



CHƯƠNG 1 GIỚI THIỆU JAVA



1



Java

Java?

- 1995
- Oracle
- Java là một trong những ngôn ngữ lập trình phổ biến
- Hỗ trợ mạnh về hướng đối tượng

2



Java

Ứng dụng

- Mobile applications (specially Android apps)
- Desktop applications
- Web applications
- Web servers and application servers
- Games
- Database connection



Java

Tại sao dùng java?

- Chạy trên nhiều platforms (Windows, Mac, Linux, Raspberry Pi, etc.)
- Rất phổ biến.
- Đơn giản và dễ dùng.
- Open-source và hoàn toàn miễn phí.
- Secure, fast, powerful.
- Hệ sinh thái cộng đồng hỗ trợ cực lớn (tens of millions of developers)
- OOP cấu trúc rõ ràng, sử dụng lại mã, giảm chi phí phát triển.
- Java gần giống với C ++ và C# nên lập trình viên dễ dàng chuyển sang Java hoặc ngược lại.



Cài đặt java trong Windows OS

Java Development Kit - JDK 1.8

- Tải JDK 1.7 tại <https://www.oracle.com/java/technologies/downloads/#jdk17-windows>
Hoặc <http://jdk.java.net/>

IDE (Integrated Development Environment)

- Netbeans: <https://netbeans.apache.org/download/>
- IntelliJ IDEA
- Eclipse

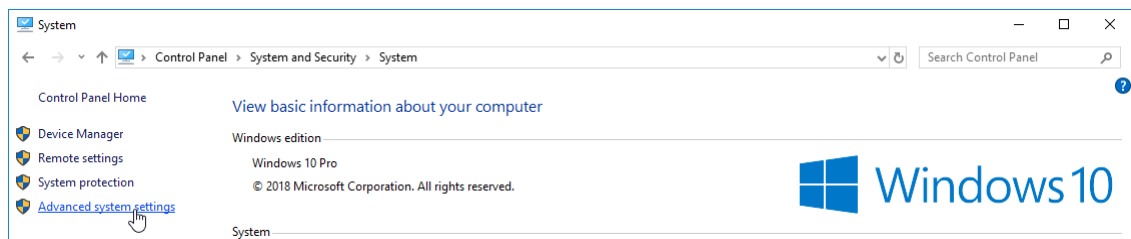
Xem video hướng dẫn: https://youtu.be/13_1XVglC1g

5



Cài đặt java trong Windows OS

• Bước 1

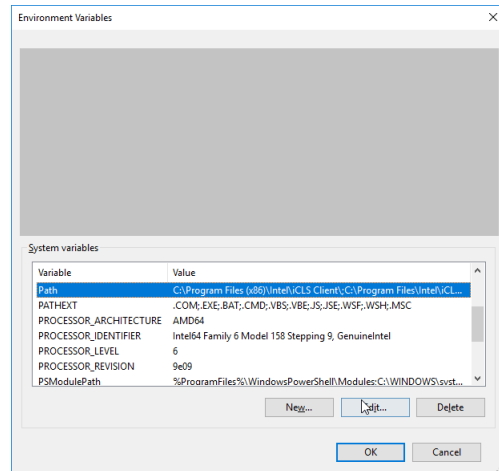
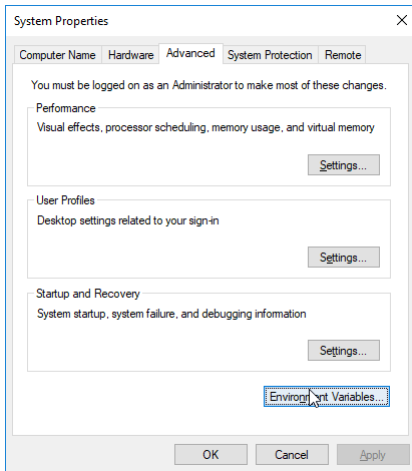


6



Cài đặt java trong Windows OS

• Bước 1



Khoa Khoa học và Kỹ thuật Thông tin – Trường ĐH CNTT Tp.HCM

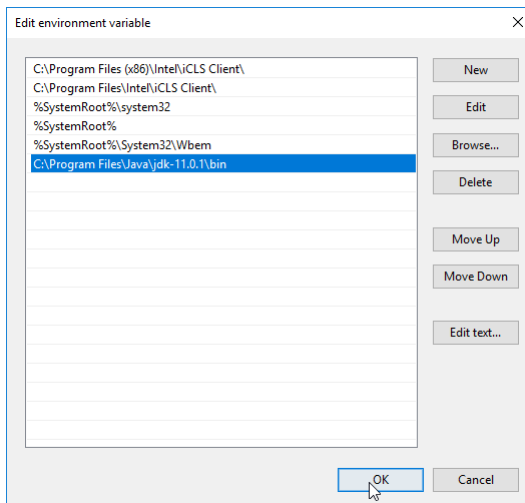
7

7



Cài đặt java trong Windows OS

• Bước 1



cmd.exe

Khoa Khoa học và Kỹ thuật Thông tin – Trường ĐH CNTT Tp.HCM

8

8



Tạo project

Project đầu tiên

- Sau khi cài đặt xong, tạo project đầu tiên “Hello World”

```
public class Main {
    public static void main(String[] args) {
        System.out.println("Hello World");
    }
}
```



Comments

Comments?

- Single-line Comments: Dùng //

```
// This is a comment
System.out.println("Hello World");
```

- Multi-line Comments: Dùng /* ... */

```
/* The code below will print the words Hello World to the screen,
and it is amazing */
System.out.println("Hello World");
```



Nhập – Xuất

Nhập dữ liệu từ bàn phím

- Nhập thư viện thư viện java.util
- Sử dụng lớp Scanner để lấy đầu vào từ người dùng
- Code:

```
Scanner ip = new Scanner(System.in);
ip.nextInt(); // Gọi hàm nhập số nguyên
```

In thông tin ra màn hình

- Cú pháp:

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



Nhập – Xuất

Các phương thức hỗ trợ nhập

Phương thức	Mô tả
nextBoolean()	Nhập vào kiểu Boolean (true – false)
nextByte()	Nhập vào kiểu dữ liệu Byte
nextShort()	Nhập vào kiểu Short (số nguyên từ -32768 đến 32767)
nextInt()	Nhập vào kiểu số nguyên từ bàn phím
nextLong()	Nhập vào số nguyên lớn
nextFloat()	Nhập vào kiểu số thực
nextDouble()	Nhập vào kiểu số thực Double
nextLine()	Nhập vào kiểu String



Nhập – Xuất

Ví dụ

```
import java.util.Scanner; // Import the Scanner class

class Main {
    public static void main(String[] args) {
        Scanner myObj = new Scanner(System.in); // Create a Scanner object

        System.out.println("Enter username");
        String userName = myObj.nextLine(); // Read user input

        System.out.println("Username is: " + userName); // Output user input
    }
}
```

13

```
import java.util.Scanner;

class Main {
    public static void main(String[] args) {
        Scanner myObj = new Scanner(System.in);
        System.out.println("Enter name, age and salary:");

        // String input
        String name = myObj.nextLine();

        // Numerical input
        int age = myObj.nextInt();
        double salary = myObj.nextDouble();

        // Output input by user
        System.out.println("Name: " + name);
        System.out.println("Age: " + age);
        System.out.println("Salary: " + salary);
    }
}
```

14



print – println – printf

- **print**: in ra màn hình, kết thúc không xuống dòng
- **println**: in ra màn hình, kết thúc xuống dòng
- **printf**: in ra màn hình, có định dạng kết quả. Sử dụng printf phải nhớ các ký hiệu định dạng. Một vài ký hiệu cơ bản:

STT	Ký hiệu	Mô tả
1	%c	Ký tự
2	%s	Chuỗi
3	%d	Số thập phân (số nguyên) (cơ số 10)
4	%f	Dấu phẩy động
5	%e	Dấu phẩy động theo cấp số nhân
6	%i	Số nguyên
7	%t	Định dạng ngày / giờ

Khoa Khoa học và Kỹ thuật Thông tin – Trường ĐH CNTT Tp.HCM

15

15



ĐẠI HỌC QUỐC GIA TP. HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN



CHƯƠNG 2 KIỂU DỮ LIỆU – TOÁN TỬ



16



NỘI DUNG

1. Biến số ✓
2. Hằng số ✓
3. Các loại kiểu dữ liệu ✗
4. Toán tử

17



Biến (variable)

Biến?

- Biến dùng để lưu trữ giá trị.

- Cú pháp:

1:

```
<data type> <variable_name>;
<variable_name> = <value>;
```

2:

```
<data type> <variable_name> = <value>;
```

- Có thể khai báo nhiều biến trên cùng một dòng.

18



Biến (variable)

Khuyến nghị

- Nên bắt đầu bằng một chữ cái viết thường. Ví dụ: id, name.
- Không nên bắt đầu bằng các ký tự đặc biệt &, \$, _.
- Nếu tên biến dài, hãy bắt đầu bằng chữ cái viết thường theo sau là chữ cái viết hoa. Ví dụ FirstName, lastName...
- Hạn chế đặt tên biến một ký tự như x, y, z.

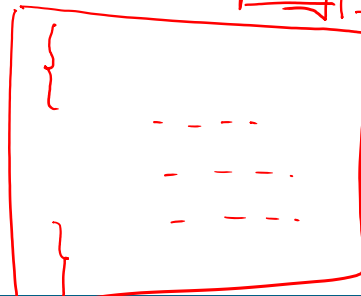


Biến (variable)

Phân loại

- **Member variable**: là một biến được kết hợp với một đối tượng cụ thể, được khai báo trong class (*trong chương OOP sẽ học thêm Instance variable, Class/static variable*).
- **Local variable**: là một biến được khai báo bên trong các block, method, constructor.

oop





Hằng số (Constant)

Hằng số?

- Hằng số là một biến được khởi tạo trước giá trị và không thể thay đổi, gán giá trị mới.

- Cú pháp

```
final datatype CONSTANTNAME = VALUE;
```

- Ví dụ:

```
final int x = 10; // declare a integer constant x = 10
```

```
final long y = 20L; // declare a long constant y = 20
```



Kiểu dữ liệu

Chia thành 2 nhóm:

- Primitive data types (*gọi là kiểu nguyên thủy/cơ sở*):

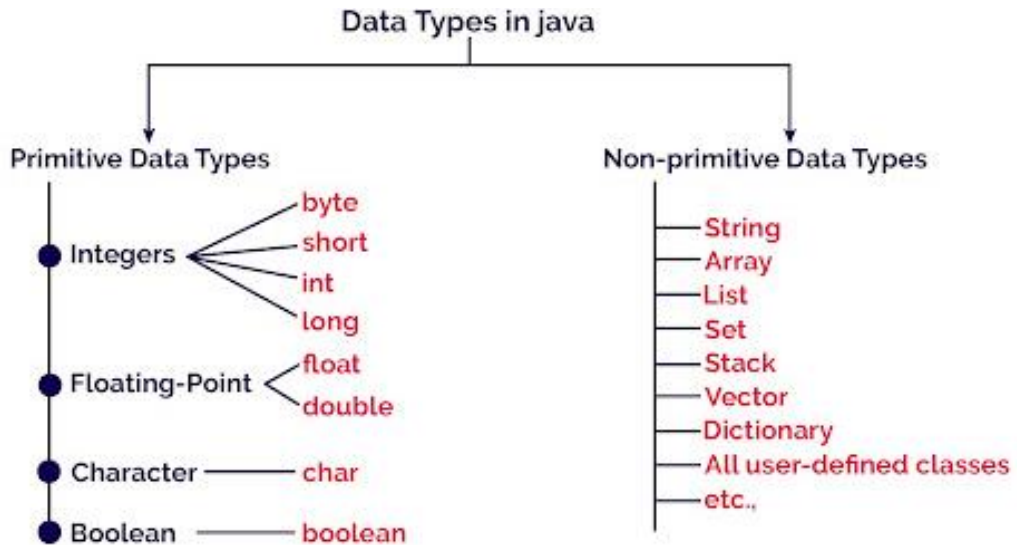
- byte
- short, int, long, float, double
- boolean
- và char

- Non-primitive data types:

- String,
- Arrays
- và Classes



Kiểu dữ liệu



Khoa Khoa học và Kỹ thuật Thông tin – Trường ĐH CNTT Tp.HCM

23

23



Kiểu dữ liệu

Độc thêm (Java bao gồm hai kiểu dữ liệu xây dựng sẵn):

1. Hướng đối tượng
 2. Không hướng đối tượng (chính là các kiểu nguyên thủy/cơ sở)
- Các kiểu dữ liệu hướng đối tượng của Java được định nghĩa thông qua việc dùng class
 - Trọng tâm của java là 8 kiểu dữ liệu cơ sở (hay nguyên thủy - primitive), các kiểu này không phải là hướng đối tượng, mà là các kiểu nhị phân bình thường.
 - Tất cả các kiểu dữ liệu khác được xây dựng từ các kiểu cơ sở này.

Khoa Khoa học và Kỹ thuật Thông tin – Trường ĐH CNTT Tp.HCM

24

24



Kiểu dữ liệu nguyên thủy trong java

STT	Kiểu dữ liệu	Ý nghĩa
1	boolean	Đại diện cho các giá trị true/false
2	char	Ký tự
3	byte	Số nguyên 8-bit
4	double	Số dấu chấm động độ chính xác kép
5	float	Số dấu chấm động độ chính xác đơn
6	int	Số nguyên
7	long	Số nguyên dài
8	short	Số nguyên ngắn



Kiểu dữ liệu nguyên thủy trong java

Data Type	Size	Description
byte	1 byte	Stores whole numbers from -128 to 127
short	2 bytes	Stores whole numbers from -32,768 to 32,767
int	4 bytes	Stores whole numbers from -2,147,483,648 to 2,147,483,647
long	8 bytes	Stores whole numbers from -9,223,372,036,854,775,808 to 9,223,372,036,854,775,807
float	4 bytes	Stores fractional numbers. Sufficient for storing 6 to 7 decimal digits
double	8 bytes	Stores fractional numbers. Sufficient for storing 15 decimal digits
boolean	1 bit	Stores true or false values
char	2 bytes	Stores a single character/letter or ASCII values



Chuyển kiểu dữ liệu (Type Casting)

Phương pháp: Trong java, có 2 loại chuyển kiểu dữ liệu

1. Widening Casting (automatically) - converting a smaller type to a larger type size

byte -> short -> char -> int -> long -> float -> double

2. Narrowing Casting (manually) - converting a larger type to a smaller size type

double -> float -> long -> int -> char -> short -> byte

- Cú pháp narrowing

<tên biến 2> = (kiểu dữ liệu) <tên biến 1>

- Ví dụ int i = (int)3.0



Chuyển kiểu dữ liệu (Type Casting)

Ví dụ (Widening Casting)

```
public class Main {
    public static void main(String[] args) {
        int myInt = 9;
        double myDouble = myInt; // Automatic casting: int to double

        System.out.println(myInt); // Outputs 9
        System.out.println(myDouble); // Outputs 9.0
    }
}
```



Chuyển kiểu dữ liệu (Type Casting)

Ví dụ (Narrowing Casting)

```
public class Main {
    public static void main(String[] args) {
        double myDouble = 9.78d;
        int myInt = (int) myDouble; // Manual casting: double to int

        System.out.println(myDouble); // Outputs 9.78
        System.out.println(myInt); // Outputs 9
    }
}
```



Lớp Math

Phương thức	Mô tả
Math.PI	Hằng số pi
Math.max(x, y)	Trả về giá trị lớn nhất của 2 số
Math.min(x, y)	Trả về giá trị nhỏ nhất của 2 số
Math.pow(x, y)	Tính x^y
Math.sqrt(x)	Lấy căn bậc 2
Math.abs(x)	Tính giá trị tuyệt đối
Math.sin(x), Math.cos(x)	Tính sin, cos
Math.random()	Trả về một số thuộc [0; 1)
Math.toDegrees(x)	Đổi radian => độ. Ngược lại, Math.toRadians()
Math.ceil(x)	Làm tròn tăng, trả về double. Ví dụ 9.234 => 10.0
Math.floor(x)	Làm tròn giảm, trả về double. Ví dụ 9.234 => 9.0



