệu, phiên bản sản phẩm cơ sở dữ liệu, tên Driver, tên của các bảng, tên view, ...
- **ResultSetMetaData:** cung cấp các phương thức để lấy siêu dữ liệu (metadata) của ResultSet như tổng số cột, tên cột, kiểu của

cột,...

Các bước truy xuất cơ sở dữ liệu

- **Bước 1:** Sử dụng lớp DirverManager để nạp trình điều khiển nhằm xác định hệ quản trị cơ sở dữ liệu cần sử dụng.
- **Bước 2:** Sử dụng lớp DirverManager để tạo đường kết nối đến cơ sở dữ liệu.
- **Bước 3:** Sử dụng đối tượng Connection để tạo ra các đối tượng Statement, PreparedStatement và CallableStatement.
- **Bước 4:** Sử dụng các đối tượng Statement, PreparedStatement và CallableStatement để thực thi các câu lênh SQL hoặc thủ tục lưu trữ.
- **Bước 5:** Xử lý dữ liệu lấy về từ cơ sở dữ liệu
- **Bước 6:** Giải phóng bộ nhớ và đóng kết nối

3.4. Lớp DirverManager

Lớp DirverManager hoạt động như một giao diện giữa người dùng và các trình điều kiển (Driver). Nó theo dõi các trình điều khiển có sẵn và xử lý việc thiết lập kết nối giữa một cơ sở dữ liệu và trình điều khiển thích hợp. Lớp DriverManager duy trì một danh sách các trình điều khiển được đăng ký bằng cách gọi phương thức DriverManager.registerDriver().

Để DriverManager nạp trình điều khiển thì trong project chúng ta phải thêm thư viện tương ứng với hệ quản trị cơ sở dữ liệu ta cần kết nối đến, Bảng 3.1 liệt kê một số thư viện điều khiển thường dùng.

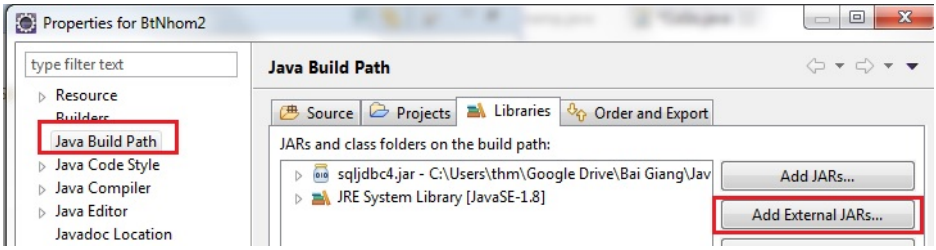
Bảng 3.1. *Một số thư viện điều khiển dùng trong JDBC*

STT	HQTCSDL	Tên thư viện điều khiển
1	MySQL	Mysql-connector-java-X.Y.Z.bin.jar (X.Y.Z là số hiệu của từng phiên bản)
2	SQL Server	sqljdbc.jar, sqljdbc4.jar
3	Oracle	jodbc5.jar, jodbc6.jar, ...
4	PostgreSQL	Postgresql-9.1-902.jdbc4.jar

5	SQLite	Sqlite-jdbc-3.7.2.jar

Ví dụ 3.4.1: Khi làm việc trên SQl Server, chúng ta cần thêm thư viện điều khiển sqljdbc4.rar vào Project theo các bước:

- Bước 1: Download thư viện điều khiển sqljdbc4.rar về máy.
- Bước 2: Tạo 1 project trên Eclipse.
- Bước 3: Nhấn Alt + Enter, chọn Java Build Path, chọn Add External JARs như Hình 3.3, sau đó chọn file sqljdbc4.rar download ở Bước 1.



Hình 3.3. *Thêm thư viện điều khiển vào Project của Eclipse*

Một số phương thức thường dùng trong lớp DirverManager

a. public static void registerDriver(Driver driver) throws SQLException: Phương thức này được sử dụng để đăng ký trình điều khiển đã cho với DriverManager.

Ví dụ 3.4.2: cần đăng ký trình điều khiển để làm việc với SqlServer:

```
1    Driver dr=new
         com.microsoft.sqlserver.jdbc.SQLServerDriver();
2    DriverManager.registerDriver(dr);
```

Ngoài cách trên, người ta thường dùng Class.forName() để đăng ký trình điều khiển đây là hướng tiếp cận chung và được dùng phổ biến để đăng ký một Driver. Cú pháp:

```
public static void Class.forName(String DatabaseDriver)
    throws ClassNotFoundException
```

Trong đó, DatabaseDriver là một chuỗi để xác đinh tên hệ quản trị cơ sỡ dữ liệu, Bảng 3.2 trình bày một số DatabaseDriver thường dùng:

Bảng 3.2. *Một số DatabaseDriver thường dùng*

STT	HQTCSDL	DatabaseDriver
1	MySQL	com.mysql.jdbc.Driver
2	SQL Server	com.microsoft.sqlserver.jdbc.SQLServerDriver
3	Oracle	oracle.jdbc.driver.OracleDriver
4	PostgreSQL	org.postgresql.Driver
5	Microsoft Access	sun.jdbc.odbc.JdbcOdbcDriver

Ví dụ 3.4.3:

Đăng ký trình điều khiển để làm việc với SqlServer

```
Class.forName("com.microsoft.sqlserver.jdbc.SQLServerDriver");
```

Đăng ký trình điều khiển để làm việc với MySql

```
Class.forName("com.mysql.jdbc.Driver");
```

2. public static void deregisterDriver(Driver driver)

Phương thức này được sử dụng để xóa Driver đã cho khỏi danh sách đăng ký với DriverManager.

Ví dụ 3.4.4: Đăng ký trình điều khiển để làm việc với SqlServer, sau đó xóa trình điều khiển ra khỏi danh sách:

```
Driver dr=new com.microsoft.sqlserver.jdbc.SQLServerDriver();
DriverManager.registerDriver(dr);
```

```
3 DriverManager.deregisterDriver(dr);
```

3. public static Connection getConnection(String DatabaseURL)

Phương thức này được sử dụng để thiết lập kết nối đến cơ sở dữ liệu, các thông tin kết nối được lưu trong DatabaseURL. DatabaseURL là một chuỗi thường lưu tên Server chứa hệ quản trị cơ sở dữ liệu, tên cơ sở dữ liệu cần kết nối. Một số DatabaseURL có thể lưu username và password để truy cập vào cơ sở dữ liệu. Một số DatabaseURL thường dùng được lưu ở Bảng 3.3.

Bảng 3.3. *Một số DatabaseDriver thường dùng*

STT	HQTCSDL	DatabaseDriver
1	MySQL	jdbc:mysql://<Tên Server>:<port>/<Tên CSDL>
2	SQL Server	jdbc:sqlserver://<Tên server>:<port>; databaseName=<Tên CSDL>;user=...; password=...
3	Oracle	jdbc:oracle:thin:@<Tên server>:<port>:<Tên CSDL>
4	PostgreSQL	jdbc:postgresql://<Tên server>:<port>/<Tên CSDL>
5	Microsoft Access	jdbc:odbc:Driver=Microsoft Access Driver (*.mdb); DBQ=<Tên CSDL.mdb>

Ví dụ 3.4.5: Viết chương trình kết nối đến CSDL: QlNv của SQLServer; Tên Server: ADMIN; Username: sa; password:123

```
1 import java.sql.*;
2 public class ViDu345 {
3 public static void main(String[] args) {
4     try {
5         //b1 Nạp trình điều khiển
6         String driver=
              "com.microsoft.sqlserver.jdbc.SQLServerDriver";
```

```
        Class.forName(driver);
        //b2: Kết nối vào CSDL
        String urlDatabase ="jdbc:sqlserver://ADMIN:1433;
            databaseName=QlNv; user=sa; password=123";
        Connection
            cn=DriverManager.getConnection(urlDatabase);
        System.out.println("Da ket noi");
    } catch (Exception e) {
        e.printStackTrace();
    }
  }
}
```

Khi chạy chương trình trên, một số lỗi thường gặp:

a. Chưa thêm thư viện sqljdbc4 vào Project: Khi chưa thêm thư viện sqljdbc4.rar vào Project sẽ hiển thị lỗi như Hình 3.4. Để khắc phục lỗi này ta thực hiện 3 bước như Ví dụ 3.4.1.

```
java.lang.ClassNotFoundException: com.microsoft.sqlserver.jdbc.SQLServerDriver
        at java.net.URLClassLoader.findClass(Unknown Source)
        at java.lang.ClassLoader.loadClass(Unknown Source)
        at sun.misc.Launcher$AppClassLoader.loadClass(Unknown Source)
        at java.lang.ClassLoader.loadClass(Unknown Source)
        at java.lang.Class.forName0(Native Method)
        at java.lang.Class.forName(Unknown Source)
        at ViDu345.main(ViDu345.java:10)
```

Hình 3.4. *Lỗi chưa thêm thư viện vào Project*

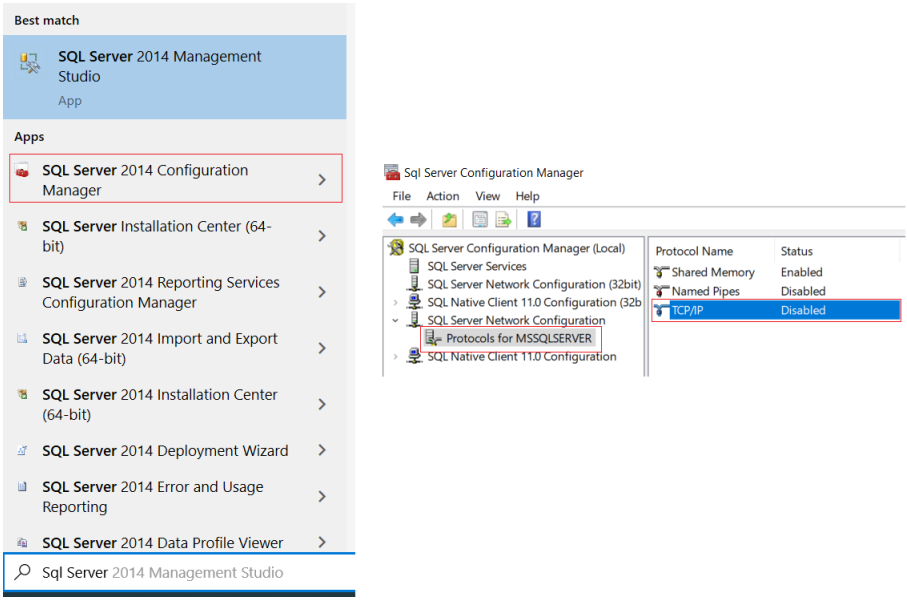
b. Chưa bật giao thức TCP và mở cổng trong SQLServer: Trong một số phiên bản của SQLServer khi cài lên sẽ chưa mở giao thức TCP và cổng 1433. Vì vậy, khi chạy chương trình sẽ báo lỗi như Hình 3.5.

```
com.microsoft.sqlserver.jdbc.SQLServerException: The TCP/IP connection to the host ADMIN, port 1433 has failed.
        at com.microsoft.sqlserver.jdbc.SQLServerException.makeFromDriverError(SQLServerException.java:130)
        at com.microsoft.sqlserver.jdbc.SQLServerConnection.connectHelper(SQLServerConnection.java:1195)
        at com.microsoft.sqlserver.jdbc.SQLServerConnection.loginWithoutFailover(SQLServerConnection.java:1054)
        at com.microsoft.sqlserver.jdbc.SQLServerConnection.connect(SQLServerConnection.java:758)
        at com.microsoft.sqlserver.jdbc.SQLServerDriver.connect(SQLServerDriver.java:842)
        at java.sql.DriverManager.getConnection(Unknown Source)
        at java.sql.DriverManager.getConnection(Unknown Source)
        at ViDu345.main(ViDu345.java:13)
```

Hình 3.5. *Lỗi chưa mở cổng và bật giao thức TCP*

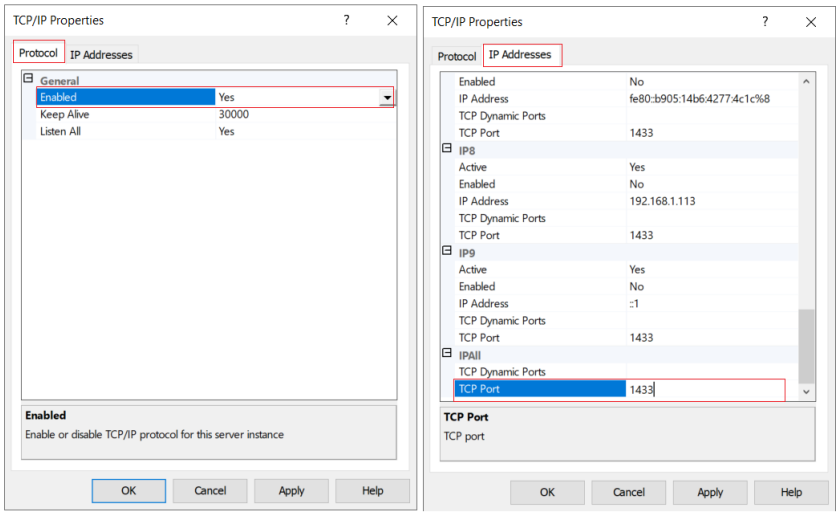
Để khắc phục lỗi này ta thực hiện theo các bước sau:

Bước 1: Nhấn Start và đi tìm và mở SQL Server Configuration Manager như Hình 3.6. Chọn SQL Server Network Configuration, chọn Protocols for ... và kích đôi lên TCP/IP.



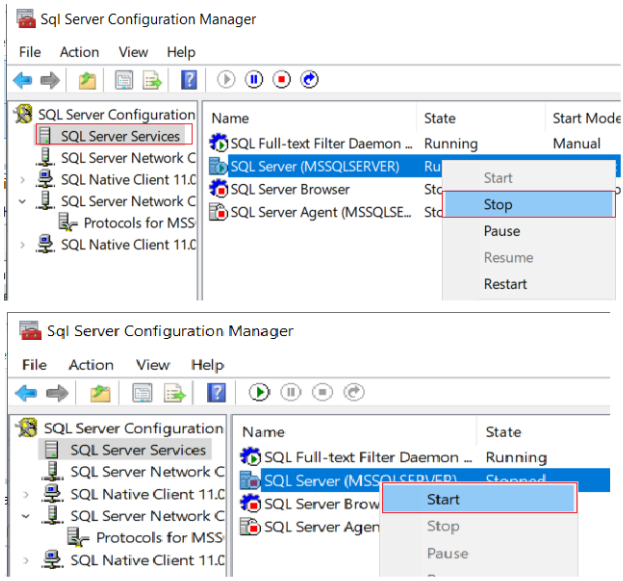
Hình 3.6. *Mở SQL Server Configuration Manager*

Bước 2: Bật giao thức TCP và mở cổng 1433 như Hình 3.7



Hình 3.7. *Bật giao thức TCP và mở cổng 1433*

Bước 3: Tắt và khởi động lại dịch vụ của SQL Server như Hình 3.8. Kích phải lên dịch vụ của SQL Server chọn Stop để tắt dịch vụ, sau đó kích phải lên dịch vụ của SQL Server chọn Start để khởi động lại dịch vụ.



