* **%JAVA\_HOME%** dùng để trỏ qua trỏ về phiên bản jdk(là gói java development kit là hộp nhà phát triển của java) của java khi trong công ty dự án có nhiều phiên bản của jdk khác nhau
* **jdk** : chứa jre: là nơi từ source code(chứa byte code và jvm) đến byte code và jvm: từ byte code đến marchne

-quy trình thực thi đoạn mã code:

+ .java--> .class-->machine code --> result

- cấu trúc khi tạo dự án Java

+ workspace->project->package->.java(class)

**\*Convention:**

**+package:** luôn viết thường

**+class:** camelcase + viết hoa chữ cái đầu tiên

Vd: Ex01MethodClass

**+method:**camelcase + viết thường chữ đầu tiên và chữ phía sau viết hoa chữ cái đầu tiên của từng từ

vd: printStarTriangle

**+variable(hàm)** : 🡪single 🡪viết thường ; hằng số 🡪 viết hoa

**-Biến** của gias trị nguyên thủy được lưu vào stack

**+compile time error là lỗi khi code bị lỗi**

**+runtime error là lỗi khi khởi chạy**

-//Hàm có cấu trúc 2 phần

+**phần khai báo**

+**phần thực thi** (phần thân hàm)

[access modifier] [static]

-Kiểu dữ liệu biến đối tượng lưu trữ các phương thức và các hàm để xử lí

- /\*\* dùng để giải thích cho 1 class

+Kiểu Biến luôn nằm ở Stack , còn giá trị nguyên thủy nằm ở stack còn đối tượng nằm ở HEAP

* compile time 🡪 typing code
* runtime 🡪 intial object at heap
* alt + shift + r : để chọn và sửa cùng lúc

+Biến đối tượng(là wapper(bao bọc)) lưu trữ nhiều thông tin , phù hợp nhu cầu thực tế.

+Phân biệt int và Interger

-int: lưu trữ ở STACK , luôn luôn có giá trị🡪lưu trữ

-Interger: lưu trữ ở HEAP , có thể có giá trị hoặc không( NULL)

🡪 lưu trữ hỗ trợ các phương thức hữu ích

-final là hằng số là những giá trị không thay đổi giá trị chỉ áp dụng với kiểu dữ liệu nguyên thủy.

-Sử dụng final để hạn chế tạo các vùng nhớ rác trên HEAP

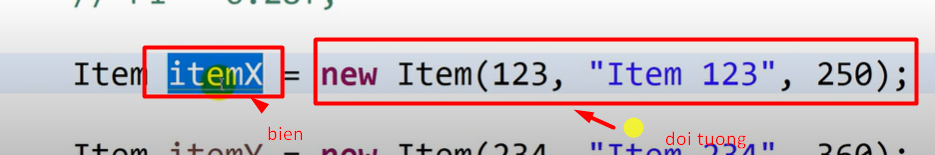
-Final ở Stack k thể cập nhập được giá trị nhưng ở HEAP thì có thể cập nhập

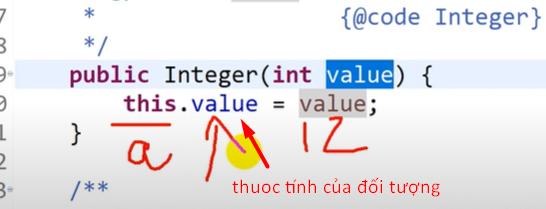
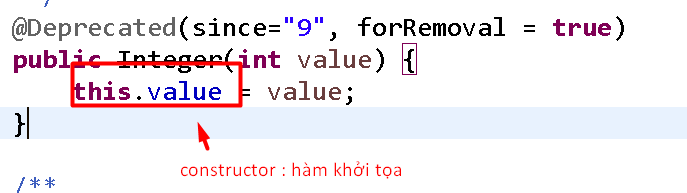
-Nếu biến là hằng số không thể thay đổi giá trị ở STACK

