**19/04/2024:**

* **Ngôn ngữ lập trình là gì ?**
* ***Ngôn ngữ lập trình*** là một ngôn ngữ được xây dựng bằng tập từ khóa và kí hiệu để người lập trình viết mã lệnh (code) giao cho máy tính xử lý công việc của người lập trình (developer, coder,…)
* **Hệ số trong máy tính:**
* Hệ nhị phân: Là hệ số sử dụng bit 0 và 1 để biểu diễn thông tin
* Đơn vị trong máy tính:
* BIT là đơn vị nhỏ nhất trong máy tính, sử dụng số 0, 1
* 0: Trạng thái đèn tắt
* 1: Trạng thái đèn sáng
* 1 Byte = 8 BIT
* 1 KB = 1024 Byte
* 1 MB = 1024 KB
* 1 GB = 1024 MB
* 1 TB = 1024 GB
* **Chuyển đổi từ số thập phân sang nhị phân:**
* Lấy số thập phân chia liên tiếp cho 2
* Mỗi lần chia thì ghi số dư của phép chia
* Khi thương bằng 0 thì dừng lại sau đó viết các số dư theo chiều từ dưới lên trên thì được số nhị phân cần tìm
* Ví dụ:
* 12 => Nhị phân
* 12/2 = 6 dư 0
* 6/2 = 3 dư 0
* 3/2 = 1 dư 1
* 1/2 = 0 dư 1

=> Số nhị phân là: 1100

* 19 => Nhị phân
* 19/2 = 9 dư 1
* 9/2 = 4 dư 1
* 4/2 = 2 dư 0
* 2/2 = 1 dư 0
* 1/2 = 0 dư 1

=> Số nhị phân 10011

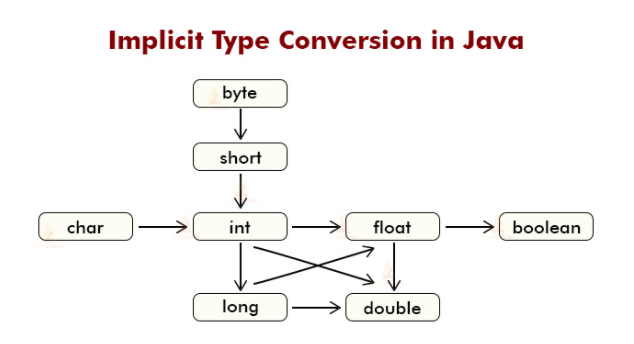
* **Chuyển từ số nhị phân về thập phân:**
* Lấy hệ số (1, 0) nhân với 2^n - 1 trong n là số bit tính đến hệ số đang xét
* 10011 => Thập phân ???
* 1\*2^4 + 0\*2^3 + 0\*2^2 + 1\*2^1 + 1\*2^0 = 16 + 2 + 1 = 19
* **Bảng mã ASCII:**Bảng mã theo tiêu chuẩn của Mỹ
* Trong đó quy định 1 kí tự tương ứng với số bao nhiêu
* Ví dụ:
* A: 65
* 65/2 = 32 dư 1
* 32/2 = 16 dư 0
* 16/2 = 8 dư 0
* 8/2 = 4 dư 0
* 4/2 = 2 dư 0
* 2/2 = 1 dư 0
* 1/2 = 0 dư 1