Lớp Math

Làm tròn số:

```

11 public class NumberRound {
12     public static void main(String[] args) {
13         double n = 8.123456789d;
14
15         System.out.println(Math.round(n));
16         System.out.println((double)Math.round(n*1000)/1000);
17     }
18 }

```

Output - Chuong02 (run) x

```

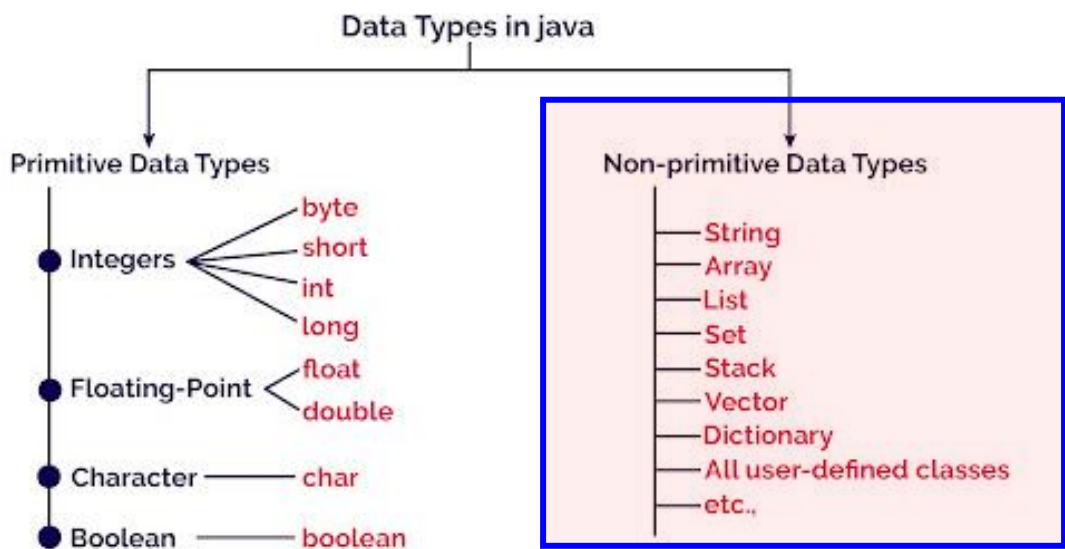
run:
8
8.123
BUILD SUCCESSFUL (total time: 0 seconds)

```

33



Non-primitive data types



34



Non-primitive data types/Reference data types

Khai báo

<Object Type> <Object Variable>;

- **Khởi tạo** (initialize)

<Object Type> <Object Variable> = new <Object Type>;

- **Truy xuất**

<Object Variable>.<Properties>

<Object Variable>.<Methods>



String

Giới thiệu

- Chuỗi được sử dụng để lưu trữ các ký tự (hoặc văn bản).
- Cú pháp khởi tạo:

String <tên_chuỗi> = “các ký tự”

- Ví dụ: String txt = "ABCDEFGH IJKLMNOPQRSTUVWXYZ";
- Chuỗi txt trở thành 1 đối tượng. Có thể gọi các phương thức đã được xây dựng sẵn.



String

Built-in method

Phương thức	Mô tả
char charAt(int index)	Trả về ký tự theo chỉ số index
int length()	Trả về độ dài chuỗi
String substring(int start)	Trả về chuỗi con tại vị trí index
String substring(start, stop)	Trả về chuỗi con từ start đến stop
boolean contains(s)	Chuỗi s có nằm trong string
static String join()	Nối nhiều chuỗi với nhau
boolean isEmpty()	Kiểm tra chuỗi rỗng
String concat(String str)	Nối chuỗi với nhau
String replace(String old, String new)	Thay thế
static String equalsIgnoreCase(String another)	So sánh chuỗi, phân biệt hoa thường
String[] split(String regex)	Tách chuỗi

Khoa Khoa học và Kỹ thuật Thông tin – Trường ĐH CNTT Tp.HCM

37

37



String

Built-in method

Phương thức	Mô tả
int indexOf(...)	Trả về vị trí của ký tự, chuỗi
String toLowerCase()	Trả về chữ thường
String toUpperCase()	Trả về chữ hoa
String trim()	Xóa khoảng trắng đầu cuối
static String valueOf(int value)	Chuyển value thành chuỗi. Ngược lại thì sao?
boolean equals(Object)	Kiểm tra chuỗi có tương đương với một object

Escape character	Result	Description
\'	'	Single quote
\"	"	Double quote
\\	\	Backslash

Khoa Khoa học và Kỹ thuật Thông tin – Trường ĐH CNTT Tp.HCM

38

38



String

Ví dụ

- Minh họa trong Netbeans



Array type

Array type?

- Array được dùng để lưu trữ nhiều giá trị trong 1 biến.
- Dùng chỉ số để lấy giá trị trong mảng.

Khai báo

```
<DT>[] <Array Name>; // mảng 1 chiều
<DT> <Array Name>[]; // mảng 1 chiều
<DT>[][] <Array Name>; // mảng 2 chiều
<DT> <Array Name>[][]; // mảng 2 chiều
```



Array type

Initialization

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

Allocating & accessing an array

```
int [] arrInt = new int[100];
int arrInt[100]; // Error.
indexes of a n-element array: from 0 to n-1
```



Toán tử (Operators)

Java chia các toán tử thành 5 nhóm sau:

- Toán tử số học (*Arithmetic operators*)
- Toán tử gán (*Assignment operators*)
- Toán tử so sánh (*Comparison operators*)
- Toán tử logic (*Logical operators*)
- Toán tử dịch bit (*Bitwise operators*)



Toán tử (Operators)

Toán tử số học (Arithmetic operators)

Operator	Name	Description	Example
+	Addition	Adds together two values	$x + y$
-	Subtraction	Subtracts one value from another	$x - y$
*	Multiplication	Multiplies two values	$x * y$
/	Division	Divides one value by another	x / y
%	Modulus	Returns the division remainder	$x \% y$
++	Increment	Increases the value of a variable by 1	$++x$
--	Decrement	Decreases the value of a variable by 1	$--x$



Toán tử (Operators)

Toán tử gán (Assignment operators)

Operator	Example	Same As
=	$x = 5$	$x = 5$
+=	$x += 3$	$x = x + 3$
-=	$x -= 3$	$x = x - 3$
*=	$x *= 3$	$x = x * 3$
/=	$x /= 3$	$x = x / 3$
%=	$x \% = 3$	$x = x \% 3$
&=	$x \& = 3$	$x = x \& 3$
=	$x = 3$	$x = x 3$
^=	$x \wedge = 3$	$x = x \wedge 3$
>>=	$x >> = 3$	$x = x >> 3$
<<=	$x << = 3$	$x = x << 3$



Toán tử (Operators)

Toán tử so sánh (Comparison operators)

Operator	Name	Example
==	Equal to	$x == y$
!=	Not equal	$x != y$
>	Greater than	$x > y$
<	Less than	$x < y$
>=	Greater than or equal to	$x >= y$
<=	Less than or equal to	$x <= y$



Toán tử (Operators)

Toán tử logic (Logical operators)

Operator	Name	Description	Example
&&	Logical and	Returns true if both statements are true	$x < 5 \ \&\& \ x < 10$
	Logical or	Returns true if one of the statements is true	$x < 5 \ \ x < 4$
!	Logical not	Reverse the result, returns false if the result is true	$!(x < 5 \ \&\& \ x < 10)$



Toán tử (Operators)

Toán tử dịch bit (Bitwise operators)

Operator	Meaning
&	AND
	OR
^	XOR
<<	Left Shift
>>	Right Shift
~	Bitwise Compliment



BÀI TẬP VỀ NHÀ

Anh/chị tìm hiểu các kiểu sau:

- Date and Time
 - ArrayList
 - LinkedList
 - HashMap
 - HashSet
 - Iterator
- Hướng dẫn: tham khảo <https://www.w3schools.com/java/default.asp>



ĐẠI HỌC QUỐC GIA TP. HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN



CHƯƠNG 3 CẤU TRÚC RÊ NHÁNH



49



Giới thiệu

Các dạng cấu trúc rẽ nhánh

if
if ... else
if ... else if
switch ... case

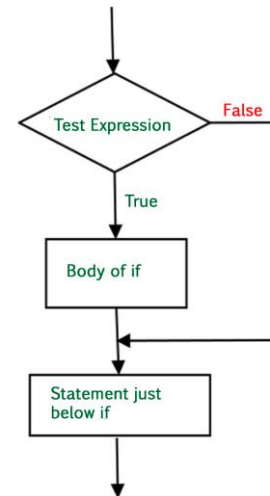
50



if

Cú pháp

```
if (condition) {
    // block of code to be executed
    // if the condition is true
}
```



if

Ví dụ

```
int x = 20;
int y = 18;

if (x > y) {
    System.out.println("x is greater than y");
}

System.out.println("Next line.");
```



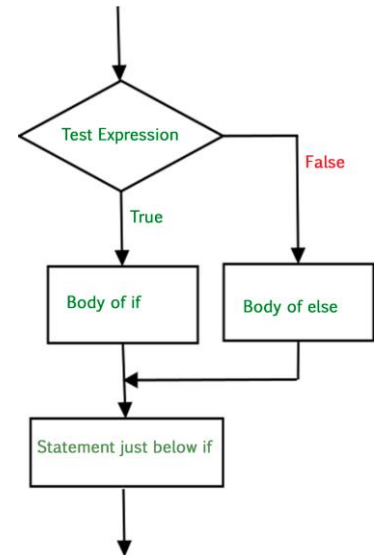
if ... else

Cú pháp

```

if (condition) {
    // block of code to be executed
    // if the condition is true
}
else {
    // block of code to be executed
    // if the condition is false
}

```



if ... else

Ví dụ

```

int time = 20;

if (time < 18) {
    System.out.println("Good day.");
}
else {
    System.out.println("Good evening.");
}

```



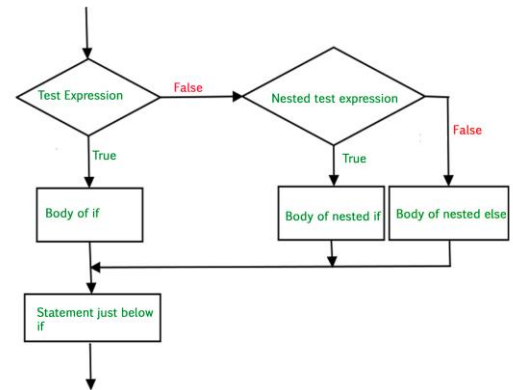
if ... else if

Cú pháp

```

if (condition1) {
    // block of code to be executed
    if condition1 is true
}
else if (condition2) {
    // block of code to be executed
    if the condition1 is false and
    condition2 is true
}
else {
    // block of code to be executed
    if the condition1 is false and
    condition2 is false
}

```



if ... else if

Ví dụ

```

int time = 22;

if (time < 10) {
    System.out.println("Good morning.");
}
else if (time < 20) {
    System.out.println("Good day.");
}
else {
    System.out.println("Good evening.");
}

```



if ... else if

Toán tử tam đoạn (ternary operator)

```
variable = (condition) ? expressionTrue : expressionFalse;
```

Ví dụ

```
int time = 20;
String result = (time < 18) ? "Good day." : "Good evening.";
System.out.println(result);
```



switch

Cú pháp:

```
switch(expression) {
    case x:
        // code block
        break;

    case y:
        // code block
        break;

    default:
        // code block
}
```

```
int day = 4;
switch (day) {
    case 1:
        System.out.println("Monday");
        break;
    case 2:
        System.out.println("Tuesday");
        break;
    case 3:
        System.out.println("Wednesday");
        break;
    case 4: System.out.println("Thursday");
        break;
    case 5:
        System.out.println("Friday");
        break;
    case 6:
        System.out.println("Saturday");
        break;
    case 7:
        System.out.println("Sunday");
        break; }
```

59

59

```
int day = 4;
switch (day) {
    case 6:
        System.out.println("Today is Saturday");
        break;
    case 7:
        System.out.println("Today is Sunday");
        break;
    default:
        System.out.println("Looking forward to the Weekend");
}
```

60

60



Bài tập

Anh/chị nhập điểm trung bình (GPA) từ bàn phím. Sau đó thông báo kết xếp hạng học lực:

- Nếu $GPA < 3.5$ thì “Học lực Kém”
- Nếu $3.5 \leq GPA < 5.0$ thì “Học lực Yếu”
- Nếu $5.0 \leq GPA < 7.0$ thì “Học lực Trung bình”
- Nếu $7.0 \leq GPA < 8.0$ thì “Học lực Khá”
- Nếu $8.0 \leq GPA < 9.0$ thì “Học lực Giỏi”
- Nếu $9.0 \leq GPA \leq 10$ thì “Học lực Xuất sắc”

Mở rộng: random điểm gpa thuộc $[0; 10]$ để test cho nhanh.



ĐẠI HỌC QUỐC GIA TP. HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN



CHƯƠNG 4 CẤU TRÚC LẬP





Giới thiệu

Các loại cấu trúc lặp

while
do ... while
for



while

Cú pháp

```
while (condition) {  
    // code block to be executed  
}
```



while

Ví dụ

```
import java.util.Scanner;

public class Demo_while {
    public static void main(String[] args) {
        Scanner in = new Scanner(System.in);
        int x = in.nextInt();
        while (x <= 0) {
            x = in.nextInt();
        }

        System.out.println("Giá trị x = " + x);
    }
}
```



do ... while

Cú pháp

```
do {
    // code block to be executed

} while (condition);
```




do ... while

Ví dụ

```

import java.util.Scanner;

public class Demo_do_while {
    public static void main(String[] args) {
        Scanner in = new Scanner(System.in);
        int x;
        do {
            x = in.nextInt();
        } while (x <= 0);

        System.out.println("Giá trị x = " + x);
    }
}

```



for

Cú pháp

```

for (statement 1; statement 2; statement 3) {
    // code block to be executed
}

```

• Ví dụ

```

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

```



For-Each Loop

Cú pháp

```
for (type variableName : arrayName) {
    // code block to be executed
}
```

• Ví dụ:

```
String[] cars = {"Volvo", "BMW", "Ford", "Mazda"};

for (String i : cars) {
    System.out.println(i);
}
```



Break và Continue

```
for (int i = 0; i < 10; i++) {
    if (i == 4) {
        break;
    }
    System.out.println(i);
}
```

```
for (int i = 0; i < 10; i++) {
    if (i == 4) {
        continue;
    }
    System.out.println(i);
}
```



Break và Continue

```
int i = 0;
while (i < 10) {
    System.out.println(i);
    i++;
    if (i == 4) {
        break;
    }
}
```

```
int i = 0;
while (i < 10) {
    if (i == 4) {
        i++;
        continue;
    }
    System.out.println(i);
    i++;
}
```



BÀI TẬP

BÀI #1

Viết chương trình tính tổng biểu thức:

$$S = n! = 1 + 2 + 3 + \dots + n$$

Code it!

Viết chương trình tính giai thừa của một số cho trước n (nguyên dương) được nhập từ bàn phím.

$$S = n! = 1.2.3...n$$



BÀI TẬP

BÀI #2

Viết chương trình in ra bản cửu chương 2 đến cửu chương 9.

Ví dụ:

$1 \times 2 = 2$
 $2 \times 2 = 4$
 $3 \times 2 = 6$
 $4 \times 2 = 8$
 $5 \times 2 = 10$
 $6 \times 2 = 12$
 $7 \times 2 = 14$
 $8 \times 2 = 16$
 $9 \times 2 = 18$
 $10 \times 2 = 20$

Code it!



BÀI TẬP

BÀI #3

Viết chương trình có thể tạo một mảng chứa các giá trị bình phương của các số thực được chọn ngẫu nhiên từ 18 đến 99. Số lượng các phần tử được chọn ngẫu nhiên từ 20 đến 30.

Code it!



BÀI TẬP

BÀI #4

Viết chương trình tạo một mảng **arr** tự động các số chẵn nguyên từ 2020 đến 3838.

Code it!