-Cấu trúc dữ liệu gồm biến, thuộc tính

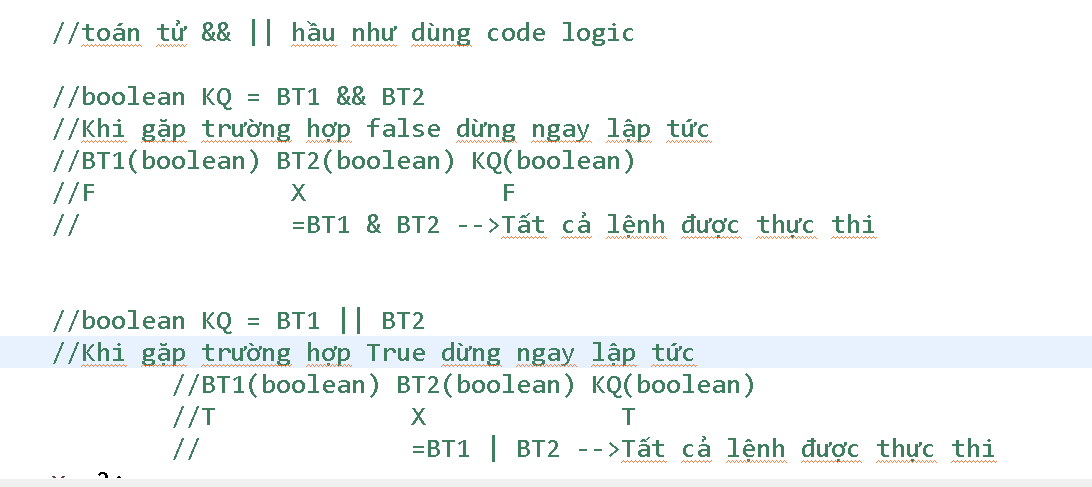
-Thuật toán : hàm , phương thức

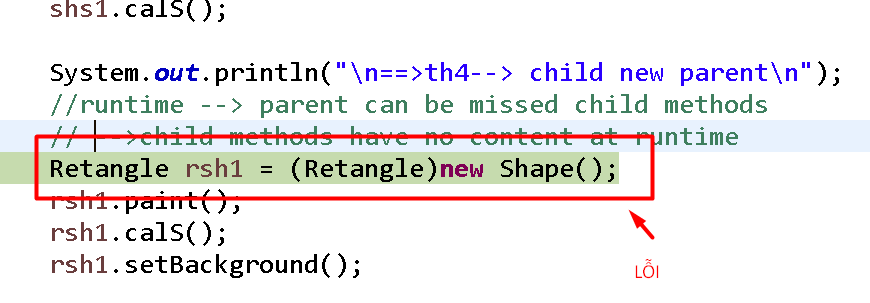
- Khi in ra biến ở kiểu đối tượng mặc định là ToString



* Toán tử = hoạt động ở STACK
* PASS BY VALUE at Stack bởi dấu =
* 
* Ở HEAP có kiểu dữ liệu Interger
* 
* phạm vi sử dụng của biến la` block scope trong {}
* 

\*\*\*\* //code --> saving memory,time -->clean



* Đối với swtitch case k có break nếu đúng trường hợp thứ k thì các trường hợp sau đó không xét mà nhảy vào phần thực thi(body) và chỉ kết thúc khi gặp break hoặc đến cuối default
* Break thoát khỏi vòng lặp hiện tại.
* continue: bỏ qua phần chưa thực thi của lần lặp hiện tại.
* đối tượng 🡪 constructor
* Class là khái niệm trừu tượng dùng để chỉ
* đối tượng dùng để mô tả các thực thể
* \* nếu để defaut thì ai cũng truy cập k đảm bảo tính OOP sẽ thành FOP(Functional oriented programming🡪
* Static của hàm để hạn chế khởi tạo quá nhiều ô nhớ trên HEAP
* Static không phụ thuộc vào đối tượng .(sử dụng cho biến toàn cục)
* //class, object: --> call static method
* //object: call non-static method
* //non-static: thuộc phạm vi của Object , static của class
* //non-static method: khi hàm phụ thuộc vào đối tượng đang gọi
* //static: không phụ thuộc vào đối tượng đang gọi mà chỉ phụ thuộc vào các tham số
* - Toán tử == dùng để so sánh 2 biến giá trị ở Stack(kiểu nguyên thủy)
* So sánh giá trị của biến đối tượng 🡪 equals của class Object, nhưng thường Class tự tạo ra phải Overrwrite lại equal
* Ctrl + D dùng để xóa nhanh;
* Static luôn tạo trước non-static vì không phụ thuộc vào đối tượng mà chỉ phụ thuộc vào tham số
* Con không thể ép kiểu sang cha Cha có thể sẽ thiếu hàm mở rộng trong con
* vd: 
* this và super thuộc object scope , this gọi các hàm non static ở class hiện tại còn super gọi các hàm từ lớp cha có thể qua lại các contructor
* singertin parten cho phép tạo ra 1 đối tượng duy nhất cho một class , buider partern là imutable object khi 1 class có yêu cầu nhìu hàm khởi tạo phức tạp , có nhiều thuộc tính truyền rối thì dùng bằng cách (.) để dùng
* Strantegy dùng khi có chung tham số kiểu dữ liệu giống nhau
* 1 interface chỉ có 1 hàm trừu tượng gọi là functional interface, anoymos funtion là fuction override hàm trừu tượng trong functional interface hay còn gọi là lamdar express tion
* \*\* Lamda expresstion là 1 cách viết ngắn gọn để mình tạo ra 1 thể hiện cho functional Interface
* Trong Java không cho phép truyền tham số là một hàm khác , chi được phép KDL , NT, DT
* Để giải quyết tạo ra Functional Interface để chứa Strategy(abstract method trong Interface đó)