TÀI LIỆU KHÓA HỌC “Java Core”

Tác giả: Hỏi Dân IT & Eric

Version: 1.0

Note cập nhật:

- Update tài liệu khóa học

Chapter 0: Introduction **3**
#0. Demo Kết quả đạt được 3
Chapter 1: Lab 01 **6**
#1. Java là gì ? 6
#2. Setup Java 7
#3. Hello world with Java 8
#4. Setup Spring Tool Suite 10
#5. Variables (Khái niệm biến) 11
#6. Các hàm xuất ra màn hình 13
#7. Các hàm toán học 15
#8. Bài tập Lab 01 16
#9. Chữa bài tập Lab 01 18
Chapter 2: Lab 02 **19**
#10. Java Data Types (Kiểu dữ liệu) 19
#11. Java Operators (Toán Tử) 22
#12. Java If...Else 23
#13. Java Switch 25
#14. Bài tập Lab 02 27
#15. Chữa bài tập Lab 02 28
Chapter 3: Lab 03 **29**
#16. Vòng lặp (Loop) 29
#17. Array 30
#18. Luyện tập Array và Loop 31
#19. Bài tập Lab 03 32
#20. Chữa bài tập Lab 03 33
Chapter 4: Lab 04 **34**
#21. Lập trình hướng đối tượng (OOP) là gì ? 34
#22. Khái niệm Class và Object 36
#23. Thực hành tạo Class/Object 38
#24. Class Attributes (Thuộc tính của Class) 39
#25. Class Method (Phương thức của class) 40
#26. Định nghĩa hàm tạo (constructor) 41
#27. Access Modifier (private/public/protected) 42
#28. Encapsulation 44
#29. Bài tập Lab 04 45
#30. Chữa bài tập Lab 04 46
Chapter 5: Lab 05 **47**
#31. Non-primitive(reference) data type 47
#32. Java Autoboxing/Unboxing (Object wrapper class) 48
#33. ArrayList là gì ? 49
#34. Java Generics (Basic) 50
#35. Bài tập Lab 05 51
#36. Chữa bài tập lab 05 52

Chapter 6: Lab 06 **53**
#37. String (Chuỗi) 53
#38. Bài tập sử dụng String 54
#39. Regular Expression 55
#40. Bài tập Regular Expression 56
Chapter 7: Lab 07 **57**
#41. Java Inheritance 57
#42. Super Keyword trong Java 59
#43. Abstract Class (Basic) 62
#44. Polymorphism (Tính đa hình) 63
#45. Bài tập Lab 07 64
#46. Chữa bài tập Lab 07 65
Chapter 8: Lab 8 **66**
#47. Interface 66
#48. Interface với Java 8 68
#49. Java Package 69
#50. Bài tập Lab 08 70
#51. Chữa bài tập Lab 08 71
Chapter 9: Lab 09 **72**
#52. Try...catch 72
#53. Keyword finally 74
#54. Keyword Throws và Throw 75
#55. Bài tập Lab 09 76
#56. Chữa bài tập Lab 09 77
Chapter 10: Lab 10 **78**
#57. Tổng quan về Java IO (Stream) 78
#58. Phân loại Stream với Java IO 79
#59. Ví dụ về Read/Write File 80
#60. Path Class 81
Chapter 11: Lab 11 **82**
#61. Processes and Threads 82
#62. Thread Objects 83
#63. What's next ? Tổng kết các kiến thức đã học 84
Lời Kết **85**

Chapter 0: Introduction
Giới thiệu về khóa học Java

#0. Demo Kết quả đạt được
Link video demo: https://youtu.be/ZplRX4di6Fg

1. Giới thiệu
- Đây là khóa học được public 100% FREE (full course), và "đúng nghĩa" dành cho beginners.
- Mình mất 2 tháng (làm liên tục) mới xong course này
+ khóa FREE nhưng chất lượng cao
+ Có tài liệu, giáo án và bài tập rõ ràng.

và, chân thành cảm ơn các bạn hội viên (memberships) đã ủng hộ để mình có thêm động lực hoàn thiện khóa học này.
(đi làm fulltime các ông ạ. haizzz)

2. Khóa học này dành cho ai ?
Khóa học này chính là quá trình mình "tự học" một ngôn ngữ lập trình, và nếu như:

- Bạn muốn xây dựng đam mê lập trình (từ số 0) ? Tâm lý không biết code :v

- Bạn muốn học ngôn ngữ Java từ số 0 ?

=> đây chính là khóa học bạn cần.

3. Các kiến thức đề cập

Chia thành 11 chapters, bao quát các chủ đề sau:

- Kiến thức nền tảng của java:
+ **Variables**:
    - Khai báo biến, data type (dữ liệu nguyên thủy/không nguyên thủy - object)
    - Kiểu dữ liệu String
    - Vòng lặp (loop)
    - Array
    - Auto boxing (wrapper class)

+ **Lập trình hướng đối tượng** : Class/Object
    - Khai báo Class
    - Tạo đối tượng
    - Access modifier: public, private, protected
    - Encapsulation: getter/setter (tính che dấu bảo vệ dữ liệu)

+ **Kế thừa**:
    - extends (class)
    - implements (interface)
    - polymorphism (tính đa hình)

+ **Exception**: xử lý ngoại lệ
    - try/catch
    - finally
+ **Java I/O**

+ **Multi Threading**

4. Tài liệu của khóa học

- Ý tưởng mình làm khóa học này (outline) là dựa vào giáo trình Lập trình Java 1 & 2 của FPT PolyTechnic

Giáo trình trên được share FREE trên internet, các bạn có thể download (google là ra)

Tuy nhiên, giáo trình của mình có 'update' thêm kiến thức của Java (version mới) và xâu chuỗi kiến thức theo 1 trình tự mà mình thấy là hợp lý cho beginners

- Ngoài ra, các trang tài liệu khác sẽ được đề cập trong từng video

Và, trong quá trình thực hiện series, sẽ không tránh khỏi các thiếu sót. Mong a/e góp ý bằng cách comment để mình update thêm videos cho mọi người:v

Về tác giả:

Mọi thông tin về Tác giả Hỏi Dân IT, các bạn có thể tìm kiếm tại đây:

Website chính thức: https://hoidanit.com.vn/

Youtube "Hỏi Dân IT" : https://www.youtube.com/@hoidanit

Tiktok "Hỏi Dân IT" : https://www.tiktok.com/@hoidanit

Fanpage "Hỏi Dân IT" : https://www.facebook.com/askITwithERIC/

Udemy Hỏi Dân IT: https://www.udemy.com/user/eric-7039/

Chapter 1: Lab 01
Làm quen với ngôn ngữ Java

#1. Java là gì ?

1. Lịch sử phát triển
1991: Ra đời với tên gọi Oak bởi Sun Microsystem
1995: Đổi tên thành java (tên hòn đảo trồng cafe tại Indonesia) => logo của java
2010: Oracle mua lại
bonus: javascript khi ra đời, tên có tiền tố "java" để lấy sự nổi tiếng :v

2. Đặc điểm của java
Write once, run anywhere (Bạn viết code 1 lần, code của bạn có thể chạy trên nhiều môi trường khác nhau, ví dụ windows, macos, linux...)

learn one, write anywhere (react)

Đặc điểm nổi bật:
- Hướng đối tượng (OOP)
- Chạy trên mọi nền tảng
- Bảo mật cao
- Đa luồng...

Java có thể làm các sản phẩm:
- Mobile app, game => android (java/kotlin)
- Web app
- Embedded devices (nhúng): atm, tivi, tủ lạnh...
...

3. Cơ hội việc làm với Java ? Java liệu đã die ?

#2. Setup Java

Link tải Java 17 và IDE sử dụng trong video (windows):
https://drive.google.com/drive/folders/11_KIENr8SJebcQMGmFJSDgr1Sli0Hv0J?usp=sharing

- Lưu ý: cài đặt java 17

1. JDK (java development kit)

https://www.oracle.com/java/technologies/downloads/archive/
https://www.oracle.com/java/technologies/javase/jdk17-archive-downloads.html

download và cài đặt java 17 (8, 11, 17, 21 -> LST: long term support)

- kiểm tra version: **java --version**

- kiểm tra java path (if needed)
https://www.w3schools.com/java/java_getstarted.asp

2. Setup IDE

IDE là công cụ giúp việc "coding" trở nên dễ dàng hơn => ví dụ sử dụng notepad :v

- Eclipse
- **IntelliJ IDEA**
- Netbeans

- Spring tool suite:
https://spring.io/tools

=> Spring Tools 4 for Eclipse

Bước 1: Download files

Bước 2: giải nén file .jar

Bước 3: giải nén file contents.zip

Bước 4: Run file sts-4.20.0.RELEASE/SpringToolSuite4.exe

#3. Hello world with Java

1. Run HelloWorld

Lưu ý:

- Đã cài đặt thành công java. Kiểm tra = cách: **java --version**
- Đã cài đặt Spring tool suite