Hình 3.8. *Tắt và khởi động lại dịch vụ của SQL Server*

c. Chưa bật cấu hình user và password để login vào SQL Server: Trong quá trình cài SQLServer ta chưa cấu hình tài khoản để login vào SQL Server. Vì vậy, khi chạy chương trình sẽ báo lỗi như Hình 3.9.

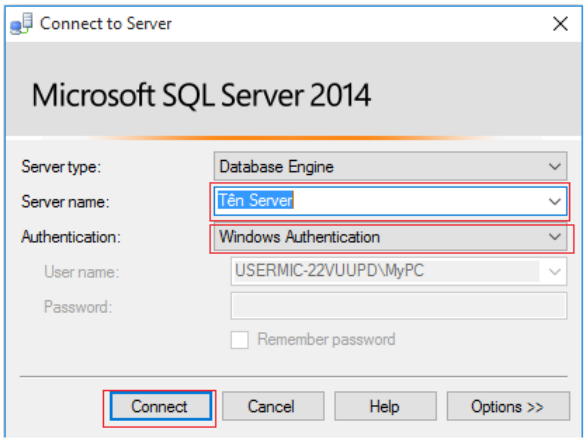
```
com.microsoft.sqlserver.jdbc.SQLServerException: Login failed for user 'sa'.
        at com.microsoft.sqlserver.jdbc.SQLServerException.makeFromDatabaseError(S(
        at com.microsoft.sqlserver.jdbc.TDSTokenHandler.onEOF(tdsparser.java:240)
        at com.microsoft.sqlserver.jdbc.TDSParser.parse(tdsparser.java:78)
```

Hình 3.9. *Chưa cấu hình tài khoản để login vào SQL Server*

Để khắc phục lỗi này ta lần lượt thực hiện theo các bước để cấu hình user và password như sau:

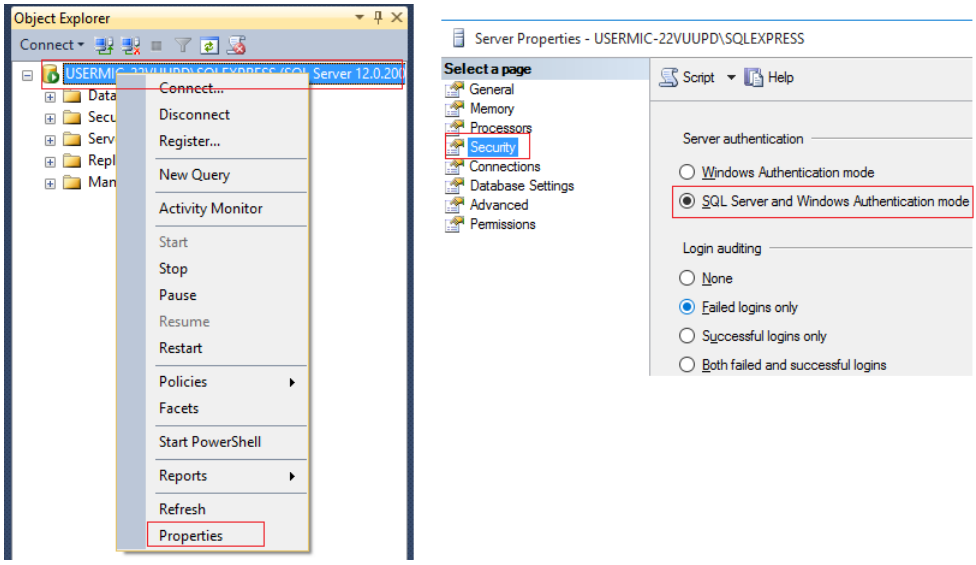
Bước 1: Khởi động SQL Server Management Studio và đăng nhập vào theo tài khoản của Window như Hình 3.10

Bước 2: Cấu hình để người dùng đăng nhập bằng tài khoản trong



Hình 3.10. *Đăng nhập vào SQL Server*

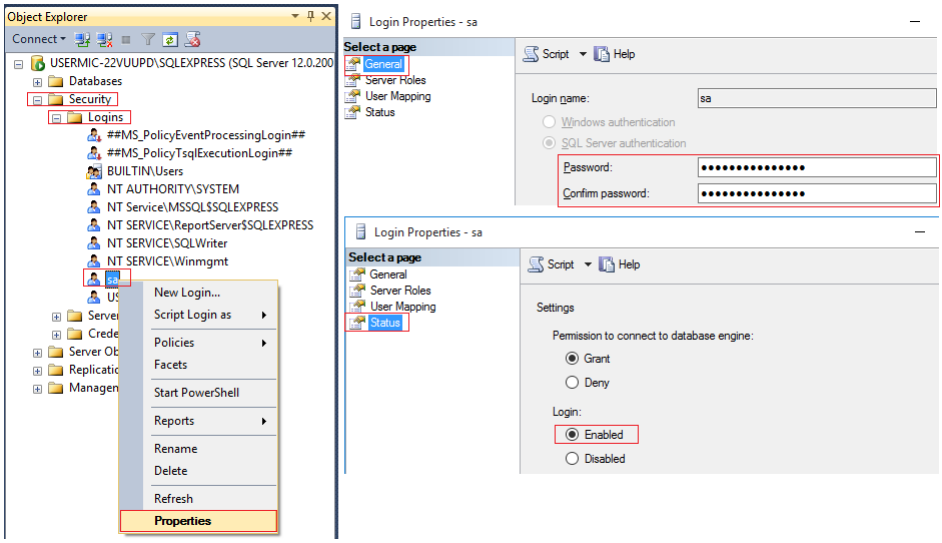
SQL Server như Hình 3.11. Kích phải lên Tên Server, chọn Properties, tại Security chọn SQL Server and Window Authentication mode.



Hình 3.11. *Cấu hình để người dùng đăng nhập bằng tài khoản*

Bước 3: Đặt mật khẩu và cho phép user: sa đăng nhập vào SQL Server như Hình 3.12. Kích phải lên user: sa, chọn Properties, Chọn General, sau đó nhập mật khẩu và nhập lại mật khẩu. Chọn Status và chọn Enabled để cho phép tài khoản sa đăng nhập vào hệ thống. **Bước 4:** Tắt và mở lại dịch vụ SQL Server. Kích phải lên Tên Server chọn

Stop, sau đó kích phải lên Tên Server chọn Start.



Hình 3.12. *Đặt mật khẩu và cho phép user: sa đăng nhập*

4. public static Connection getConnection(String url,String userName,String password)

Trong trường hợp một số DatabaseURL không chứa username, và password thì dùng phương thức này để thiết lập kết nối đến CSDL.

Ví dụ 3.4.6. Viết chương trình kết nối đến CSDL: QlNv của MySql; Tên Server: ADMIN; Username: root; password:123

```
import java.sql.Connection;
import java.sql.DriverManager;
public class ViDu346 {
public static void main(String[] args) {
    try {
        //b1 Nạp trình điều khiển
        String driver="com.mysql.jdbc.Driver";
        Class.forName(driver);
        //b2: Kết nối vào CSDL
        String urlDatabase ="jdbc:mysql://localhost/QlNv";
        Connection  cn=DriverManager.getConnection(
```

```
            urlDatabase,"root","123");
12          System.out.println("Da ket noi");
13      } catch (Exception e) {
14      e.printStackTrace();
15      }
16  }
17 }
```

Sau khi kết nối vào CSDL thì phương thức getConnection sẽ trả về một đối tượng Connection, sử dụng đối tượng Connection để tạo ra các đối tượng Statement, PreparedStatement và CallableStatement.

5. public static void setLoginTimeout(int second)

Phương thức này thiết lập thời gian (bằng giây) tối đa mà một Driver sẽ đợi trong khi cố gắng kết nối với một CSDL.

6. public static int getLoginTimeout()

Phương thức này lấy thời gian (bằng giây) tối đa mà một Driver sẽ đợi trong khi cố gắng truy cập vào một CSDL.

3.5. Đối tượng Connection

Sau khi đã nạp trình điều khiển để xác định hệ quản trị cơ sở dữ liệu và đã kết nối vào cơ sở dữ liệu, bước tiếp theo chúng ta cần lấy dữ liệu về để xử lý và lưu lại dữ liệu vào cơ sở dữ liệu. Để thực hiện được các việc này, chúng ta sử dụng các phương thức của Connection, một số phương thức của Connection:

1. public Statement createStatement()

Phương thức này tạo đối tượng Statement để thực thi các truy vấn SQL. Khi lấy dữ liệu từ cơ sở dữ liệu về chúng ta chỉ duyệt các bản ghi từ trên xuống, không thể di chuyển con trỏ đến bản ghi bất kỳ và không thể cập nhật lại dữ liệu.

2. public Statement createStatement(int resultSetType,int resultSetConcurrency) throws SQLException

Phương thức này cũng tạo ra một đối tượng Statement khắc phục các nhược điểm của của phương thức createStatement(), với phương thức reateStatement(int resultSetType,int resultSetConcurrency) ta có

thể di chuyển con trỏ đến vị trí bất kỳ, cho phép người dùng cập nhật lại dữ liệu trên các bản ghi được lấy về, trong đó:

resultsetType là 1 trong 3 kiểu:

- ResultSet.TYPE _FORWARD _ONLY: chỉ di chuyển con trỏ đi tới.
- ResultSet.TYPE _SCROLL _INSENSITIVE: cho phép di chuyển con trỏ nhưng không cho phép cập nhật lại dữ liệu
- ResultSet.TYPE _SCROLL _SENSITIVE: cho phép di chuyển con trỏ và cho phép cập nhật lai dữ liệu.

resultSetConcurrency là 1 trong 2 kiểu:

- ResultSet.CONCUR _READ _ONLY: dữ liệu lấy về chỉ đọc không thể cập nhật được.
- ResultSet.CONCUR _UPDATABLE: Cho phép cập nhật lại dữ liệu.

3. public PreparedStatement prepareStatement(String sql) throws SQLException Trong đó: chuỗi sql chứa câu lệnh SQL.

Giống như phương thức Statement() nó được dùng để thực thi truy vấn SQL nhưng có thể sử dụng tham số trong câu lênh SQL. Thực tế, ta nên dùng phương này ngay cả khi không cần phải truyền tham số bởi nó thực thi nhanh hơn do được nạp trước (preload) khi thực thi. PreparedStatement được kế thừa từ Statement, vì vậy mọi phương thức của Statement đều có thể dùng ở đây.

4. public PreparedStatement prepareStatement(String sql, int resultSetType, int resultSetConcurrency)throws SQLException

Phương thức này và các tham số của nó giống như phương thức createStatement(int resultSetType,int resultSetConcurrency) như ta có thể truyền tham số vào câu lệnh sql.

Ví dụ 3.4.7. Đã có cơ sở dữ liệu QlNv trong SqlServer, trong cơ sở dữ liệu này đã có bảng DonVi(**MaDonVi**, TenDonVi), đoạn mã dưới đây sử dụng phương thức prepareStatement để tạo ra một câu lệnh

SQL với các tham số nhằm mục đích chuẩn bị thêm vào 1 đơn vị vào bảng DonVi.

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
public class ViDu347 {
public static void main(String[] args) {
    try {
        //b1 Nạp trình điều khiển
        String driver=
            "com.microsoft.sqlserver.jdbc.SQLServerDriver";
        Class.forName(driver);
        //b2: Kết nối vào CSDL
        String urlDatabase ="jdbc:sqlserver://ADMIN:1433;
            databaseName=Qlnv;user=sa;password=123";
        Connection
            cn=DriverManager.getConnection(urlDatabase);
        //b3: Tạo câu lệnh PreparedStatement
        String sql="insert into DonVi values (?,?)";
        PreparedStatement cmd=cn.prepareStatement(sql);
    } catch (Exception e) {
        e.printStackTrace();
    }
 }
}
```

5. public CallableStatement prepareCall(" call tên_thủ_tục(các tham số của thủ tục)");

Được sử dụng để thực thi một thủ tục lưu trữ (Stored Procedure) trong SQL Server. Ta có thể sử dụng dấu ? để truyền vào các tham số của thủ tục. CallableStatement được thừa kế từ PreparedStatement.

5. public void setAutoCommit(boolean autoCommit) throws

SQLException

Phương thức này thiết lập Connection trong chế độ auto-commit. Nếu một Connection trong chế độ tự động ký thác thì tất cả các lệnh SQL của nó sẽ được thực thi và ký thác sau mỗi giao tác. Nếu tham số autoCommit được thiết lập là true tức là kích hoạt chế độ auto-commit, nếu là false là vô hiệu hóa chế độ này.

6. public void commit() throws SQLException

Phương thức này lưu các thay đổi đã được thực hiện trước đó. Phương thức này nên chỉ được sử dụng khi chế độ auto-commit đã bị vô hiệu hóa.

7. public void rollback()

Phương thức này xóa tất cả các thay đổi đã được thực hiện trước đó và quay về trạng thái trước khi thực hiện thay đổi. Phương thức này nên chỉ được sử dụng khi chế độ auto-commit đã bị vô hiệu hóa.

8. public void close()

Phương thức này đóng kết nối và giải phóng tài nguyên.

Ví dụ 3.4.8. Ví dụ này trình bày các bước để quyết định thực hiện các công việc hoặc phục hồi lại các công việc nếu chương trình bị lỗi.

```
1  import java.sql.Connection;
2  import java.sql.DriverManager;
3  import java.sql.PreparedStatement;
4  public class ViDu348 {
5  public static void main(String[] args) {
6      Connection  cn=null;
7      try {
8          //b1 Nạp trình điều khiển
9          String driver=
               "com.microsoft.sqlserver.jdbc.SQLServerDriver";
10         Class.forName(driver);
11         //b2: Kết nối vào CSDL Qlnv, tên Server: ADMIN,
               user:sa, password=123
12         String urlDatabase ="jdbc:sqlserver://ADMIN:1433;
```

```
            databaseName=Qlnv;user=sa; password=123";
        cn=DriverManager.getConnection(urlDatabase);
        //b3: Đặt chế độ AutoCommi=false
        cn.setAutoCommit(false);
        //b4: Thực thi các công việc như thêm, xóa, cập nhật
            vào cơ sở dữ liệu
          ...
        //b5: Quyết định thực thi các công việc ở B4
        cn.commit();
        cn.close();//Đóng kết nối
        System.out.println("Đã hoàn thành các công việc");
    } catch (Exception e) {
        System.out.print("Chương trình bị lỗi");
        try {
            //b6: Phục hồi lại công việc ở ở B4
            cn.rollback();
            cn.close();//Đóng kết nối
        } catch (Exception e2) {
        System.out.print("Không thể phục hồi");
    }
  }
 }
}
```

Ví dụ 3.4.9. Để đơn giản B1 và B2 ta viết một lớp DungChung và xây dựng một hàm tĩnh để kết nối và trả về đường kết nối đến cơ sở dữ liệu Qlnv trên máy chủ ADMIN với user và password để login vào CSDL là: sa và 123.

