**Chuẩn bị:**

**1. Phương thức là gì? Tại sao phải có phương thức (ý nghĩa của phương thức)?**

**Phương thức** là một tập hợp bao gồm các câu lệnh, gom lại các câu lệnh thực hiện cho mục đích nào đó.

**Có phương thức vì**: làm cho chương trình bớt phức tạp, phân tách được vấn đề. Không phải viết đi viết lại nhiều lần câu lệnh mà chỉ cần gọi một lần phương thức.

**2 .Các thành phần của phương thức.**

Một phương thức trong Java (và trong hầu hết các ngôn ngữ lập trình) bao gồm các thành phần sau:

Kiểu trả về (Return Type): Kiểu trả về xác định loại dữ liệu mà phương thức sẽ trả về sau khi hoàn thành việc thực hiện. Nếu phương thức không trả về giá trị nào, bạn sử dụng từ khóa void. Ví dụ:

public int add(int a, int b) {

return a + b;

}

Tên phương thức (Method Name): Tên của phương thức đặt sau kiểu trả về và phải tuân theo các quy tắc đặt tên trong Java. Tên phương thức được sử dụng để gọi phương thức khi cần.

public void printMessage() {

System.out.println("Hello, World!");

}

Danh sách tham số (Parameter List): Danh sách các tham số là các giá trị mà bạn truyền vào phương thức khi gọi nó. Tham số giúp truyền dữ liệu vào phương thức để xử lý. Mỗi tham số bao gồm kiểu dữ liệu và tên.

public double calculateArea(double radius) {

return Math.PI \* radius \* radius;

}

Thân phương thức (Method Body): Thân phương thức chứa các lệnh và câu lệnh Java thực hiện công việc cụ thể của phương thức. Thân phương thức được bao quanh bởi cặp dấu ngoặc nhọn {}.

public void printMessage() {

System.out.println("Hello, World!");

}

Quyền truy cập (Access Modifier): Quyền truy cập xác định ai có thể gọi phương thức và truy cập nó. Các quyền truy cập phổ biến trong Java bao gồm public, private, protected, và mặc định (không sử dụng bất kỳ quyền truy cập nào). Ví dụ:

public void doSomething() {

}

Sử dụng từ khóa static: Nếu một phương thức được đánh dấu là static, nó thuộc về lớp chứ không thuộc về đối tượng cụ thể. Bạn có thể gọi phương thức static bằng cách sử dụng tên lớp.

public static void staticMethod() {

}

**3. Thế nào là truyền tham số bằng giá trị (pass by value)?**

Truyền tham số bằng giá trị (pass by value) trong Java là cách mà giá trị của biến được truyền vào phương thức, và bất kỳ thay đổi nào thực hiện trên tham số trong phương thức không ảnh hưởng đến giá trị ban đầu của biến ngoài phương thức. Điều này có nghĩa là biến gốc không thay đổi sau khi phương thức hoàn thành.

Ví dụ:

public class giaTri {

public static void main(String[] args) {

int a = 5;

System.out.println("Giá trị trước: a = " + a);

suaGiaTri(a);

System.out.println("Giá trị sau: a = " + a);

}

public static void suaGiaTri (int value) {

value = value \* 2;

System.out.println("Giá trị trong phương thức: value = " + value);

}

}

Kết quả của ví dụ trên như sau:

Giá trị trước: a = 5.

Giá trị sau: a = 10.

Giá trị trong phương thức: value = 5.

Câu 3: Thế nào là overloading (của phương thức)? Cho ví dụ.

Overloading (nạp chồng) là tính năng cho phép định nhiều phương thức cùng tên trong một lớp, nhưng với các danh sách tham số khác nhau. Compiler sẽ xác định phương thức cần gọi dựa trên đối số được truyền vào.

Ví dụ:

public class MathOperations {

public int add(int a, int b) {

return a + b;

}

public double add(double a, double b) {

return a + b;

}

}

**5. Thế nào là biến cục bộ (local variable). Phạm vi sử dụng của biến cục bộ.**

**Biến cục bộ (local variable)** trong Java là một biến được khai báo và chỉ tồn tại trong phạm vi cụ thể, chẳng hạn như bên trong một phương thức hoặc một khối lệnh. Điều quan trọng là biến cục bộ chỉ có phạm vi sử dụng trong khối lệnh hoặc phương thức mà nó được khai báo.

Điểm chính về biến cục bộ:

**Phạm vi sử dụng:** Biến cục bộ chỉ có thể truy cập và sử dụng bên trong khối lệnh hoặc phương thức mà nó được khai báo. Bên ngoài phạm vi đó, biến này không tồn tại và không thể truy cập được.

Không có giá trị mặc định: Biến cục bộ không có giá trị mặc định. Bạn phải gán giá trị cho biến trước khi sử dụng nó trong mã.

Hủy sau khi phạm vi kết thúc: Biến cục bộ bị hủy sau khi khối lệnh hoặc phương thức mà nó thuộc về kết thúc thực thi. Điều này đảm bảo rằng không có sự xung đột về tên biến giữa các phạm vi khác nhau trong chương trình.

VD:

public class viDuBienCucBo {

public void phuongThuc() {

int x = 10; // Biến cục bộ x được khai báo trong phương thức

System.out.println(x); // Có thể sử dụng biến cục bộ x ở đây

}

public static void main(String[] args) {

viDuBienCucBo viDu = new viDuBienCucBo();

viDu.phuongThuc(); // Gọi phương thức

// Không thể truy cập biến cục bộ x ở đây

}

}