1. Tạo thêm mảng lưu các số chia hết cho 9 từ mảng **arr**.
2. Các số thu được sẽ được in thành chuỗi trên một dòng, cách nhau bằng 1 tab.



BÀI TẬP

BÀI #5

1. Viết lệnh kiểm tra giá trị **n** nhập vào từ bàn phím thuộc đoạn [-68; 108]. Nếu nhập sai thì bắt nhập lại.

Code it!

2. Từ giá trị **n** được nhập vào, tính biểu thức sau:

$$S(n) = 1 * (1 + 2) * (1 + 2 + 3) * \dots * (1 + 2 + 3 + \dots + n) \text{ với } n > 1$$

PHẦN BÀI TẬP TỔNG HỢP

77

77



BÀI TẬP TỔNG HỢP

BÀI #1:

Từ một chuỗi **str_input** được nhập vào từ bàn phím. Hãy viết script thực hiện các chức năng sau:

- Tính độ dài chuỗi.
- Đếm và in các ký tự đặc biệt: ! @ # \$ % ^ & * () - = + . /
- Đếm và in các ký tự chữ cái từ [a-z]
- Đếm và in các ký tự chữ số từ [0-9]
- Đếm và in các ký tự chữ [A-Z]

Code it!

Font 0, pwrix_database

A	B	C	D	E	F	G	H	I	J
K	L	M	N	O	P	Q	R	S	T
U	V	W	X	Y	Z	a	b	c	d
e	f	g	h	i	j	k	l	m	n
o	p	q	r	s	t	u	v	w	x
y	z	1	2	3	4	5	6	7	8
9	0	-	=	\	_	!	@	#	\$
%	^	&	*	()	~	+		~
[]	{	}	:	;	'	"	.	.
/	<	>	?						

78



BÀI TẬP TỔNG HỢP

BÀI #2:

Viết hàm kiểm tra để phát hiện chuỗi nhập vào có phải là một e-mail hay không?

- Chú ý: Nếu là các e-mail xuất phát từ Gmail, Yahoo, Hotmail, Outlook... (gọi tắt là tập luật tên miền e-mail) thì trước ký tự @ là một chuỗi tối thiểu 6 ký tự, không khoảng trắng và ký tự đặc biệt.
- Khảo sát và tìm thêm các tên miền e-mail để bổ sung vào tập luật giới hạn trên.

Code it!



BÀI TẬP TỔNG HỢP

BÀI #3:

Viết hàm kiểm tra quá trình đăng nhập ID user và password khi login vào hệ thống phải thỏa mãn các tiêu chí sau:

- ID User:
 - Là một chuỗi, có ít nhất 6 ký tự, tối đa 24 ký tự.
 - Không chứa các ký tự: ! @ # \$ % ^ & * () - = +
 - Không khoảng trắng.
- Password:
 - Ít nhất 1 chữ cái nằm trong [a-z]
 - Ít nhất 1 số nằm trong [0-9]
 - Ít nhất 1 ký tự nằm trong [A-Z]
 - Ít nhất 1 ký tự nằm trong [\$ # @]
 - Độ dài mật khẩu tối thiểu 6 ký tự
 - Độ dài mật khẩu tối đa 24 ký tự

Code it!





BÀI TẬP TỔNG HỢP

BÀI #4:

- Viết chương trình mô phỏng trò chơi **Kéo - Búa - Bao** giữa người và máy.

Code it!

Qui ước:

- Kéo > Bao
- Búa > Kéo
- Bao > Búa
- GV hướng dẫn thêm cách chơi nâng cao.



BÀI TẬP TỔNG HỢP

BÀI #5:

- Nâng cấp từ **Bài 4** (nhiều người chơi tự động với nhau)
- Số lượng người được chọn ngẫu nhiên từ 8 đến 20 người.

Code it!



BÀI TẬP TỔNG HỢP NÂNG CAO MINH HỌA VÉ SỐ VIETLOT 6/45

Người mua:

- Chọn 6 số bất kỳ từ 1 đến 45, các số không trùng nhau trong 1 vé. Gọi là 1 dãy số trên 1 vé. Ví dụ 10-18-26-33-39-44
- Có thể mua nhiều vé số tự chọn (n vé). Các vé có thể trùng dãy số với nhau.
- Giá bán mỗi vé 10.000 đ

Máy xổ số:

- Random 6 số bất kỳ từ 1 đến 45, các số không trùng nhau. Gọi là dãy số trúng thưởng.

Dò kết quả: So sánh dãy số người mua và dãy số trúng thưởng:

- Nếu trùng nhau 3 con số thì người chơi nhận: 30.000 đ
- Nếu trùng nhau 4 con số thì người chơi nhận: 300.000 đ
- Nếu trùng nhau 5 con số thì người chơi nhận: 10.000.000 đ
- Nếu trùng nhau 6 con số thì người chơi nhận: 10.000.000.000 đ

Thống kê kết quả người chơi nhận được bao nhiêu tiền khi dò kết quả.

Code it!

Khoa Khoa học và Kỹ thuật Thông tin – Trường ĐH CNTT Tp.HCM

83

83



ĐẠI HỌC QUỐC GIA TP. HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN



CHƯƠNG 5 Phương thức - Method



84



Phương thức

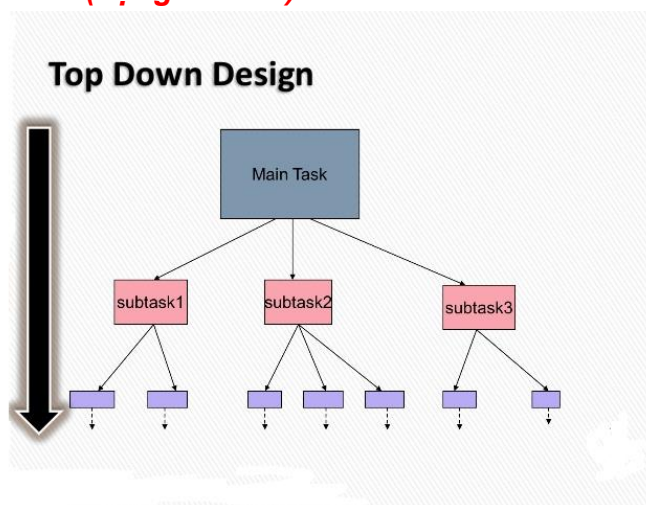
Giới thiệu

- Phương thức còn gọi là hàm.
- Trong OOP, phương thức thể hiện hành vi của một đối tượng.
- Chứa một khối các dòng code có hệ thống và tổ chức.
- Có thể tái sử dụng để thực hiện một công việc có liên quan nhau.
- Hàm dùng để thực hiện, xử lý các công việc đã được **chia nhỏ ra từ công việc lớn**, việc phân chia này giúp người lập trình có khả năng kiểm soát lỗi và dễ phát hiện để sửa lỗi.
- Khuyến nghị khi thiết kế phương thức, nên tuân theo nguyên tắc sau:
Input – Process – Output.
- Viết một lần, sử dụng lại nhiều lần.



Phương thức

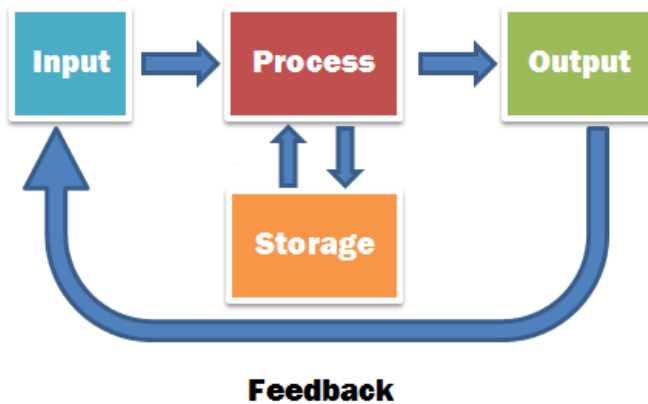
Nguyên tắc thiết kế (dạng cơ bản)





Phương thức

Nguyên tắc thiết kế (dạng cơ bản)



Khoa Khoa học và Kỹ thuật Thông tin – Trường ĐH CNTT Tp.HCM

87

87



Phương thức

Cú pháp chung

```

<quyền riêng tư> <kiểu dữ liệu> tenPhuongThuc(<khai báo input>)
{
    // Lệnh
    // Hoặc khối lệnh
    // return <giá trị>
}
  
```

- <quyền riêng tư>: public, private, protected
- <kiểu dữ liệu>: void, int, float, double, String...

Khoa Khoa học và Kỹ thuật Thông tin – Trường ĐH CNTT Tp.HCM

88

88



Phương thức

Ví dụ

```
public class Main {
    static void myMethod() {
        System.out.println("I just got executed!");
    }

    public static void main(String[] args) {
        myMethod();
        myMethod();
        myMethod();
    }
}
```



Phương thức

Ví dụ: (có Parameters, Arguments)

```
public class Main {
    static void myMethod(String fname, int age) {
        System.out.println(fname + " is " + age);
    }

    public static void main(String[] args) {
        myMethod("Liam", 5);
        myMethod("Jenny", 8);
        myMethod("Anja", 31);
    }
}
```



Phương thức

Ví dụ

```
public class Main {
    static int myMethod(int x) {
        return 5 + x;
    }

    public static void main(String[] args) {
        System.out.println(myMethod(3));
    }
}
```



Phương thức

Ví dụ

```
public class Main {
    static int myMethod(int x, int y) {
        return x + y;
    }

    public static void main(String[] args) {
        int z = myMethod(5, 3);
        System.out.println(z);
    }
}
```



Nạp chồng phương thức - Method Overloading

Giới thiệu nạp chồng phương thức - Method Overloading

- Phương thức có cùng tên, nhưng khác nhau về số lượng tham số truyền vào.
- Kỹ thuật overloading làm tăng sự linh hoạt cho các phương thức.

```
void sum(int x, int y) {

}

void sum(int x, int y, int z) {

}
```



Nạp chồng phương thức - Method Overloading

Ví dụ

```
static int plusMethodInt(int x, int y) {
    return x + y;
}

static double plusMethodDouble(double x, double y) {
    return x + y;
}

public static void main(String[] args) {
    int myNum1 = plusMethodInt(8, 5);
    double myNum2 = plusMethodDouble(4.3, 6.26);

    System.out.println("int: " + myNum1);
    System.out.println("double: " + myNum2);
}
```



Nạp chồng phương thức - Method Overloading

Ví dụ

```
static int plusMethod(int x, int y) {
    return x + y;
}

static double plusMethod(double x, double y) {
    return x + y;
}

public static void main(String[] args) {
    int myNum1 = plusMethod(8, 5);
    double myNum2 = plusMethod(4.3, 6.26);

    System.out.println("int: " + myNum1);
    System.out.println("double: " + myNum2);
}
```



Java Scope/Block

```
public class Main {
    public static void main(String[] args) {
        // Code here CANNOT use x
        { // This is a block
            // Code here CANNOT use x

            int x = 100;
            // Code here CAN use x

            System.out.println(x);
        } // The block ends here
        // Code here CANNOT use x
    }
}
```



BÀI TẬP TỔNG HỢP NÂNG CAO MINH HỌA VÉ SỐ VIETLOT 6/45

Người mua:

- Chọn 6 số bất kỳ từ 1 đến 45, các số không trùng nhau trong 1 vé. Gọi là 1 dãy số trên 1 vé. Ví dụ 10-18-26-33-39-44
- Có thể mua nhiều vé số tự chọn (n vé). Các vé có thể trùng dãy số với nhau.
- Giá bán mỗi vé 10.000 đ

Máy xổ số:

- Random 6 số bất kỳ từ 1 đến 45, các số không trùng nhau. Gọi là dãy số trúng thưởng.

Dò kết quả: So sánh dãy số người mua và dãy số trúng thưởng:

- Nếu trùng nhau 3 con số thì người chơi nhận: 30.000 đ
- Nếu trùng nhau 4 con số thì người chơi nhận: 300.000 đ
- Nếu trùng nhau 5 con số thì người chơi nhận: 10.000.000 đ
- Nếu trùng nhau 6 con số thì người chơi nhận: 10.000.000.000 đ

Thống kê kết quả người chơi nhận được bao nhiêu tiền khi dò kết quả.

Code it!

Khoa Khoa học và Kỹ thuật Thông tin – Trường ĐH CNTT Tp.HCM

97

97



ĐẠI HỌC QUỐC GIA TP. HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN



CHƯƠNG 6 XỬ LÝ NGOẠI LỆ (Exceptions – try ... catch)



99



try ... catch

Cú pháp

```
try {
    // Block of code to try
} catch(Exception e) {
    // Block of code to handle errors
}
```



try ... catch

Xét minh họa

```
public class Main {
    public static void main(String[] args) {
        int[] myNumbers = {1, 2, 3};
        System.out.println(myNumbers[10]); // error!
    }
}
```

```
Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: 10
    at Main.main(Main.java:4)
```



try ... catch

Sử dụng try ... catch để bắt ngoại lệ

```
public class Main {
    public static void main(String[] args) {
        try {
            int[] myNumbers = {1, 2, 3};
            System.out.println(myNumbers[10]);
        } catch (Exception e) {
            System.out.println("Something went wrong.");
        }
    }
}
```



try ... catch ... finally

- Có lỗi hay không có lỗi thì các lệnh trong finally vẫn thực hiện.

```
public class Main {
    public static void main(String[] args) {
        try {
            int[] myNumbers = {1, 2, 3};
            System.out.println(myNumbers[10]);
        } catch (Exception e) {
            System.out.println("Something went wrong.");
        } finally {
            System.out.println("The 'try catch' is finished.");
        }
    }
}
```



throw và throws

Giới thiệu

- throw cho phép tạo ra một trường hợp lỗi
- throws cho phép xác định các trường hợp liên quan đến phương thức

throw	throws
Used to throw an exception for a method	Used to indicate what exception type may be thrown by a method
Cannot throw multiple exceptions	Can declare multiple exceptions
<ul style="list-style-type: none"> •Syntax:throw is followed by an object (new type) •used inside the method 	<ul style="list-style-type: none"> •Syntax:throws is followed by a class and used with the method signature



throw và throws

- Có nhiều loại exception trong java: ArithmeticException, ClassNotFoundException, ArrayIndexOutOfBoundsException, SecurityException...

Cú pháp

```
static void method_name(arguments) throws ArithmeticException {
    throw new ArithmeticException("Thông báo lỗi");
}
```



throw và throws

Ví dụ

```
public class Main {
    static void checkAge(int age) throws ArithmeticException {
        if (age < 18) {
            throw new ArithmeticException("Access denied-You must be at least 18 years old.");
        }
        else {
            System.out.println("Access granted - You are old enough!");
        }
    }

    public static void main(String[] args) {
        checkAge(15); // Set age to 15 (which is below 18...)
    }
}
```

Khoa Khoa học và Kỹ thuật Thông tin – Trường ĐH CNTT Tp.HCM

106

106



ĐẠI HỌC QUỐC GIA TP. HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN



ÔN TẬP MẢNG 1 CHIỀU VÀ 2 CHIỀU



107

107



Mảng - Array

Giới thiệu

- Mảng được sử dụng để lưu trữ nhiều giá trị trong một biến duy nhất, thay vì khai báo nhiều biến riêng biệt cho từng giá trị.
- Dùng chỉ số để lấy giá trị trong mảng.



Mảng

Cú pháp

<kiểu dữ liệu> [] tenMang;

<kiểu dữ liệu> [] tenMang = {<giá trị khởi tạo cách nhau dấu , >};

- <kiểu dữ liệu> có thể là int, String, float, double...

Ví dụ

- String[] cars;
- String[] cars = {"Volvo", "BMW", "Ford", "Mazda"};
- int[] myNum = {10, 20, 30, 40};



Mảng

Độ dài mảng

- Dùng thuộc tính `length` để lấy độ dài mảng.

