1. Các kiểu dữ liệu trong Java (Giá trị mặc định của các kiểu dữ liệu ấy)

- Kiểu dữ liệu nguyên thủy (Primitive Data Types) và giá trị mặc định:

+ byte: 0

+ short: 0

+ int: 0

+ long: 0L

+ float: 0.0f

+ double: 0.0d

+ boolean: false

+ char: '\u0000'

- Kiểu dữ liệu tham chiếu (Reference Types):

+ Array

+ Class

+ Interface

2. Quy tắc ép kiểu, cho ví dụ

- Có 2 cách ép kiểu:

+ ép kiểu ngầm định: kiểu muốn ép sang phải lớn hơn kiểu đầu

Ví dụ: int a = 10;

float b = a;

+ ép kiểu tường minh: ngược lại kiểu trên, phạm vi kiểu hẹp đi nên bị mất dữ liệu

Ví dụ: float a = 10.3f;

int b = (int) a;

3. Toán tử trong kiểu dữ liệu nguyên thủy và hỗn hợp kiểu dữ liệu nguyên thủy và Wrapper class

\*\*\* Toán tử trong kiểu dữ liệu nguyên thủy:

- Toán tử số học

+ phép cộng: +

+ phép trừ: -

+ phép nhân: \*

+ phép chia: /

+ phép chia lấy phần dư: %

+ phép lượng gia: ++

+ phép lượng giảm: --

- Toán tử quan hệ:

+ == kiểm tra giá trị của 2 toán hạng

+ != phủ định của =

+ >

+ <

+ >=

+ <=

- Toán tử logic

+ && và

+ || hoặc

+ ! phủ định

+ ^ XOR

- Toán tử gán

+ =

+ +=

+ -=

+ /=

+ %=

+ &=

+ |=

- Toán tử Bit

+ ~ phủ định NOT

+ & toán tử AND

+ | toán tử OR

+ ^ toán tử Exclusive OR

+ >> dịch phải

+ << dịch trái

\*\*\* kiểu dữ liệu nguyên thủy và Wrapper class

- Wrapper class sử dụng để chuyển đổi kiểu dữ liệu nguyên thủy thành 1 đối tượng và ngược lại

- Wrapper class giá trị mặc định gán null

- kiểu dữ liệu nguyên thủy giá trị mặc định theo như kiểu dữ liệu

- nếu chuyển đổi kiểu dữ liệu nguyên thủy về wrapper class thì trình biên dịch java tự chuyển đổi => autoboxing

- nếu chuyển kiểu wrapper class về kiểu nguyên thủy => unboxing

4. Các lỗi có thể xảy ra với kiểu dữ liệu nguyên thủy

- giá trị vượt quá giới hạn giá trị của kiểu dữ liệu

- Ép kiểu dữ liệu tường minh, dẫn đến dữ liệu bị sai

5. Luồng xử lý cơ bản của JVM với file .java (tìm class thế nào? tìm hàm main thế nào)

- file .java qua biên dịch javac thì thành Bytecode file .class, sau đó JVM tải mã lệnh và kiểm tra mã lệnh,

kiểm tra class nào public và có tên trùng với tên file, sau đó vào tìm hàm public static void main(cú pháp) chạy

6. Sự khác biệt của biến local, biến static, biến instance(cho ví dụ)

- Biến local:

+ khai báo trong các method, hàm constructor, trong block. Biến mất khi kết thúc method, constructor, block.

public deMoBienLocal(){

int c = 10; // biến local

}

+ được lưu trên vùng nhớ của Stack.

+ bên ngoài method không truy cập được, k sử dụng access modifier khi khai báo biến local

+ biến local không có giá trị mặc định, và khởi tạo giá trị mặc định cho biến trước khi sử dụng

- Biến instance:

+ khai báo trong class, bên ngoài method, constructor, block

public class Demo{

public int tuoi; // biến instance

String ten ; // biến instance

}

+ được lưu trong bộ nhớ heap

+ biến instance được tạo khi 1 đối tượng được new

+ có thể sử dụng access modifier khi khai báo biến instance, mặc định là default

+ có giá trị mặc định phụ thuôc vào kiểu dữ liệu.