Bước 1: tạo java project

File -> New -> Java Project
(hoặc chọn trực tiếp "Create a Java project")

Bước 2:

Project name: hoidanit
- Có thể lựa chọn location của project

- sử dụng JRE là java 17

-> chọn finish

Mặc định, project java được tạo sẵn, sẽ bao gồm:
- JRE System library: các thư viện viết sẵn của java khi cài đặt java
- src : (package) thư mục nơi chứa code

Bước 3:

Tạo file: Nhấn chuột phải vào src -> New -> Class -> HelloWorld -> finish

-> mặc định, sẽ có 1 file tên là HelloWorld.java được tạo bên trong src/hoidanit

```
//HelloWorld.java
package hoidanit;

public class HelloWorld {

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

```
}
}
```

Bước 4: Chạy chương trình

Cách 1: Nhấn chuột phải vào HelloWorld.java -> Run as -> Java application

Cách 2: Open file location -> gõ lệnh: java HelloWorld.java

2. Write once, run anywhere

//

#4. Setup Spring Tool Suite

1. auto format code
truyền thống (thổ dân):
select block code -> nhấn chuột phải -> Source -> Format (ctrl +shift + F)

Cấu hình:
https://stackoverflow.com/a/234625

2. Chỉnh font
https://stackoverflow.com/a/4923243

3. Chỉnh theme light/dark
Windows -> preferences -> Appearance -> theme

4. Bảo vệ mã nguồn với GIT
//todo
Link git của khóa học:
https://gitlab.com/public-starter-projects1/05-java-core/01-java-starter

#5. Variables (Khái niệm biến)

Ví dụ:

```
package hoidanit;

public class TestClass {

	public static void main(String[ ] args) {
		int a = 6;
		int b = 9;
		int c = a + b;
		System.out.println("Sum: " + c);
	}

}
```

Ý nghĩa:
- Khối code trên (block code) tạo giá trị cho a, b và tính tổng c, sau đấy in ra màn hình.

a, b, c gọi là "biến số nguyên"

=> biến số (variables) là dữ liệu chương trình sử dụng để tính toán (mục đích tạo ra kết quả nào đấy)

1. Quy tắc khai báo

<type> <variable_name> = <initial_value>
<kiểu_dữ_liệu> <tên_biến> = <giá_trị_khởi_tạo>

Ví dụ:
int a = 6;
<type>: int -> kiểu dữ liệu số nguyên
<variable_name> : a -> đặt tên biến là "a"
<initial_value>: 6 -> giá trị khởi tạo của biến a là 6

Lưu ý: có thể khai báo biến mà không cần khởi tạo giá trị ban đầu.
Ví dụ:
int a;
a = 10;

2. Quy tắc đặt tên biến

- Sử dụng ký tự alphabet (a-zA-Z), dấu $ hoặc dấu gạch dưới(_)

- Tên biến có phân biệt HOA/thường

- không đặt tên bắt đầu bởi số, và không dùng các từ khóa được sử dụng để xây dựng ngôn ngữ java, ví dụ:
new, class, const...

ví dụ:
Tên biến nào sau đây không hợp lệ:
abc
1abc
ab_c
_abc
$abc
ab-c
_123
$123
if
$if

3. Các toán tử thường gặp (phép toán số học)

+ -> phép cộng. ví dụ: int a = 6 + 9
- -> phép trừ. ví dụ: int b = 6 - 9;
* -> phép nhân . ví dụ: double c = 6 * 9
/ -> phép chia. ví dụ: double d = 6/9;

Toán tử dùng để thực hiện phép tính số học, thực hiện theo quy tắc ưu tiên sau:
- 1. Nhân và chia
- 2. Cộng và trừ
- 3. Trái sang phải

Ví dụ: 5 + 9*3 - 10/2 = ?

#6. Các hàm xuất ra màn hình

1. In ra màn hình console

```
//in ra kết quả nhưng "không xuống dòng": print
System.out.print( );

//in ra kết quả rồi xuống dòng: print line
System.out.println( );


//in ra kết quả có định dạng: print format
System.out.printf( );
%d: số nguyên
%f: số thực
 - mặc định là 6 số lẻ
 %.3f định dạng 3 số lẻ
%s: chuỗi

Ví dụ:
System.out.print("Hỏi Dân IT");
System.out.println("với Eric");
System.out.printf("có %d người đăng ký", 40000);


Ví dụ:
Khai báo 2 biến name và age. Sử dụng 3 hàm trên để in ra màn hình:
My name is <name>, age = <age>
```

2. Nhập từ bàn phím

java.util.Scanner là công cụ được java hỗ trợ sẵn, dùng để nhập dữ liệu từ bàn phím

Bước 1: Tạo đối tượng Scanner

```
Scanner scanner = new Scanner(System.in);
```

Bước 2: sử dụng các phương thức được cung cấp sẵn

```
scanner.nextLine() => nhận 1 dòng nhập từ bàn phím => trả ra dữ liệu string
scanner.nextInt() => nhận 1 số nguyên từ bàn phím => trả ra dữ liệu số nguyên
```

scanner.nextDouble() => nhận 1 số thực từ bàn phím => trả ra dữ liệu số thực
Ví dụ:
Khai báo 2 biến name và age.
Nhập name và age từ bàn phím
Sử dụng 3 hàm trên để in ra màn hình:
My name is <name>, age = <age>

```
public static void main(String[] args) {
		Scanner scanner = new Scanner(System.in);
		System.out.println("Your name is: ");
		String name = scanner.nextLine();
		System.out.println("Your age is: ");
		int age = scanner.nextInt();
		System.out.println("My name is " + name + ", age = " + age);
		scanner.close();
	}
```

https://stackoverflow.com/questions/12519335/resource-leak-in-is-never-closed

#7. Các hàm toán học

Các hàm toán học có thể kể tới như:

- Tính Căn bậc 2
- Tính lũy thừa
- Làm tròn số...

https://www.w3schools.com/java/java_math.asp

https://www.w3schools.com/java/java_ref_math.asp

#8. Bài tập Lab 01

Mục tiêu:

- Biết cách tạo 1 java project đơn giản với Spring Tool Suite
- Đọc dữ liệu từ bàn phím
- In giá trị ra console

Bài 1:

Viết chương trình cho phép nhập **tên sinh viê**n, **điểm trung bình** từ bàn phím.
In ra màn hình với định dạng:
<tên_sinh_viên> có điểm = <điểm>

Ví dụ: "Eric" có điểm = 9.6
<tên sinh viên>: Eric
<điểm> : 10

Gợi ý:

- Chạy spring tool suite để tạo dự án java
- tạo class Lab1Bai1
- tạo hàm main

- Đọc kết quả từ bàn phím:

```
Scanner scanner = new Scanner(System.in);
String name = scanner.nextLine();
```

- In kết quả với println:

```
System.out.println("Your name is: " + name);
```

Bài 2: Viết chương trình nhập từ bàn phím 2 cạnh của 1 hình chữ nhật.
Tính toán và in ra console:

- chu vi hình chữ nhật
- diện tích hình chữ nhật
- cạnh nhỏ nhất của hình chữ nhật

Gợi ý:

chu vi = (dài + rộng) x 2
diện tích = dài x rộng

cạnh nhỏ nhất = Math.min(dài, rộng)

Bài 3: Viết chương trình nhập vào bàn phím cạnh của một khối lập phương. Tính và xuất ra thể tích của hình lập phương.

Gợi ý:
thể tích = cạnh x cạnh x cạnh = cạnh^3 = Math.pow(canh, 3)

#9. Chữa bài tập Lab 01

Source code video này:
https://gitlab.com/public-starter-projects1/05-java-core/01-java-starter/-/commit/491206b63046cc7e573a8de6b939d2604f575361

Chapter 2: Lab 02
Thực hành câu điều kiện với Java

#10. Java Data Types (Kiểu dữ liệu)
https://www.w3schools.com/java/java_data_types.asp

```
int myNum = 5;              // Integer (whole number)
float myFloatNum = 5.99f;   // Floating point number
char myLetter = 'D';        // Character
boolean myBool = true;      // Boolean
String myText = "Hello";    // String
```

Kiểu dữ liệu được chia thành 2 nhóm:

Primitive data types (dữ liệu nguyên thủy), bao gồm: byte, short, int, long, float, double, boolean, char
Non-primitive data types : String, Arrays and Classes (sẽ học sau)

1. Kiểu dữ liệu nguyên thủy

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

2. Java Numbers
https://www.w3schools.com/java/java_data_types_numbers.asp
Được chia thành 2 nhóm:
- Integer Types: lưu trữ số nguyên, bao gồm: byte, short, int và long

- Float point types: lưu trữ số thực, bao gồm: float, double

2.1 Byte
ví dụ:

byte myNum = 100;
System.out.println(myNum);

=> lưu trữ dữ liệu từ -128 tới 127 (dùng để save memory, vì byte chiếm 1 byte)

2.2 Short

=> lưu trữ dữ liệu từ -32768 to 32767:

short myNum = 5000;
System.out.println(myNum);

2.3 Int

-2147483648 to 2147483647
2^(N-1) => 2^31

2.4 Long

-9223372036854775808 to 9223372036854775807 (2^63)
=> được sử dụng khi int không đủ giá trị để lưu trữ

long myNum = 15000000000L;
System.out.println(myNum);

=> lưu ý: cần thêm chữ L, nếu không sẽ quan niệm là "integer"

3. Floating Point Types

- Được dùng để lưu trữ số thực (số thập phân)

Lưu ý: java phân biệt số nguyên và số thực.
Với 1 vài ngôn ngữ, ví dụ **javascript**, không có sự phân biệt trên, chỉ quan niệm là "number"

ví dụ về float:
float myNum = 5.75f;
System.out.println(myNum);

ví dụ về double:
double myNum = 19.99d;
System.out.println(myNum);

=> nếu không có ký tự "f/d" ở cuối biến số => java sẽ coi là biến "double"

=> tùy thuộc vào giá trị của biến mà lựa chọn kiểu lưu trữ phù hợp. float (4 bytes); double (8 bytes)

4. Boolean types