Ví dụ

```
String[] cars = {"Volvo", "BMW", "Ford", "Mazda"};
System.out.println(cars.length);
// Outputs 4
```



Mảng

Thay đổi giá trị

```
tenMang[<chỉ số>] = <giá trị mới>;
```

Ví dụ

```
String[] cars = {"Volvo", "BMW", "Ford", "Mazda"};
cars[0] = "Opel";
System.out.println(cars[0]);
// Now outputs Opel instead of Volvo
```



Mảng

Xuất toàn bộ giá trị

- Dùng **for**

```
String[] cars = {"Volvo", "BMW", "Ford", "Mazda"};
for (int i = 0; i < cars.length; i++) {
    System.out.println(cars[i]);
}
```



Mảng

Xuất toàn bộ giá trị

- Dùng **For-Each**
- Cú pháp

```
for (type variable : arrayname) {
    ...
}
```

```
String[] cars = {"Volvo", "BMW", "Ford", "Mazda"};
for (String i : cars) {
    System.out.println(i);
}
```



Mảng 2 chiều

Giới thiệu

	Cột 0	Cột 1	Cột 2	Cột 3
Dòng 0	M[0][0] 7	M[0][1] 2	M[0][2] 9	M[0][3] 0
Dòng 1	M[1][0] 9	M[1][1] 5	M[1][2] 4	M[1][3] 1
Dòng 2	M[2][0] 8	M[2][1] 0	M[2][2] 3	M[2][3] 6

Tên mảng ————— Chỉ số cột
 ————— Chỉ số dòng



Mảng 2 chiều

Khai báo

<kiểu dữ liệu> [][] tenMang;

- <kiểu dữ liệu> có thể là int, String, float, double...

Ví dụ

```
int[][] myNumbers = { {1, 2, 3, 4}, {5, 6, 7} };
int x = myNumbers[1][2];
System.out.println(x); // Outputs 7
```




Mảng 2 chiều

Xuất mảng 2 chiều

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

        int[][] myNumbers = { {1, 2, 3, 4}, {5, 6, 7} };

        for (int i = 0; i < myNumbers.length; ++i) {
            for(int j = 0; j < myNumbers[i].length; ++j) {
                System.out.println(myNumbers[i][j]);
            }
        }
    }
}
```

Khoa Khoa học và Kỹ thuật Thông tin – Trường ĐH CNTT Tp.HCM

116

116



ĐẠI HỌC QUỐC GIA TP. HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN



CHƯƠNG 7 Object-Oriented Programming



117

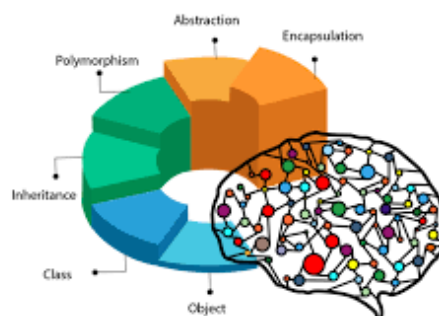


Giới thiệu OOP

Object-Oriented Programming (OOP)

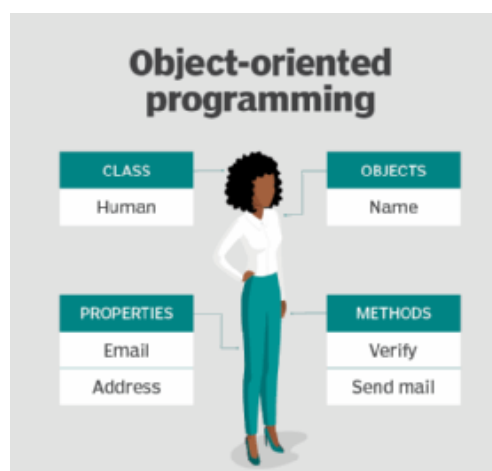
- Ý tưởng OOP?
- Là một phương pháp phân tích bài toán rất tiên tiến.
- Bài toán sẽ được phân tích thành các đối tượng mang đặc điểm và hành vi.
- OOP là phương pháp tư duy gần với thực tế cuộc sống của con người, xây dựng bài toán dựa vào mối quan hệ giữa các đối tượng tham gia vào bài toán.

OOPs (Object-Oriented Programming System)



Các khái niệm


- Đối tượng - Objects
- Lớp - Class
- Thuộc tính - Properties
- Phương thức - Methods





Đối tượng

- Đối tượng (object) là một thực thể tham gia vào bài toán, có các đặc điểm và hành vi.
- Ví dụ: nhân viên, sinh viên, xe...

Object	Properties	Methods
	car.name = Fiat	car.start()
	car.model = 500	car.drive()
	car.weight = 850kg	car.brake()
	car.color = white	car.stop()

Khoa Khoa học và Kỹ thuật Thông tin – Trường ĐH CNTT Tp.HCM

120

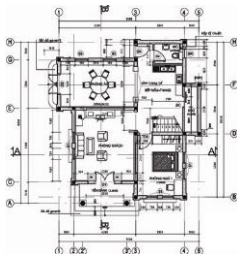
120



Lớp - Class

Giới thiệu

- Class là một bản thiết kế được dùng để khởi tạo đối tượng.



Khoa Khoa học và Kỹ thuật Thông tin – Trường ĐH CNTT Tp.HCM

121

121



Lớp - Class

Cú pháp

```
class <TenLop> {
    //Khai báo thuộc tính

    //Constructor

    //Khai báo phương thức

}
```

Khuyến nghị:

- Viết hoa các chữ cái đầu của từ.
- Dùng các chữ cái a-z, A-Z, 0-9 để đặt tên đối tượng.
- KHÔNG đặt số ở đầu



Lớp - Class

Ví dụ

```
class SinhVien {
    //Khai báo thuộc tính

    //Constructor

    //Khai báo phương thức

}
```



Từ khóa khai báo quyền riêng tư

- Khi khai báo/xây dựng một class có thể dùng các từ khóa để thể hiện **quyền riêng tư*** của class.
- <quyền riêng tư>:
 - public
 - private
 - protected
- Ngoài ra, **quyền riêng tư** còn được sử dụng để thể hiện mức độ riêng tư trong khai báo thuộc tính, constructor, phương thức...

```
<quyền riêng tư> class <TenLop> {
    //Khai báo thuộc tính

    //Constructor

    //Khai báo phương thức

}
```

* Chú ý: phần quyền riêng tư sẽ được giải thích, giảng kỹ trong bài "Tính đóng gói và che giấu dữ liệu".



Thuộc tính - Attribute

Giới thiệu

- Thuộc tính (attribute) là các đặc điểm của một đối tượng cụ thể.
- Có một giá trị cụ thể và có miền giá trị đã được xác định.

Ví dụ về một số thuộc tính

- Hệ số lượng là một đặc điểm của một nhân viên, chúng có miền giá trị xác định, là số thực có giá trị như 2.34, 2.69...
- Điểm là một đặc điểm của một sinh viên, chúng có miền giá trị xác định, là số thực có giá trị như 7.75, 8, 9.5...
- Khối lượng xe là một đặc điểm của một xe, chúng có miền giá trị xác định, là số thực có giá trị như 879 kg, 950 kg...



Thuộc tính - Attribute

Cú pháp khai báo thuộc tính

```
<Kiểu dữ liệu> tenThuocTinh;  
<Kiểu dữ liệu> tenThuocTinh = <giá trị>;
```

Nếu thuộc tính là hằng số:

```
final <Kiểu dữ liệu> tenThuocTinh = <giá trị>;
```

Khuyến nghị

- Không viết hoa chữ cái đầu tiên của từ.
- Dùng các chữ cái a-z, A-Z, 0-9 để đặt tên.
- KHÔNG đặt số ở đầu.
- Đối với thuộc tính hằng thì dùng A-Z, 0-9 và “_”



Thuộc tính - Attribute

Ví dụ

```
class SinhVien {  
    //Khai báo thuộc tính  
    final int ID_MAHOA = 521342;  
    int maSV;  
    string hoTen;  
  
    //Constructor  
  
    //Khai báo phương thức  
  
}
```



Phương thức khởi tạo - Constructor

Constructor là gì?

- Constructor là một phương thức đặc biệt dùng để khởi tạo một đối tượng.
- Phương thức khởi tạo này có thể được sử dụng để khởi tạo thuộc tính.
- Không bắt buộc phải có phương thức khởi tạo trong class.

Cú pháp

- Constructor phải có cùng tên với class
- Không có kiểu dữ liệu trả về

```
class TinhToan {
    //Khai báo Thuộc tính
    double a, b;

    //Khai báo constructor
    public TinhToan(double a, double b) {
        this.a = a;
        this.b = b;
    }

    //Khai báo các phương thức
}
```



Phương thức khởi tạo - Constructor

Ví dụ

```
class TinhToan {
    //Khai báo Thuộc tính
    double a, b;

    //Khai báo constructor
    public TinhToan(double a, double b) {
        this.a = a;
        this.b = b;
    }

    //Khai báo các phương thức
}
```



Phương thức khởi tạo - Constructor

Nạp chồng khởi tạo – Constructor overloading

- Constructor overloading là một kỹ thuật cho phép trong một class có nhiều constructor.
- Các constructor khác nhau về số lượng đối số truyền vào.



Phương thức khởi tạo - Constructor

Ví dụ nạp chồng khởi tạo

```
class TinhToanSo {
    //Khai báo Thuộc tính
    double a, b, c;

    //Khai báo constructor
    TinhToanSo() {
    }
    TinhToanSo(double a, double b) {
        this.a = a;
        this.b = b;
    }
    TinhToanSo(double a, double b, double c) {
        this.a = a;
        this.b = b;
        this.c = c;
    }
    //Khai báo các phương thức
}
```




Phương thức - Method

Giới thiệu

- Phương thức là một hành vi của đối tượng (thể hiện hành động).
- Đối với mỗi class/đối tượng, ngoài các phương thức đã xây dựng để giải quyết vấn đề thì luôn có 2 phương thức:
 - Nhập dữ liệu
 - Xuất, in ra dữ liệu
- Nhập dữ liệu là phương thức nhập các giá trị đầu vào cho các thuộc tính
- Xuất, in ra dữ liệu là phương thức in ra các thông tin của các thuộc tính

Ví dụ về phương thức:

- Phương thức `tinhDTB()` thực hiện tính điểm trung bình cho các môn học.



Phương thức - Method

Cú pháp khai báo phương thức

```
<kiểu dữ liệu> tenPhuongThuc(<các khai báo input>){
    //khối lệnh
}
```

Khuyến nghị

- Không viết hoa chữ cái đầu tiên của từ.
- Dùng các chữ cái a-z, A-Z, 0-9 để đặt tên.
- KHÔNG đặt số ở đầu.



Phương thức - Method

Ví dụ

```
class SinhVien {
    //Khai báo thuộc tính
    final int idMaHoa = 521342;
    int maSV;
    string hoTen;

    //Khai báo phương thức
    void xuatSV() {

    }
}
```



Khởi tạo đối tượng

Cú pháp

ClassName <tên đối tượng> = new ClassName();



Ví dụ

```
SinhVien sv = new SinhVien();
sv.nhapSV();
sv.xuatSV();
```



BÀI TẬP ÁP DỤNG

Code it!

1. Giải phương trình bậc nhất $aX + b = 0$
2. Giải phương trình bậc 2 $aX^2 + bX + c = 0$
3. Giải phương trình trùng phương $aX^4 + bX^2 + c = 0$
4. Chương trình kiểm tra số nguyên tố
5. In ra dãy số nguyên tố bé hơn N

Chú ý:

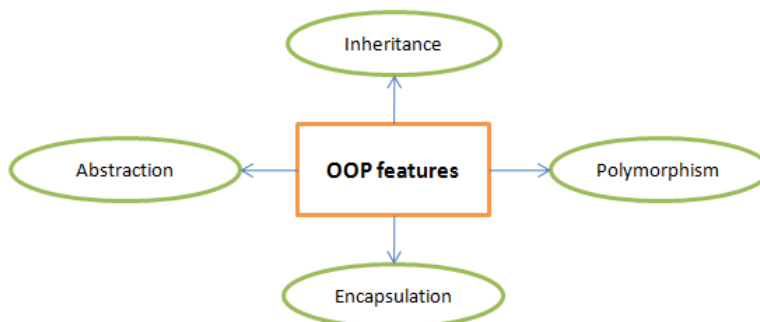
- Tất cả các bài được thiết kế theo phương pháp hướng đối tượng.



Đặc tính của OOP

Giới thiệu

- Các tính chất quan trọng ==> điểm mạnh của OOP





Tính kế thừa

Giới thiệu

- Kế thừa là sự thừa hưởng lại các thuộc tính, phương thức mà không cần khai báo lại các thuộc tính, phương thức.
- Kế thừa giúp tái sử dụng lại mã nguồn giúp lập trình viên dễ dàng rà soát bảo trì mã nguồn.
- Dùng từ khóa **extends** để thể hiện kế thừa.
- Chú ý: Java không có đa kế thừa.

```
class A{
    ...
}

class B extends A{
    ...
}
```

Khoa Khoa học và Kỹ thuật Thông tin – Trường ĐH CNTT Tp.HCM

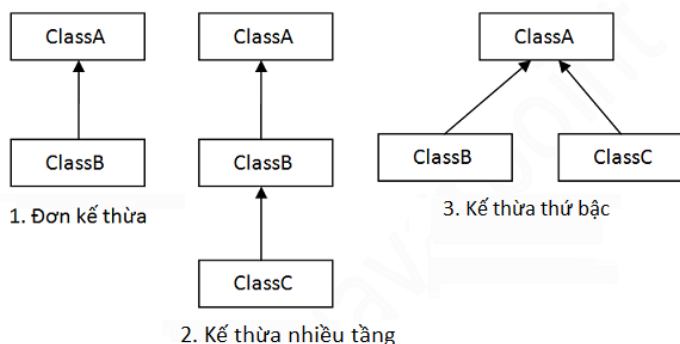
138

138



Tính kế thừa

Các kiểu kế thừa



Chú ý:

- Đa kế thừa trong java không được hỗ trợ qua thiết kế class.
- Muốn đa kế thừa thì dùng dùng interface.

Khoa Khoa học và Kỹ thuật Thông tin – Trường ĐH CNTT Tp.HCM

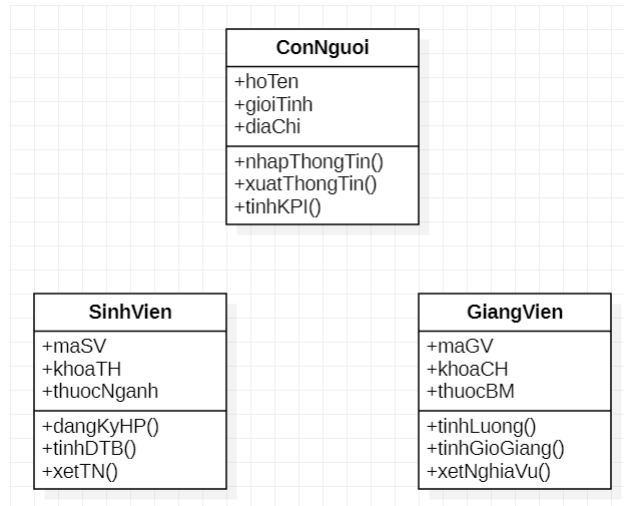
139

139



Tính kế thừa

Minh họa



Khoa Khoa học và Kỹ thuật Thông tin – Trường ĐH CNTT Tp.HCM

140

140



Tính kế thừa

Nạp chồng phương thức?

- Phương thức có cùng tên, nhưng khác nhau về số lượng tham số/đối số truyền vào.
- Kỹ thuật overloading làm tăng sự linh hoạt cho các phương thức.