=> Số nhị phân: 01000001

* B: 66
* …
* **Khai báo biến trong lập trình Java:**
* Là việc người lập trình chỉ ra cho máy tính cần cấp phát 1 vùng nhớ để lưu trữ thông tin phục vụ công việc của người lập trình như nhập thông tin, in thông tin, tính toán, so sánh,…
* ***Cú pháp khai báo:***
* [Kiểu dữ liệu] <Tên biến> = [Giá trị khai báo hoặc mặc định ban đầu];
* Ví dụ: Khai báo 1 biến kiểu số nguyên để chứa tuổi của sinh viên
* Khai báo biến
  + - * + int tuoiSV = 20;
* Gán giá trị mới cho biến
  + - * + [Tên biến] = [Giá trị mới cần gán];
        + tuoiSV = 21;
        + tuoiSV = 22;
* **Quy tắc đặt tên biến trong Java:**
* Tên biến là 1 chuỗi kí tự liên tiếp, không khoảng cách, không tiếng việt có dấu và không chứa kí tự đặc biệt, ví dụ: tuoiSV, soLuong, tenCongTy,…
* Tên biến bắt đầu bằng kí tự chữ hoặc dấu \_, ví dụ : 123Abc là sai, 12 = 13; \_abc,abc,…là đúng
* Tên biến phân biệt hoa thường, ví dụ: Abc khác abc
* Tên biến không được đặt trùng với từ khóa, ví dụ tên biến: if, break, switch, for, while,…là sai vì trùng với từ khóa
* Tên biến, tên hàm là gồm nhiều từ trong đó chữ đầu viết thường và các chữ sau thì kí tự đầu tiên viết hoa, ví dụ: tenCongTy, soLuongCay, soConGa, luongCoBan,….(Java Convention)
* **Kiểu dữ liệu trong lập trình Java:**
* Khác nhau ở kích thước cấp phát vùng nhớ
* ***Byte:***
* 1 byte để chứa thông tin = 8 bit
* Trong 8 bit:
  + - * + 1 bit chứa dấu
        + 7 bit chứa giá trị
        + Miền giá trị: -2^7 => 2^7 - 1 => -128 => 127
* ***Short:***
* 2 byte để chứa thông tin = 16 bit
* Trong 16 bit:
  + - * + 1 bit chứa dấu (-)
        + 15 bit chứa giá trị
        + Miền giá trị: -2^15 => 2^15 - 1 => -32768 => 32767

**20/4/2024:**

* **Đặt tên cho Package trong Java:** Không được đặt trùng với tên từ khóa của package, ví dụ java.abc
* **Khai báo biến:**
* int soLuong = 100;
* System.out.printf("Số lượng là: %d", soLuong); với %d (kiểu số) được thay bằng giá trị của tên biến “soLuong”;
* **Khai chuỗi:**
* String tenCongTy = “Stanford – Dạy lập trình”;
* System.out.printf("Số lượng là: %s", tenCongTy); với %s (kiểu chuỗi, kiểu kí tự) được thay bằng giá trị của tên biến “soLuong”;
* **Quy tắc chuyển đổi kiểu dữ liệu trong Java:**



**24/4/2024:**

* **Cấu trúc điều kiện if else trong Java:**
* Nếu biểu thức điều kiện đúng (true) thì thực hiện trong if, ngược lại nếu biểu thức sai (false) và có else thì thực hiện trong else;
* Ví dụ:
* if(Đèn xanh)//true

{

Được phép đi

}

* else

{

Dừng xe

}

**05/5/2024:**

* **Declare an object to retrieve information from keyboard:**
* Scanner sc = new Scanner(System.in);
* **Declare a constant:**
* final double PI = 3.14;
* **Declare variables for a character**
* Char kiTu;
* Enter a character: kiTu = sc.Next().charAt(0);
* **Convert Char to Int (ASCII):**
* (int)kiTu
* **Convert String to Int (from character):**
* String buffer = "abc";
* char ch = buffer.charAt(0/1/2); (0=a), (1=b), (2=c) (only choose one)
* System.out.println((int)ch);
* **Convert String to Int, Double:**
* String strTuoi = ""; (String must be entered as digits, not characters)
* int tuoi = 0;
* double tuoi = 0;
* tuoi = Integer.parseInt(strTuoi);
* tuoi = Double.parseDouble(strTuoi);
* **Convert String to Int, Double (other method):**
* String strTuoi = "";
* tuoi = Integer.parseInt(strTuoi,16/8); (convert from String (16=hexa,8=octal) to integer tuoi (decimal))
* **Method to catch error, use:**
* Try {Content needed to be entered correctly}
* Catch (NumberFormatException ex)