```
import java.sql.Connection;
import java.sql.DriverManager;
public class DungChung {
   public static Connection KetNoi() throws Exception{
    //b1 Nạp trình điều khiển
```

```
6    String
         driver="com.microsoft.sqlserver.jdbc.SQLServerDriver";
7    Class.forName(driver);
8     //b2: Kết nối vào CSDL Qlnv, tên Server: ADMIN, user:sa,
          password=123
9    String urlDatabase ="jdbc:sqlserver://ADMIN:1433;
         databaseName=Qlnv;user=sa; password=123";
10   return DriverManager.getConnection(urlDatabase);
11  }
12 }
```

3.6. Đối tượng Statement

Statment được tạo ra bởi đối tượng Connection dùng để thực thi các câu lệnh SQL, một số phương thức thường dùng:

1. public ResultSet executeQuery(String sql)

Thực thi một lệnh SQL đã cho và trả về một đối tượng Resultset. Tham số sql là một chuỗi chứa câu lệnh SQL: SELECT.

2. public int executeUpdate(String sql)

Thực thi lệnh sql đã cho, câu lệnh sql có thể là INSERT, UPDATE, DELETE hoặc một lệnh SQL mà không trả về giá trị như lệnh SQL về định nghĩa dữ liệu: CREATE TABLE, CREATE DATABASE, ...

Phương thức trả về số mẫu tin có hiệu lực, ví dụ trả về số mẫu tin thêm được hoặc xóa được hoặc cập nhật được. Nếu câu lệnh SQL chứa các câu lệnh SQL về định nghĩa dữ liệu hàm sẽ trả về bằng 0.

Ví dụ 3.6.1: Chèn thêm 1 dòng vào bảng DonVi có cấu trúc như ví dụ 3.4.7. Đoạn mã sau chưa kiểm tra trùng MaDonVi.

```
1 import java.sql.*;
2 public class ViDu361 {
3 public static void main(String[] args) {
4 Connection  cn=null;
5 try {
```

```
6       //b1, b2
7       cn=DungChung.KetNoi();
8       //b3: Đặt chế độ AutoCommi=false
9       cn.setAutoCommit(false);
10      //b4: Thêm vào bảng đơn vị 1 bản ghi với mã đơn vị là cntt
            và tên đơn vị là Công nghệ thông tin
11      String madv="cntt";
12      String tendv="Công nghệ thông tin";
13      //Thiết lập câu lệnh sql
14      String sql="insert into DonVi values ('" +madv+"',N'"+
            tendv+"')";
15      Statement cmd=cn.createStatement();
16      int smt=cmd.executeUpdate(sql);
17      System.out.print("Số mẫu tin thêm được: "+ smt);
18      //b5: như trong ví dụ 3.4.8
19      ...
20  } catch (Exception e) {
21      System.out.print("Chương trình bị lỗi");
22      try {
23          //b6: Phục hồi lại công việc ở ở B4
24          cn.rollback();
25          cn.close();//Đóng kết nối
26      } catch (Exception e2) {
27          System.out.print("Không thể phục hồi");
28      }
29    }
30   }
31  }
```

Chú ý: Theo ví dụ trên nếu Tendv="Baker's Chocolate" thì câu lệnh SQL sẽ bị lỗi cú pháp vì sau khi ghép câu lệnh sql sẽ là "insert into DonVi values ('cntt',N'Baker's Chocolate'). Để khắc phục nhược điểm này ta dùng đối tượng PreparedStatement thay vì dùng đối tượng Statement.

3. void close() throws SQLException

Phương thức được sử dụng để đóng đối tượng Statement và giải phóng tài nguyên ngay lập tức. Khi đã đóng đối tượng Statement rồi, thì các lời gọi phương thức nào tới đối tượng đó sẽ không hoạt động. Khi một đối tượng Statement đã bị đóng thì đối tượng ResultSet của nó cũng bị đóng theo.

3.7. Hạn chế của đối tượng Statement

Khi sử dụng đối tượng Statement, thường nối (ghép) các chuỗi lại với nhau để thiết lập câu lệnh SQL.

Ví dụ 3.7.1: Để tìm kiếm một đơn vị trong bảng DonVi thì câu lệnh SQL là "Select * from DonVi where TenDonVi=N'"+**tendv**+"'", trong đó **tendv** do người dùng nhập vào.

Giả sử người dùng nhập **tendv=Công nghệ thông tin** thì sẽ tìm ra các đơn vị có tên là Công nghệ thông tin. Nhưng nếu người dùng nhập vào **tendv=Baker's Chocolate** thì câu lệnh SQL sẽ bị sai cú pháp vì bị dư dấu ' trong câu lệnh SQL.

Ví dụ 3.7.2: Giả sử đã có bảng Login(**username**, password) để người dùng đăng nhập vào hệ thống. Câu lệnh SQL để kiểm tra người dùng đăng nhập thành công hay không là: "select * from Login where username ='"+**un**+ "' and password='"+**pass**+"'", trong đó **un** và **pass** do người dùng nhập vào. Như vậy nếu người dùng nhập vào **un=nhập bất kỳ** và **pass= nhập bất kỳ' or '1'='1** khi đó điều kiện của câu lệnh SQL luôn đúng: select * from Login where username ='nhập bất kỳ' and password='nhập bất kỳ' **or '1'=1'**". Điều này có nghĩa là cho dù người dùng không có tài khoản để đăng nhập họ vẫn đăng nhập thành công vào hệ thống. Đây được gọi là SQL Injection:

SQL injection là một kỹ thuật cho phép những kẻ tấn công lợi dụng lỗ hổng của việc kiểm tra dữ liệu đầu vào trong các ứng dụng và các thông báo lỗi của hệ quản trị cơ sở dữ liệu trả về để inject (tiêm vào) và thi hành các câu lệnh SQL bất hợp pháp. SQL injection có thể cho phép những kẻ tấn công thực hiện các thao tác, delete, insert, update, v.v. trên cơ sở dữ liệu của ứng dụng, thậm chí là server mà ứng dụng đó đang chạy. SQL injection thường được biết đến như là một vật trung gian tấn

công trên các ứng dụng có dữ liệu được quản lý bằng các hệ quản trị cơ sở dữ liệu như SQL Server, MySQL, Oracle, DB2, Sysbase...

Qua hai ví dụ trên ta thấy được các tác hại và hạn chế khi sử dụng Statement. Statement không có tham số hóa ? nên muốn truyền vào tham số chúng ta cần thêm dấu + để nối các chuỗi lại với nhau. Từ đó hacker hay người dùng xấu có thể cố tình truy cập vào hệ thống.

Vì vậy PreparedStatement sinh ra để khắc phục nhược điểm trên, PreparedStatement không nối chuỗi và dùng các tham số ? để truyền vào câu lệnh SQL, ví dụ:

String sql = "SELECT * FROM users WHERE username = ? AND password = ?";

Từ đó ta có thể đặt giá trị cho các biến được tham số hóa dấu ?. PreparedStatement sẽ tự loại bỏ những ký tự đặc biệt và phù hợp với câu lệnh truy vấn, tránh xảy ra SQL Injection.

3.8. Đối tượng PreparedStatement

Như đã trình bày ở trên, PreparedStatement được tạo ra để khắc phục các hạn chế của Statement. Đối tượng này được sử dụng khi chúng ta muốn thực thi 1 truy vấn SQL có tham số. Nó là 1 đối tượng được kế thừa từ Statement(public interface PreparedStatement extends Statement) cho nên mọi phương thức của Statement đều có thể dùng cho PreparedStatement. Một số phương thức thường dùng trong PreparedStatement:

1. public int executeUpdate()

Thực thi truy vấn SQL trong đối tượng PreparedStatement, câu lệnh SQL gồm INSERT, UPDATE hoặc DELETE, hoặc một lệnh SQL mà không trả về bất cứ cái gì, chẳng hạn như câu lệnh SQL định nghĩa dữ liệu như CREATE, ALTER,...

2. public ResultSet executeQuery() throws SQLException Phương thức này thực thi truy vấn SQL trong đối tượng PreparedStatement, câu lệnh SQL thường dùng là SELECT. Phương thức trả về đối tượng ResultSet được tạo bởi truy vấn.

3. public void setString(int paramIndex, String giaTri)

Thiết lập tham số đã cho thành giá trị String trong Java đã cung

cấp. Trình điều khiển sẽ chuyển đổi giá trị này thành một kiểu varchar hoặc nvarchar khi nó gửi giá trị tới CSDL.

Ví dụ 3.8.1: Thêm vào bảng DonVi(MaDonVi,TenDonVi) một bảng ghi và đổi tên một đơn vị dựa vào MaDonVi. MaDonVi và TenDonVi có kiểu nvarchar(50). Đoạn mã dưới đây chưa kiểm tra trùng MaDonVi khi thêm.

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
public class ViDu381 {
public static void main(String[] args) {
Connection  cn=null;
try {
    //b1, b2
    cn=DungChung.KetNoi();
    //b3: Đặt chế độ AutoCommi=false
    cn.setAutoCommit(false);
    String madv="toan";
    String tendv="Khoa Toán";
//b4: Thêm vào một đơn vị có mã đơn vị là: toan, tên đơn vị
    là Khoa Toán
    //Thiết lập câu lệnh SQL
    String sql="insert into DonVi values(?,?)";
    PreparedStatement cmd= cn.prepareStatement(sql);
    cmd.setString(1,madv);//Truyền tham số vào câu lệnh SQL
    cmd.setString(2,tendv);
    int smt=cmd.executeUpdate();//Thực thi câu lệnh
    System.out.println("Số mẫu tin thêm được: "+ smt);
//B5: Sửa tên đơn vị thành Tin học của đơn vị có mã là: cntt
    madv="cntt";
    tendv="Tin học";
    //Thiết lập câu lệnh SQL
```

```
27      sql="update DonVi set TenDonVi=? where MaDonVi=?";
28      cmd=cn.prepareStatement(sql);
29      cmd.setString(1,tendv);//Truyền tham số vào câu lệnh SQL
30      cmd.setString(2,madv);
31      smt=cmd.executeUpdate();//Thực thi câu lệnh
32      System.out.println("Số mẫu tin sửa được: "+ smt);
33      cn.commit();//Quyết định thực thi các công việc ở B4,B5
34      cmd.close();//Giải phóng bộ nhớ
35      cn.close();//Đóng kết nối
36      System.out.println("Đã hoàn thành các công việc")
37    } catch (Exception e) {
38      System.out.print("Chương trình bị lỗi");
39      try {
40   //b6: Phục hồi lại công việc ở ở B4
41          cn.rollback();
42          cn.close();//Đóng kết nối
43      } catch (Exception e2) {
44         System.out.print("Không thể phục hồi");
45      }
46    }
47   }
48  }
```

4. public void setInt(int paramIndex, int giaTri)

Thiết lập tham số đã cho tới giá trị nguyên trong Java đã cung cấp. Driver sẽ chuyển đổi giá trị này thành một giá trị nguyên trong SQL khi nó gửi giá trị tới CSDL.

5. public void setFloat(int paramIndex, float giaTri)

Thiết lập tham số đã cho thành giá trị float trong Java đã cung cấp. Driver chuyển đổi giá trị này thành một giá trị REAL trong SQL khi nó gửi giá trị tới CSDL.

6. public void setDouble(int paramIndex, double giaTri)

Thiết lập tham số đã cho thành giá trị double trong Java đã cung

cấp. Driver chuyển đổi giá trị này thành một giá trị DOUBLE trong SQL khi nó gửi giá trị tới CSDL.

Ngoài các phương thức trên, PreparedStatement hỗ trợ các phương thức thiết lập tham số cho các kiểu Boolean, Date, Byte, Long như setBoolean(int paramIndex, int giaTri), setDate(int paramIndex, int giaTri),setByte(int paramIndex, int giaTri), setLong(int paramIndex, int giaTri).

Ví dụ 3.8.2: Viết chương trình để tạo ra bảng NhanVien trong CSDL QlNv có cấu trúc như Hình 3.13, sau đó nhập vào 1 nhân viên.

	Column Name	Data Type	Allow Nulls
🔑	Manv	nvarchar(50)	☐
	HoTen	nvarchar(50)	☑
	gioitinh	bit	☑
	NgaySinh	date	☑
	Hsl	float	☑
	MaDv	nvarchar(50)	☑

Hình 3.13. *Cấu trúc của bảng NhanVien*

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.text.SimpleDateFormat;
import java.util.Date;
public class ViDu382 {
public static void main(String[] args) {
Connection  cn=null;
try {
    //b1, b2
    cn=DungChung.KetNoi();
    //b3: Đặt chế độ AutoCommi=false
    cn.setAutoCommit(false);
    //b4: Tạo bảng
        NhanVien(MaNv,HoTen,GioiTinh,NgaySinh,HSL,Madv)
```

```
16      //Tạo câu lệnh SQL để tạo bảng NhanVien
17      String sql="CREATE TABLE NhanVien(MaNv nvarchar(15) NOT
            NULL,HoTen nvarchar(50) NULL,GioiTinh bit NULL,
            NgaySinh date NULL, Hsl real NULL, MaDv [nvarchar](50)
            NULL, PRIMARY KEY( MaNv ))";
18      PreparedStatement cmd= cn.prepareStatement(sql);
19      cmd.executeUpdate();//Thực thi câu lệnh
20      //b5: Thêm vào một nhân viên
21      //Tạo câu lệnh SQL
22      sql="insert into nhanvien(Manv,HoTen,GioiTinh,NgaySinh,
            Hsl, Madv) values(?,?,?,?,?,?)";
23      cmd= cn.prepareStatement(sql);
24      //Thiết lập các tham số
25      cmd.setString(1,"Nv1");
26      cmd.setString(2,"Nguyễn Hoàng Hà");
27      cmd.setBoolean(3,true);
28      String ngaysinh ="18/11/1976";
29      //Đổi chuỗi ngaysinh ra kiểu ngày java.util.Date;
30      SimpleDateFormat dd= new SimpleDateFormat("dd/MM/yyyy");
31      Date NsUtil=dd.parse(ngaysinh);
32      //Đổi ngày java.util.Date ra java.sql.Date
33      java.sql.Date NsSql= new java.sql.Date(NsUtil.getTime());
34      //Thiết lập tham số cho trường NgaySinh
35      cmd.setDate(4,NsSql);
36      cmd.setDouble(5, 2.34);
37      cmd.setString(6, "cntt");
38      cmd.executeUpdate();//Thực thi câu lệnh
39      //b6 Quyết định thực thi các công việc ở b4,b5
40      cn.commit();//
41      cmd.close();//Giải phóng bộ nhớ
42      cn.close();//Đóng kết nối
43      System.out.println("Đã hoàn thành các công việc");
44
45  } catch (Exception e) {
```

