|  |  |
| --- | --- |
| **1. Nhắc lại method là gì? Mục đích sử sụng metod?** | - Phương thức (method) là một nhóm các câu lệnh thực hiện một nhiềm vụ nhất định  - M**ethod** (hay còn gọi là phương thức/hàm) được sử dụng để **tổ chức mã lệnh một cách gọn gàng, dễ hiểu và tái sử dụng**. |
| **2. Cú pháp tạo methd trong java ? Cách gọi method trong java?** | - Cú pháp:  Modifier returnValueType methodName(listof parameters){  //Method body;  }  Trong đó :  + Modifier có thể là các từ khóa để quy định các tính chất khác nhau của phương thức  + RetumValueType là kiểu dữ liệu trả về của phương thức  + MethodName là tên gọi của phương thức  + List of paramaters là danh sách các tham số của phương thức  + Method body là phần thân của phương thức  Cách gọi methd trong java:  Gọi trực tiếp bằng **tên class** (không cần tạo object).  TenClass.tenMethod(); |
| **3. Giải thích các thành phần trong một method?** | Trong đó :  Method Header (Phần đầu phương thức)   1. **Modifier (public static)**    * public: Phạm vi truy cập — method này có thể được gọi từ bất cứ đâu.    * static: Method này có thể được gọi mà không cần tạo đối tượng của lớp. 2. **Return Value Type (int)**    * int: Kiểu trả về của method. Method này trả về một số nguyên. 3. **Method Name (max)**    * max: Tên của method — dùng để gọi hoặc tham chiếu đến method. 4. **Formal Parameters (int num1, int num2)**    * Đây là hai **tham số đầu vào** cho method, mỗi cái có kiểu int.    * Chúng là biến tạm thời chỉ tồn tại trong method này. 5. **Method Signature**    * Là phần kết hợp giữa **tên method** và **danh sách tham số**: max(int num1, int num2)    * Dùng để phân biệt các method (nhất là khi có overloading — quá tải phương thức).   **🔸 Method Body (Phần thân phương thức)**   1. **Local Variable Declaration**    * int result; Khai báo biến cục bộ (local variable) chỉ sử dụng trong method này. 2. **If-Else Statement**    * Dùng để so sánh hai số num1 và num2, gán số lớn hơn vào result. 3. **Return Statement**    * return result; Trả lại giá trị của biến result cho nơi gọi method. |
| **4. Các loại method ?** | |  |  | | --- | --- | | **1. Instance Method** | Gọi thông qua đối tượng. Dùng để thao tác trên dữ liệu cụ thể của object. | | **2. Static Method** | Gọi trực tiếp qua lớp, không cần tạo object. Dùng cho logic chung. | | **3. Void Method** | Không trả giá trị (dùng từ khóa void). | | **4. Return Method** | Trả về một giá trị (dùng return và chỉ định kiểu trả về như int, String, v.v.). | | **5. Overloaded Method** | Các method cùng tên nhưng khác tham số (số lượng hoặc kiểu). | | **6. Abstract Method** | Chỉ khai báo, không có thân method. Phải được override trong lớp con | |
| **5. Trong một class tạo được bao nhiêu method cùng tên? Điều kiện tạo được method cùng tên?** | Trong một class Java: - Có thể tạo bao nhiêu method cùng tên cũng được: Nhưng phải thỏa điều kiện sau: 1. Khác nhau về danh sách tham số(parameter list) - Khác số lượng tham số - Khác kiểu dữ liệu tham số - Khác thứ tự kiểu tham số Chỉ khác return type(kiểu trả về) là KHÔNG ĐỦ , phải khác tham số |
| **6. Tìm bộ nhớ stack và heap trong java ? Dữ liệu nào lưu ở stack và dữ liệu nào lưu ở heap?** | Stack Memory: +Là vùng nhớ nhỏ, cực kỳ nhanh. + Dùng để lưu biến nguyên thủy , biến local , biến tham chiếu. + Quản lý kiểu LIFO(Last In First Out) - vào sau ra trước + Khi một hàm kết thúc, các biến trên Stack của nó sẽ tự động bị xóa - Heap Memory: + Là vùng nhớ lớn hơn, truy cập chậm hơn +Dùng để lưu trữ object(đối tượng) và instancev ariables(các thuộc tính của đối tượng) +Các object trên Heap sẽ tồn tại cho tới khi không còn tham chiếu nào tớ nó 🡪 lúc đó Garbage Collector(trình thu gom rác) sẽ thu hồi bộ nhớ. |
| **7. Tìm hiểu tham trị và tham chiếu trong java?** | - Tham trị(pass-by-value): Khi bạn truyền một giá trị có kiểu dữ liệu nguyên thủy, Java sẽ tạo ra một bản sao của giá trị đó.Mọi thay đổi lên biến trong phương thức không ảnh hưởng tới biến gốc bên ngoài.  - Tham chiếu(Pass-by-Rerence): Khi bạn truyền vào một kiểu dữ liệu tham chiếu thì Java cũng copy giá trị tham chiếu. Có nhĩa là cả biến gốc và biến trong hàm đều trỏ tới cùng một object trong bộ nhớ. Thay đổi nội dung của object thì sẽ ảnh hưởng tới biến gốc.  - Trong java chỉ có chuyền tham trị không có chuyền tham chiếu |