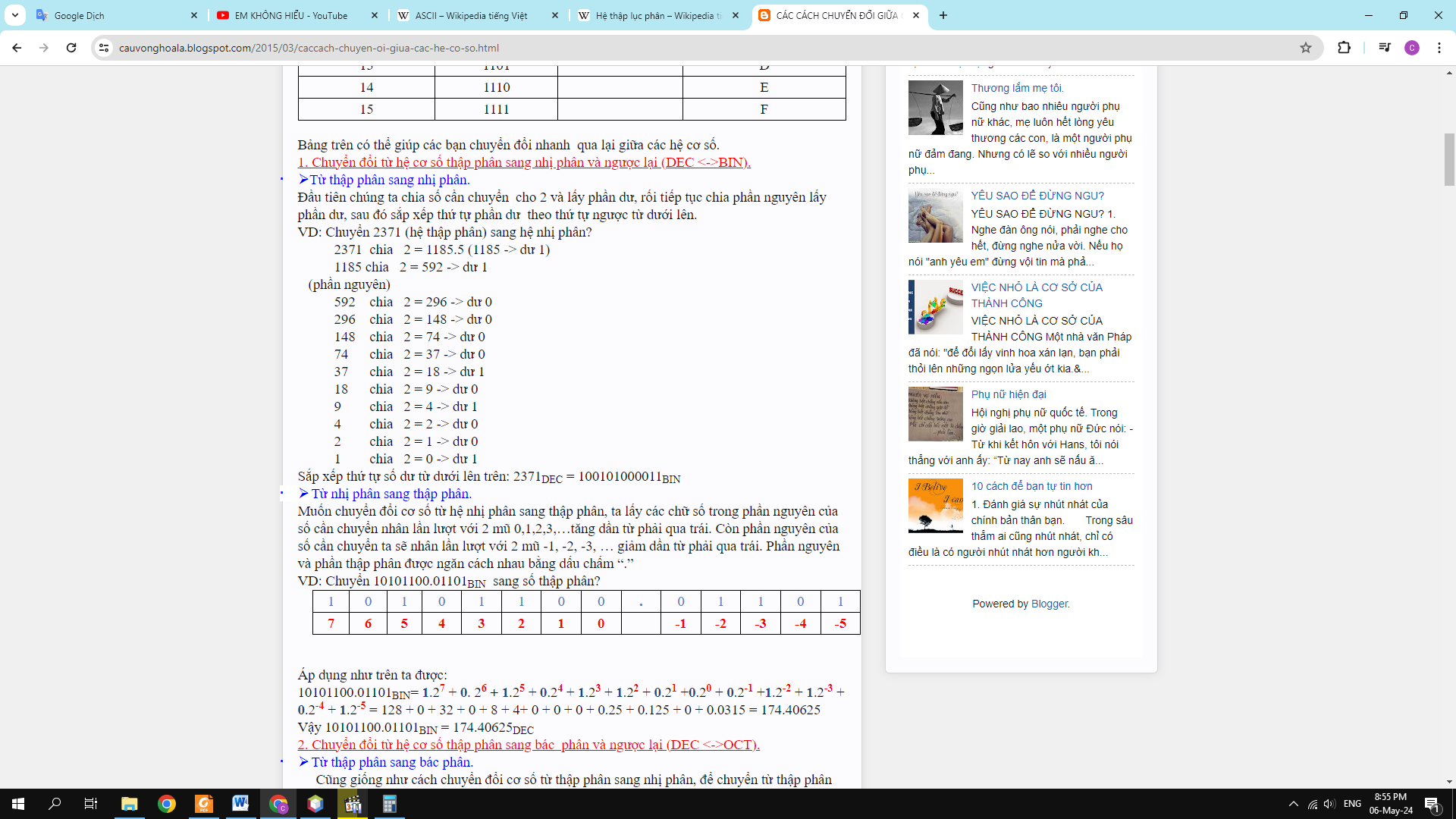
{If the content entered is incorrect, catch will operate)