```
boolean isJavaFun = true;
boolean isFishTasty = false;
System.out.println(isJavaFun);     // Outputs true
System.out.println(isFishTasty);   // Outputs false
```

5. Characters

https://www.w3schools.com/java/java_data_types_characters.asp

5.1 Char

=> được dùng để lưu trữ "single character"

```
char myGrade = 'B';
System.out.println(myGrade);
```

5.2 String

=> lưu ý (chữ S viết hoa, khác biệt với các kiểu dữ liệu đã học ở trên)

```
String greeting = "Hello World";
System.out.println(greeting);
```

#11. Java Operators (Toán Tử)
https://www.w3schools.com/java/java_operators.asp

1. Toán tử số học (Arithmetic Operators)
cộng: +
trừ: -
nhân: *
chia: /
chia lấy số nguyên: % chia lấy dư
tăng 1: ++
giảm 1: --

2. Toán tử so sánh (trả về true/false)
bằng: ==
khác: !=
lớn hơn: >
nhỏ hơn: <
lớn hơn or bằng: >=
nhỏ hơn or bằng: <=

3. Toán tử logic
điều kiện và: &&
điều kiện hoặc: ||
not (lấy giá trị phủ định): !

#12. Java If...Else
https://www.w3schools.com/java/java_conditions.asp

1. If
Cú pháp:
if(<điều kiện>) {
<công_việc>
}

=> Nếu <điều_kiện> == true => <công_việc> được thực hiện
ví dụ:
int score = 3;
if (score < 4) {
System.out.println("Tạch rồi, học lại đêeeeeeee");
}

int score = 5;
if (score < 4) {
System.out.println("Tạch rồi, học lại đêeeeeeee");
}

2. If...else

if(<điều_kiện>) {
<công_việc_1>
}
else {
<công_việc_2>
}

=> nếu <điều_kiện> == true => <công_việc_1> được thực hiện. ngược lại, <công_việc_2> được thực hiện

int score = 3;
if (score < 4) {
System.out.println("Tạch rồi, học lại đêeeeeeee");
}else {
System.out.println("Qua môn rồi mày ơiiiiii");

}

3. Nhiều điều kiện

```
if(<điều_kiện_1>){
<công_việc_1>
}else if (<điều_kiện_2) {
<công_việc_2>
}
....
...
else {
<công_việc_n>
}
```

Bài tập: Viết chương trình tính thuế

Viết chương trình cho phép nhập vào số tiền thuế (đơn vị là triệu đồng), và in ra kết quả:
- dưới 10M: không đóng thuế
- từ 10M tới 15M: thuế 10%
- từ 15 tới 30M : thuế 20%
- trên 30M: thuế 50%

Ví dụ: nhập vào 25M -> in ra kết quả "thuế 20%"

#13. Java Switch
https://www.w3schools.com/java/java_switch.asp

- Keyword: break; default

```
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;
}
```

// Outputs "Thursday" (day 4)

Bài tập: Viết chương trình lựa chọn chức năng

```
System.out.println(">> LỰA CHỌN TÍNH NĂNG <<");
System.out.println("++ ---------- ++");
System.out.println("| 1. Cộng      |");
System.out.println("| 2. Trừ       |");
System.out.println("| 3. Kết thúc  |");
System.out.println("++ ---------- ++");
Scanner scanner = new Scanner(System.in);
System.out.println("Lựa chọn của bạn là : ");
```

Yêu cầu:
Nếu nhập vào 1 => in ra "Bạn đã lựa chọn phép cộng"
Nếu nhập vào 2 => in ra "Bạn đã lựa chọn phép trừ"
Nếu nhập vào 3 => in ra "Bạn đã lựa chọn thoát chương trình" => System.exit(0);

#14. Bài tập Lab 02

Mục tiêu:

- Sử dụng các loại toán tử
- Sử dụng câu điều kiện if/switch

Bài 1: Cho phương trình *a*x + b = 0

Viết chương trình nhập vào 2 số nguyên *a* và b. Tính toán nghiệm của phương trình trên

Gợi ý:

- Nếu *a* = 0, b = 0 => thông báo "phương trình có vô số nghiệm"
- Nếu *a* = 0, b # 0 => thông báo "phương trình vô nghiệm"
- Còn lại: x = -b/a => thông báo x = ?

Bài 2: Cho phương trình: *a*x^2 + bx + c = 0

Viết chương trình nhập vào 3 số nguyên *a*, b, c. Tính toán nghiệm của phương trình trên

Gợi ý:

- Nếu *a* = 0 => làm tương tự ví dụ bài 1
- Nếu a # 0:
  - Tính delta = b^2 - 4ac.
  - Nếu delta < 0 => thông báo "phương trình vô nghiệm"
  - Nếu delta = 0 => thông báo "nghiệm kép x = -b/(2*a)
  - Nếu delta > 0 => thông báo có 2 nghiệm riêng biệt
    x1 = (-b + căn(delta))/(2*a)
    x2 = (-b - căn(delta))/(2*a)

Bài 3: Viết chương trình tính số tiền điện (in kết quả ra console)
Yêu cầu: nhập vào số điện, sau đấy tính số tiền dựa vào quy luật sau:

- Nếu số điện sử dụng từ 0 tới 100, giá mỗi số điện là 1000
- Từ số điện 101 (tức là > 100), giá mỗi số điện là 1500

Gợi ý:
- Nếu số điện <= 100 => số tiền = số điện x 1000
- Nếu số điện > 100
=> số tiền = 100 * 1000 + (số điện - 100) * 1500

Bài 4: Viết chương trình tổ chức menu:

```
System.out.println(">> LỰA CHỌN TÍNH NĂNG <<");
System.out.println("++ ------------------------------------ ++");
System.out.println("| 1. Giải phương trình bậc nhất        |");
System.out.println("| 2. Giải phương trình bậc hai         |");
System.out.println("| 3. Tính số tiền điện                 |");
System.out.println("| 4. Kết thúc                          |");
System.out.println("++ ---------------- ++");
```

Yêu cầu:
- Khi nhập vào "tính năng", sẽ thực hiện theo 3 bài tập đã làm ở trên

#15. Chữa bài tập Lab 02

Source code Part 1:
https://gitlab.com/public-starter-projects1/05-java-core/01-java-starter/-/commit/b9796b1105003fdcbe51bbe66a5216a3fb0e3b17

Source code Part 2:
https://gitlab.com/public-starter-projects1/05-java-core/01-java-starter/-/commit/a6b0df45a6c70d16e5688aa30ae5c020f783716f

Chapter 3: Lab 03
Thực hành sử dụng vòng lặp với Java

#16. Vòng lặp (Loop)
1. Vòng lặp For
https://www.w3schools.com/java/java_for_loop.asp

2. Vòng lặp While/Do...While
https://www.w3schools.com/java/java_while_loop.asp

//phân biệt While và Do...While

4. Break/Continue
https://www.w3schools.com/java/java_break.asp

#17. Array
https://www.w3schools.com/java/java_arrays.asp

https://docs.oracle.com/javase/tutorial/java/nutsandbolts/arrays.html

//thêm hình ảnh minh họa cho array
https://docs.oracle.com/javase/tutorial/java/nutsandbolts/arrays.html

1. Định nghĩa

Array là cách chúng ta lưu trữ "nhiều giá trị" bên trong 1 biến duy nhất (store multiple values in single variable)
=> không tốn công khai báo nhiều lần.

Để định nghĩa array, khai báo "type" và cặp dấu đóng/mở ngoặc vuông (square brackets).

Chiều dài (length) của array được tạo ra khi gán giá trị cho array

Ví dụ: **String [ ] cars**; // khai báo array và không gán giá trị (chưa khởi tạo không gian lưu trữ)
=> nếu chỉ khai báo như trên, compiler sẽ báo lỗi

String[ ] cars = {"Volvo", "BMW", "Ford", "Mazda"};
=> Array string, length = 4

Int[ ] array = new int[10] //khởi tạo memory chứa 10 phần tử int

2. Truy cập phần tử của mảng

Để truy cập phần tử của mảng, sử dụng "chỉ số của mảng" (index)

Chỉ số của mảng bắt đầu từ 0

=> cars[0] = "Volvo"
cars[3] = "Mazda"

3. Loop Through an Array (duyệt tất cả phần tử của mảng)

https://www.w3schools.com/java/java_arrays_loop.asp

#18. Luyện tập Array và Loop

1. Các method của Array
- array.length
Array.sort
Arrays.toString(a)

2. Array sort