```
46      System.out.print("Chương trình bị lỗi");
47      try {
48          //b6: Phục hồi lại công việc ở ở B4
49          cn.rollback();
50          cn.close();//Đóng kết nối
51      } catch (Exception e2) {
52          System.out.print("Không thể phục hồi");
53      }
54     }
55    }
56  }
```

3.9. Đối tượng ResultSet

Phương thức executeQuery của PrepareStatement hoặc Statement trả về một con trỏ trỏ đến một tập kết quả lấy về từ cơ sở dữ liệu, con trỏ này có kiểu là ResultSet. Người lập trình sử dụng con trỏ này để lấy dữ liệu về, di chuyển con trỏ, cập nhật lại dữ liệu, ...

3.9.1. Lấy giá trị của mẫu tin tại vị trí của con trỏ.

Để lấy giá trị của mẫu tin tại vị trí của con trỏ ta sử dụng các phương thức getXXX. Cú pháp chung của các phương thức này có dạng:

public XXX getXXX(int columnIndex)

hoặc

public XXX getXXX(String columnName)

Trong đó, **XXX** là tên kiểu dữ liệu, **columnIndex** và **columnName** là chỉ số của cột hoặc tên cột cần lấy dữ liệu ra.

Ví dụ: getInt, getString, getDouble, getBoolean, getDate, ...

Các phương thức sau lấy ra số nguyên và chuỗi tại vị trí con trỏ:

1. public int getInt(int columnIndex): được sử dụng để lấy về

một số nguyên trên cột thứ columnIndex tại vị trí của con trỏ. Chú ý: cột đầu tiên có số thứ tự là 1.

2. public int getInt(String columnName): được sử dụng để lấy về một số nguyên trên cột có tên columnName tại vị trí của con trỏ.

3. public String getString(int columnIndex): được sử dụng để lấy về một chuỗi trên cột thứ columnIndex tại vị trí của con trỏ.

4. public String getString(String columnName): được sử dụng để lấy về một chuỗi trên cột có tên columnName tại vị trí của con trỏ.

3.9.2. Các phương thức di chuyển con trỏ mẫu tin

1. public boolean next(): được sử dụng để di chuyển con trỏ đến mẫu tin tiếp theo từ vị trí hiện tại. Hàm trả về false nếu con trỏ đang đứng mẫu tin cuối cùng mà tiếp tục di chuyển, ngược lại hàm trả về true.

Chú ý: Để duyệt từ mẫu tin đầu tiên đến cuối cùng trong ResultSet: rs ta thường dùng vong lặp while:

```
1  while (rs.next()){
2    //Xử lý dữ liệu
3  }
```

Ví dụ 3.9.1: Giả sử bảng NhanVien đã có dữ liệu như Hình 3.14

	Manv	HoTen	gioiti...	NgaySinh	Hsl	MaDv
1	Nv1	Nguyễn Hoàng Hà	1	1976-11-18	2.34	cntt
2	Nv2	Lê Văn Trung	1	1980-10-10	3.33	cntt
3	Nv3	Trần Nguyên Phong	1	1976-12-10	3.33	cntt
4	Nv4	Lê Viết Mẫn	1	1992-10-12	3.66	toan
5	Nv5	Hà Lê Nguyên	0	1980-04-03	2.67	van
6	Nv6	Phạm Bình Minh	1	1979-02-09	4.32	van

Hình 3.14. *Dữ liệu của bảng NhanVien*

Viết chương trình hiển thị thông tin của tất cả các nhân viên trong bảng NhanVien.

```
import java.sql.*;
import java.text.SimpleDateFormat;
import java.util.Date;
public class ViDu391 {
public static void main(String[] args) {
    Connection  cn=null;
    try {
        cn=DungChung.KetNoi();
        //Tạo câu lệnh SQL Lấy về tất cả các nhân viên
        String sql="select * from nhanvien";
        PreparedStatement cmd= cn.prepareStatement(sql);
        ResultSet rs= cmd.executeQuery();//Thực thi câu lệnh
        int i=1;
        //Duyệt qua từng mẫu tin trong rs
        while(rs.next()){
            String Manv=rs.getString("Manv");//Lấy ra MaNv
            String Hoten=rs.getString("HoTen");//Lấy ra HoTen
            Boolean GioiTinh=rs.getBoolean("GioiTinh");
            Date NgaySinh =rs.getDate("NgaySinh");
            String Madv=rs.getString("MaDv");
            System.out.println("Thông tin của nhân viên
                thứ: "+ i++);
            System.out.print(Manv);
            System.out.print(" "+ Hoten);
            System.out.print(" "+ GioiTinh);
            System.out.print(" "+ NgaySinh);
            System.out.println(" "+Madv);
            System.out.println("--------------------");
        }
        rs.close();// Đóng ResultSet
        cn.close();//Đóng kết nối
        System.out.println("Đã hoàn thành các công việc");

```

```
33      } catch (Exception e) {
34          System.out.print("Chương trình bị lỗi");
35          try {
36              cn.close();//Đóng kết nối
37          } catch (Exception e2) {
38              System.out.print("Không thể phục hồi");
39          }
40      }
41    }
42  }
```

Kết quả hiển thị của chương trình:

```
Thông tin của nhân viên thứ:  1
Nv1 Nguyễn Hoàng Hà true 1976-11-18 cntt
--------------------
Thông tin của nhân viên thứ:  2
Nv2 Lê Văn Trung true 1980-10-10 cntt
--------------------
Thông tin của nhân viên thứ:  3
Nv3 Trần Nguyên Phong true 1976-12-10 cntt
--------------------
Thông tin của nhân viên thứ:  4
Nv4 Lê Viết Mẫn true 1992-10-12 toan
--------------------
Thông tin của nhân viên thứ:  5
Nv5 Hà Lê Nguyên false 1980-04-03 van
--------------------
Thông tin của nhân viên thứ:  6
Nv6 Phạm Bình Minh true 1979-02-09 van
--------------------
Đã hoàn thành các công việc
```

Hình 3.15. *Kết quả chương trình*

2. public boolean previous(): được sử dụng để di chuyển con trỏ đến một mẫu tin trước đó từ vị trí hiện tại.

3. public boolean first(): được sử dụng để di chuyển con trỏ đến mẫu tin đầu tiên.

4. public boolean last(): được sử dụng để di chuyển con trỏ đến mẫu tin cuối cùng.

5. public boolean absolute(int row): được sử dụng để di chuyển

con trỏ đến số mẫu tin thứ row.

6. public boolean relative(int row): được sử dụng để di chuyển con trỏ đến mẫu tin tương đối. Nếu row là số âm thì di chuyển con trỏ đến mẫu tin trước mẫu tin hiện tại, ngược lại di chuyển mẫu tin đến sau mẫu tin hiện tại.

7. public void moveToInsertRow()

Di chuyển con trỏ tới một hàng đặc biệt trong ResultSet mà có thể được sử dụng để chèn một hàng mới vào trong cơ sở dữ liệu.

Chú ý: Để di chuyển con trỏ mẫu tin bằng các phương thức previous(), first(), last(), absolute(), relative(), moveToInsertRow() ta phải chỉ rõ loại ResultSet (resultsetType) và kiểu của ResultSet (resultSetConcurrency) trong phương thức prepareStatement:

public PreparedStatement prepareStatement(String sql, int resultSetType, int resultSetConcurrency)throws SQLException

Trong đó: resultsetType: chỉ dùng 1 trong 2 kiểu:

- ResultSet.TYPE _SCROLL _INSENSITIVE
- ResultSet.TYPE _SCROLL _SENSITIVE

resultSetConcurrency là 1 trong 2 kiểu:

- ResultSet.CONCUR _READ _ONLY:
- ResultSet.CONCUR _UPDATABLE:

Ví dụ 3.9.2. Giả sử bảng NhanVien đã có dữ liệu như Hình 3.14. Hãy hiển thị họ tên của người đầu tiên, người cuối cùng, người thứ 4, người đứng trước 2 mẫu tin tính từ mẫu tin số 4. người đứng sau 2 mẫu tin tính từ mẫu tin số 4.

```
1 import java.sql.Connection;
2 import java.sql.DriverManager;
3 import java.sql.PreparedStatement;
4 import java.sql.ResultSet;
5
```

```
6  import dao.DungChung;
7  public class ViDu392 {
8  public static void main(String[] args) {
9      Connection  cn=null;
10     try {
11         cn=DungChung.KetNoi();
12         //Tạo câu lệnh SQL Lấy về tất cả các nhân viên
13         String sql="select * from nhanvien";
14         PreparedStatement cmd= cn.prepareStatement(sql,
               ResultSet .TYPE_SCROLL_INSENSITIVE,
               ResultSet.CONCUR_READ_ONLY);
15         //Thực thi câu lệnh
16         ResultSet rs= cmd.executeQuery();
17         //Đưa con trỏ về mẫu tin đầu tiên;
18         rs.first();
19         //Lấy ra HoTen mẫu tin thứ 1
20         String Hoten=rs.getString("HoTen");
21         System.out.println("Họ tên người đầu tiên: "+ Hoten);
22         System.out.println("--------------------");
23
24         //Đưa con trỏ về mẫu tin cuối cùng;
25         rs.last();
26         //Lấy ra HoTen mẫu tin thứ cuối cùng
27         Hoten=rs.getString("HoTen");
28         System.out.println("Họ tên người cuối cùng: "+ Hoten);
29         System.out.println("--------------------");
30         //Đưa con trỏ về mẫu tin thứ 4;
31         rs.absolute(4);
32         //Lấy ra HoTen mẫu tin thứ 4
33         Hoten=rs.getString("HoTen");
34         System.out.println("Họ tên người người thứ
               4: "+ Hoten);
35         System.out.println("--------------------");
36
```

```
37          //Đưa con trỏ về trước 2 mẫu tin tính từ mẫu tin số 4
38          rs.absolute(4);
39          rs.relative(-2);
40          //Lấy ra HoTen mẫu tin thứ 2
41          Hoten=rs.getString("HoTen");
42          System.out.println("Họ tên người thứ 2: "+ Hoten);
43          System.out.println("--------------------");
44
45          //Đưa con trỏ về sau 2 mẫu tin tính từ mẫu tin số 4
46          rs.absolute(4);
47          rs.relative(2);
48          //Lấy ra HoTen mẫu tin thứ 6
49          Hoten=rs.getString("HoTen");
50          System.out.println("Họ tên người thứ 6: "+ Hoten);
51          System.out.println("--------------------");
52          rs.close();// Đóng ResultSet
53          cn.close();//Đóng kết nối
54          System.out.println("Đã hoàn thành các công việc");
55
56      } catch (Exception e) {
57          e.printStackTrace();
58          System.out.print("Chương trình bị lỗi");
59      }
60  }
61 }
```

Kết quả hiển thị của chương trình:

3.9.3. Các phương thức cập nhật dữ liệu

Để cập nhật dữ liệu trong ResultSet, ta sử dụng phương thức có dạng:

public void updateXXX(int columnIndex, XXX value);

hoặc

```
Họ tên người đầu tiên: Nguyễn Hoàng Hà
---------------------------------
Họ tên người cuối cùng: Phạm Bình Minh
---------------------------------
Họ tên người người thứ 4: Lê Viết Mẫn
---------------------------------
Họ tên người thứ 2: Lê Văn Trung
---------------------------------
Họ tên người thứ 6: Phạm Bình Minh
---------------------------------
Đã hoàn thành các công việc
```

Hình 3.16. *Kết quả chương trình*

public void updateXXX(String columnName, XXX value);

Trong đó, **XXX** là tên kiểu dữ liệu; **columnIndex** và **columnName** là chỉ số của cột hoặc tên cột cần cập nhật dữ liệu; value là giá trị cần cập nhật. ví dụ: updateInt, updateString, updateDouble, updateDate, ...

Để cập nhật dữ liệu vào CSDL, khi lấy dữ liệu về ta phải sử dụng kiểu con trỏ (resultSetType) là ResultSet.TYPE_SCROLL_ SENSITIVE và loại con trỏ (resultSetConcurrency) là ResultSet.CONCUR_ UPDATABLE.

Các phương thức update này không cập nhật vào cơ sở dữ liệu. Để cập nhật dữ liệu vào cơ sở dữ liệu, ta sử dụng thêm phương thức updateRow() hoặc insertRow() nếu thêm mới 1 mẫu tin.

Một số phương thức thường dùng để cập nhật dữ liệu:

1. public void updateInt(int columnIndex hoặc String columnName, int x):

Cập nhật giá trị nguyên **x** vào mẫu tin hiện tại của cột có chỉ số **columnIndex** hoặc có tên cột là **columnName**.

2. public void updateString(int columnIndex hoặc String columnName, String x)

Cập nhật chuỗi **x** vào mẫu tin hiện tại của cột có chỉ số **columnIndex** hoặc có tên cột là **columnName**.

3. public void updateBoolean(int columnIndex hoặc String columnName, Boolean x)

Cập nhật giá trị **x** vào mẫu tin hiện tại của cột có chỉ số **columnIndex** hoặc có tên cột là **columnName**

4. public void updateDate(int columnIndex hoặc String columnName, Date x)

Cập nhật ngày **x** vào mẫu tin hiện tại của cột có chỉ số **columnIndex** hoặc có tên cột là **columnName**

5. public void updateDouble(int columnIndex hoặc String columnName, Double x)

Cập nhật số thực **x** vào mẫu tin hiện tại của cột có chỉ số **columnIndex** hoặc có tên cột là **columnName**

Ví dụ 3.9.3. Viết chương trình để nhập vào 1 mã nhân viên. Tìm xem có nhân viên có mã này hay không. Nếu có thì nhập vào 1 ngày sinh và cập nhập lại ngày sinh cho nhân viên tìm được.