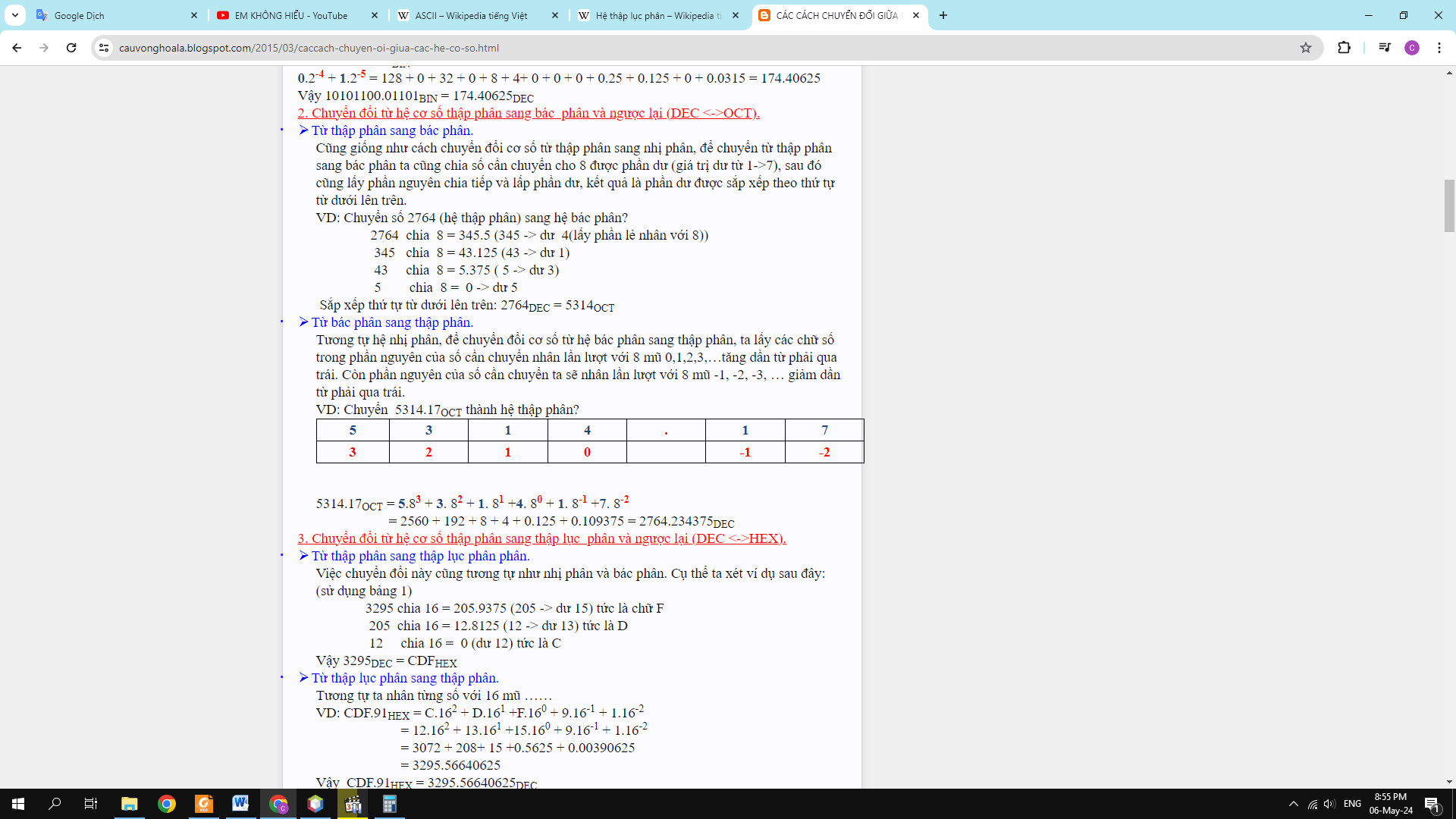
* **Covnert Int to String (2: Binary, 08: Octal, 10: Decimal, 16: Hexa …)**
* int i = 32;
* String a = "";
* a = Integer.toString(i,2);