```
int[] a = { 10, 5, 20, 30, 1, 3, 69, 12, 2 };
for (int i = 0; i < a.length - 1; i++) {
        for (int j = i + 1; j < a.length; j++) {
                if (a[i] > a[j]) {
                        int temp = a[i];
                        a[i] = a[j];
                        a[j] = temp;
                }
        }
}
System.out.println("a = " + Arrays.toString(a));
```

#19. Bài tập Lab 03
https://stackoverflow.com/a/50753931

Mục tiêu:
- Sử dụng Vòng lặp và câu lệnh Break/Continue
- Sử dụng Array

Bài 1: Viết chương trình nhập vào 1 số nguyên từ bàn phím. Hiển thị kết quả số đấy là số nguyên tố hay không ?

Gợi ý:
- Số nguyên tố là số chỉ chia hết cho 1 và chính nó (tối đa 2 ước)

- Dùng vòng lặp (2, Số nguyên -1). Nếu số nguyên chia hết cho i => ko là số nguyên tố

- Sử dụng toán tử % để biết có chia hết hay không ?

Bài 2: Viết bảng cửu chương của 1 số nguyên bất kỳ
Gợi ý:
- Dùng vòng lặp từ 1 tới 10
```
sysout("  %d x %d = %d ", x, i, x*i)
```

Bài 3: Viết chương trình nhập vào 1 mảng số nguyên bất kỳ:
- Sắp xếp và xuất mảng vừa nhập ra màn hình
- Xuất phần tử có giá trị nhỏ nhất
- Xuất phần tử có giá trị lớn nhất

Gợi ý:
- Sử dụng Arrays.sort
- Sử dụng Math.min, Math.max

#20. Chữa bài tập Lab 03

Source code video này:
https://gitlab.com/public-starter-projects1/05-java-core/01-java-starter/-/commit/35ca4bd4dafbee59d073856696c4d1675a099b46

Chapter 4: Lab 04
Thực hành sử dụng OOP (lập trình hướng đối tượng)

#21. Lập trình hướng đối tượng (OOP) là gì ?

OOP: Object-Oriented Programming

At the end of the day: chương trình/phần mềm/app là tập hợp của các khối code (block code) và máy tính thực hiện tuần tự theo yêu cầu của người dùng.

Máy tính chỉ có thể "chạy từng dòng lệnh"

Có rất nhiều cách để hoàn thiện 1 chương trình:
- procedural programming (pascal) /structure programming
- functional programming (react hook vs react class component)
- object oriented programming
...
=> Cách làm nào không quan trọng, điều quan trọng là nó giải quyết tốt vấn đề bạn gặp phải.

OOP có các lợi thế:
- Clear Structure : cấu trúc rõ ràng
- Reuse application code : tính tái sử dụng code cao => dễ dàng maintain (bảo trì), modify (sửa đổi) và debug (fix bug)
- **designing large and complex programs** (tương tự angular) : mục tiêu giải quyết các bài toán phức tạp

Vậy OOP là gì ?

OOP là cách chúng ta mô phỏng thế giới thực (real world) vào chương trình (program) máy tính.
Tất cả các "đối tượng" tham gia vào chương trình chính là những thành phần chủ chốt.

Ví dụ với website hoidanit.com.vn (phân tích chức năng khóa học):
Các "đối tượng"/thành phần tạo lên website:
+ Danh sách khóa học
    + khóa học riêng lẻ

+ nội dung của khóa học : tiêu đề, giá cả...
+ giáo án của khóa học

=> OOP thiên hướng "mô phỏng" những điều bạn thấy (sử dụng) ở thế giới thực tế vào thế giới programing (coding).
but how ?

#22. Khái niệm Class và Object

Để mô phỏng "real world", chúng ta cần các "đối tượng" (target)
Đối với 1 sinh vật/đồ vật trong thế giới thực, điều có điều miêu tả bởi 2 thứ:
- Đặc điểm (thuộc tính)
- Hành vi đặc trưng của nó

Ví dụ:

Miêu tả con sư tử: (sinh vật)
- Đặc điểm (thuộc tính):
+ Có lông
+ thuộc họ mèo

- Hành vi:
+ đi săn mồi

Miêu tả 1 bạn sinh viên: (sinh vật)
- Đặc điểm:
+ có mã số sinh viên
+ có địa chỉ sinh sống
+ có tên lớp học

- Hành vi:
+ ôn tủ cho qua môn
+ ngủ nướng qua ngày
+ xem youtube lấy kiến thức qua môn :v

Miêu tả 1 cái máy tính (đồ vật)
- Đặc điểm:
+ có dung lượng ổ cứng (SSD/HDD)
+ có dung lượng bộ nhớ (RAM)

- Hành vi:
+ có thể on/off/restart

=> Sử dụng "đối tượng" để mô hình hóa.
Tuy nhiên, chúng ta không "miêu chỉ chi tiết", chỉ miêu tả "đối tượng tổng quát" (đây gọi là **tính trừu tượng hóa/abstraction**)

Ví dụ, thay vì miêu tả cụ thể sinh viên A, B, C, D ... chúng ta gọi chung là sinh viên:

Đối tượng tổng quát được gọi là **Class** (Lớp), còn mỗi sinh viên cụ thể gọi là **Object** (đối tượng).

Ví dụ:

class SinhVien gồm:

- Thuộc tính (đặc điểm)
+ Tên
+ mã số sinh viên

- Hành vi:
+ ăn/ngủ/nghỉ/ôn thi

=> Để có tính tái sử dụng code cao, OOP chỉ "định nghĩa" lớp tổng quart. còn muốn có chi tiết => chúng ta tự tạo ra

Ví dụ, để tạo ra sinh viên A thì:
+ gán Tên = A
+ gán mã số sinh viên = 1234

Tương tự, tạo ra sinh viên B thì:
+ gán Tên = B
+ gán mã số sinh = 6789
...

Khái niệm:
- **Class** chính là "đối tượng tổng quát" của real-word được mô phỏng
- **Object** là các thực tể chi tiết của class đấy (instance)

#23. Thực hành tạo Class/Object

Khái niệm:

- Class chính là "đối tượng tổng quát" của real-word được mô phỏng
- Object là các thực tể chi tiết của class đấy

Ví dụ về class:

```
public class Student {
	String name;
	int age;

	public void learnJava(){
	  System.out.println("Learn Java with Youtube Hỏi Dân IT");
	}

}
```

Làm sao để tạo ra các đối tượng Student cụ thể:

```
	public static void main(String[] args) {
		Student a = new Student();
		a.age = 20;
		a.name = "Hỏi Dân IT";

		System.out.println("Student a with name = " + a.name + " and age = " +
a.age);
	}
```

Ở block code trên thì:

- Student là tên Class

a là Object, một "đối tượng" chi tiết của class Student (instance) /1 bản thể/ 1 bản sao

a được tạo ra thông qua keyword "new" => có thể tạo ra bao nhiêu đối tượng tùy thích (từ 1 class ban đầu)

a được gán giá trị thông qua "dot.attribute"

#24. Class Attributes (Thuộc tính của Class)

Attributes/Fields là các biến dùng để thể hiện giá trị cho class
=> thường được khai báo bên trong class (đầu hàm)

- Không giới hạn số lượng thuộc tính của 1 class
- Để truy cập/sửa đổi => sử dụng object.attribute

- Tên thuộc tính là danh từ, và viết camelCase

#25. Class Method (Phương thức của class)

1. Định nghĩa method thông thường

Method là cách tượng trưng cho hành động của object.
Tên method thường viết theo camelCase, và là các động từ (chỉ hành động)

Cú pháp:
<return_type> <method_name> ([parameters]) { //code}

<kiểu_trả_về> <tên_method> ([danh_sách_tham_số]) { //code}

ví dụ:

```
void getName(){
//
}

int getTuoi(){

///
}
```

- Với void => return "nothing"

2. Method Overloading (Nạp chồng phương thức)

- Chúng ta có thể định nghĩa các method với tên giống nhau (nhưng khác nhau về tham số)

```
void method( ){ }
void method(int a){ }
void method(int a, String b){ }
```

#26. Định nghĩa hàm tạo (constructor)

1. Constructor

Hàm tạo là một "method đặc biệt" dùng để tạo ra object

- Constructor "phải có tên giống với class", và không thể có "return type"

- Tất cả các class đều cần có "hàm tạo". Nếu bạn không tạo => java sẽ tự động tạo cho bạn (không có tham số đầu vào)

Ví dụ:

```
 public Student (String name, int age){
        this.name = name;
        this.age = age;
 }
```

2. Keyword this

- Dùng để "tham chiếu" tới giá trị của instance/object hiện tại (tương tự con trỏ của C :v)

//tạo mới object với hàm tạo

#27. Access Modifier (private/public/protected) Quyền hạn gì với 1 tham số, hay func
https://www.w3schools.com/java/java_modifiers.asp
https://stackoverflow.com/questions/215497/what-is-the-difference-between-public-protected-package-private-and-private-in

1. Khái niệm package
package === folder
=> được sử dụng để "gom nhóm" file lại với nhau => cho gọn gàng, dễ tái sử dụng

- Keyword import:
+ được sử dụng để import (tái sử dụng lại) các class đã được định nghĩa trong các package

2. Access Modifier
Access Modifier : quyền truy cập sử dụng/sửa đổi

Trong java, có 4 thuộc tính hay dùng nhất:
- **private**: chỉ được sử dụng trong nội bộ class

- **public** : công khai hoàn toàn (100%)
- (default): là public đối với class trong cùng package. là private khi khác package

- **protected**: được sử dụng với lớp cha/con (tính chất kế thừa : sẽ đề cập sau)

=> public và private là 2 thuộc tính được sử dụng nhiều nhất

Mức độ che dấu tăng dần theo mũi tên:
public -> protected -> {default} -> private

3. Ví dụ về sử dụng access modifier
```
package p1;
public class P {
	public int a;
	int b;
	protected int c;
	private int d;
}
```

trường hợp 1: (cùng package) private khác class không truy cập được

```
package p1
public class P1 {

void f() {

P pObject = new P();
}
}
```

trường hợp 2: (khác package) public luôn hoạt động khi khác package
- default:
 + cùng package: ~ public
 + khác package: ~ private

```
package p2
public class P2 {

void f() {

P pObject = new P();
}
}
```

trường hợp 3 (kế thừa):

```
package p3
public class P3 extends P {

void f() {

P1 pObject = new P1();
}
}
```

#28. Encapsulation đóng gói => che dấu datai
Để tăng tính bảo mật cho dữ liệu, bạn cần:
- hạn chế quyền "modify" data
- che dấu data : public/private...

ví dụ:
```
public class Student {
        public String name;
        public int age;
}

public class MyClass{

        main(){

Student a = new Student();
a.age = 1500; //gán sai dữ liệu

}
}
```

1. Sử dụng keyword private
thay vì public => sử dụng private

2. Getter
//

3. Setter
//

#29. Bài tập Lab 04

Mục tiêu:

- Tạo 1 class hoàn chỉnh
- Biết cách sử dụng object (instance từ class đã tạo)

Yêu cầu:

Tạo class **Product** gồm các thuộc tính:

- name: String //tên sản phẩm
- price: double //giá sản phẩm
- tax : double //thuế sản phẩm

Bài 1:

- Tạo class với yêu cầu ở trên, đặt tên là Product.java
- khai báo các thuộc tính: name, price, tax
- Tạo thêm 2 method cho class trên:

+ nhapThongTin(): void (dùng để nhập thông tin object)
+ xuatThongTin(): void (in ra thông tin object đã tạo)
+ getTaxPrice(): double

số tiền thuế = price * tax

Bài 2: Kế thừa bài 1, và bổ sung thêm:

- Trong hàm main, tạo 2 object Products và in thông tin ra màn hình

=> sử dụng constructor

Bài 3: Kế thừa bài tập 2, bổ sung thêm:

+ getter/setter
+ sử dụng keyword private

#30. Chữa bài tập Lab 04

Source code video này:

https://gitlab.com/public-starter-projects1/05-java-core/01-java-starter/-/commit/d42ede24400531e34d96d8666403e7eadcdf6644

Chapter 5: Lab 05
Java ArrayList

#31. Non-primitive(reference) data type
1. Primitive type (Kiểu dữ liệu nguyên thủy)
https://www.w3schools.com/java/java_data_types.asp
https://docs.oracle.com/javase/tutorial/java/nutsandbolts/datatypes.html

Bao gồm: (8 loại)
byte, short, int, long (số nguyên)
float, double (số thực/số thập phân)
char (kí tự)
boolean

Thực tế, **kiểu dữ liệu String không là kiểu dữ liệu nguyên thủy**. cơ mà chúng ta hay dùng nó như là dữ liệu nguyên thủy =))
String (chuỗi)

2. Non-Primitive type (reference data type)
https://stackoverflow.com/questions/29284402/why-do-reference-data-types-point

Kiểu dữ liệu "không nguyên thủy" (kiểu dữ liệu tham chiếu) là các kiểu dữ liệu do "lập trình viên" định nghĩa ra, bao gồm:
Class, Interface, Array (**gọi chung là Object**)
=> Non-Primitive type "viết hoa chữ cái đầu tiên"

#32. Java Autoboxing/Unboxing (Object wrapper class)
Java là OOP (lập trình hướng đối tượng), và đa phần các công cụ hỗ trợ "Object" và không hỗ trợ primitive type
ví dụ: Generic (từ java v5) không hỗ trợ "kiểu dữ liệu nguyên thủy" (hiểu đơn giản là đảm bảo hiệu năng cao)
https://stackoverflow.com/questions/2721546/why-dont-java-generics-support-primitive-types

=> cần có 1 giải pháp để convert "primitive type" => "non-primitive-type"
=> object wrapper class ra đời.

boolean, byte, short, char, int, long, float, double
=> được convert thành:
Boolean, Byte, Short, Character, Integer, Long, Float, Double

Tạo wrapper class:

```
Integer object = new Integer(1);  //lưu ý là có keyword new (sẽ học trong chapter sau)
```

=> convert ngược lại: int val = object.intValue();

1. Auto boxing và unboxing
"Boxing": convert từ primitive value (giá trị nguyên thủy) => chuyển thành object (wrapper class)
"unboxing": là quá trình ngược lại, từ object => giá trị nguyên thủy
=> Công việc Boxing/Unboxing do compiler "tự động làm"
(nên gọi là autoboxing/unboxing :v)

Ví dụ:
https://docs.oracle.com/javase/tutorial/java/data/autoboxing.html

#33. ArrayList là gì ?
Tài liệu: https://www.w3schools.com/java/java_arraylist.asp

Bên cạnh kiểu dữ liệu String, Array là một trong các kiểu dữ liệu được sử dụng nhiều nhất trong lập trình.

Ví dụ về array:
int [ ] myArray = { 1, 2, 9, 6, 10};

Tuy nhiên, nếu sử dụng Array thuần túy, đang tồn tại các nhược điểm sau:
- Số lượng phần tử của Array là cố định (không thể thêm/xóa bớt phần tử)
ở ví dụ trên, myArray có length = 5
- Nếu khai báo Array có nhiều phần tử, mà không dùng hết slot => lãng phí bộ nhớ

=> ArrayList ra đời để khắc phục nhược điểm trên (flexible: thêm/xóa phần tử...)

1. Định nghĩa
- Không khai báo type (hạn chế sử dụng):
ArrayList a = new ArrayList();
a.add("Hỏi Dân IT") ; //String
a.add(26); //int => autoboxing, compiler tự động convert từ int sang Integer

Integer x = (Integer) a.get(1); //cần ép kiểu

- Khai báo type cụ thể:
ArrayList<String> b = new ArrayList<String>();
b.add("Eric")
b.add("Hỏi Dân IT")

String name = b.get(1); //không cần ép kiểu

2. Các method hay dùng

add : thêm phần tử vào cuối
remove: xóa
clear: xóa tất cả
get: truy xuất phần tử tại vị trí
size

#34. Java Generics (Basic)
Tài liệu:
https://docs.oracle.com/javase/tutorial/java/generics/index.html

Lưu ý: Generic là 1 topic rộng, video này chỉ là cơ bản (type parameters). phần còn lại có thể tham khảo ở link trên, và học thông qua thực hành project.

1. Generic là gì ?
Có bao giờ bạn thắc mắc:
ArrayList<Integer> a = ... // <Integer> là cú pháp generic
ArrayList<String> b = ... // <String> là cú pháp generic

Generic là cú pháp được giới thiệu vào v5 của Java, giúp "cải thiện chất lượng" của code.

The following code snippet without generics requires casting:
List list = new ArrayList();
list.add("hello");
String s = (String) list.get(0);

When re-written to use generics, the code does not require casting:
List<String> list = new ArrayList<String>();
list.add("hello");
String s = list.get(0); // no cast

2. Tại sao cần Generic
- Generic giúp "báo lỗi" khi "dịch code " (code gồm 2 giai đoạn: compile then run)
- Generic giúp tái sử dụng code (ví dụ như viết các hàm, class tổng quát)
https://docs.oracle.com/javase/tutorial/java/generics/types.html

3. Java Diamond
được giới thiệu từ Java v7, khi sử dụng constructor với generic, không bắt buộc phải truyền type (vì compiler sẽ tự đống gán/đoán type)
Ví dụ: Box<Integer> box = new Box<>( );
//thay vì: Box<Integer> box = new Box<Integer>( );

#35. Bài tập Lab 05

Mục tiêu:

- Sử dụng ArrayList

Bài 1:

Nhập danh sách số thực với số lượng tùy ý từ bàn phím. Xuất ra danh sách vừa nhập và tính tổng của nó

Gợi ý: sử dụng ArrayList<double> để lưu trữ