```
1  import java.sql.Connection;
2  import java.sql.PreparedStatement;
3  import java.sql.ResultSet;
4  import java.text.SimpleDateFormat;
5  import java.util.Date;
6  import java.util.Scanner;
7
8  import dao.DungChung;
9  public class ViDu3_9_3 {
10
11 public static void main(String[] args) {
12 Connection  cn=null;
13 try {
14     //Kết nối vào CSDL
15     cn=DungChung.KetNoi();
16     //Nhập vào từ bàn phím 1 mã nhân viên
17     System.out.println("Nhập vào 1 mã nhân viên:");
18     Scanner nhap= new Scanner(System.in);
19     String manv=nhap.next();
```

```
20
21      //Tạo câu lệnh SQL để lấy về nhân viên có mã là manv
22      String sql="select * from nhanvien where Manv=?";
23      //Tạo câu lệnh prepareStatement với:
24      //Kiểu con trỏ: TYPE_SCROLL_SENSITIVE
25      // và loại con trỏ: CONCUR_UPDATABLE
26      PreparedStatement cmd=
            cn.prepareStatement(sql,ResultSet.TYPE_SCROLL_SENSITIVE,
            ResultSet.CONCUR_UPDATABLE);
27      //Truyền tham số vào câu lệnh
28      cmd.setString(1, manv);
29      //Thực thi câu lệnh
30      ResultSet rs= cmd.executeQuery();
31
32      //Nếu trong rs không có dữ liệu
33      if (!rs.next())
34          System.out.println("Không tim ra nhân viên này");
35      else{
36          System.out.println("Nhập vào 1 ngày sinh theo dạng:
                yyyy-MM-dd");
37          String ngay=nhap.next();
38          //Đổi chuỗi sang ngày của Util
39          SimpleDateFormat dd= new SimpleDateFormat(
                "yyyy-MM-dd");
40          Date ngayUtil=dd.parse(ngay);
41          //Đổi ngày của Util sang ngày của sql
42          //và cập nhật lại ngày sinh
43          rs.updateDate("NgaySinh", new java.sql.Date(
                ngayUtil.getTime()));
44          rs.updateRow();
45      }
46      rs.close();// Đóng ResultSet
47      cn.close();//Đóng kết nối
48  } catch (Exception e) {
```

```
49      e.printStackTrace();
50      System.out.print("Chương trình bị lỗi");
51     }
52    }
53  }
```

3.9.4. Thêm và xóa một mẫu tin

Để thêm một mẫu tin vào cơ sở dữ liệu thì ta thực hiện các bước:

- Di chuyển con trỏ về sau mẫu tin cuối cùng và tạo ra 1 mẫu tin: moveToInsertRow();
- Cập nhật lại dữ liệu vào mẫu tin mới thêm, sử dụng các phương thức updateXXX(int columnIndex, XXX value);
- Lưu dữ liệu vào cơ sở dữ liệu, sử dụng phương thức insertRow();

Ví dụ 3.9.4. Viết chương trình để thêm vào 1 nhân viên.

```
1  import java.sql.*;
2  import java.text.SimpleDateFormat;
3  import java.util.Date;
4  import java.util.Scanner;
5
6  import dao.DungChung;
7  public class ViDu3_9_4 {
8
9  public static void main(String[] args) {
10 Connection  cn=null;
11 try {
12     //Kết nối vào CSDL
13     cn=DungChung.KetNoi();
14     //Nhập vào từ bàn phím 1 mã nhân viên
15     System.out.println("Nhập vào 1 mã nhân viên:");
16     Scanner nhap= new Scanner(System.in);
17     String manv=nhap.nextLine();
```

```
18      //Tạo câu lệnh SQL để lấy về nhân viên có mã là manv
19      String sql="select * from nhanvien where Manv=?";
20      //Tạo câu lệnh prepareStatement
21      PreparedStatement cmd=
            cn.prepareStatement(sql,ResultSet.TYPE_SCROLL_SENSITIVE,
            ResultSet.CONCUR_UPDATABLE);
22      //Truyền tham số vào câu lệnh
23      cmd.setString(1, manv);
24      //Thực thi câu lệnh
25      ResultSet rs= cmd.executeQuery();
26      //Nếu có dữ liệu trong rs
27      if (rs.next())
28          System.out.println("Đã tồn tại nhân viên có mã:
                "+manv);
29      else{
30          System.out.println("Nhập họ tên nhân viên:");
31          String HoTen=nhap.nextLine();
32
33          System.out.println("Nhập giới tính (true/false):");
34          Boolean GioiTinh=Boolean.parseBoolean(nhap.nextLine());
35          System.out.println("Nhập vào 1 ngày sinh theo dạng:
                yyyy-MM-dd");
36          String ngay=nhap.nextLine();
37          //Đổi chuỗi sang ngày của Util
38          SimpleDateFormat dd= new
                SimpleDateFormat("yyyy-MM-dd");
39          Date ngayUtil=dd.parse(ngay);
40
41          System.out.println("Nhập hệ số lương:");
42          Double hsl= Double.parseDouble(nhap.nextLine());
43
44          System.out.println("Nhập mã đơn vị:");
45          String Madv=nhap.nextLine();
46          //Di chuyển con trỏ về sau mẫu tin cuối cùng và tạo ra
                1 mẫu tin
```

```
47          rs.moveToInsertRow();
48          //Cập nhật lại dữ liệu vào mẫu tin mới thêm
49          rs.updateString("Manv", manv);
50          rs.updateString("HoTen", HoTen);
51          rs.updateBoolean("GioiTinh", GioiTinh);
52          rs.updateDate("NgaySinh", new
                java.sql.Date(ngayUtil.getTime()));
53          rs.updateDouble("Hsl", hsl);
54          rs.updateString("MaDv", Madv);
55          //Lưu vào CSDL
56          rs.insertRow();
57      }
58
59      rs.close();// Đóng ResultSet
60      cn.close();//Đóng kết nối
61  } catch (Exception e) {
62      e.printStackTrace();
63      System.out.print("Chương trình bị lỗi");
64     }
65    }
66  }
```

Để xóa một mẫu tin ra khỏi cơ sở dữ liệu thì ta thực hiện các bước:

- Di chuyển con trỏ đến mẫu tin cần xóa (tìm mẫu tin cần xóa)
- Xóa và lưu dữ liệu vào cơ sở dữ liệu, sử dụng phương thức deleteRow().

Ví dụ 3.9.5. Viết chương trình để nhập vào 1 mã nhân viên và xóa nhân viên có mã này ra khỏi CSDL.

```
1  import java.sql.Connection;
2  import java.sql.PreparedStatement;
3  import java.sql.ResultSet;
4  import java.util.Scanner;
```

```
import dao.DungChung;
public class ViDu3_9_5 {
public static void main(String[] args) {
Connection  cn=null;
try {
        cn=DungChung.KetNoi();//Kết nối vào CSDL
    //Nhập vào từ bàn phím 1 mã nhân viên
    System.out.println("Nhập vào 1 mã nhân viên:");
    Scanner nhap= new Scanner(System.in);
    String manv=nhap.nextLine();
    //Tạo câu lệnh SQL để lấy về nhân viên có mã là manv
    String sql="select * from nhanvien where Manv=?";
    //Tạo câu lệnh prepareStatement
    PreparedStatement cmd=
        cn.prepareStatement(sql,ResultSet.TYPE_SCROLL_SENSITIVE,
        ResultSet.CONCUR_UPDATABLE);
    cmd.setString(1, manv);//Truyền tham số vào câu lệnh
    //Thực thi câu lệnh
    ResultSet rs= cmd.executeQuery();
    //Nếu không tim thấy
    if (!rs.next())
        System.out.println("Không tìm ra nhân viên có
            mã: "+manv);
    else{
        // Xóa dòng hiện tại và lưu vào CSDL
        rs.deleteRow();
    }
    rs.close();// Đóng ResultSet
    cn.close();//Đóng kết nối
} catch (Exception e) {
    e.printStackTrace();
    System.out.print("Chương trình bị lỗi");
    }
  }
```

```
}
```

3.9.5. Lấy về bảng mô tả trong ResultSet

Các phần trên tập trung vào truy xuất dữ liệu trong ResultSet, để lấy thông tin (metadata) trong ResultSet như tên bảng, tổng số cột, tên cột, kiểu của cột, ...ResultSet hỗ trợ hàm:

public ResultSetMetaData getMetaData();

Một số phương thức trong ResultSetMetaData:

1. public int getColumnCount()throws SQLException: trả về tổng số cột trong đối tượng ResultSet.

2. public String getColumnName(int index)throws SQLException: trả về tên cột tại chỉ mục đã cho.

3. public String getColumnTypeName(int index)throws SQLException : trả về tên kiểu của cột tại chỉ mục đã cho.

4. public Boolean isAutoIncrement(int column)

Xác định xem cột đã cho có phải là tự động tăng (auto-increment) hay không.

Ví dụ 3.9.6. Viết hàm GetBang(String TenBang) để hiển thị tên cột và dữ liêu của bảng: TenBang. Sau đó viết chương trình gọi lại hàm GetBang để hiển thi tên cột vào dữ liệu của bảng NhanVien và DonVi.

```
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.ResultSetMetaData;

public class ViDu3_9_6 {
public static void GetBang(String TenBang) throws Exception{
    Connection cn=DungChung.KetNoi();//Tạo đường kết nối
    // Tạo câu lênh SQl để lấy về dữ liệu của bảng: TenBang
    String sql="select * from "+ TenBang;
    //Tạo và thực thi câu lệnh
```

```
11      PreparedStatement cmd=cn.prepareStatement(sql);
12      ResultSet rs=cmd.executeQuery();
13      // Lấy về thông tin của rs (metadata)
14      ResultSetMetaData meta=rs.getMetaData();
15      //' Lấy về tổng số cột
16      int sc=meta.getColumnCount();
17      // Hiển thị tên các cột của bảng: TenBang
18      for(int i=1;i<=sc;i++)
19          System.out.printf("%-20s",meta.getColumnName(i));
20      System.out.println();
21      // Duyệt qua các mẫu tin trong rs
22      while(rs.next()){
23          // Duyệt qua các cột trên mỗi mẫu tin
24          for(int i=1;i<=sc;i++)
25              // Hiển thị giá trị trên cột i
26              System.out.printf("%-20s",rs.getString(i));
27          System.out.println();
28      }
29      rs.close();
30      cn.close();
31  }
32  public static void main(String[] args) {
33  try {
34      System.out.println("Bảng Nhân viên:");
35      DungChung.GetBang("NhanVien");
36      System.out.println("Bản Đơn Vị:");
37      DungChung.GetBang("DonVi");
38  } catch (Exception e) {
39      e.printStackTrace();
40    }
41   }
42  }
```

Kết quả hiển thị của chương trình như Hình 3.17

Bảng Nhân viên:

MaNv	HoTen	GioiTinh	NgaySinh	Hsl	MaDv
Nv1	Nguyễn Hoàng Hà	1	1976-11-18	2.34	cntt
Nv2	Lê Văn Trung	1	1980-10-10	3.33	cntt
Nv3	Trần Nguyên Phong	1	1976-12-10	3.33	cntt
Nv4	Lê Việt Mẫn	1	1992-10-12	3.66	toan
Nv5	Hà Lê Nguyên	0	1980-04-03	2.67	van
Nv6	Phạm Bình Minh	1	1979-02-09	4.32	van

Bản Đơn Vị:

MaDonVi	TenDonVi
cntt	Tin học
cntt3	Công nghệ thông tin2
toan	Khoa Toán

Hình 3.17. *Dữ liệu bảng Nhân viên và Đơn vị*

3.10. Bài tập thực hành

Để quản lý nhân viên của một công ty người ta quản lý đơn vị và nhân viên của các đơn vị. Đối với đơn vị cần quản lý mã đơn vị và tên đơn vị, đối với nhân viên người ta quản lý các thông tin: mã nhân viên, họ tên, ngày sinh, giới tính, hệ số lương (hsl), mã đơn vị và mật khẩu.
Yêu cầu của bài toán:
Đối với đơn vị cần có các chức năng:

1. Hiển thị tất cả các đơn vị.
2. Thêm, xóa và sửa các đơn vị.
3. Tìm kiếm theo tên đơn vị.

Đối với nhân viên cần có các chức năng:

1. Thêm, xóa và đổi mật khẩu các nhân viên.
2. Hiển thị các nhân viên theo đơn vị.
3. Tìm kiếm nhân viên theo mã đơn vị hoặc họ tên.
4. Tìm kiếm nhân viên theo mã nhân viên.
5. Cho nhân viên đăng nhập vào hệ thống.

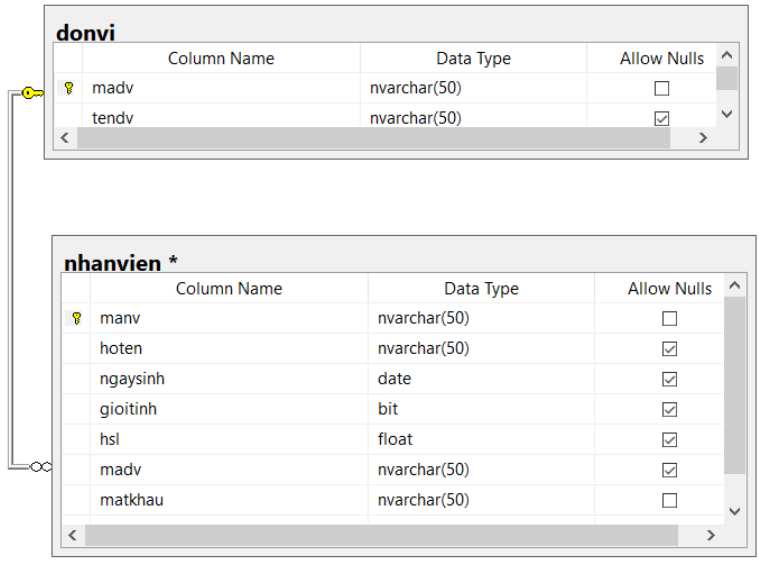
Để thực hiện các chức năng của bài toán, sinh viên làm các bài thực hành sau:

3.10.1. Bài thực hành 1

- Hãy thiết lập cấu hình để SQL Server cho phép đăng nhập bằng tài khoản sa với mật khẩu bất kỳ.

- Hãy mở giao thức TCP/IP và cổng 1433 cho SQL Server.

- Tạo cơ sở dữ liệu: Qlnv bao gồm 2 bảng donvi và nhanvien có quan hệ như Hình 3.18:



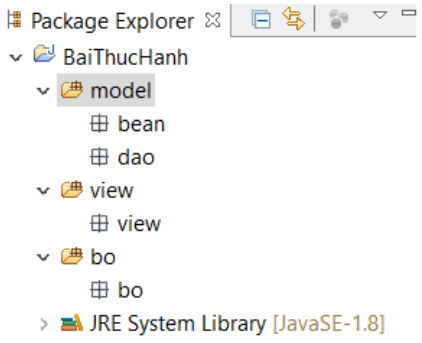
Hình 3.18. *Lược đồ quan hệ giữa các bảng*

- Hãy nhập dữ liệu vào bảng donvi và nhanvien.

3.10.2. Bài thực hành 2

1. Tạo các gói

Sử dụng Eclipse để tạo các Source Folder và các gói như Hình 3.19.



Hình 3.19. *Các gói trong Project*

Nhiệm vụ của mỗi gói:

model: Chứa các gói và lớp tách riêng với tầng bo (business objects) và tầng view:

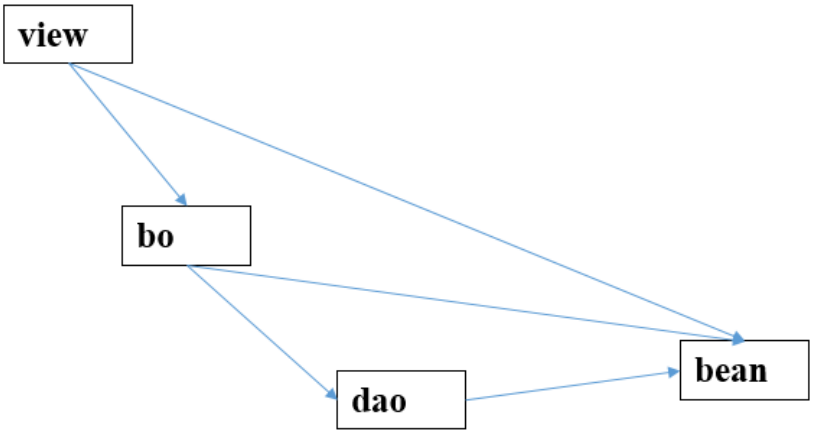
model.bean: Chứa các thực thể, gồm các trường (private), kèm theo các phương thức set/get, hàm tạo, ...

model.dao: Thực hiện các công việc liên quan đến CSDL như kết nối, lấy dữ liệu, truy vấn, chỉnh sửa, thêm xóa dữ liệu trực tiếp với cơ sở dữ liệu

bo: Truyền yêu cầu từ view chuyển qua dao, lấy dữ liệu từ dao về cho view. Nhiệm vụ của tầng này là xử lý các yêu cầu nghiệp vụ.

view: Dùng để nhập xuất dữ liệu của người dùng.

Một số quy tắc tương tác giữa các gói như thể hiện ở Hình 3.20:



Hình 3.20. *Quan hệ giữa các gói*

Trong đó:

bean chứa các thực thể, có thể sử dụng ở bất cứ nơi nào cần thiết như: **dao**, **bo** và **view**.

Các kết nối đến CSDL chỉ thực hiện ở **model.dao**, các tầng khác không liên quan đến CSDL.

dao chỉ cho phép gọi từ **bo**, các nơi khác không được gọi đến **dao**.

bo chỉ cho phép gọi từ **view**, các nơi khác không được gọi **bo**.

view: chỉ nhận và gửi dữ liệu từ **bo**.

2. Lập trình trên gói bean

Tại **model.bean**, hãy tạo ra lớp sau:

a. Lớp **donvibean** bao gồm các trường và phương thức như sau:

```
package bean;
public class donvibean {
    // Khai báo 2 trường madv và tendv
    private String madv;
    private String tendv;
    // Tạo constructor không tham số
    public donvibean() {
        super();
    }
    //Tạo constructor có đầy đủ tham số
    public donvibean(String madv, String tendv) {
        super();
        this.madv = madv;
        this.tendv = tendv;
    }
    //lần lượt viết các hàm để lấy giá tri ra (get)
    //và gán giá trị vào (set) các trường
    public String getMadv() {
        return madv;
    }
    public void setMadv(String madv) {
        this.madv = madv;
    }
    public String getTendv() {
        return tendv;
    }
    public void setTendv(String tendv) {
        this.tendv = tendv;
```

```
29      }
30      //Viết hàm tostring() để hiển thị giá trị các trường
31      @Override
32      public String toString() {
33          return madv + " " + tendv ;
34  }
35  }
```

Để kiểm tra các trường và phương thức trên lớp **donvibean**, mở lớp **donvibean** và thêm vào hàm **main()** để tạo ra 1 đơn vị và hiển thị thông tin của đơn vị vừa tạo:

```
1  public static void main(String[] args) {
2      // tạo ra một đơn vị
3       donvibean dv= new donvibean("cntt","Công nghệ thông tin");
4       // hiển thị thông tin của đơn vị
5       System.out.println(dv.toString());
6  }
```

Chú ý: Trên Eclipse có sẵn chức năng phát sinh ra mã lệnh, đối với mỗi lớp trên **model.bean** chúng ta chỉ cần gõ tên trường, sau đó Eclipse sẽ tự động sinh ra hàm tạo, hàm toString(), các hàm get/set tương ứng với từng trường.

Ví dụ: trong lớp **donvibean** sau khi gõ 2 trường madv và tendv, chúng ta phát sinh ra hàm tạo, hàm get/set, hàm toString() như sau:

Chọn vị trí cần phát sinh ra mã lệnh, kích chuột phải và chọn Source, sau đó chọn các chức năng phát sinh tương ứng như Hình 3.21, trong đó:

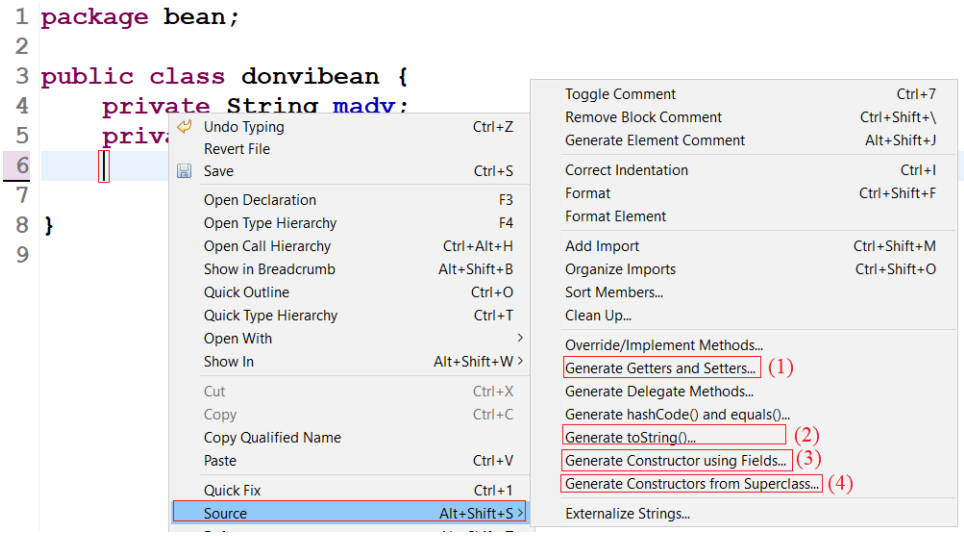
(1): phát sinh các hàm get/set;

(2): phát sinh hàm toString();

(3): phát sinh hàm tạo với các tham số;

(4): phát sinh hàm tạo không tham số.

b. Lớp **nhanvienbean**



Hình 3.21. *Phát sinh các hàm trên Eclipse*

Tương tự như lớp **donvibean**, hãy tạo lớp **nhanvienbean** bao gồm các trường và phát sinh các phương thức như sau:

```
1  package bean;
2  import java.util.Date;
3  public class nhanvienbean {
4      private String manv;
5      private String hoten;
6      private Date ngaysinh;
7      private Boolean gioitinh;
8      private Double hsl;
9      private String madv;
10     private String matkhau;
11     public nhanvienbean() {
12         super();
13     }
14     public nhanvienbean(String manv, String hoten, Date
           ngaysinh, Boolean gioitinh,Double hsl, String madv,
           String matkhau) {
15         super();
```

```
16          this.manv = manv;
17          this.hoten = hoten;
18          this.ngaysinh = ngaysinh;
19          this.gioitinh = gioitinh;
20          this.hsl = hsl;
21          this.madv = madv;
22          this.matkhau = matkhau;
23      }
24      public String getManv() {
25          return manv;
26      }
27      public void setManv(String manv) {
28          this.manv = manv;
29      }
30      public String getHoten() {
31          return hoten;
32      }
33      public void setHoten(String hoten) {
34          this.hoten = hoten;
35      }
36      public Date getNgaysinh() {
37          return ngaysinh;
38      }
39      public void setNgaysinh(Date ngaysinh) {
40          this.ngaysinh = ngaysinh;
41      }
42      public Boolean getGioitinh() {
43          return gioitinh;
44      }
45      public void setGioitinh(Boolean gioitinh) {
46          this.gioitinh = gioitinh;
47      }
48      public Double getHsl() {
49          return hsl;
```

```
50      }
51      public void setHsl(Double hsl) {
52          this.hsl = hsl;
53      }
54      public String getMadv() {
55          return madv;
56      }
57      public void setMadv(String madv) {
58          this.madv = madv;
59      }
60      public String getMatkhau() {
61          return matkhau;
62      }
63      public void setMatkhau(String matkhau) {
64          this.matkhau = matkhau;
65  }
66  }
```

Yêu cầu bổ sung:

Để kiểm tra các trường và phương thức trên lớp **nhanvienbean**, mở lớp **nhanvienbean** và sinh viên hãy thêm vào hàm **main()** để tạo ra 1 nhân viên bất kỳ và hiển thị thông tin của nhân viên vừa tạo.

3. Lập trình trên gói dao

Tại **model.dao**, hãy tạo ra lớp để thực hiện các công việc liên quan đến CSDL như kết nối, lấy dữ liệu, truy vấn, chỉnh sửa, thêm xóa dữ liệu trực tiếp với cơ sở dữ liệu. Trong gói **dao** này sẽ tạo ra các lớp sau:

- Lớp **DungChung**: gồm các phương thức dùng chung trong gói **dao** như kết nối vào cơ sở dữ liệu, lấy về tên cột của một bảng, lấy về dữ liệu của một bảng.
- Lớp **donvidao**: gồm các phương thức thao tác trên bảng **donvi** như lấy về tất cả các đơn vị, thêm, xóa, ...
- Lớp **nhanviendao**: gồm các phương thức thao tác trên bảng **nhanvien** như lấy về tất cả các nhân viên, thêm, ...

a. Xây dựng lớp DungChung

Lớp này bao gồm các phương thức:

- Kết nối vào cơ sỡ dữ liệu;
- Lấy dữ liệu của một bảng bất kỳ;
- Lấy về các tên cột của một bảng bất kỳ.

```
1  package dao;
2
3  import java.sql.Connection;
4  import java.sql.Driver;
5  import java.sql.DriverManager;
6  import java.sql.PreparedStatement;
7  import java.sql.ResultSet;
8  import java.sql.ResultSetMetaData;
9  import java.util.ArrayList;
10
11 public class DungChung {
12     //khai báo toàn cục một đường kết nối: cn
13     //đường kết nối này dùng chung trong toàn bộ Project
14     public static Connection cn;
15
16     //xây dựng hàm để kết nối vào máy chủ:server
17     //tên cơ sở dữ liệu: database, tên đăng nhập: un
18     //và mật khẩu: pass
19     public static void ketNoi(String server,String database,
           String un, String pass) throws Exception{
20         //b1: nạp trình điều khiển: SQL Server
21         Class.forName(
22         "com.microsoft.sqlserver.jdbc.SQLServerDriver");
23         System.out.println("Da xac xac dinh HQTCSDL");
24         //b2: tạo chuỗi kết nối và kết nối vào CSDL
25         String url=String.format("jdbc:sqlserver://%s:1433;
```

```
            databaseName=%s;user=%s; password=%s",
26          server,database,un,pass);
27          cn=DriverManager.getConnection(url);
28          System.out.println("Da ket noi");
29      }
30
31      //xây dựng hàm để lấy dữ liệu của bảng: tb
32      //hàm trả về 1 ResultSet chứa dữ liệu lấy về
33      public static ResultSet getBang(String tb) throws
            Exception{
34          //tạo câu lệnh SQL
35          String sql="select * from "+ tb;
36          //thực hiện câu lệnh SQL
37          PreparedStatement cmd= cn.prepareStatement(sql);
38          return cmd.executeQuery();
39      }
40
41      //Hàm trả về một mảng chứa tên cột của bảng: tb
42      public static ArrayList<String> getTenCot(String tb)
            throws Exception{
43          //Tạo ra 1 ArrayList để lưu tên các trường
44          ArrayList<String> ds= new ArrayList<String>();
45          // Lấy dữ liệu của bảng: tb về lưu vào 1 Resultset
46          ResultSet rs=getBang(tb);
47          // Lấy về bảng thông tin (metadata) của rs
48          ResultSetMetaData meta=rs.getMetaData();
49          // Lấy về tổng số cột
50          int socot=meta.getColumnCount();
51          // Lưu tên cột vào mảng
52          for(int i=1;i<=socot;i++)
53              ds.add(meta.getColumnName(i));
54          rs.close();
55          return ds;
56  }
```

```
}
```

Để kiểm tra các phương thức trên lớp, mở lớp **DungChung** và thêm vào 1 hàm **main()** để kiểm tra 3 phương thức vừa tạo:

```
public static void main(String[] args) {
try {
    DungChung dc= new DungChung();
    // kiểm tra hàm kết nối vào cơ sở dữ liệu
    dc.ketNoi("nhập tên server" , "qlnv", "sa", "nhập mật
        khẩu");
    //hiển thị tất cả tên cột của bảng donvi
    for(String tc: dc.getTenCot("donvi"))
        System.out.printf("%-20s",tc);
    // lấy về toàn bộ dữ liệu của bảng donvi
    ResultSet rs= dc.getBang("donvi");
    //hiển thị dữ liệu của bảng donvi
    while(rs.next())
        System.out.printf("%-20s %-20s \n",
            rs.getString("madv"), rs.getString("tendv"));
    rs.close();
    } catch (Exception e) {
        e.printStackTrace();
    }
}
```

b. Xây dựng lớp donvidao

Lớp **donvidao**: gồm các phương thức thao tác trên bảng **donvi** như lấy về tất cả các đơn vị, thêm một đơn vị, xóa một đơn vị theo mã đơn vị:

```
package dao;

import java.sql.PreparedStatement;
```

```
4  import java.sql.ResultSet;
5  import java.util.ArrayList;
6
7  import bean.donvibean;
8
9  public class donvidao {
10     //Xây dựng hàm getdv để lấy về tất cả các đơn vị
11     //hàm trả về 1 ArrayList<donvibean> chứa các đơn vị trong
12     //bảng donvi
13     public ArrayList<donvibean> getdv() throws Exception{
14         //Tạo ra 1 ArrayList để lưu các đơn vị
15         ArrayList<donvibean> ds= new ArrayList<donvibean>();
16         //Lấy về tất cả các đơn vị lưu vào 1 ResultSet
17         ResultSet rs= DungChung.getBang("donvi");
18         //Duyệt qua các đơn vị trong ResultSet: rs
19         while(rs.next()){
20             //lấy về mã đơn vị từ rs
21             String madv=rs.getString("madv");
22             //lấy về tên đơn vị từ rs
23             String tendv=rs.getString("tendv");
24             //tạo ra 1 đơn vị
25             donvibean dv= new donvibean(madv, tendv);
26             //lưu đơn vị vào ArrayList:ds
27             ds.add(dv);
28         }
29         //đóng ResultSet: rs
30         rs.close();
31         // trả về ArrayList: ds
32         return ds;
33     }
34     //Xây dựng hàm Them để thêm 1 đơn vị vào bảng donvi
35     //Hàm trả về số mẫu tin thêm được
36     public int Them(String madv, String tendv) throws
           Exception{
```