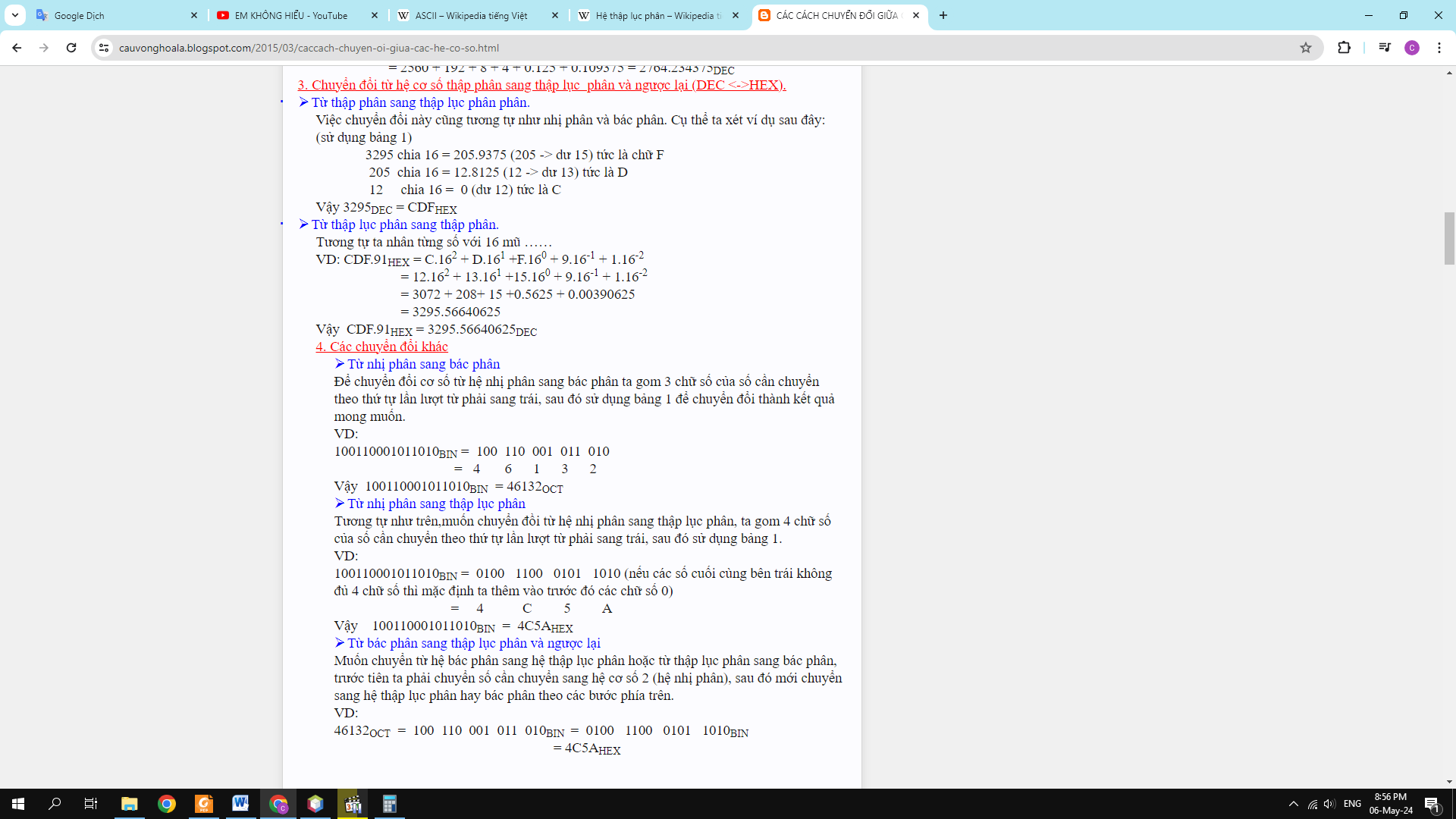
**06/5/2024:**

* **[txtMatKhau.getPassword()] will return an array, so use new String() to convert an array to String.**
* //Declare variables
* String matKhau = "";
* Take user information entered on the interface
* matKhau = new String (txtMatKhau.getPassword());
* **[getSelectedItem()] will return an object, so use** "" **+ cboPhepTinh.getSelectedItem() to convert an object to String.**
* //Declare variables
* String phepTinh = "";
* //Take user information entered from keyboard
* phepTinh = "" + cboPhepTinh.getSelectedItem();
* **Convert Decimal to Octal, Hexa, Binary …**