```
//nhập số lượng tùy thích
While(true){
        Double x = scanner.nextDouble();
        list.add(x);
        sysout("Nhập thêm (Y/N))?
if(scanner.nextLine().equals("N"){ // why using equals ?
break;
}
}
//tính tổng = vòng lặp for
```

#36. Chữa bài tập lab 05

Link fix bug trong video: https://stackoverflow.com/questions/13102045/scanner-is-skipping-nextline-after-using-next-or-nextfoo

Source code video này:

https://gitlab.com/public-starter-projects1/05-java-core/01-java-starter/-/commit/d2af8df93338b1767a9d22b7aba20ce7c5e8b787

Chapter 6: Lab 06
Java String

#37. String (Chuỗi)
Why String ?
String là loại dữ liệu được dùng nhiều nhất trong ngôn ngữ lập trình.

1. String data type
- String được dùng để lưu trữ "chuỗi ký tự".
Nếu như **char** là lưu trữ "ký tự đơn lẻ", String dùng để lưu trữ nhiều ký tự (chuỗi)

Với java, **String** (chữ S viết hoa) là một "Class" được xây dựng sẵn (non-primitive data type)

Ví dụ: String a = "Hello World";

Một vài ký tự đặc biệt:
https://www.w3schools.com/java/java_strings_specchars.asp

\t : ký tự tab
\r : Về đầu dòng
\n : xuống dòng
\\ : \
\" : " (in ra dấu nháy đôi)

2. Các method thường dùng
String là Object (Class), nên nó sẽ có các method được xây dựng sẵn:
length() : lấy độ dài của chuỗi
trim() : bỏ đi dấu space đầu/cuối của chuỗi
toLowerCase() : viết thường tất cả ký tự
toUpperCase(): viết hoa tất cả ký tự
substring(): cắt chuỗi thành chuỗi con
startsWith bắt đầu bằng chuỗi

#38. Bài tập sử dụng String

Ví dụ 1: Nhập username và password từ bàn phím. Nếu username = "hoidanit" và password > 6 ký tự thì hợp lệ
gợi ý: sử dụng equals để so sánh; length để check chiều dài

Ví dụ 2: Ứng dụng quản lý sinh viên
- Tạo class Student với thuộc tính name, id

- Tại hàm main, tạo 5 students với tên khác nhau (sử dụng constructor)
eg: new Student("Nam", 1); new Student("Eric", 2);

Yêu cầu: tìm và xuất ra:
- các student có tên bắt đầu là "Nguyễn"

#39. Regular Expression

Có bao giờ bạn thắc mắc, khi đăng ký tài khoản Gmail, bạn nhập:
abc@gmail.com => tên hợp lệ
eric@hoidanit.com.vn => hợp lệ

nhưng bạn nhập : tentoila@blabla => tên không hợp lệ ?

Làm sao để máy tính có thể hiểu 1 chuỗi (String), có thỏa mãn 1 điều kiện nào đó hay không ?
Ở ví dụ trên, là kiểm tra tính hợp lệ của email ?

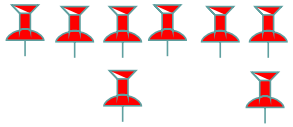
Tài liệu:
https://www.w3schools.com/java/java_regex.asp 

1. Regular Expression là gì ?
Regular Expression (Viết tắt là RegEx : Biểu thức chính quy) là cách tạo ra một chuỗi các ký tự "đặc biệt"
=> **phục vụ mục đích: text search / text replace**
Ví dụ về RegEx: "^[A-Z0-9._%+-]+@[A-Z0-9.-]+\\.[A-Z]{2,6}$"

Java hỗ trợ gói: **java.util.regex** để xử lý regular expression, bao gồm các class chính:
- **Pattern Class**: định nghĩa "hình thức để search" đầu vào
- **Matcher Class**: được sử dụng để search
- **PatternSyntaxException** Class: xử lý exception về lỗi cú pháp

#40. Bài tập Regular Expression

Part 1: Learn
https://regexone.com/



Part 2:

https://stackoverflow.com/a/24304858

Viết các hàm để check String:

- Số CCCD : chỉ gồm ký tự số, chứa 12 ký tự, không chứa ký tự là chữ số hoặc ký tự đặc biệt
Ví dụ hợp lệ: 099145697413
không hợp lệ: 09914569741a (chứa ký tự là chữ số)

- Mật khẩu: có ít nhất 6 ký tự
ví dụ hợp lệ: 123456
không hợp lệ: 12345 (chứa 5 ký tự)

- Email : cần có ký tự @ và dấu chấm (.)
ví dụ hợp lệ: eric@hoidanit.com.vn
không hợp lệ: abc#blabla (không chứa @ và dấu chấm)

https://stackoverflow.com/a/8204716

Chapter 7: Lab 07
Java Inheritance (Subclass and Superclass)

#41. Java Inheritance
Tài liệu: https://www.w3schools.com/java/java_inheritance.asp

Với Java, chúng ta có thể "kế thừa/thừa hưởng" (inherit) thuộc tính và phương thức (attribute & method) từ 1 class khác.

Kế thừa được phân cấp thành:
- subclass (child) : đứa con, sẽ kế thừa/thừa hưởng lại "thành quả" của class khác
- superclass (parent): cha mẹ, sẽ cho đi "tài sản" để class khác kế thừa lại.

1. Nguyên tắc kế thừa
- 1 class chỉ có thể kế thừa "duy nhất" 01 class khác (đơn kế thừa)
(nếu muốn kế thừa nhiều, sử dụng interface - sẽ học sau)

- Để kế thừa, sử dụng keyword "**extends**"
Cú pháp: Subclass extends SuperClass { ... }

Ví dụ:
```
class Vehicle {
  protected String brand = "Ford";       // Vehicle attribute
  public void honk() {                   // Vehicle method
    System.out.println("Tuut, tuut!");
  }
}
class Car extends Vehicle {
  private String modelName = "Mustang";    // Car attribute
  public static void main(String[] args) {
    // Create a myCar object
    Car myCar = new Car();
    // Call the honk() method (from the Vehicle class) on the myCar object
    myCar.honk();
    // Display the value of the brand attribute (from the Vehicle class) and the value of the
modelName from the Car class
    System.out.println(myCar.brand + " " + myCar.modelName);
  }
```

}//Lưu ý về keyword "Protected"

2. Mục đích của kế thừa

Mục đích của thừa kế là "tái sử dụng" code.
(không cần lặp lại code, tăng tính reuse, đồng thời phản ánh "thực tế vào code")

- Lớp con (childClass) có thể sử dụng thuộc tính và phương thức của lớp cha (superClass)

=> lưu ý về access modifier (chỉ kế thừa public/protected, không kế thừa private)

- Lớp con "không kế thừa" hàm tạo của lớp cha (Constrcuctor)

#42. Super Keyword trong Java

//part 1:

Bài toán 1: (Sử dụng hàm tạo không có tham số === không khởi tạo constructor)

Cần tạo 2 class là:

SinhVienIT:

- id //mã số sinh viên
- name //tên
- language //ngôn ngữ lập trình
- price //học phí
- tax //tỉ lệ thuế

Method: getPriceTax = price * tax

SinhVienCoKhi

- id //mã số sinh viên
- name //tên
- skill //kỹ năng
- price //học phí
- tax //tỉ lệ thuế

Các bước cần làm:

Bước 1: xác định điểm chung/riêng

subClass:

SinhVienIT, SinhVienCoKhi

superClass: SinhVien

=> định nghĩa lại:

SinhVienIT:

- language

SinhVienCoKhi:

- Skill

SinhVien:

- id //mã số sinh viên
- name //tên
- price //học phí
- tax //tỉ lệ thuế

Method: getPriceTax = price * tax

1. Sử dụng keyword super

Sử dụng keyword super để truy cập tới thuộc tính và method của cha
ví dụ: function: getPriceTax

2. Method Overriding

Vấn đề: sử dụng this.getPriceTax hay super.getPriceTax kết quả đều như nhau
Nếu class child cũng có method getPriceTax ???

- Đây là cách ghi đè method
- Tên method (số lượng tham số) giống hệt như giữa child/parent

Lưu ý: phân biệt method overloading và method overriding ghi đè lại

overloading: định nghĩa nhiều method có tên giống nhau nhưng tham số truyền vào khác nhau

//part 2

Bài toán 2: (Sử dụng hàm tạo có tham số)
Kế thừa bài tập 1

- Yêu cầu: khởi tạo đối tượng SinhVienIT, SinhVienCoKhi thông qua hàm tạo

=> keyword super cần gọi đầu tiên (bên trong constructor)

#43. Abstract Class (Basic)
Tài liệu:
https://docs.oracle.com/javase/tutorial/java/IandI/abstract.html

Đôi khi, chúng ta muốn có sự ràng buộc chặt chẽ giữa 'cha/con', ví dụ: tất cả con kế thừa, cần phải có 1 phương thức
ví dụ: bắt buộc 2 sv ghi đè lại tinhDiem
sinhVienIT => method TinhDiem()
sinhVienCoKhi => method TinhDiem()

=> abstract là cách chúng ta định nghĩa lớp trừu tượng

#44. Polymorphism (Tính đa hình)
Tài liệu: https://www.javatpoint.com/runtime-polymorphism-in-java
https://docs.oracle.com/javase/tutorial/java/IandI/polymorphism.html

1. Upcasting vs downcasting in java
//todo

2. Polymorphism (many-form)
//todo

#45. Bài tập Lab 07

Kế thừa video #42

Tạo 3 class:

- **SinhVien** (super class)

+ getDiem (phương thức trừu tượng)

- **SinhVienIT** (child class)

+ scoreJava : double

+ scoreHTML : double

getDiem = (scoreJava * 2 + scoreHTML)/3;

- **SinhVienCoKhi** (child class)

+ scoreCNC : double

+ scorePLC : double

getDiem = (scoreCNC + scorePLC)/2;

#46. Chữa bài tập Lab 07

//todo

Source code video này: https://gitlab.com/public-starter-projects1/05-java-core/01-java-starter/-/commit/ef6ca528d6542e9f612a0007bf6262b137a1b197

Chapter 8: Lab 8
Java Interface
#47. Interface
Tài liệu:
https://www.w3schools.com/java/java_interface.asp

1. Interface là gì ?
- Interface là một cách khác để thực hiện tính "kế thừa/Inherit" trong java (bên cạnh abstract class)
- Là một tập hợp của các methods (không có body)
(không đề cập tới default method => đề cập tại video sau)

Ví dụ về interface:
```
// interface
interface Animal {
  public void animalSound( ); // interface method (does not have a body)
  public void run( ); // interface method (does not have a body)
}
```

Để truy cập/sử dụng 1 interface, chúng ta sử dụng keyword "**Implements**" (thay vì **extends**), và class sử dụng interface bắt buộc phải ghi đè các methods có trong interface (ngoại trừ default method)

ví dụ:
```
// Interface
interface Animal {
  public void animalSound( ); // interface method (does not have a body)
  public void sleep( ); // interface method (does not have a body)
}