```
37          //b1:thiết lập câu lệnh SQL để thêm 1 đơn vị
38          String sql="Insert into donvi(madv,tendv)
                values(?,?)";
39          //b2: tạo 1 PreparedStatement để thực thi câu lệnh SQL
40          PreparedStatement cmd=
                DungChung.cn.prepareStatement(sql);
41          //b3: truyền tham số vào câu lệnh SQL
42          cmd.setString(1, madv);
43          cmd.setString(2, tendv);
44          //thực thi câu lệnh SQL và trả về số mẫu tin thêm được
45          return cmd.executeUpdate();
46      }
47      public int Xoa(String madv) throws Exception{
48          //b1:thiết lập câu lệnh SQL để thêm 1 đơn vị
49          String sql="delete from donvi where madv=?";
50          //b2: tạo 1 PreparedStatement để thực thi câu lệnh SQL
51          PreparedStatement cmd=
                DungChung.cn.prepareStatement(sql);
52          //b3: truyền tham số vào câu lệnh SQL
53          cmd.setString(1, madv);
54          //thực thi câu lệnh SQL và trả về số mẫu tin thêm được
55          return cmd.executeUpdate();
56   }
57  }
```

Để kiểm tra các phương thức trên lớp **donvidao**, sinh viên tạo thêm 1 hàm **main()** để kiểm tra 3 phương thức vừa tạo như sau:

```
1  public static void main(String[] args) {
2      try {
3          //tạo lớp donvidao
4          donvidao dvdao= new donvidao();
5          //tạo lớp DungChung để gọi hàm kết nối vào CSDL và
6          //hàm getTenCot
```

```
7           DungChung dc= new DungChung();
8           //kết nối vào CSDL
9           dc.ketNoi("nhập tên server", "qlnv", "sa", "nhập mật
                khẩu");
10
11          //thêm vào đơn vị có mã: dv123 và tên: Phòng Đào tạo
12          dvdao.Them("dv123", "Phòng Đào tạo");
13
14          //xóa đơn vị có mã đơn vị là dv1
15          dvdao.Xoa("dv1");
16
17          //hiển thị tên các cột của bảng đơn vị
18          for(String tc:dc.getTenCot("donvi"))
19              System.out.printf("%-20s",tc);
20
21          System.out.println();
22          //hiển thị ra tất cả các đơn vị
23          for(donvibean dv:dvdao.getdv())
24              System.out.printf("%-20s %-20s\n",
                    dv.getMadv(),dv.getTendv());
25      }catch(Exception e){
26      e.printStackTrace();
27      }
28  }
```

Yêu cầu bổ sung:

- Lớp **donvidao** ở trên chỉ có phương thức lấy về các đơn vị, thêm 1 đơn vị và xóa 1 đơn vị. Dựa vào phương thức Them hãy viết thêm phương thức Sua để sửa lại tên đơn vị của 1 đơn vị nào đó.

c. Xây dựng lớp nhanviendao

Lớp **nhanviendao** sẽ xây dựng các phương thức thao tác trên bảng **nhanvien** như lấy về tất cả các nhân viên, thêm một nhân viên:

```
1  package dao;
2
3  import java.sql.PreparedStatement;
4  import java.sql.ResultSet;
5  import java.util.ArrayList;
6  import java.util.Date;
7
8  import bean.donvibean;
9  import bean.nhanvienbean;
10
11 public class nhanviendao {
12     //Xây dựng hàm getnv để lấy về tất cả các nhân viên
13     //hàm trả về 1 ArrayList<nhanvienbean> chứa các
14     //nhân viên trong bảng nhanvien
15     public ArrayList<nhanvienbean> getnv() throws Exception{
16         //Tạo ra 1 ArrayList để lưu các nhân viên
17         ArrayList<nhanvienbean> ds= new
               ArrayList<nhanvienbean>();
18         //Lấy về tất cả các nhân viên lưu vào 1 ResultSet
19         ResultSet rs= DungChung.getBang("nhanvien");
20         //Duyệt qua các nhân viên trong ResultSet: rs
21         while(rs.next()){
22             //lấy về dữ liệu các trường trong rs
23             String manv=rs.getString("manv");
24             String hoten=rs.getString("hoten");
25             Date ngaysinh=rs.getDate("ngaysinh");
26             Boolean gioitinh=rs.getBoolean("gioitinh");
27             Double hsl=rs.getDouble("hsl");
28             String madv=rs.getString("madv");
29             String matkhau=rs.getString("matkhau");
30             //tạo ra 1 nhân viên
31             nhanvienbean nv = new nhanvienbean(manv, hoten,
                   ngaysinh, gioitinh, hsl, madv, matkhau);
32             //lưu nhân viên vào ArrayList:ds
```

```
33              ds.add(nv);
34          }
35          rs.close();//đóng ResultSet: rs
36          // trả về ArrayList: ds
37          return ds;
38      }
39      //Xây dựng hàm Them để thêm 1 nhân viên vào bảng nhanvien
40      //Hàm trả về số mẫu tin thêm được
41      public int Them(String manv, String hoten, Date ngaysinh,
            Boolean gioitinh,
42      Double hsl, String madv, String matkhau) throws Exception{
43          //b1:thiết lập câu lệnh sql để thêm 1 nhân viên
44          String sql="insert into
                nhanvien(manv,hoten,ngaysinh,gioitinh,hsl,madv,
                matkhau)values(?,?,?,?,?,?,?)";
45          //b2:tạo 1 PreparedStatement để thực thi câu lệnh sql
46          PreparedStatement cmd=
                DungChung.cn.prepareStatement(sql);
47          //b3: truyền 7 tham số vào câu lệnh sql
48          cmd.setString(1, manv);
49          cmd.setString(2, hoten);
50          //Đổi ngày util sang ngày sql
51          cmd.setDate(3, new java.sql.Date(
                ngaysinh.getTime()));
52          cmd.setBoolean(4, gioitinh);
53          cmd.setDouble(5, hsl);cmd.setString(6, madv);
54          cmd.setString(7, matkhau);
55          //thực thi câu lệnh sql và trả về số mẫu tin thêm được
56          return cmd.executeUpdate();
57  }
58  }
```

Yêu cầu bổ sung:

- Sinh viên tự viết các phương thức để: xóa một nhân viên theo mã nhân viên; đổi mật khẩu cho nhân viên.
- Để kiểm tra các phương thức trên lớp **nhanviendao**, sinh viên hãy tạo thêm hàm **main()** để gọi lại các phương thức trên lớp **nhanviendao**.

4. Lập trình trên gói bo

Nhiệm vụ của gói **bo**: khi có yêu cầy từ gói **view**, **bo** sẽ lấy dữ liệu về từ gói **dao**, xử lý tác nghiệp, chuyển kết quả ra cho **view**.

Tại **bo**, hãy tạo ra lớp để thực hiện các chức năng của bài toán:

- Lớp **HeThongbo** thực hiện các chức năng: kết nối vào CSDL và lấy về tên các cột của một bảng.
- Lớp **donvibo** thực hiện các chức năng: lấy về toàn bộ đơn vị, tìm kiếm theo tên đơn vị, thêm và xóa một đơn vị.
- Lớp **nhanvienbo** thực hiện các chức năng: lấy về toàn bộ nhân viên, tìm kiếm nhân viên theo mã đơn vị, tìm kiếm tương đối nhân viên theo họ tên hoặc mã đơn vị, thêm vào một nhân viên.

a. Xây dựng lớp HeThongbo

```
1  package bo;
2
3  import java.util.ArrayList;
4
5  import dao.DungChung;
6
7  public class HeThongbo {
8      //hàm này sẽ gọi hàm ketNoi() của lớp DungChung
9      //để kết nối vào CSDL
10     public static void KetNoi(String tenServer, String
           database, String user, String pass) throws Exception{
11         DungChung.ketNoi(tenServer, database, user, pass);
12     }
```

```
13      //hàm này sẽ gọi hàm getTenCot() của lớp DungChung
14      //để lấy về các tên cột của một bảng
15      public static ArrayList<String> getTenCot(String TenBang)
            throws Exception{
16          return DungChung.getTenCot(TenBang);
17      }
18  }
```

b. Xây dựng lớp donvibo

Lớp **donvibo** sẽ xây dựng các phương thức để quản lý đơn vị như:

- getdv(): lấy về tất cả các đơn vị.
- timdv(): tìm kiếm theo tên đơn vị.
- timdv(): thêm vào một đơn vị.

```
1   package bo;
2
3   import java.util.ArrayList;
4
5   import bean.donvibean;
6   import dao.donvidao;
7
8   public class donvibo {
9       //donvibo sẽ gọi các hàm của donvidao để lấy dữ liệu về
10      donvidao dvdao= new donvidao();
11      //các đơn vị lấy về sẽ lưu vào 1 mảng
12      ArrayList<donvibean> ds;
13
14      //xây dựng hàm getdv để lấy toàn bộ đơn vị về
15      // hàm trả về một mảng chứa các đơn vị
16      public ArrayList<donvibean> getdv() throws Exception{
17          ds=dvdao.getdv();
18          return ds;
```

```
19      }
20
21      //xây dựng hàm timdv() để tìm tương đối tên đơn vị
22      public ArrayList<donvibean> timdv(String key){
23          //tạo ra 1 mảng để lưu các đơn vị tìm được
24          ArrayList<donvibean> tam= new ArrayList<donvibean>();
25          //duyệt qua các đơn vị để tìm tên đơn vị
26          for(donvibean dv: ds)
27              if(dv.getTendv().toLowerCase().contains(
                    key.toLowerCase()))
28                  tam.add(dv);//nếu tìm được lưu vào mảng tam
29          return tam;
30      }
31
32      //xây dựng hàm để thêm vào 1 đơn vị
33      //kiểm tra xem có trùng mã hay không, nếu không:
34      //thêm vào 2 vị trí:
35          //1. Thêm vào bộ nhớ: ds
36          //2. gọi hàm thêm của donvidao để thêm vào csdl
37      // hàm trả về số đơn vị thêm được
38      public int Them(String madv, String tendv) throws
            Exception{
39          //kiểm tra trùng mã
40          for(donvibean dv: ds)
41          //nếu trùng mã đơn vị thì thoát ra khỏi hàm
42              if(dv.getMadv().equals(madv))
43                 return 0;
44          //tao thêm 1 đơn vị
45          donvibean dv_moi= new donvibean(madv, tendv);
46          //Thêm vào bộ nhớ: ds
47          ds.add(dv_moi);
48          //gọi hàm them của donvidao để thêm vào CSDL
49          return dvdao.Them(madv, tendv);
50      }
```

```
}
```

Yêu cầu bổ sung:

Trong **donvibo** sinh viên bổ sung thêm các chức năng như:

- Xóa một đơn vị theo mã đơn vị.
- Sửa tên đơn vị theo mã đơn vị.
- Để kiểm tra các phương thức trên lớp **donvibo**, sinh viên hãy tạo thêm hàm **main()** để gọi lại các phương thức trên lớp **donvibo**.

c. Xây dựng lớp nhanvienbo

Lớp **nhanvienbo** sẽ xây dựng các phương thức để quản lý nhân viên như:

- getdv(): lấy về tất cả các nhân viên.
- timdv(): tìm kiếm nhân viên theo mã đơn vị.
- tim(): tìm kiếm tương đối nhân viên theo họ tên hoặc mã đơn vị.
- them(): thêm vào một nhân viên.

```
package bo;

import java.util.ArrayList;
import java.util.Date;

import bean.nhanvienbean;
import dao.nhanviendao;

public class nhanvienbo {
    //nhanvienbo sẽ gọi các hàm từ nhanviendao
    nhanviendao nvdao= new nhanviendao();
    //các nhân viên lấy về sẽ lưu vào 1 mảng
    ArrayList<nhanvienbean> ds;

```

```
15      //xây dựng hàm getnv để lấy về toàn bộ nhân viên
16      // hàm trả về một mảng chứa các nhân viên
17      public ArrayList<nhanvienbean> getnv() throws Exception{
18          ds=nvdao.getnv();
19          return ds;
20      }
21
22      //xây dựng hàm timdv(),
23      // hàm trả về tất cả nhân viên của một đơn vị nào đó
24      public ArrayList<nhanvienbean> timdv(String madv){
25          //tạo ra 1 mảng để lưu các nhân viên tìm được
26          ArrayList<nhanvienbean> tam= new
                ArrayList<nhanvienbean>();
27          //duyệt qua các nhân viên trong ds để tìm ra nhân viên
28          // có mã đơn vị: madv
29          for(nhanvienbean nv: ds)
30              //nếu tìm được lưu vào mảng tam
31              if(nv.getMadv().equals(madv))
32                  tam.add(nv);
33          return tam;
34      }
35
36      //xây dựng hàm tìm nhân viên theo họ tên hoặc mã đơn vị
37      //hàm trả về 1 mảng chứa các nhân viên tìm được
38      public ArrayList<nhanvienbean> tim(String key){
39          ArrayList<nhanvienbean> tam= new
                ArrayList<nhanvienbean>();
40          for(nhanvienbean nv: ds)
41              if(nv.getHoten().toLowerCase().contains(
                    key.toLowerCase()) ||
                    nv.getMadv().toLowerCase().contains(
                    key.toLowerCase()))
42              tam.add(nv);
43          return tam;
```

```
44      }
45
46      //xây dựng hàm để thêm vào 1 nhân viên
47      //kiểm tra xem có trùng mã hay không, nếu không:
48      //thêm vào 2 vị trí:
49      //1. Thêm vào bộ nhớ: ds
50      //2. gọi hàm thêm của nhanviendao để thêm vào csdl
51      // hàm trả về số nhân viên thêm được
52      public int Them(String manv, String hoten, Date ngaysinh,
            Boolean gioitinh, Double hsl, String madv, String
            matkhau) throws Exception{
53          //kiểm tra trung mã
54          for(nhanvienbean nv: ds)
55              if(nv.getManv().equals(manv))
56                  return 0;
57          //tạo ra 1 nhân viên và thêm vào mảng: ds
58          nhanvienbean nv= new nhanvienbean(manv, hoten,
                ngaysinh, gioitinh, hsl, madv, matkhau);
59          ds.add(nv);
60          //gọi hàm thêm của nhanviendao để thêm vào csdl
61          return nvdao.Them(manv, hoten, ngaysinh, gioitinh,
                hsl, madv, matkhau);
62      }
63  }
```

Yêu cầu bổ sung:

Trong **nhanvienbo** sinh viên bổ sung thêm các chức năng như:

- Xóa một nhân viên theo mã nhân viên.
- Đổi mật khẩu cho một nhân viên.
- Tìm một nhân viên theo mã nhân viên, hàm trả về thông tin một nhân viên tìm được: **public nhanvienbean TimManv(String manv) throws Exception;**

- Dựa vào **manv** và **matkhau**, hãy tạo một phương thức để kiểm tra đăng nhập cho nhân viên:
  public nhanvienbean KtDangnhap(String manv, String matKhau) throws Exception
  nếu đăng nhập đúng hàm trả về thông tin của nhân viên, đăng nhập sai hàm trả về null.
- Để kiểm tra các phương thức trên lớp **nhanvienbo**, sinh viên hãy tạo thêm hàm **main()** để gọi lại các phương thức trên lớp này.

5. Lập trình trên gói view

Nhiệm vụ của gói **view** là để xuất nhập dữ liệu. Gói **view** sẽ gọi các phương thức từ gói **bo** để xuất dữ liệu hoặc truyền dữ liệu từ **view** vào **bo**

Tại **view**, hãy tạo ra lớp để thực hiện các chức năng của bài toán:

- Lớp **donviview** thực hiện các chức năng:
  - Hiển thị tất cả các đơn vị.
  - Tìm kiếm một đơn vị theo tên đơn vị.
  - Thêm vào cơ sở dữ liệu một đơn vị.
  - Một số chức năng bổ sung.
- Lớp **nhanvienview** thực hiện các chức năng:
  - Hiển thị tất cả các nhân viên.
  - Tìm kiếm các nhân viên theo mã đơn vị.
  - Tìm kiếm các nhân viên theo mã đơn vị hoặc họ tên.
  - Thêm vào cơ sở dữ liệu một nhân viên.
  - Một số chức năng bổ sung.
- Lớp **ChuongTrinh** xây dựng một menu để hiển thị các chức năng trong **donviview** và **nhanvienview**.

a. Xây dựng lớp donviview