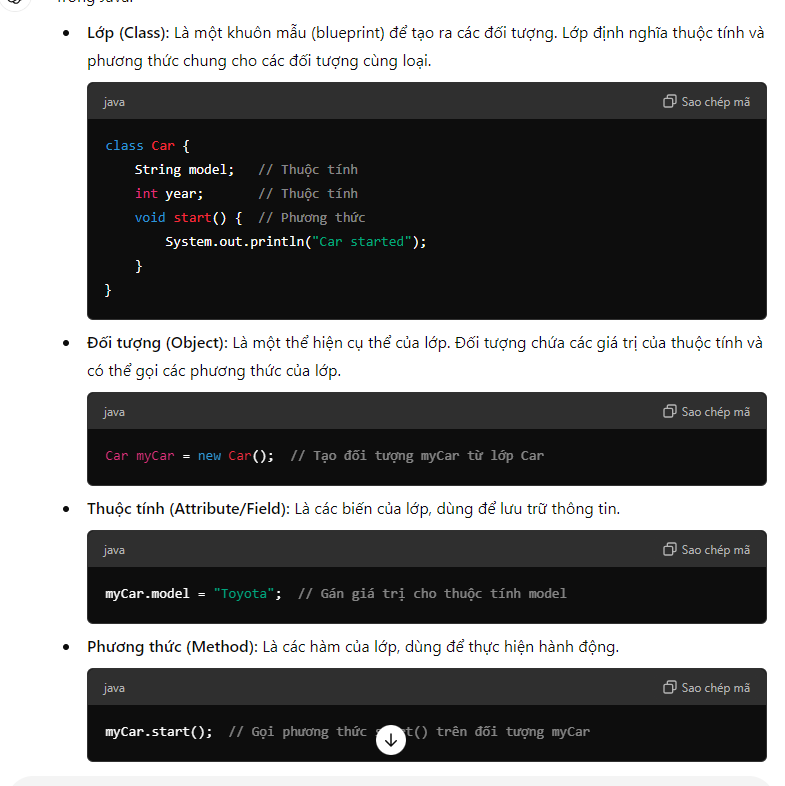


**10/5/2024:**

* **Mảng trong lập trình Java là gì ?**
* Mảng là tập hợp các phần tử có cùng kiểu dữ liệu.
* Đặc điểm mảng:
* Gồm tập hợp các phần tử cùng kiểu dữ liệu.
* Khai báo 1 biến mảng có thể chứa được nhiều giá trị.
* Các phần tử trong mảng được cấp phát liên tiếp trong bộ nhớ máy tính.
* Để xác định được giá trị của phần tử tại vị trí thứ i trong mảng a ta sử dụng cú pháp a[i]. Trong đó i gọi là chỉ số (index) và bắt đầu bằng số 0.

**29/5/2024:**

* **Lập trình hướng đối tượng: Là lập trình hướng đến 1 đối tượng để phân tích về**
* **Thuộc tính** chính là những thông tin, đặc điểm của đối tượng. Ví dụ: Con người có các đặc điểm như mắt, mũi, tay, chân,…
* **Phương thức (hàm)** là những thao tác, hành động mà đối tượng đó có thể thực hiện. Ví dụ: Một người có thể thực hiện hành động nói, đi, ăn, uống, …



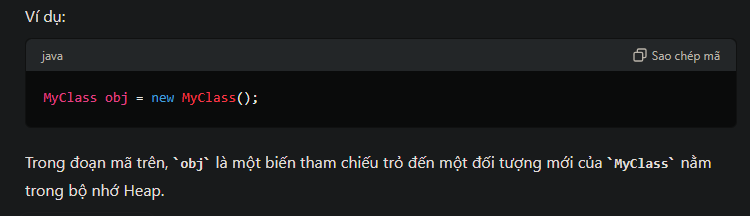
* **Tại sao phải tạo Data Provider:**
* **Tạo Data Provider về** ý nghĩa tương tự như tạo đối tượng để sử dụng các thuộc tính và phương thức trong hàm.
* **Điểm khác biệt của Data Provider** với đối tượng tạo bằng (new) là Data Provider có thể mang giá trị (được khởi tạo, hoặc đã được chỉnh sửa …) đi xuyên suốt chương trình. Đối tượng tạo bằng (new) thì không.