// Pig "implements" the Animal interface
class Pig implements Animal {
  public void animalSound( ) {
    // The body of animalSound( ) is provided here
    System.out.println("The pig says: wee wee");
  }
  public void sleep( ) {
    // The body of sleep( ) is provided here
    System.out.println("Zzz"); } }
```

```
class Main {
  public static void main(String[] args) {
    Pig myPig = new Pig( );  // Create a Pig object
    myPig.animalSound( );
    myPig.sleep( );
  }
}
```

2. Đặc điểm của interface
- Interface không có body => body được ghi đè lại tại các class "implements"
- Khi kế thừa interface, cần phải ghi đè tất cả method của interface (ngoại trừ default method)
- Mặc định, tất cả method của interface là "abstract" + "public"
=> thay vì viết: abstract public void animalSound( );
chuyển thành:
void animalSound( )

- Interface không thể có constructor
=> không thể khởi tạo object từ interface (ví dụ là không thể tao Animal object)

- 1 class có thể kế thừa cùng lúc nhiều interface (abstract class là kế thừa 1:1)

3. Kế thừa nhiều interface
1 class có thể kế thừa nhiều interface
https://stackoverflow.com/questions/2801878/implementing-two-interfaces-in-a-class-with-same-method-which-interface-method

#48. Interface với Java 8

1. Điểm mới trong Java 8

thằng con không cần phải ghi đè lại

Có rất nhiều update trong java 8 => chỉ đề cập default method

https://docs.oracle.com/javase/tutorial/java/IandI/defaultmethods.html

https://www.baeldung.com/java-static-default-methods

- static: giúp không cần tạo mới 1 đối tượng, cô động trong interface mà không phải tạo thêm file mới

2. So sánh Interface và Abstract Class

https://docs.oracle.com/javase/tutorial/java/IandI/abstract.html

Type vs interface

- Nên dùng interface khi nào ko dùng được interface thì dùng Abstract

- Abs: nếu muốn nhóm code bởi các class lquan tới nhau: vd SV, svit, svck | đ/n có nhiều thuộc tính protected, private)

- Inter: có thể dùng ở vơi bất kỳ clas nào hết, ko pbiet bố con tk nào thik là kế thừa thôi (ko có mối quan hệ họ hàng) | đa kế thừa | static và default

#49. Java Package

1. Package là gì ?

Hiểu đơn giản, package === folder => nơi chứa code
- Package được tạo nên bởi tập hợp: class/interface
- Package có thể chứa sub-package (cha/con)

2. Truy cập package

- Các class mà dự định sẽ được sử dụng ngoài package => định danh public (private/default/protected/public)

- Các package khác nhau có thể khai báo các class "trùng tên"

3. Sử dụng packages

Cú pháp:
import package_name.class_name;

ví dụ:
import java.util.Scanner;
package: java, sub-package: util , class: Scanner

import java.util.* ;
=> import tất cả class có trong package java/util

#50. Bài tập Lab 08

- Mục tiêu: làm quen & sử dụng interface

Bonus: hàm toString( )
Java to String method: https://stackoverflow.com/a/29140403

Bài tập:

- Tạo interface IPerson với 2 method:
  + void input( )
  + void display( )

- Tạo class Person kế thừa interface ở trên, và có thêm:
  protected String id;
  protected String name;
  protected int age;
  + viết constructor, setter, getter

- Tạo class Student kế thừa class Person, và có thêm:
  private int mark;
  private String grade;
  + viết getter cho mark, grade
  + viết setter cho mark
  + viết setter cho grade theo công thức:trong constructor không cần khởi tạo grade
  nếu mark >=8 thì grade = "A"
  nếu mark >=7 thì grade = "B"
  nếu mark >=6 thì grade = "C"
  nếu mark >=5 thì grade = "D"
  nếu mark <5 thì grade = "Tạch cmnr"

#51. Chữa bài tập Lab 08

Source code video này:
https://gitlab.com/public-starter-projects1/05-java-core/01-java-starter/-/commit/7ecb426d51a1afd61d4fac7a6cca94ac41b1fc02

Chapter 9: Lab 09
Exception (Ngoại lệ)

#52. Try...catch
Tài liệu: https://docs.oracle.com/javase/tutorial/essential/exceptions/definition.html

Bài toán:
Cho user nhập input từ bàn phím, và thực hiện phép chia : 10/x (với x nhập từ bàn phím)

Nếu nhập vào "ký tự" (ví dụ a,b, c...) hoặc nhập vào số 0, chương trình sẽ báo lỗi và "dừng lại".

Làm sao để:
- khi nhập vào ký tự ko hợp lệ, chương trình báo lỗi và "tiếp tục chạy"
- User tiếp tục nhập input cho tới khi hợp lệ

1. Exception (ngoại lệ) là gì ?
chương trình khi được chạy gồm 2 bước:
1 là compile (từ code java -> byte code). nếu có lỗi, thì chương trình báo ngay (ko đề cập tới lỗi này)
2 là run (từ bytecode -> JVM run). nếu có lỗi, chương trình ngỏm luôn

=> Ngoại lệ là các lỗi xảy ra sau khi "run chương trình" và khiến chương trình của chúng ta "stop" => ngỏm

Phân loại exceptions:
https://www.geeksforgeeks.org/exceptions-in-java/

=> Exception là cách chúng ta handle các exception "có thể biết trước được " (checked exception)

2. Try.. catch để xử lý ngoại lệ
Cú pháp:

```
Try {
//your code
}catch(...){
```

```
//handle exception
}

//ví dụ chia cho 0
//todo

//dùng try với nhiều catch
try {
        int b = 0;
        int a = 6/0; //exception 1: ArithmeticException

        int c[] = {9, 6};
        c[2] = 10; //exception 2: ArrayIndexOutOfBoundsExceptions

}catch(ArithmeticException e){
//todo
}catch(ArrayIndexOutOfBoundsExceptions e){
//todo