```

void sum(int x, int y) {

}

void sum(int x, int y, int z) {

}
  
```

Khoa Khoa học và Kỹ thuật Thông tin – Trường ĐH CNTT Tp.HCM

141

141



Tính kế thừa

Vấn đề **nạp chồng** phương thức trong class con

1. Một phương thức a() được xây dựng trong class cha (class A).
 2. Tiếp theo, class B kế thừa class A.
 3. Và trong class B, xây dựng thêm phương thức a() cùng tên.
 4. Trong phương thức a() tại class B, gọi lại phương thức a() của class A.
- => Hai phương thức trùng tên này gọi là quá trình nạp chồng phương thức

```
public class A {
    void a(int x) {
    }
}
```

```
public class B extends A {
    void a(int x, int y) {
        //Các lệnh xử lý
        a(x);
    }
}
```

Khoa Khoa học và Kỹ thuật Thông tin – Trường ĐH CNTT Tp.HCM

142

142



Tính kế thừa

Vấn đề **ghi đè** phương thức trong class con

```
public class A {
    void a() {
        System.out.println("Phương thức a()");
    }
}
```

```
public class B extends A {
    void a() {
    }
}
```

Vì phạm nạp chồng
phương thức

```
public class B extends A {
    @Override
    void a() {
    }
}
```

```
public class B extends A {
    void a() {
    }
}
```

Add @Override Annotation

Khoa Khoa học và Kỹ thuật Thông tin – Trường ĐH CNTT Tp.HCM

145

145



Tính kế thừa

Vấn đề ghi đè phương thức trong class con

• Ghi đè hoàn toàn:

- Là việc xóa bỏ và xây dựng lại toàn bộ nội dung của phương thức trong class cha.
- Khi nào phải ghi đè hoàn toàn?

• Ghi đè không hoàn toàn:

- Là kế thừa lại một phần phương thức trong class cha và xây dựng thêm nội dung cho phương thức con.
- Sử dụng từ khóa `super.tenPhuongThuc()` khi ghi đè không hoàn toàn để gọi đến phương thức trong class cha



Tính kế thừa

Minh họa ghi đè phương thức

```
public class A {
    void a() {
        System.out.println("Phương thức a()");
    }
}

public class B extends A {
    @Override
    void a() {
        System.out.println("Định nghĩa lại a()");
    }
}
```



Tính kế thừa

Minh họa ghi đè phương thức

```
public class A {
    void a() {
        System.out.println("Phương thức a()");
    }
}

public class C extends A {
    @Override
    void a() {
        //Gọi đến phương thức của class cha bằng từ khóa super
        super.a();

        //Sau đó, định nghĩa thêm nội dung cho a() trong class C
    }
}
```

Khoa Khoa học và Kỹ thuật Thông tin – Trường ĐH CNTT Tp.HCM

148

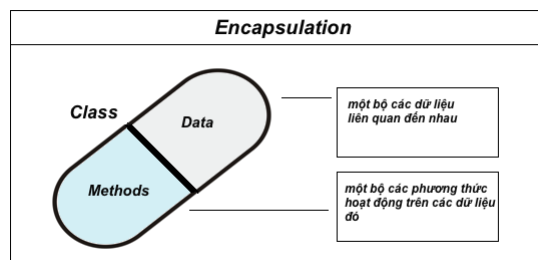
148



Tính đóng gói và che giấu dữ liệu

Tính đóng gói

- Là việc phân chia các đối tượng có cùng chung đặc điểm về cùng một nhóm để dễ quản lý.
- Tính đóng gói giúp chương trình trở nên mạch lạc, rõ ràng. Vì vậy, khả năng bảo trì, sửa chữa nâng nắp dễ dàng hơn.



Khoa Khoa học và Kỹ thuật Thông tin – Trường ĐH CNTT Tp.HCM

150

150



Tính đóng gói và che giấu dữ liệu

Cách xây dựng, đặt tên package

com.<tên tổ chức>.<tên project>

Giải thích:

- com: viết tắt của company
- Chỉ dùng a-z, 0-9 để đặt tên các package
- <tên tổ chức>: có thể dùng tên tổ chức hoặc cá nhân như: viettel, uit, nhihy

Ví dụ: com.intel.vinqa, com.nhihy.kbb

- Khi xây dựng xong các package thì tiến hành gom nhóm các class có cùng đặc điểm vào cùng một package
- Sử dụng cú pháp package <tên package> ở đầu mỗi class



Tính đóng gói và che giấu dữ liệu

Minh họa

- *Hướng dẫn trên lớp.*



Tính đóng gói và che giấu dữ liệu

Tính che giấu dữ liệu

- Che giấu dữ liệu là việc quy định **phạm vi** và **quyền hạn** chia sẻ thuộc tính và phương thức.
- Ví dụ một đối tượng A được quy định quyền riêng tư về thuộc tính và phương thức thì các đối tượng khác sẽ có mức độ truy cập vào đối tượng A là khác nhau.

Lợi ích của tính che giấu dữ liệu

- **Giới hạn** tính toàn vẹn dữ liệu, **hạn chế** việc tùy ý truy cập mọi thuộc tính và phương thức.
- Làm nổi bật vai trò của đối tượng này so với đối tượng khác trong quan hệ sau: cùng một class, package, giữ các package, mối quan hệ kế thừa.



Tính đóng gói và che giấu dữ liệu

Cú pháp khai báo

- Khai báo quyền riêng tư cho class, thuộc tính, phương thức
 - <quyền riêng tư>** class TenClass {.....}
 - <quyền riêng tư>** **<kiểu dữ liệu>** tenThuocTinh;
 - <quyền riêng tư>** **<kiểu dữ liệu>** tenPhuongThuc() {}
- **<quyền riêng tư>** gồm các từ khóa **private**, **protected**, **public...**



Tính đóng gói và che giấu dữ liệu

Quyền riêng tư và phạm vi truy cập

	Internal class	Trong cùng package	Khác package	Kế thừa
private	<input checked="" type="checkbox"/>			
default (để trống)	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>		
protected	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>		<input checked="" type="checkbox"/>
public	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>

Bảng này rất quan trọng.



Tính đóng gói và che giấu dữ liệu

Quyền riêng tư và phạm vi truy cập

Ví dụ minh họa

- Xem demo Chuong07



Tính đóng gói và che giấu dữ liệu

Quy tắc khai báo quyền riêng tư cho thuộc tính, phương thức

- **Đối với thuộc tính** (chỉ được khai báo 2 quyền):
 - **private**: nếu thuộc tính chỉ dùng riêng trong một class/đối tượng
 - **protected**: nếu thuộc tính đó được các đối tượng con kế thừa
- **Đối với phương thức** (tùy thuộc vào phạm vi mà có thể khai báo quyền):
 - **private**: nếu các phương thức chỉ sử dụng trong nội bộ
 - **default (để trống, không cần khai báo)**: nếu các phương thức chỉ sử dụng trong cùng một package
 - **protected**: nếu các phương thức được sử dụng trong đối tượng con
 - **public**: nếu các phương thức được sử dụng ở mọi nơi, kể cả việc khác package



Tính đóng gói và che giấu dữ liệu

Quy tắc khai báo quyền riêng tư cho thuộc tính, phương thức

- **Đối với thuộc tính là hằng số** :
 - Có thể khai báo quyền **public static final float** khi cần thiết để các class khác có thể truy cập đến.
 - Ví dụ:


```
public static final float PI = 3.14F;
```



Tính đóng gói và che giấu dữ liệu

Tại sao nên khai báo quyền riêng tư cho thuộc tính ở mức private hoặc protected?

- **Bảo vệ toàn vẹn dữ liệu** của một đối tượng, ngăn không cho đối tượng khác có thể truy cập để thay đổi dữ liệu.
- Nếu đối tượng B truy cập đến thuộc tính của đối tượng A sẽ có 2 trường hợp:
 - Trường hợp 1: Lấy thuộc tính của đối tượng A
 - Trường hợp 2: Sửa thuộc tính của đối tượng A



Tính đóng gói và che giấu dữ liệu

Trường hợp 1: Lấy thuộc tính của đối tượng A

- Trong class A khai báo quyền riêng tư thuộc tính là private thì tại class B sẽ không truy cập được thuộc tính.
- Trong trường hợp này, muốn lấy thông tin thuộc tính trong A thì phải **xây dựng thêm phương thức get()** để trả về giá trị thuộc tính.
- Cú pháp:

```
public <kiểu dữ liệu> getTenThuocTinh() {
    return tenThuocTinh;
}
```



Tính đóng gói và che giấu dữ liệu

Ví dụ

```
public class A {
    private int x;

    public A() {
    }

    public A(int x) {
        this.x = x;
    }

    public int getX() {
        return x;
    }
}
```



Tính đóng gói và che giấu dữ liệu

Trường hợp 2: Sửa thuộc tính của đối tượng A

- Trong class C khai báo quyền riêng tư thuộc tính là **private** thì tại class B sẽ không truy cập được thuộc tính.
- Trong trường hợp này, muốn sửa thông tin thuộc tính trong C thì phải xây dựng thêm phương thức set() để **gán** giá trị thuộc tính.
- Cú pháp:

```
public void setTenThuocTinh(<kiểu dl> gtThuocTinh) {
    this.tenThuocTinh = gtThuocTinh;
}
```

- Chú ý: việc thiết kế phương thức set không được tùy tiện.



Tính đóng gói và che giấu dữ liệu

Ví dụ

```
public class C {
    private int x;

    public C() {
    }

    public C(int x) {
        this.x = x;
    }

    public void setX(int x) {
        this.x = x;
    }
}
```

Khoa Khoa học và Kỹ thuật Thông tin – Trường ĐH CNTT Tp.HCM

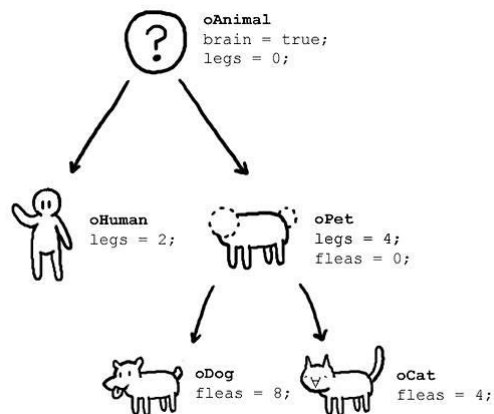
163

163



Tính trừu tượng

Ý tưởng?



Hình (Internet)

Khoa Khoa học và Kỹ thuật Thông tin – Trường ĐH CNTT Tp.HCM

164

164



Tính trừu tượng

Tính trừu tượng?

- Là tính chất không thể hiện cụ thể mà chỉ nêu tên vấn đề.
- Cho phép phớt lờ, không chú ý đến các đặc điểm của đối tượng. Thay vào đó, chỉ tập trung vào các đặc điểm cốt lõi của đối tượng mà bài toán đề cập.

Ưu điểm

- Tập trung vào những thông tin cốt lõi của đối tượng, thay vì quan tâm đến cách nó thực hiện.
- Tính trừu tượng cung cấp nhiều chức năng mở rộng thêm nếu kết hợp với **tính đa hình và kế thừa**.



Tính trừu tượng

Phương pháp

- Tính trừu tượng được xây dựng thông qua lớp trừu tượng (Abstract class) và các giao diện (Interface)
- Lớp trừu tượng không dùng để tạo ra các đối tượng như những lớp khác.
- Lớp trừu tượng đóng vai trò như một cái sườn để tạo ra các lớp con kế thừa từ cái sườn này.



Tính trừu tượng

Lớp trừu tượng (Abstract class)

- Từ khóa **abstract** đứng trước tên của **class**
- Class trừu tượng không dùng trực tiếp để tạo ra đối tượng mà phải viết 1 class **kế thừa** từ class trừu tượng đó
- Class trừu tượng **không** cần có **phương thức khởi tạo**.
- Trong class trừu tượng có hoặc không có **phương thức trừu tượng**. Một khi có phương thức trừu tượng thì class phải khai báo trừu tượng.
- Khi một class kế thừa từ một class trừu tượng thì các phương thức phải override lại.



Tính trừu tượng

Cú pháp

```
<quyền riêng tư> abstract class TenClass {
    //Khai báo thuộc tính

    //Khai báo phương thức (đa phần PT trừu tượng)
}
```



Tính trừu tượng

Phương thức trừu tượng (Abstract method)

- Sử dụng từ khóa **abstract** khi khai báo tên phương thức.
- Chỉ cần khai báo phương thức và dấu chấm phẩy “;” để kết thúc, không cần định nghĩa thân phương thức và cặp {}.

Cú pháp khai báo phương thức trừu tượng

<quyền riêng tư> abstract <kiểu trả về> TenPT(<tham số>;

Chú ý:

- <quyền riêng tư> không **private**
- Class con phải khai báo **override** phương thức trừu tượng này.



Tính trừu tượng

Đặt bài toán

Anh/chị tính **chu vi** và **diện tích** của hình vuông, chữ nhật, bình hành... từ **các điểm tọa độ**.



Tính trừu tượng

Ví dụ

```
public class Dinh {
    protected int x, y;

    public Dinh(int x, int y) {
        this.x = x;
        this.y = y;
    }

    public void xuatDinh() {
        System.out.printf("(%d,%d) \n", x, y);
    }
}

public abstract class TuGiac {
    protected Dinh a, b, c, d;
    protected double chuVi, dienTich;
    protected abstract void tinhChuVi();
    protected abstract void tinhDienTich();
    protected void ketQua() {...8 lines }
}
```



Tính trừu tượng

```
Ví dụ public class HinhVuong extends TuGiac{
    private double canhHV;

    public HinhVuong(Dinh a, Dinh b, Dinh c, Dinh d) {...6 lines }

    public void tinhCanh() {...3 lines }

    @Override
    protected void tinhChuVi() {...3 lines }

    @Override
    protected void tinhDienTich() {...3 lines }

    @Override
    protected void ketQua() {...3 lines }
}
```



Tính trừu tượng

Ví dụ

```
public class Main {
    public static void main(String[] args) {
        Dinh a = new Dinh(0, 0);
        Dinh b = new Dinh(5, 0);
        Dinh c = new Dinh(0, 5);
        Dinh d = new Dinh(5, 5);

        HinhVuong hv = new HinhVuong(a, b, c, d);

        hv.tinhCanh();
        hv.tinhChuVi();
        hv.ketQua();
    }
}
```



Tính đa hình

Giới thiệu

- Là việc gọi đến các phương thức được ghi đè từ class cha trong các class con
- Cùng là một phương thức được định nghĩa hoặc được xây dựng trong class cha thì các class con khi kế thừa cha sẽ có thể ghi đè lại các phương thức đó. Suy ra, các phương thức đó, khi các class con thực thi sẽ theo cách khác nhau. Quá trình này được gọi là đa hình.

Ý nghĩa của tính đa hình

- Quản lý các class có **mối quan hệ kế thừa**.
- Kiểm soát các phương thức dùng chung của các class giống hoặc khác nhau về **hành vi bản chất**.



Tính đa hình

Ví dụ

Class	Phương thức
TuGiac	//Phương thức trừu tượng: protected abstract void tinhCV();
HinhVuong	@Override public void tinhCV(){ chuVi = canh * 4; }
HinhChuNhat	@Override public void tinhCV(){ chuVi = (dai + rong) * 2; }

Nhận xét:

- Cùng là một phương thức tên **tinhCV** nhưng chúng lại thực thi khác nhau.

Khoa Khoa học và Kỹ thuật Thông tin – Trường ĐH CNTT Tp.HCM

181

181

```
public abstract class TuGiac {
    //Định nghĩa phương thức trừu tượng tính chu vi
    protected abstract void tinhCV();
}
```

```
public class HinhChuNhat extends TuGiac{
    private double chieuRong;
    private double chieuDai;

    HinhChuNhat(double rong, double dai){
        this.chieuRong = rong;
        this.chieuDai = dai;
    }

    @Override
    public void tinhCV(){
        double chuVi = (chieuDai + chieuRong) * 2;
        System.out.println(chuVi);
    }
}
```

```
public class HinhVuong extends TuGiac{
    private double doDaiCanh;

    HinhVuong(double canh){
        doDaiCanh = canh;
    }

    @Override
    public void tinhCV(){
        double chuVi = doDaiCanh * 4;
        System.out.println(chuVi);
    }
}
```

}

183

183



Tính đa hình

```

public class Main {
    public static void main(String[] args) {
        //Tách riêng từng loại
        HìnhChuNhat hcn = new HìnhChuNhat(3, 4);
        HìnhVuong hv = new HìnhVuong(5);

        hcn.tinhCV();
        hv.tinhCV();