**24/6/2024:**

* **Kế thừa là gì:**
* **Một lớp thừa hưởng đặc tính từ lớp khác được gọi là kế thừa.**
* **Mục đích của kế thừa:**
* Lớp con (child class) có thể tái sử dụng những cái đã có trong lớp cha (class parent) .
* Lớp con (lớp kế thừa) có thể mở rộng thêm đặc tính mà không ảnh hưởng đến lớp cha (lớp được kế thừa).
* **Overload (nạp chồng) – Đa hình trong quá trình biên dịch (Runtime):**
* **Là các phương thức nằm trong cùng 01 lớp, có cùng tên với nhau nhưng có đối số/giá trị truyền vào khác nhau.**
* **Override (ghi đè) – Đa hình trong quá trình thực thi (Compile):**
* **Khi định nghĩa 01 phương thức ở lớp cơ sở (lớp cha), thì mình có thể viết lại phương thức đó ở lớp con thông qua override.**
* **Khi đối tượng thuộc lớp con thì sẽ chọn lựa và chạy theo phương thức ở lớp con.**
* **Nếu lớp con không có phương thức đó thì mới chạy theo phương thức ở lớp cha.**
* **Polymorphism (sự đa hình): Là khả năng cho phép một hành vi (method) có thể được thể hiện dưới nhiều hình thức khác nhau mà không quan tâm tới chi tiết.**
* **Đa hình (polymorphism) trong lập trình Java là khả năng một đối tượng có thể được xử lý như thể nó là một đối tượng của nhiều kiểu khác nhau. Có hai loại đa hình chính:**
* **Đa hình thời gian biên dịch (Compile-time polymorphism):** Còn gọi là đa hình tĩnh, đạt được thông qua nạp chồng phương thức (method overloading).
* **Đa hình thời gian chạy (Runtime polymorphism):** Còn gọi là đa hình động, đạt được thông qua ghi đè phương thức (method overriding).
* **Abstract – Làm nền cho lớp con để kế thừa:**
* **Lớp trừu tượng (lớp ảo không có thực), và phương thức abstract sử dụng để khai báo định danh, không có nội dung xử lý.**
* **Trong 01 lớp chỉ cần có 01 phương thức abstract thì bắt buộc lớp đó phải là lớp abstract.**
* **01 phương thức được khai báo abstract từ lớp superclass (lớp cha) bắt buộc phải được @override từ lớp con.**
* **Không thể sử dụng đối tượng thuộc lớp abstract ra hàm main để chạy (vì nó là lớp ảo).**
* **Interface – Hiểu nôm na là bản thiết kế của ngôi nhà:**
* **Khai báo những gì chúng ta cần sử dụng, sau đó triển khai theo đúng những hạng mục mà chúng ta đã khai báo trong interface (như 1 bảng kế hoạch).**
* **Khai báo tất cả những phương thức trong interface đều phải là public, nếu không khai báo public thì vẫn mặc định là public.**
* **Chỉ khai báo định danh, không có phần body.**
* **Các hàm implements (hàm triển khai từ lớp interface) bắt buộc phải @override lại tất cả các phương thức trong lớp interface, chỉ cần có định danh phương thức, không quan tâm nội dung bên trong (tương tự abstract) – Kiểu như nhìn vào 01 bản thiết kế chính (lớp interface), và lớp implements từ lớp interface sẽ làm đúng từng hạng mục trong bản thiết kế chính đó.**
* **Sự giống nhau và khác nhau giữa abstract và interface:**
* **Giống nhau:**
* Đều phải @override lại tất cả các phương thức trong lớp cha.
* **Khác nhau:**
* Interface có thể tạo ra 01 lớp được triển khai từ nhiều interface khác nhau (đa).
* Abstract thì dùng những đối tượng/lớp có liên quan đến nhau, và tạo ra 01 lớp chỉ dùng được từ 1 abstract (đơn).
* **JDBC (hay còn gọi là API trong lập trình Web): Là tập những cái lớp/thư viện giúp chúng ta làm việc với hệ cơ sở quản trị dữ liệu.**
* **Project:**
* **Ưu điểm:**
* Ngăn không cho người dùng nhấp đôi vào JTable và sửa nội dung tạm thời, lúc đó sẽ không xem chi tiết được thông tin muốn xem.
* Đảm bảo bắt buộc người dùng nhập đầy đủ/ đúng thông tin thì mới cho tiếp tục chương trình. Ví dụ họ tên không được chứa kí tự chữ số, số điện thoại không được chứa kí tự chữ cái.
* Với bảng mã khoa, có thể xem nhanh chi tiết khoa bằng cách nhấn vào khoa cần xem.
* Khi người dùng muốn sửa thông tin, bắt buộc người dùng phải nhấn vào thông tin cần xem để lấy đúng dữ liệu.