}
```

#53. Keyword finally
Tài liệu: https://docs.oracle.com/javase/tutorial/essential/exceptions/finally.html

Lưu ý: **finally** dùng với try/catch trong handle exception, không phải là keyword final (constant: hằng số)

1. Sử dụng trong handle Exception
cú pháp:
```
try{
//code
}catch(e) {
//code
}finally {
//code
}
```

=> Finally (settle down/onSettled) : dù có ngoại lệ hay không, thì hàm finally luôn được chạy
(cho dù có sử dụng: return, continue, or break ở đâu đi nữa)

2. Try-with-resources
Tài liệu:
https://docs.oracle.com/javase/tutorial/essential/exceptions/tryResourceClose.html

Bắt đầu từ java v1.7 (version 7), thay vì dùng finally, chúng ta có thể dùng trực tiếp với try.
//ví dụ về scanner

#54. Keyword Throws và Throw

1. Throws (số nhiều)
https://docs.oracle.com/javase/tutorial/essential/exceptions/declaring.html

Ngoài việc xử lý ngoại lệ trực tiếp try/catch, chúng ta có thể dùng Throws để ném ra ngoại lệ
=> Có tác dụng rất nhiều khi viết functions/class tổng quát (để re-use) vì chúng ta không biết nên làm gì với try/catch

Ví dụ: https://stackoverflow.com/questions/18491020/what-does-throws-do-and-how-is-it-helpful

2. Throw (số ít)
Đôi khi muốn ném lỗi trực tiếp (không dùng try/catch)
ví dụ:

```
public double divide(double a, double b) {
   if (b == 0) {
      throw new ArithmeticException("Divider cannot be equal to zero!");
   }
   return a / b;
}
```

#55. Bài tập Lab 09

Tạo class SinhVien bao gồm:
masv: String
hoten: String
diem: double
age: int

- Viết setter, getter, constructor, toString

- Nhập dữ liệu sinh viên từ bàn phím và in ra màn hình (với hàm toString)
- Xử lý ngoại lệ cho các trường hợp
+ 0< điểm < 10. điểm là số thực
+ 18 < age < 100. tuổi là số nguyên
=> nếu có ngoại lệ thì thông báo lỗi và yêu cầu nhập lại

#56. Chữa bài tập Lab 09

Source code video này:
https://gitlab.com/public-starter-projects1/05-java-core/01-java-starter/-/commit/139499a14b2b2c11f89c6b779460a096752bd84b

Chapter 10: Lab 10
Xử lý Input/Output với Java

#57. Tổng quan về Java IO (Stream)
Nguyên tắc không đổi : máy tính chỉ hiểu được ký tự 0 và 1
(không có khái niệm image/video/audio/files...)

Những khái niệm images/video/audio/files... là dành cho con người

Vậy làm sao để human và computer có thể giao tiếp với nhau ?

Ví dụ:
human: cung cấp 1 file video (input)
computer: chạy và hiển thị video đấy (output)

=> **Bước 1**: Input ban đầu (raw) cần được biến đổi thành định dạng 0 và 1 (**dạng A**) => máy tính hiểu (READ)
Bước 2: dạng A (tại bước 1) cần biến đổi thành d**ạng B**(images/video/audio/files...)=> Output con người hiểu (WRITE)

Để có thể thực hiện hiệu quả quá trình trên, chúng ta sử dụng "Stream"

Tại sao gọi là Streaming (Livestream) mà không sử dụng tên gọi khác ?

Stream (dòng nước), là cách chúng ta chia data thành các đoạn/khúc (chunk) để xử lý

Ví dụ: 10101010101010000011111100001111000000.....
Thay vì xử lý "tất cả data" cùng 1 lúc, chúng ta chia nhỏ ra thành các chunk:
chunk 1: 101010101
chunk 2: 000111100
chunk n: ....

Ưu điểm:
- Tiết kiệm tài nguyên (memory/cpu). File càng lớn, càng tốn nhiều tài nguyên
- Giảm thiểu thời gian loading qua network (Livestream chất lượng càng cao, load càng chậm nếu mạng yếu ...)

Nhược điểm: sử dụng nhiều thao tác IO (đọc ghi nhiều lần :v)

#58. Phân loại Stream với Java IO

Nhắc lại khái niệm về stream:
https://docs.oracle.com/javase/tutorial/essential/io/streams.html

1. Byte Streams vs Character Streams
Tài liệu: https://docs.oracle.com/javase/tutorial/essential/io/bytestreams.html

Tham khảo:
https://stackoverflow.com/questions/3013996/byte-stream-and-character-stream

2. Buffered Stream
https://docs.oracle.com/javase/tutorial/essential/io/buffers.html

3. Data Streams và Object Streams (Giới thiệu)
//todo

#59. Ví dụ về Read/Write File

```
		FileReader inputStream = null;
		FileWriter outputStream = null;
		System.out.println(System.getProperty("user.dir"));
		try {
//			inputStream = new FileReader(
//					"C:/Users/tuan.pv/Documents/workspace-spring-tool-
suite-4-4.20.0.RELEASE/hoidanit/hoidanit");
			inputStream = new FileReader("hoidanit");

			outputStream = new FileWriter("characteroutput.txt");

			int c;
			while ((c = inputStream.read()) != -1) {
				outputStream.write(c);
			}
		} finally {
			if (inputStream != null) {
				inputStream.close();
			}
			if (outputStream != null) {
				outputStream.close();
			}
		}
```

#60. Path Class
Tài liệu: https://docs.oracle.com/javase/tutorial/essential/io/path.html

Relative Path

Absolute Path

Chapter 11: Lab 11
Xử lý Đa luồng với Java

#61. Processes and Threads
Tài liệu:
https://docs.oracle.com/javase/tutorial/essential/concurrency/procthread.html

Các khái niệm khi "chạy chương trình" với máy tính (môn hệ điều hành @@)

- **Application**: chương trình chạy trên máy tính
ví dụ: spring tool suite, google chrome...

- **Task**: các nhiệm vụ cần thực hiện
multi-tasking: làm nhiệm vụ cùng lúc
ví dụ: task 'learn java'
trong 1 lần sử dụng máy tính, bạn sử dụng song song đồng thời: spring tool suite + google chrome (mở cùng lúc)

- **Processes** (tiến trình)
Để chạy 1 applications, cần 1 (hoặc nhiều) processes.

Đặc điểm: Mỗi process "có không gian riêng" để chạy => không chia sẻ memory

Ví dụ với Google Chrome
1 tab = 1 process . Kill 1 process với Task manager => kill tab

- **Threads** (luồng)
1 Process cần ít nhất là 1 thread (main thread)
=> process chứa (bao gồm) thread

Đặc điểm: Các thread chia sẻ nguồn tài nguyên

Hiểu đơn giản:
- Thread là đơn vị nhỏ nhất để thực thi code
- Thread giúp thực hiện các "nhiệm vụ" của chương trình, ví dụ: in ra màn hình, xuất file...

- Multi-thread là cách thực thi nhiều thread song song

#62. Thread Objects

Tài liệu: https://docs.oracle.com/javase/tutorial/essential/concurrency/threads.html

```
public class HelloWorld extends Thread {

	@Override
	public void run() {
		System.out.println(" start thread");
		long sum = 0L;
		for (long i = 0L; i < 10000000000L; i++) {
			sum += i;
		}
		System.out.println(" end thread");
		// TODO Auto-generated method stub

	}

	public static void main(String[] args) throws IOException {
		int MAX = 2;
		for (int i = 0; i < MAX; i++) {
			(new HelloWorld()).start();
		}

		System.out.println("finish");
	}

}
```

#63. What's next ? Tổng kết các kiến thức đã học

1. Tổng kết các kiến thức đã học

- Kiến thức nền tảng của java:
+ **Variables**:
    - Khai báo biến, data type (dữ liệu nguyên thủy/không nguyên thủy - object)
    - Kiểu dữ liệu String
    - Vòng lặp (loop)
    - Array
    - Auto boxing (wrapper class)

+ **Lập trình hướng đối tượng** : Class/Object
    - Khai báo Class
    - Tạo đối tượng
    - Access modifier: public, private, protected
    - Encapsulation: getter/setter (tính che dấu bảo vệ dữ liệu)
+ **Kế thừa**:
    - extends (class)
    - implements (interface)
    - polymorphism (tính đa hình)

+ **Exception**: xử lý ngoại lệ
    - try/catch
    - finally
+ **Java I/O**
+ **Multi Threading**

2. What's next

- Về kiến thức java:
+ Collections: Lists, Maps, Stacks, Queues

- Về cú pháp java:
+ Streams APIs & Functional Programming (lamda)

- Về project thực hành:
+ Java Swing/Java FX
+ Spring Framework

Lời Kết

Như vậy là chúng ta đã cùng nhau trải qua hơn 60+ videos về học kiến thức cơ bản & cốt lõi của Java.

Tất cả các kiến thức mình chia sẻ, đều được lấy từ kinh nghiệm đi làm của mình và các trang tài liệu có liên quan của Java.

Dĩ nhiên rằng, trong quá trình quá trình thực hiện khóa học này, mình sẽ không thể tránh khỏi những sai sót.

Vì vậy, nếu thấy sai sót, các bạn cứ thoải mái đóng góp qua Fanpage Hỏi Dân IT nhé.
https://www.facebook.com/askITwithERIC

Hoặc comment dưới video :v

Hẹn gặp lại các bạn ở các khóa học tiếp theo ….
Hỏi Dân IT (Eric)