        //Quy về một loại
        TuGiac tg1 = new HìnhChuNhat(3, 4);
        TuGiac tg2 = new HìnhVuong(5);

        tg1.tinhCV();
        tg2.tinhCV();
    }
}

```

Khoa Khoa học và Kỹ thuật Thông tin – Trường ĐH CNTT Tp.HCM

184

184



ĐẠI HỌC QUỐC GIA TP. HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN



CHƯƠNG 8 Java Swing



186

186



Giới thiệu

Lập trình giao diện? GUI (Graphical User Interface) programming

- Sử dụng các đối tượng đã được xây dựng sẵn để thiết kế thành các giao diện trực quan.

Các thư viện hỗ trợ

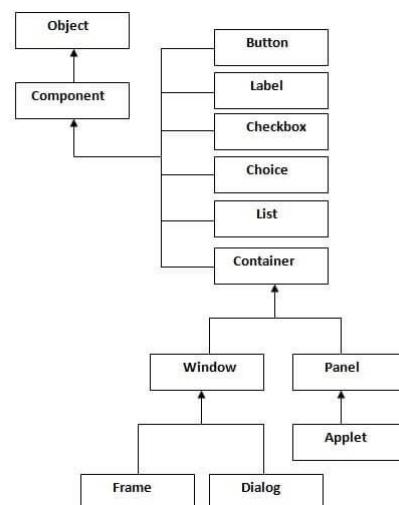
- Java có 2 bộ thư viện dùng để lập trình giao diện là AWT và SWING
- Swing được phát triển sau Awt, được xem là phiên bản tối ưu cho lập trình giao diện
- Swing loại bỏ những thứ cồng kềnh của thư viện Awt giúp tạo ra giao diện đẹp và tốt hơn



AWT

AWT – Abstract Window Toolkit

- Là bộ thư viện được Java xây dựng từ phiên bản JDK 1.0 để hỗ trợ thiết kế giao diện người dùng.
- Cồng kềnh, giao diện trên một số hệ điều hành.
- Không nên chọn AWT để lập trình giao diện
- Kiến trúc của một đối tượng AWT ==>

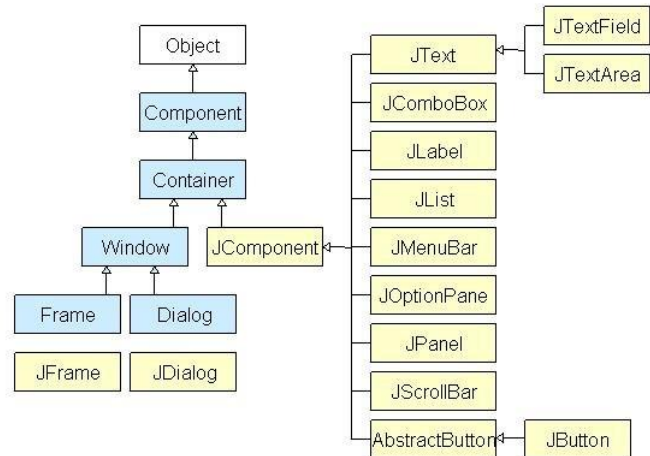




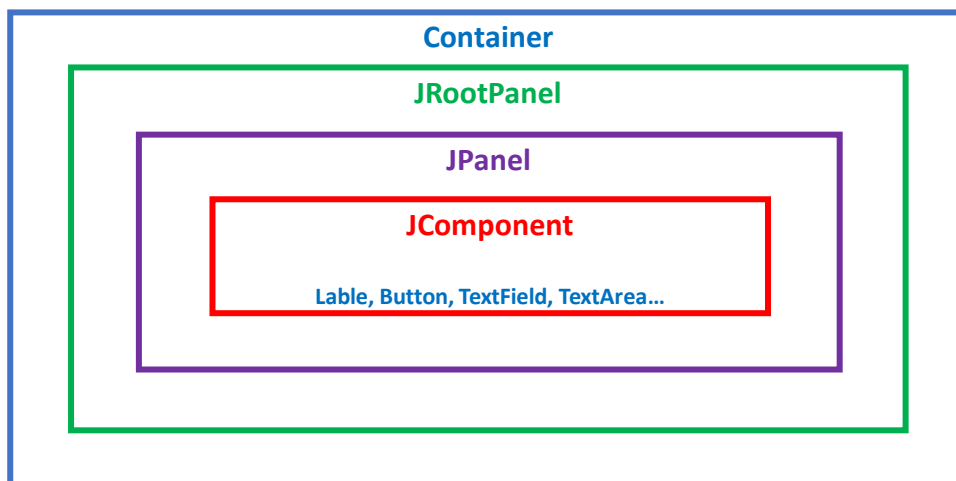
SWING

Giới thiệu Swing

- Là bộ thư viện hỗ trợ lập trình giao diện và đồ họa được phát triển dựa trên AWT, được tích hợp vào phiên bản JDK 1.1.2
- Tối ưu và thuận tiện cho việc thiết kế giao diện, phù hợp trên nhiều hệ điều hành.



Kiến trúc giao diện người dùng





Kiến trúc giao diện người dùng

Giải thích

- **Jcomponent**: chứa các đối tượng cụ thể JButton, JLabel, JTextField...
- **Jpanel**: tầng làm nền để chứa các đối tượng cụ thể, **phân chia** và **quản lý** các **giao diện** người dùng
- **JRootPanel**: được xem là 1 khung chứa chung cho tất cả các đối tượng Jpanel, kiểm soát và cầu nối chia sẻ dữ liệu giữa các Panel khác nhau.
- **Container**: đối tượng bao bọc ngoài cùng, là một khung chưa có chức năng hiển thị giao diện người dùng lên trên màn hình trong các ngữ cảnh khác nhau như: màn hình chức năng, màn hình thông báo, màn hình web.



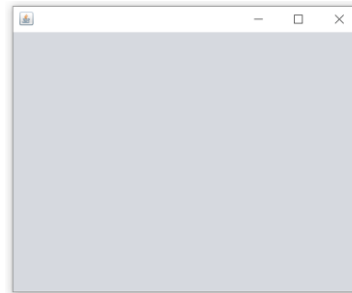
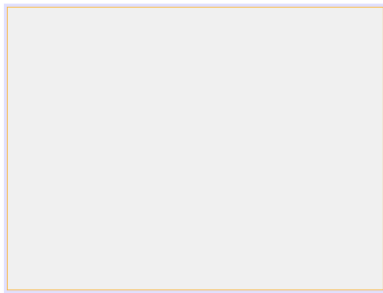
Ví dụ



JFrame

Định nghĩa

- Là một đối tượng kế thừa từ Container giúp hiển thị giao diện người dùng trong một ngữ cảnh nào đó.
- Có thể gọi là một màn hình chức năng cho phép người dùng tương tác.



JFrame

Cách xây dựng

```
public class GUI extends JFrame{
    public int cao = 300;
    public int rong = 500;

    public GUI(String title){
        initGUI(title);
    }

    private void initGUI(String title) {
        setTitle(title);
        setSize(rong, cao);
        setResizable(false);
    }
}

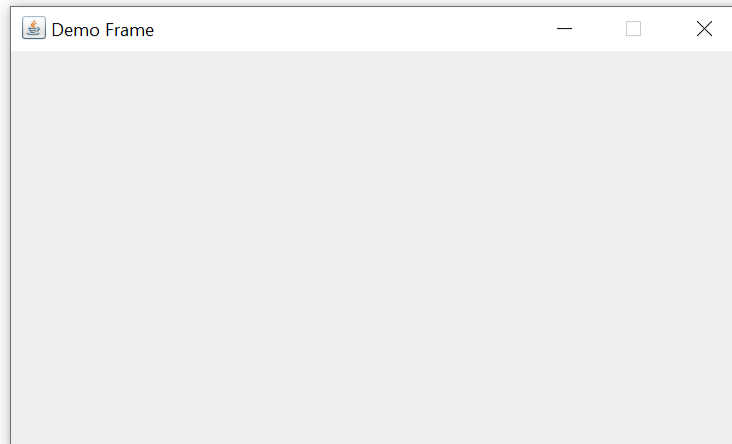
public class Main {
    public static void main(String[] args) {
        GUI gui = new GUI("Demo Frame");
        gui.setVisible(true);

        //new GUI("Demo Frame").setVisible(true);
    }
}
```



JFrame

Kết quả



JFrame

Xử lý sự kiện cho JFrame

- Là việc lắng nghe và bắt thao tác của người dùng khi tương tác với JFrame.

Các loại sự kiện:

- **WindowListener**: lắng nghe các sự kiện khi trạng thái frame bị thay đổi như: bị đóng lại, mở ra, thu nhỏ lại...
- **MouseListener**: lắng nghe các sự kiện khi người dùng sử dụng chuột tương tác như: click, press, move, di chuyển chuột vào frame, di chuyển chuột ra khỏi frame
- **KeyListener**: lắng nghe các sự kiện khi người dùng nhấn các phím để tương tác như: lên, xuống, trái, phải



JFrame

Phương pháp 1

- Khởi tạo các đối tượng từ các class:
 - **WindowListener** **wL** = **new WindowListener ();**
 - **MouseListener** **mL** = **new MouseListener ();**
 - **KeyListener** **kL** = **new KeyListener ();**
- Override (hết) các phương thức có sẵn để triển khai chi tiết.
- Gọi phương thức thêm sự kiện cho đối tượng vừa tạo:


```
addWindowListener(wL);
addMouseListener(mL);
addKeyListener(kL);
```



JFrame

Minh họa

```
public class GUI extends JFrame{
    public int cao = 300;
    public int rong = 500;

    public GUI(String title) {
        initGUI(title);
        addEvents();
    }

    private void initGUI(String title) {...5 lines }
    private void addEvents() {
        WindowListener wL = new WindowListener() {
            @Override
            public void windowOpened(WindowEvent e) {...3 lines }
            @Override
            public void windowClosing(WindowEvent e) {...3 lines }
            @Override
            public void windowClosed(WindowEvent e) {...3 lines }
            @Override
            public void windowIconified(WindowEvent e) {...3 lines }
            @Override
            public void windowDeiconified(WindowEvent e) {...3 lines }
            @Override
            public void windowActivated(WindowEvent e) {...3 lines }
            @Override
            public void windowDeactivated(WindowEvent e) {...3 lines }
        };
        addWindowListener(wL);
    }
}
```



JFrame

Phương pháp 2

- Khởi tạo các đối tượng từ các class trừu tượng, cho phép sử dụng một vài sự kiện, chứ không lấy tất cả:
 - **WindowListener** **wL** = new **WindowAdapter()**;
 - **MouseListener** **mL** = new **MouseAdapter()**;
 - **KeyListener** **kL** = new **KeyAdapter()**;
- Override các phương thức có sẵn cần xây dựng sự kiện.
- Gọi phương thức thêm sự kiện cho đối tượng vừa tạo:


```
addWindowListener(wL);
addMouseListener(mL);
addKeyListener(kL);
```



JFrame

Minh họa

```
public class GUI extends JFrame{
    public int cao = 300;
    public int rong = 500;

    public GUI(String title) {
        initGUI(title);
        addEvents();
    }
    private void initGUI(String title) {...5 lines }
    private void addEvents() {
        WindowListener wL = new WindowAdapter() {
            @Override
            public void windowClosing(WindowEvent e) {
                super.windowClosing(e);
                System.out.println("Bạn đã đóng frame");
            }
            //Ghi đè thêm các phương thức khác
            //nếu muốn xây dựng thêm sự kiện
        };
        addWindowListener(wL);
    }
}
```



JFrame

WindowListener

Sự kiện	Giải thích
windowOpened(WindowEvent e)	Lắng nghe sự kiện khi frame hiển thị thành công
windowClosing(WindowEvent e)	Lắng nghe frame được đóng lại
windowClosed(WindowEvent e)	Lắng nghe khi frame đóng lại thành công
windowIconified(WindowEvent e)	Lắng nghe khi frame bị thu nhỏ lại
windowDeiconified(WindowEvent e)	Lắng nghe khi frame được phóng to
windowActivated(WindowEvent e)	Lắng nghe khi frame được focus
windowDeactivated(WindowEvent e)	Lắng nghe khi frame mất focus



JFrame

MouseListener

Sự kiện	Giải thích
mouseClicked(MouseEvent e)	Lắng nghe sự kiện khi click chuột trên frame
mousePressed(MouseEvent e)	Nhấn và giữ chuột trên frame
mouseReleased(MouseEvent e)	Di chuyển chuột từ trong frame ra ngoài frame
mouseEntered(MouseEvent e)	Di chuyển chuột từ ngoài frame vào trong frame
mouseExited(MouseEvent e)	Click hoặc press chuột và nhả tay ra



JFrame

KeyListener

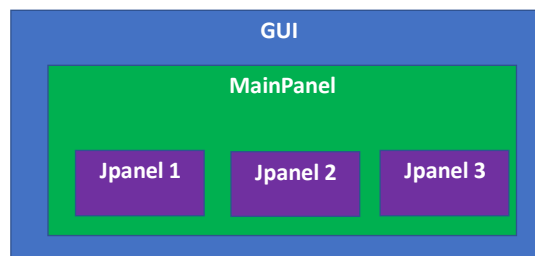
Sự kiện	Giải thích
keyTyped(KeyEvent e)	Lắng nghe sự kiện khi nhấn phím trên frame
keyPressed(KeyEvent e)	Nhấn các phím unicode
keyReleased(KeyEvent e)	Nhấn phím sau đó nhả tay ra



Thêm đối tượng giao diện vào JFrame

Định nghĩa thiết kế

- Là khai báo các thuộc tính Jpanel, giao diện cụ thể từng màn và bố trí chúng lên JFrame.
- Nên thiết kế theo mẫu sau:



Giải thích:

- Mô hình trên, sẽ có một đối tượng Jpanel là (MainPanel) đây là đối tượng duy nhất được add vào GUI
- Thiết kế này giúp cho việc trao đổi dữ liệu các Jpanel dễ dàng hơn, kiến trúc layout trở nên tường minh, dễ quản lý.



Jbutton – Jlable – JTextField – JTextArea

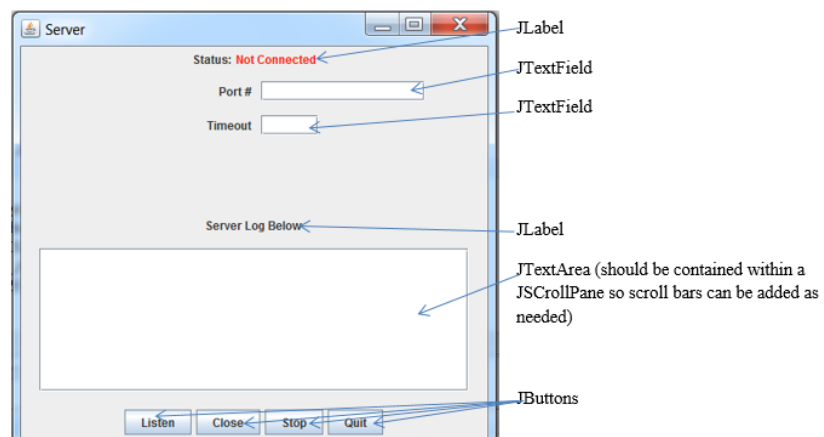
Giới thiệu

- **JLabel**: là đối tượng giao diện thể hiện một tiêu đề có tác dụng ghi chú thích hoặc thông báo trên frame.
- **JButton**: là đối tượng giao diện thể hiện một nút lệnh để người dùng có thể click vào.
- **JTextField**: là đối tượng giao diện thể hiện một khung để người dùng nhập dữ liệu vào. Dữ liệu được nhập trên 1 dòng.
- **JTextArea**: là đối tượng giao diện thể hiện một khung nhiều dòng cho phép người dùng nhập nhiều đoạn văn bản.
- Cả JButton – JLabel – JTextField – JTextArea đều là JComponent giao diện cụ thể.
- Nhận các sự kiện về MouseListener, KeyListener



Jbutton – Jlable – JTextField – JTextArea

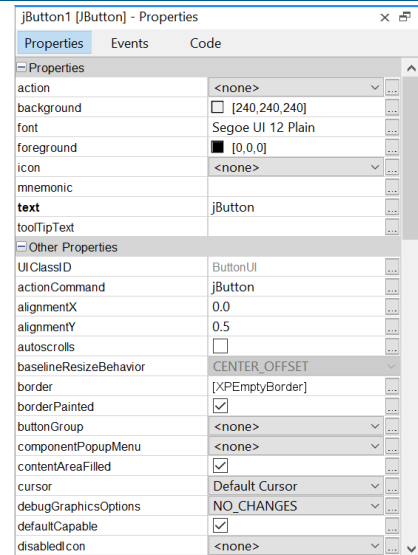
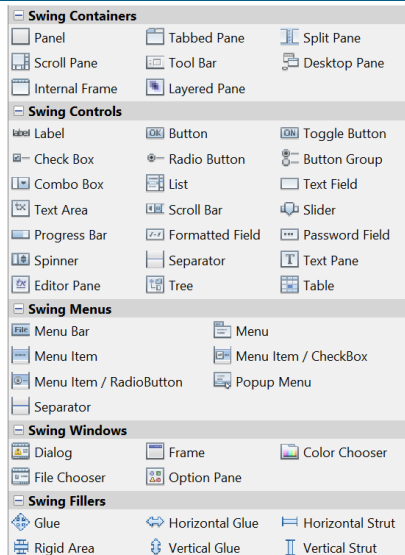
Minh họa





Jbutton – Jlable – JtextField – JtextArea

Hướng dẫn



Khoa Khoa học và Kỹ thuật Thông tin – Trường ĐH CNTT Tp.HCM

211

211



Jbutton – Jlable – JtextField – JtextArea

Các phương thức

Phương thức	Mô tả	Ví dụ
setSize(int width, int height)	Kích thước rộng và cao	setSize(300, 100)
setLocation(int x, int y)	Vị trí hiển thị trong JPanel	setLocation(20, 20)
setBackground(Color c)	Màu sắc nền	getContentPanael.setBackground(Color.RED)
setBounds(int x, int y, int width, int height)	Vị trí và kích thước	setBounds(20, 20, 300, 100)
setVisible(Boolean value)	Ẩn (false) và hiện (true)	setVisible(true)
setText(String text)	Chuỗi hiển thị	setText("Nhập tên")
setFont(Font font)	Định nghĩa font	setFont(Font font)
setForeground(Color c)	Màu chuỗi text	setForeground(Color.GREEN)

Khoa Khoa học và Kỹ thuật Thông tin – Trường ĐH CNTT Tp.HCM

212

212



Jbutton – Jlable – JTextField – JTextArea

Các sự kiện

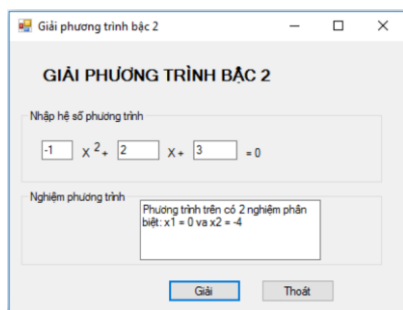
Sự kiện	Mô tả
<code>addMouseListener(MouseListener mL)</code>	Lắng nghe các sự kiện dùng chuột để tương tác như: click, press, move, di chuột vào, di chuột ra
<code>addKeyListener(KeyListener kL)</code>	Lắng nghe các sự kiện dùng chuột để tương tác như: lên, xuống, trái, phải
<code>MouseEvent</code>	Là đối tượng chứa các thông số về chuột, tương ứng với các trạng thái mà <code>Jcomponent</code> lắng nghe được.
<code>KeyEvent</code>	Chứa các thông tin về phím mà người dùng vừa thao tác lắng nghe được, có thể thông qua đối tượng này lấy các thông tin như mã, nội dung vừa nhấn phím gì.
<code>ActionListener</code>	Lắng nghe khi người dùng tương tác vào bằng phím khi ở trạng thái được focus
<code>ActionEvent</code>	Chứa các thông tin cần thiết về đối tượng đang được tương tác



Jbutton – Jlable – JTextField – JTextArea

Bài tập áp dụng

- Thiết kế giao diện giải phương trình bậc 1
- Thiết kế giao diện giải phương trình bậc 2





JCheckBox

Giới thiệu

- Là một đối tượng giao diện thể hiện lựa chọn của người dùng khi có nhiều phương án lựa chọn.



Khoa Khoa học và Kỹ thuật Thông tin – Trường ĐH CNTT Tp.HCM

215

215



JCheckBox

Cách khởi tạo