**STACK ANH HEAP MEMORY**

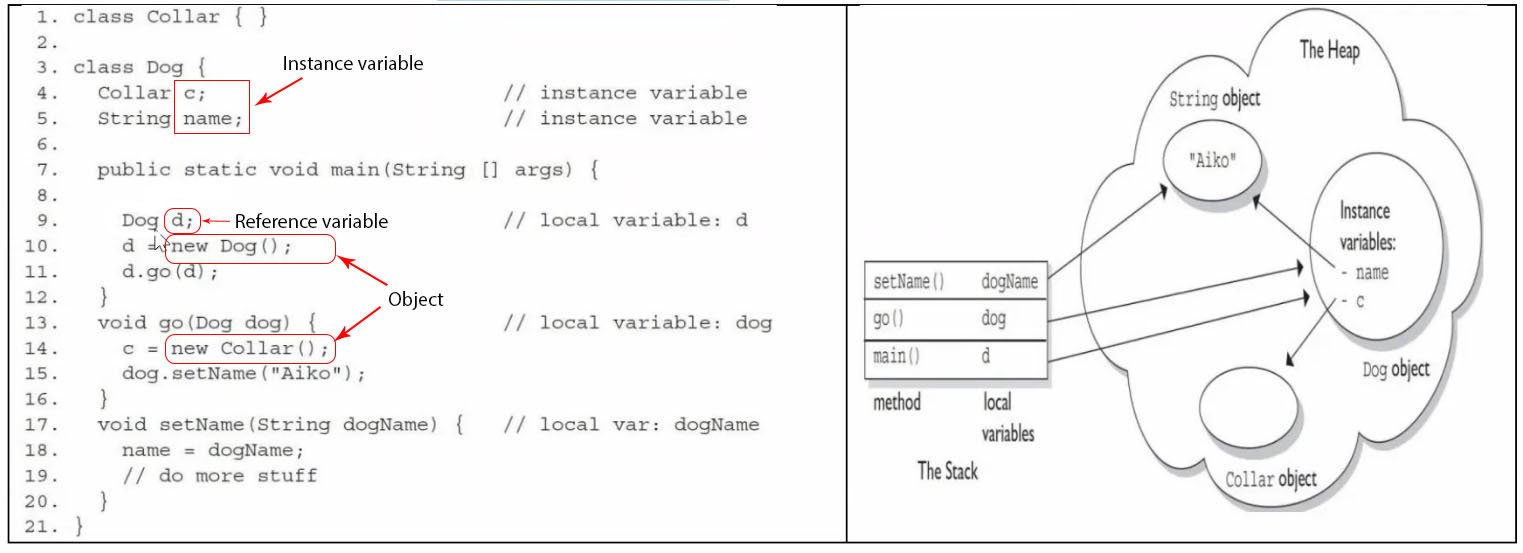
* **Chương trình JAVA khởi chạy sẽ dùng bộ nhớ RAM để lưu trữ các biến các hàm, và trong bộ nhớ đó được phân làm 2 phân vùng chính:**
* **Bộ nhớ STACK:**
* Có dung lượng nhỏ hơn bộ nhớ HEAP, dùng để lưu các **biến cục bộ** (***local variable***: Biến cục bộ trong Java là các biến được khai báo bên trong một phương thức, constructor hoặc khối lệnh và chỉ có thể được sử dụng trong phạm vi của phương thức, constructor hoặc khối lệnh đó. Biến cục bộ được khởi tạo khi phương thức hoặc khối lệnh bắt đầu thực thi và bị hủy khi phương thức hoặc khối lệnh kết thúc) , các **hàm**, các **đối số truyền vào hàm** và các **biến tham chiếu** (***reference variables***: Biến tham chiếu trong Java là các biến lưu trữ địa chỉ bộ nhớ của các đối tượng thay vì lưu trữ trực tiếp giá trị của đối tượng).
* Ví dụ biến tham chiếu:



* **Bộ nhớ HEAP:**
* Dùng để lưu các **Objects (đối tượng)** và **Instance Variables** (***biến thể hiện***: là biến khai báo bên trong 1 lớp nhưng bên ngoài bất kì phương thức, constructor (hàm khởi tạo) hoặc khối lệnh nào) (bất cứ khi nào Object được tạo ra bằng từ khóa ***“new”*** thì nó sẽ được cấp phát vùng nhớ trên bộ nhớ Heap, và dùng các biến tham chiếu để trỏ đến các đối tượng đó để sử dụng).
* Bộ nhớ HEAP sẽ tăng dần nếu số lượng Object được khởi tạo tăng dần.
* Bộ nhớ Heap được quản lý bởi ***Garbage Collector (trình dọn rác của JAVA)*** bất cứ khi nào đối tượng không được sử dụng, giúp thu hồi bộ nhớ của các đối tượng không còn được sử dụng để tránh rò rỉ bộ nhớ.

**\* Cả STACK và HEAP đều được quyết định sử dụng bao nhiêu tài nguyên ở RAM ở trong quá trình Runtime, lúc Compile thì cần sử dụng tài nguyên ở RAM.**

* **Ví dụ minh họa:**



* **Trong hình trên, c và name (instance variable) thuộc object Dog (cũng như là các biến thuộc class Dog khi khởi tạo đối tượng) và được chứa trong bộ nhớ HEAP.**