+ sử dụng để truyền đối số vào cho method

+ bên trong class khai báo biến instance thì có thể gọi trực tiếp bằng tên khi sử dụng ở khắp nơi bên trong class đó

- Biến static:

+ được khai báo bên trong class sử dụng từ khóa static, phía bên ngoài method, constructor, block

public class Demo{

public int tuoi = =24; // biến static

String ten = "Nguyen van a"; // biến static

}

+ có duy nhất 1 bản sao biến static được tạo ra dù tạo bao nhiêu đối tượng từ lớp tương ứng

+ được lưu trong bộ nhớ static riêng (permgen) (bên trong heap)

+ biến static được tạo khi chương trình bắt đầu và mất khi dừng chương trình

+ giá trị mặc định phụ thuộc vào kiểu giá trị khai báo

+ truy cập thông qua class chứa nó

+ phương thức khai báo biến static thì gọi tên k cần new mới đối tượng

7. Bộ nhớ stack có lỗi kinh điển nào. giải thích và cho ví dụ

- lỗi Stack Overflow: Do là bộ nhớ Stack lưu trữ dữ liệu theo kiểu vào trước thì ra sau, vào sau thì ra trước

chia thành các ô nhớ. Stack nơi chứa tham số và biến local khai báo bên trong method. Mỗi lần gọi 1 method thì giá trị được lưu vào ô trong Stack, các biến local

=> mỗi lần gọi hàm sẽ thì ô chứa trong Stack mất đi, ô chứa được trả lại khi thoát khỏi hàm. Nếu càng gọi thì ô nhớ càng nhỏ đi, và lúc nào đó sẽ đầy, không thể gọi hàm được nữa

==> xảy ra lỗi Stack Overflow

- Ví dụ:

public class Demo{

public static void method(){

method();

}

public static void main(String args[]) {

method1();

}

}

8. So sánh đệ quy và vòng lặp (for, while)

- Đệ quy là gọi 1 hàm tương tự trong 1 hàm, ngăn xếp được sử dụng để lưu trữ hàm, biến local, giá trị. thực hiện tốc độ chậm hơn, bên trong phức tạp hơn vòng lặp.

- vòng lặp: cấu trúc điều khiển thực thi lệnh nhiều lần nếu điều kiện đúng, loop không sử dụng stack. thực hiện tốc độ nhanh

9. Thuộc tính của đối tượng lưu trữ dữ liệu như thế nào trên bộ nhớ Heap

- khi một đối tượng được khởi tạo thì biến lưu vào Stack, còn đối tượng được lưu bên Heap. biến của đối tượng được khởi tạo sẽ trỏ đến đối tượng trong Heap và có địa chỉ riêng

Nếu đôi tượng có giá trị thì giá trị được trỏ ra chỗ khác trong heap từ địa chỉ dối tượng này trong Heap

- biến khai báo theo kiểu String thì giá trị được lưu trong String pool trong Perm của bộ nhớ Heap

10. Tại sao nói không có final object. Cho ví dụ

- có thể thay đổi được giá trị của object final, nhưng không thể khởi tạo object đó lại 1 lần nữa

- Ví dụ:

final Mentor mentor = new Mentor("dongpt",33);

=> có thể thay đổi "dongpt" thành 1 tên khác, nhưng k thể new mới 1 Mentor

mentor lưu trong Stack với 1 địa chỉ, "dongpt" lưu trong Heap với 1 địa chỉ và trỏ ra giá trị "dongpt",

nếu thay đổi thì thay đổi được địa chỉ "dongpt" lưu trong heap, còn k thay đổi đc địa chỉ mentor lưu trong stack

11. Vẽ hình minh họa các đối tượng trên bộ nhớ heap và stack (các bước nếu có hiện tượng hàm trong hàm)

public static void main(String [] args){

String name = "dongpt";

int age = 18;

innerMethod();

User u1 = new (name, age);

}

public static innerMethod(){

String color = "RED";

System.out.println("color");