```
JCheckBox checkbox = new JCheckBox();
```

Các phương thức phổ biến

Phương thức	Mô tả	Ví dụ
setSize(int width, int height)	Kích thước rộng và cao	setSize(300, 100)
setLocation(int x, int y)	Vị trí hiển thị trong JPanel	setLocation(20, 20)
setBackground(Color c)	Màu sắc nền	getContentPanael.setBackground(Color.RED)
setBounds(int x, int y, int width, int height)	Vị trí và kích thước	setBounds(20, 20, 300, 100)
setVisible(Boolean value)	Ẩn (false) và hiện (true)	setVisible(true)

Khoa Khoa học và Kỹ thuật Thông tin – Trường ĐH CNTT Tp.HCM

216

216



JCheckBox

Các phương thức phổ biến (tiếp theo)

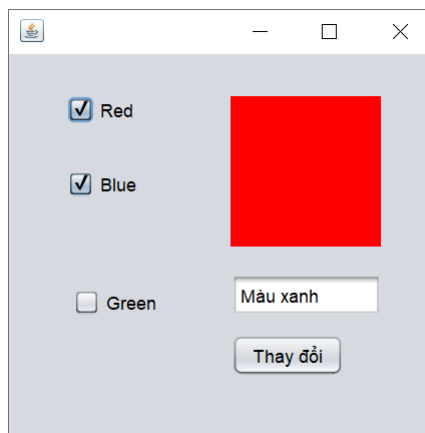
Phương thức	Mô tả
<code>addChangeListener(ChangeListener cl)</code>	Lắng nghe các sự kiện khi người dùng thay đổi trạng thái của check box
<code>addKeyListener(KeyListener kL)</code>	Lắng nghe các sự kiện dùng chuột để tương tác như: lên, xuống, trái, phải
<code>ChangeEvent</code>	Chứa các thông số cần thiết về đối tượng tương ứng với trạng thái JCheckBox lắng nghe được
<code>KeyEvent</code>	Chứa các thông tin về phím mà người dùng vừa thao tác lắng nghe được, có thể thông qua đối tượng này lấy các thông tin như mã, nội dung vừa nhấn phím gì.
<code>setSelected(Boolean value)</code>	Set trạng thái cho check box
<code>isSelected()</code>	Trả về true nếu check box được chọn, false nếu không chọn



JCheckBox

Minh họa

- Xem minh họa trên lớp





JCheckBox

Bài tập áp dụng

Tạo giao diện thỏa yêu cầu sau

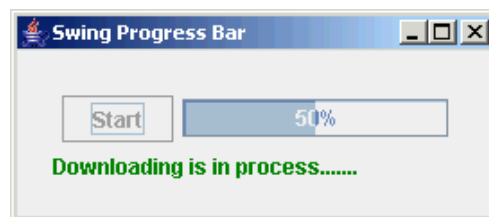
- Bên trái là các check box thể hiện chọn các font chữ Bold, Italic, Plain
- Bên phải là một text area thể hiện các nội dung đã chọn bên các check box khi click vào một button



JProgressBar

Giới thiệu

- Là đối tượng giao diện giúp thể hiện tiến độ của công việc





JProgressBar

Cách khởi tạo

```
JProgressBar bar = new JProgressBar()
```

Các phương thức phổ biến

Phương thức	Mô tả	Ví dụ
setSize(int width, int height)	Kích thước rộng và cao	setSize(300, 100)
setLocation(int x, int y)	Vị trí hiển thị trong JPanel	setLocation(20, 20)
setBackground(Color c)	Màu sắc nền	getContentPanael.setBackground(Color.RED)
setBounds(int x, int y, int width, int height)	Vị trí và kích thước	setBounds(20, 20, 300, 100)
setVisible(Boolean value)	Ẩn (false) và hiện (true)	setVisible(true)



JProgressBar

Các phương thức phổ biến (tiếp theo)

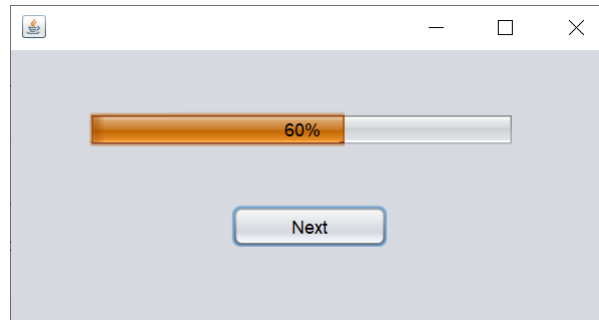
Phương thức	Mô tả
addMouseListener(ChangeListener cl)	Lắng nghe các sự kiện khi người dùng sử dụng chuột để tương tác như click, press, move, di chuột vào JProgressBar, di chuột ra khỏi JProgressBar
addKeyListener(KeyListener kl)	Lắng nghe các sự kiện dùng chuột để tương tác như: lên, xuống, trái, phải
MouseEvent	Chứa các thông số cần thiết về chuột tương ứng với trạng thái JProgressBox lắng nghe được
KeyEvent	Chứa các thông tin về phím mà người dùng vừa thao tác lắng nghe được, có thể thông qua đối tượng này lấy các thông tin như mã, nội dung vừa nhấn phím gì.
setStringPaint(boolean value)	Hiển thị phần trăm tiến độ trên thanh JProgressBox
setMaximum(int value)	Hiển thị giá trị max của thanh tiến độ
setMinimum(int value)	Hiển thị giá trị min của thanh tiến độ
setValue(int value)	Set giá trị hiện tại của thanh tiến độ



JProgressBar

Minh họa

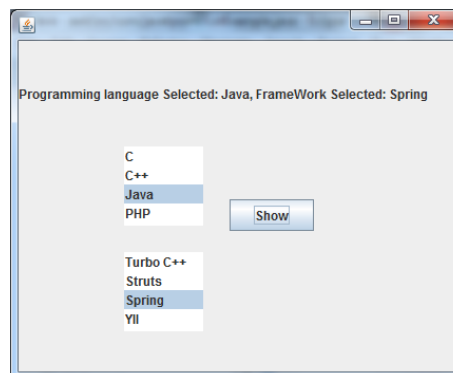
- Xem hướng dẫn xây dựng demo trên lớp



JList

Giới thiệu

- Là đối tượng giao diện thể hiện dưới dạng một danh sách.
- Có thể chọn 1 hoặc nhiều phần tử trong danh sách đó.



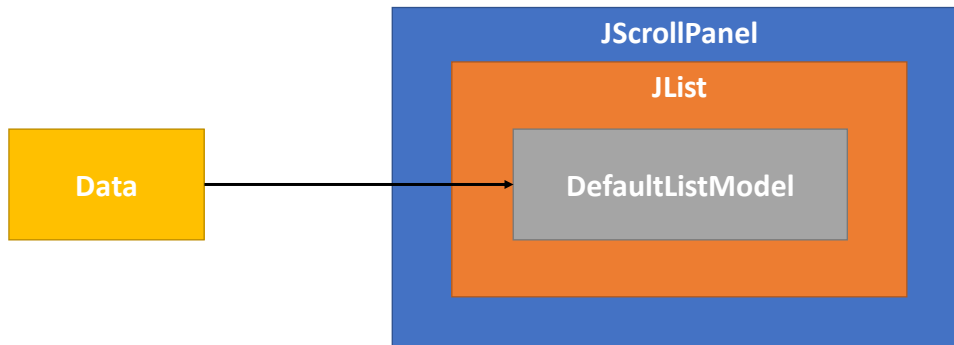


JList

Cú pháp

```
JList <data type> listName = new JList();
```

Cấu trúc



JList

DefaultListModel

- Là một đối tượng dùng để quản lý dữ liệu cho Jlist
- Tương tự ArrayList

Cú pháp:

- `DefaultListModel <data type> listModel = new DefaultListModel();`



JList

Các phương thức quan trọng

- `setModel(ListModel <kiểu dữ liệu> model)`: hiển thị dữ liệu trong JList
- Demo hướng dẫn minh họa tạo JList



JComboBox

Giới thiệu

- Là đối tượng giao diện để hiển thị dữ liệu dưới dạng một danh sách dropdown
- Được phép chọn bất kỳ một phần tử trong danh sách đó.

Before selecting an item:

Effective Java ▼

Easy ▼

Developer ▼

After selecting an item:

Effective Java ▼
Effective Java
Head First Java
Thinking in Java
Java for Dummies

Easy ▼
Easy
Normal
Hard
Hardest

Consultant| ▼
Developer
Designer
Architect
Team Leader

JComboBox in Java look-and-feel

JComboBox in Windows look-and-feel

Editable JComboBox

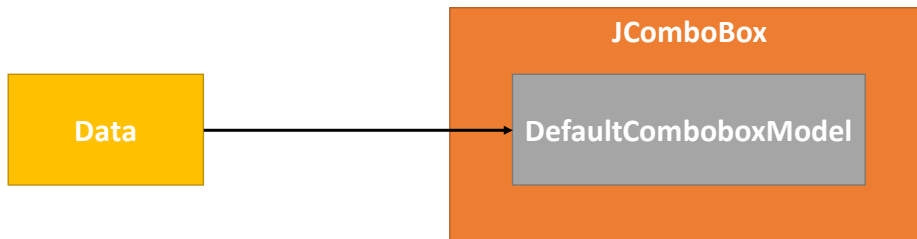


JComboBox

Cú pháp

```
JList <data type> list = new JList()
```

Cấu trúc



JComboBox

DefaultComboboxModel

- Là đối tượng quản lý dữ liệu cho combobox.
- Có các phương thức tương tự như ArrayList.

Cú pháp