```java
package view;

import java.util.ArrayList;
import java.util.Scanner;

import bean.donvibean;
import bo.HeThongbo;
import bo.donvibo;

public class donviview {
    //view sẽ lấy dữ liệu từ bo để hiển thị
    //tạo ra hethongbo để gọi hàm LaytenCot
    HeThongbo ht= new HeThongbo();
    //tạo ra lớp donvibo để gọi hàm getdv(), timdv() và them()
    donvibo dvbo=new donvibo();

    //khai báo 1 mảng lưu các tên cột của bảng đơn vị
    ArrayList<String> dsTencot;
    //khai báo 1 mảng để lưu các đơn vị
    ArrayList<donvibean> dsDonvi;
    public donviview(){
        try {
            // lấy ra tên các cột của bảng donvi
            dsTencot=ht.getTenCot("donvi");
            // lấy về thông tin của tất cả các đơn vị
            dsDonvi=dvbo.getdv();
        } catch (Exception e) {
           e.printStackTrace();
        }
    }
    //hàm này hiển thị tên cột của bảng donvi và
    // tất cả các đơn vị
    public void HienThiDonVi() throws Exception{
        System.out.println("Danh sách các đơn vị");
```

```
35          //hiển thị ra tên cột của bảng donvi
36          for(String tc:dsTencot)
37             System.out.printf("%-20s",tc);
38          System.out.println();
39          //hiển thị ra tất cả các đơn vị
40          for(donvibean dv:dsDonvi)
41             System.out.printf("%-20s %-20s\n"
                ,dv.getMadv(),dv.getTendv());
42      }
43
44      //hàm nhập vào tên đơn vị, sau đó hiển thị
45      //ra các đơn vị tìm được
46      public void timdv() throws Exception{
47          System.out.println("Nhập tên đơn vị cần tìm");
48          Scanner nhap= new Scanner(System.in);
49          String key=nhap.nextLine();
50
51          System.out.println("Các đơn vị tìm được:");
52
53          //hiển thị ra tên cột của bảng donvi
54          for(String tc:dsTencot)
55             System.out.printf("%-20s",tc);
56
57          System.out.println();
58          //hiển thị ra tất cả các đơn vị tìm được
59          for(donvibean dv:dvbo.timdv(key))
60             System.out.printf("%-20s %-20s\n"
                ,dv.getMadv(),dv.getTendv());
61      }
62
63      //hàm nhập vào mã đơn vị và tên đơn vị, sau đó thêm vào
64      // 1 đơn vị
65      public void them() throws Exception{
66          Scanner nhap= new Scanner(System.in);
```

```
67          System.out.println("Nhập mã đơn vị");
68          String madv=nhap.nextLine();
69          System.out.println("Nhập tên đơn vị");
70          String tendv=nhap.nextLine();
71          //gọi hàm them của donvibo để thêm vào một đơn vị
72          int kq = dvbo.Them(madv, tendv);
73          if(kq==0)
74             System.out.println("Mã dv: " + madv + " đã có trong
                cơ sở dữ liêu");
75          else
76             System.out.println("Thêm thành công");
77      }
78  }
```

Yêu cầu bổ sung:

Trong **donviview** sinh viên bổ sung thêm các chức năng như:

- Nhập vào 1 mã đơn vị, sau đó xóa đơn vị này.
- Nhập vào mã 1 mã đơn vị và tên đơn vị mới, sau đó sửa tên đơn vị này.
- Để kiểm tra các phương thức trên lớp **donviview**, sinh viên hãy tạo thêm hàm **main()** để gọi lại tất cả các phương thức trên lớp **donviview**.

b. Xây dựng lớp nhanvienview

Lớp **nhanvienview** sẽ gọi các phương thức trên lớp **nhanvienbo** thực hiện các chức năng:

- Hiển thị ra màn hình tất cả các nhân viên.
- Nhập vào 1 mã đơn vị, sau đó hiển thị các các nhân viên có mã đơn vị vừa nhập.
- Nhập vào 1 mã đơn vị hoặc 1 học tên, sau đó tìm và hiển thị các nhân viên có mã đơn vị hoặc họ tên vừa nhập.

```
1  package view;
2
3  import java.util.ArrayList;
4  import java.util.Scanner;
5
6  import bean.donvibean;
7  import bean.nhanvienbean;
8  import bo.HeThongbo;
9  import bo.nhanvienbo;
10
11
12 public class nhanvienview {
13     //view sẽ lấy dữ liệu từ bo
14     //tạo ra hethongbo để gọi hàm LaytenCot
15     HeThongbo ht= new HeThongbo();
16     //tạo ra lớp nhanvienbo để gọi các hàm:
17     //hàm getnv(), timdv(), tim() và them()
18     nhanvienbo nvbo=new nhanvienbo();
19     //khai báo 1 mang lưu các tên cột của bảng nhân viên\
20     ArrayList<String> dsTencot;
21     //khai báo 1 mảng để lưu các nhân viên
22     ArrayList<nhanvienbean> dsNhanvien;
23     // xây dựng hàm tạo để lấy về tên các cột và tất cả
24     // các nhân viên
25     public nhanvienview(){
26         try {
27             // lấy ra tên các cột của bảng nhân viên
28             dsTencot=ht.getTenCot("nhanvien");
29             // lấy về thông tin của tất cả các nhân viên
30             dsNhanvien=nvbo.getnv();
31         } catch (Exception e) {
32         e.printStackTrace();
33         }
```

```
34      }
35
36      //Hiển thị ra tên cột của bảng nhanvien
37      //và tất cả các nhân viên
38      public void HienthiNhanvien() throws Exception{
39          System.out.println("Danh sách các nhân viên");
40          //Hiển thị ra tên các cột của bảng nhanvien
41          for(String tc:dsTencot)
42              System.out.printf("%-15s",tc);
43
44          System.out.println();
45          // hiển thị ra các nhân viên
46          for(nhanvienbean nv:dsNhanvien)
47              nv.HienThi();
48      }
49      //hàm này nhập vào mã đơn vị, hiển thị ra tên các cột
50      // và các nhân viên có mã đơn vị: madv
51      public void timdv() throws Exception{
52          System.out.println("Nhập mã đơn vị:");
53          Scanner nhap= new Scanner(System.in);
54          String madv=nhap.nextLine();
55          System.out.println("Danh sách nhân viên có mã đơn vị:
                "+ madv);
56          //Hiển thị ra tên các cột của bảng nhanvien
57          for(String tc:dsTencot)
58              System.out.printf("%-15s",tc);
59          System.out.println();
60          //gọi hàm timdv() của nhanvienbo để hiển thị ra
61          // các nhân viên tìm được
62          for(nhanvienbean nv:nvbo.timdv(madv))
63              nv.HienThi();
64      }
65
66      //hàm này nhập vào họ tên hoặc mã đơn vị, hiển thị ra
```

```
67      // các nhân viên có mã đơn vị hoặc họ tên vừa nhập
68      public void tim() throws Exception{
69          System.out.println("Nhập mã đơn vị hoặc họ tên:");
70          Scanner nhap= new Scanner(System.in);
71          String key=nhap.nextLine();
72          System.out.println("Danh sách nhân viên tìm được:");
73          //Hiển thị ra tên các cột của bảng nhanvien
74          for(String tc:dsTencot)
75              System.out.printf("%-15s",tc);
76          System.out.println();
77          //gọi hàm tim() của nhanvienbo để hiển thị ra các nhân
                viên tìm được
78          for(nhanvienbean nv:nvbo.tim(key))
79              nv.HienThi();;
80      }
81      public void Them() throws Exception{
82          //tương tự hàm them của donviview
83          //sinh viên tự nhập vào 7 thông tin của 1 nhân viên
84          //gọi hàm them của nhanvienbo để thêm một nhân viên
85      }
86  }
```

Yêu cầu bổ sung:

Trong **nhanvienview** sinh viên bổ sung thêm các chức năng như:

- Nhập vào 1 mã nhân viên, sau đó xóa nhân viên này ra khỏi cơ sở dữ liệu.
- Nhập vào 1 mã nhân viên, sau đó đổi lại mật khẩu cho nhân viên này.
- Nhập vào 1 mã nhân viên, sau đó in ra thông tin của nhân viên này.
- Nhập vào mã nhân viên và mật khẩu, kiểm tra nhân viên này đăng nhập đúng hay không ? nếu đúng hiển thị ra thông tin của nhân viên này.

- Để kiểm tra các phương thức trên lớp **nhanvienview**, sinh viên hãy tạo thêm hàm **main()** để gọi lại các phương thức trên lớp này.

c. Xây dựng lớp ChuongTrinh

Sau khi đã có 2 lớp **donviview** và **nhanvienview**, xây dựng lớp **ChuongTrinh** để:

- Kết nối vào CSDL
- Tạo ra 1 menu gọi lại các chức năng của **donviview** và **nhanvienview** như sau:

```
1  package view;
2
3  import java.util.Scanner;
4
5  import bo.HeThongbo;
6
7  public class ChuongTrinh {
8      public static void main(String[] args) {
9         // tạo ra lớp HeThongbo để gọi hàm KetNoi vào CSDL
10         HeThongbo htbo= new HeThongbo();
11         donviview dvview = new donviview();
12         nhanvienview nvview= new nhanvienview();
13         try {
14             Scanner nhap= new Scanner(System.in);
15             System.out.println("Nhập tên Server: ");
16             String tenServer=nhap.nextLine();
17             System.out.println("Nhập tên mật khẩu: ");
18             String matKhau=nhap.nextLine();
19             htbo.KetNoi(tenServer, "qlnv", "sa", matKhau);
20             while(true){
21                 System.out.println();
```

```
22                  System.out.println("-- Menu Quản lý đơn vị
                        --");
23                  System.out.println("1. Hiển thị đơn vị ");
24                  System.out.println("2. Tìm đơn vị ");
25                  System.out.println("3. Thêm 1 đơn 1");
26                  System.out.println("-- Menu Quản lý các nhân
                        viên --");
27                  System.out.println("4. Hiển thị nhân viên ");
28                  System.out.println("5. Hiển thị nhân viên theo
                        đơn vi");
29                  System.out.println("6. Tìm họ tên hoặc mã đơn
                        vị ");
30                  System.out.println("7. Thêm 1 đơn vị");
31                  System.out.println("0. Kết thúc chương trình");
32                  System.out.print("Chọn 1 chức năng: ");
33
34                  int key=Integer.parseInt(nhap.nextLine());
35                  switch (key) {
36                      case 1: {
37                         dvview.HienThiDonVi();
38                         nhap.nextLine();
39                         break;
40                      }
41                      case 2: {
42                         dvview.timdv();
43                         nhap.nextLine();
44                         break;
45                      }
46                      case 3: {
47                         dvview.them();
48                         nhap.nextLine();
49                         break;
50                      }
51                      case 4: {
52                          nvview.HienthiNhanvien();
```

```
53                          nhap.nextLine();
54                          break;
55                      }
56                      case 5: {
57                         nvview.timdv();
58                         nhap.nextLine();
59                         break;
60                      }
61                      case 6: {
62                          nvview.tim();
63                          nhap.nextLine();
64                          break;
65                      }
66                      case 7: {
67                          System.out.println(" Chức năng này đang
                                cập nhật";
68                          nhap.nextLine();
69                          break;
70                      }
71
72                      case 0: return;
73                      default:
74                          break;
75                      }
76              }
77          } catch (Exception e) {
78              e.printStackTrace();
79          }
80
81      }
82  }
```

Yêu cầu bổ sung:

- Sau khi người dùng nhập tên Server, mật khẩu và đã kết nối

được vào CSDL. Sinh viết gọi hàm kiểm tra đăng nhập trên **nhanvienview** để kiểm tra xem người dùng đăng nhập thành công hay không? Nếu đăng nhập thành công thì mở ra menu để người dùng lựa chọn các chức năng, nếu đăng nhập sai thì thông báo "Đăng nhập sai"

- Sinh viên bổ sung các mục trên menu để gọi các chức năng trên yêu cầu bổ sung của mục **a. Xây dựng lớp donviview** và **b. Xây dựng lớp nhanvienview**.

NHÀ XUẤT BẢN ĐẠI HỌC HUẾ

07 Hà Nội, Huế - Điện thoại: 0234.3834486; Fax: 0234.3819886

Website: http://nhaxuatban.hueuni.edu.vn

Chịu trách nhiệm xuất bản

Quyền Giám đốc: TS. Trần Bình Tuyên

Chịu trách nhiệm nội dung

Quyền Tổng biên tập: TS. Nguyễn Chí Bảo

Phản biện giáo trình

PGS. TS. Trương Công Tuấn

TS. Nguyễn Công Hào

Biên tập viên

Tôn Nữ Quỳnh Chi

Biên tập kỹ thuật

Ngô Văn Cường

Trình bày, minh họa

Minh Hoàng

Sửa bản in

Nguyễn Hoàng Hà

Đối tác liên kết xuất bản

Đại học Huế, 03 Lê Lợi, thành phố Huế

GIÁO TRÌNH
XML VÀ ỨNG DỤNG

In 180 bản khổ 17 × 25 cm tại công ty TNHH MTV in và dịch vụ Thanh Minh, 99 Phan Văn Trường, phường Vĩ Dạ, thành phố Huế. Số xác nhận đăng ký xuất bản: 1548 - 2019/CXBIPH/07-12/ĐHH. Quyết định xuất bản số: 38/QĐ/ĐHH-NXB cấp ngày 21/05/2019. In xong và nộp lưu chiểu năm 2019.

Mã số ISBN: 978-604-974-152-4