```
DefaultComboBoxModel <kdl> listModel = new DefaultComboboxModel();
```

- Các phương thức: addElement(item), getSize(), getSelectedItem(int index)



JComboBox

Minh họa

- Hướng dẫn trên lớp



JTable

Giới thiệu

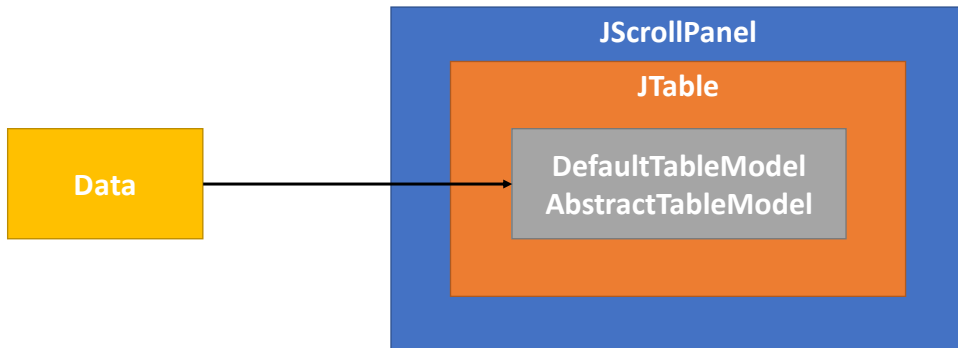
- Là một đối tượng giao diện để hiển thị dữ liệu dưới dạng bảng.
- Thích hợp cho việc biểu diễn nhiều thuộc tính của đối tượng trong danh sách.
- Các cột là các thuộc tính của đối tượng, các hàng là các đối tượng trong danh sách.
- Cú pháp: `JTable table = new JTable();`

No#	Name	Age	Job
1	John	30	Developer
2	Jane	31	Designer
3	Peter	28	Programmer
4	Mary	35	Consultant
5	Kim	27	Developer
6	George	32	Leader
7	Dash	36	Analyst
8	Tim	25	Designer
9	Ana	29	Developer
10	Tom	41	Manager
11	Sam	40	Consultant
12	Patrick	38	Manager
13	Jeremy	24	Programmer
14	David	25	Programmer
15	Steve	26	Designer



JTable

Cấu trúc JTable



JTable

DefaultTableModel

- Là một đối tượng dùng để quản lý dữ liệu JTable dưới dạng mảng 2 chiều

AbstractTableModel

- Là một đối tượng dùng để quản lý dữ liệu từng dòng cho Jtable dưới dạng mảng 2 chiều

DefaultTableColumnTable

- Là đối tượng quản lý danh sách tiêu đề các cột của bảng.



JTable

Phương thức quan trọng

- `setModel(<dữ liệu>)`: Hiển thị dữ liệu cho Jtable

- Minh họa demo trên lớp

Mã SV	Họ và tên	Điểm TB
19520123	Phạm Văn A	6.7
19520124	Nguyễn Văn B	7.8
19520125	Lê Thị C	7.0

Nhập Nhập 2



Ôn thi

Bài 01

- For
- While => Lab02

Bài 02

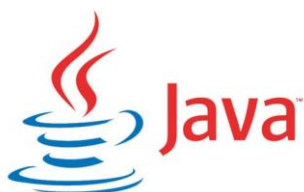
- OOP

Bài 03

- Thiết kế giao diện



CHƯƠNG 9 KẾT NỐI CƠ SỞ DỮ LIỆU



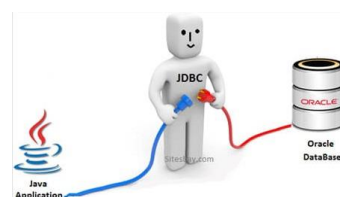
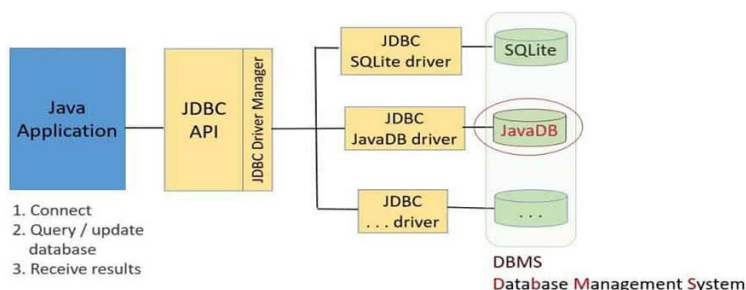
240



Java DataBase Connectivity - JDBC

Giới thiệu JDBC

- JDBC là viết tắt của Java Database Connectivity là một API dùng để kết nối và thực thi các câu lệnh SQL xuống database.
- Truy xuất các cơ sở dữ liệu như MS Access, SQL Server, Oracle,... trong các ứng dụng Java bằng ngôn ngữ truy vấn SQL.
- Các hàm truy xuất cơ sở dữ liệu với JDBC nằm trong gói `java.sql`.*



241



Microsoft JDBC Driver for SQL Server

Download Microsoft JDBC Driver for SQL Server

- Đường dẫn: <https://docs.microsoft.com/vi-vn/sql/connect/jdbc/download-microsoft-jdbc-driver-for-sql-server>

Download Microsoft JDBC Driver for SQL Server

Bài viết • 09/12/2021 • 2 phút để đọc • Trang này có hữu ích không?

The Microsoft JDBC Driver for SQL Server is a Type 4 JDBC driver that provides database connectivity through the standard JDBC application program interfaces (APIs) available on the Java platform. The driver downloads are available to all users at no extra charge. They provide access to SQL Server from any Java application, application server, or Java-enabled applet.

Download

Version 9.4 is the latest general availability (GA) version. It supports Java 8, 11, and 16. If you need to use an older Java runtime, see the [Java and JDBC specification support matrix](#) to see if there's a supported driver version you can use. We're continually improving Java connectivity support. As such we highly recommend that you work with the latest version of Microsoft JDBC driver.

Download Microsoft JDBC Driver 9.4 for SQL Server (zip)

Download Microsoft JDBC Driver 9.4 for SQL Server (tar.gz)

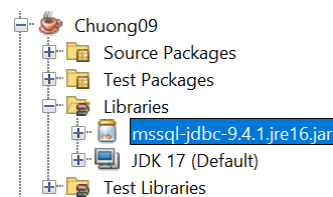
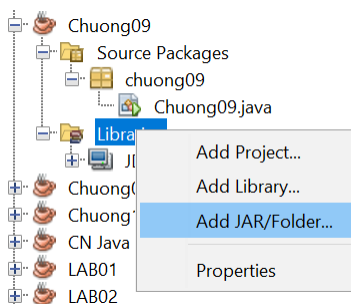
Khoa Khoa học và Kỹ thuật Thông tin – Trường ĐH CNTT Tp.HCM

242

242



Thêm Microsoft JDBC driver vào Java project



Khoa Khoa học và Kỹ thuật Thông tin – Trường ĐH CNTT Tp.HCM

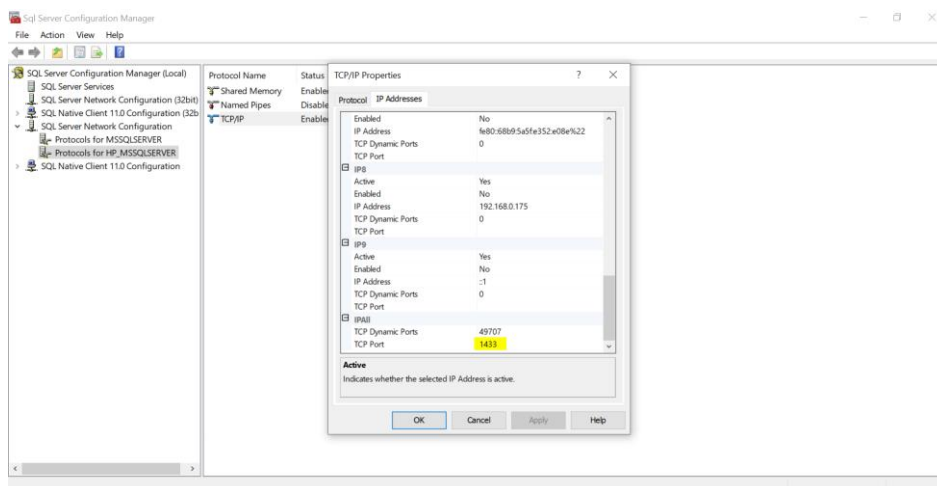
243

243



Kiểm tra port

Mở SQL Server 2019 Configuration Manager



Khoa Khoa học và Kỹ thuật Thông tin – Trường ĐH CNTT Tp.HCM

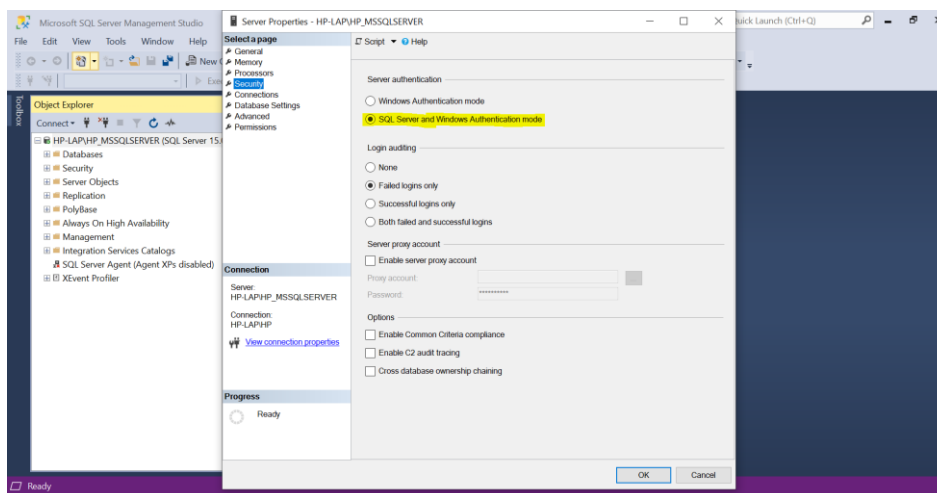
244

244



Kích hoạt tài khoản sa

1. Thiết lập SQL Server cho phép đăng nhập tài khoản của SQL Server



Khoa Khoa học và Kỹ thuật Thông tin – Trường ĐH CNTT Tp.HCM

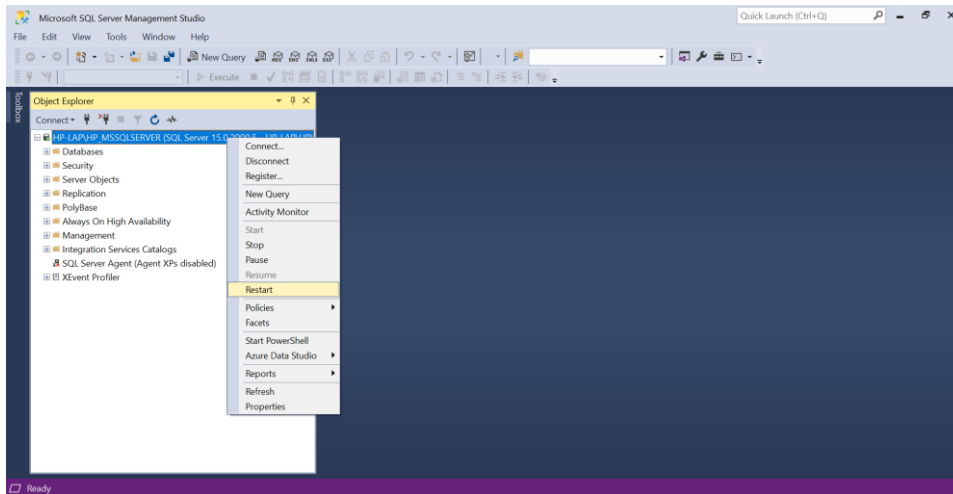
245

245



Kích hoạt tài khoản sa

2. Khởi động lại SQL Server



Khoa Khoa học và Kỹ thuật Thông tin – Trường ĐH CNTT Tp.HCM

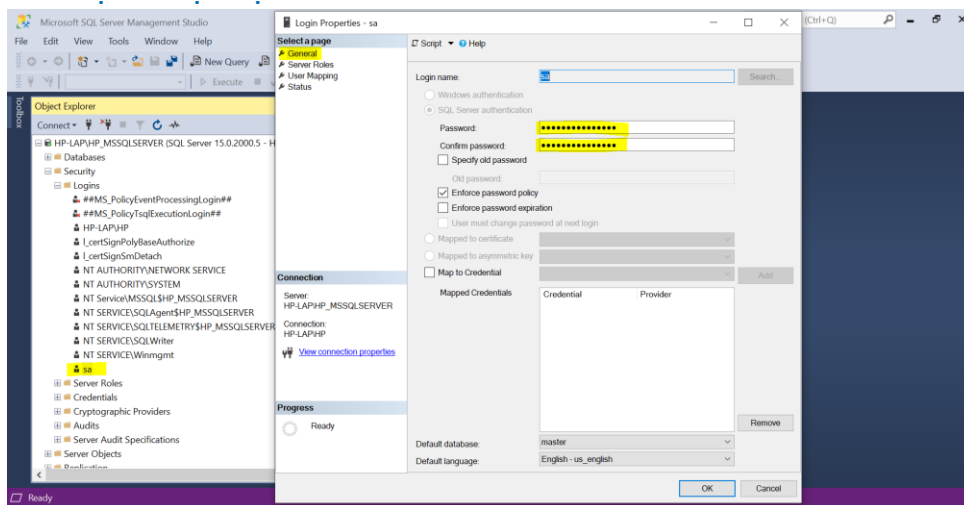
246

246



Kích hoạt tài khoản sa

3. Kích hoạt và đặt mật khẩu cho tài khoản sa



Khoa Khoa học và Kỹ thuật Thông tin – Trường ĐH CNTT Tp.HCM

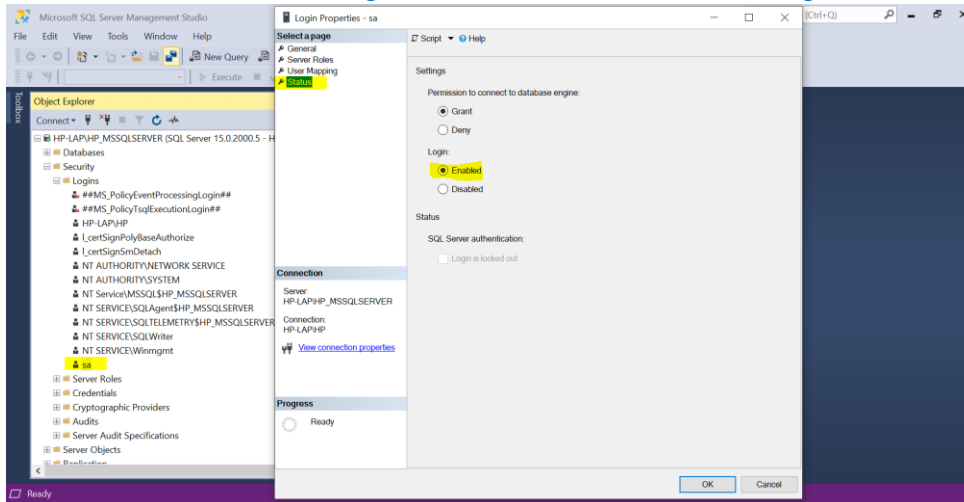
247

247



Kích hoạt tài khoản sa

4. Kích hoạt tài khoản sa bằng cách chọn Enable tại mục Login



Khoa Khoa học và Kỹ thuật Thông tin – Trường ĐH CNTT Tp.HCM

248

248



Kết nối MS SQL Server

Cách 1

```
public class Chuong09_03 {

    public static void main(String[] args) {
        // TODO code application logic here
        SQLServerDataSource ds = new SQLServerDataSource();

        ds.setUser("sa");
        ds.setPassword("sa");
        ds.setServerName("HP-LAP\\HP_MSSQLSERVER");
        ds.setPortNumber(Integer.parseInt("1433"));
        ds.setDatabaseName("QLGV");

        try {
            Connection con = ds.getConnection();
            System.out.println(con);
        } catch (SQLServerException ex) {
        }
    }
}
```

Khoa Khoa học và Kỹ thuật Thông tin – Trường ĐH CNTT Tp.HCM

249

249



Kết nối MS SQL Server

Cách 2

```
public class NewClass {
    public static void main(String[] args) throws SQLException {
        String dbURL = "jdbc:sqlserver://localhost;"
            + "databaseName=QLGV;"
            + "user=sa;password=sa";

        Connection conn = DriverManager.getConnection(dbURL);
        if (conn != null) {
            System.out.println("Connected");
        }
    }
}
```



Kết nối MS SQL Server

```
public class Chuong10 {
    public static void main(String[] args) {
        try {
            String dbURL = "jdbc:sqlserver://localhost:1433;"
                + "databaseName=QLGV7;user=sa;password=sa";
            Connection conn = DriverManager.getConnection(dbURL);
            if (conn != null) {
                System.out.println("Connected");
                DatabaseMetaData dm = (DatabaseMetaData) conn.getMetaData();
                System.out.println("Driver name: " + dm.getDriverName());
                System.out.println("Driver version: " + dm.getDriverVersion());
                System.out.println("Product name: " + dm.getDatabaseProductName());
                System.out.println("Product version: " + dm.getDatabaseProductVersion());
            }
        } catch (SQLException ex) {
            System.err.println("Cannot connect database, " + ex);
        }
    }
}
```



Thực thi câu truy vấn

```
public class ThucThi {
    public static void main(String[] args) {
        try {
            String dbURL = "jdbc:sqlserver://localhost:1433;"
                + "databaseName=QLGV7;user=sa;password=sa";
            Connection conn = DriverManager.getConnection(dbURL);
            if (conn != null) {
                System.out.println("Connected");
                DatabaseMetaData dm = (DatabaseMetaData) conn.getMetaData();
                System.out.println("Driver name: " + dm.getDriverName());
                System.out.println("Driver version: " + dm.getDriverVersion());
                System.out.println("Product name: " + dm.getDatabaseProductName());
                System.out.println("Product version: " + dm.getDatabaseProductVersion());
            }

            CallableStatement cstmt = conn.prepareCall("SELECT * from GIAOVIEN");
            ResultSet rs = cstmt.executeQuery();

            while (rs.next()) {
                System.out.println(rs.getString("MAGV") + ", " + rs.getString("HOTEN"));
            }

        } catch (SQLException ex) {
            System.err.println("Cannot connect database, " + ex);
        }
    }
}
```