**Task2: Biến và các toán tử trong Java**

**1. Khai báo biến trong Java**

* Cú pháp để khai báo biến:

DataType varName [ = value] [, varName2] [ = value2]...;

Trong đó, DataType là kiểu dữ liệu của biến, verName là tên biến.

* Quy tắc đặt tên biến trong Java:

+ Chỉ được bắt đầu bằng một ký tự(chữ), hoặc một dấu gạch dưới(\_) hoặc một ký tự dollar($)

+ Tên biến không được chứa khoảng trắng

+ Bắt đầu từ ký tự thứ hai, có thể dùng ký tự(chữ), dấu gạch dưới(\_), hoặc ký tự dollar($)

+ Không được trùng với các từ khóa

+ Có phân biệt chữ hoa và chữ thường

* Ví dụ khai báo biến:

**public** **class** Bien {

**public** **static** **float** PI = 3.14f;   // Đây là biến static

**int** n;                            // Đây là biến instance

**public** Bien () {

**char** c = 'c';                 // Đây là biến local

    }

}

**2. Biến Local trong Java**

* Biến local được khai báo trong các phương thức, hàm contructor hoặc trong các block.
* Biến local được tạo bên trong các phương thức, contructor, block và sẽ bị phá hủy khi kết thúc các phương thức, contructor và block.
* Không được sử dụng "access modifier" khi khai báo biến local.
* Các biến local được lưu trên vùng nhớ stack của bộ nhớ.
* Bạn cần khởi tạo giá trị mặc định cho biến local trước khi có thể sử dụng.

**2.1 Ví dụ**

Ví dụ 1: Khởi tạo biến Local:

**public** **class** Bien {

**public** **void** sayHello() {

**int** n = 10;                     // Đây là biến local

        System.out.println("Gia tri cua n la: " + n);

    }

**public** **static** **void** main(String[] args) {

        Bien bienLocal = **new** Bien();

        bienLocal.sayHello();

    }

}

Kết quả:



Ví dụ 2: Không khởi tạo biến Local:

**public** **class** Bien {

**public** **void** sayHello() {

**int** n;                 // Đây là biến local

        System.out.println("Gia tri cua n la: " + n);

    }

**public** **static** **void** main(String[] args) {

        Bien bienLocal = **new** Bien();

        bienLocal.sayHello();

    }

}

Kết quả:



(Khi không khởi tạo biến Local, chương trình Java sẽ báo lỗi khi biên dịch)

**3. Biến Instance(Biến toàn cục) trong Java**

* Biến instance được khai báo trong một lớp(class), bên ngoài các phương thức, constructor và các block.
* Biến instance được lưu trong bộ nhớ heap.
* Biến instance được tạo khi một đối tượng được tạo bằng việc sử dụng từ khóa “new” và sẽ bị phá hủy khi đối tượng bị phá hủy.
* Biến instance có thể được sử dụng bởi các phương thức, constructor, block, ... Nhưng nó phải được sử dụng thông qua một đối tượng cụ thể.
* Bạn được phép sử dụng "access modifier" khi khai báo biến instance, mặc định là "default".
* Biến instance có giá trị mặc định phụ thuộc vào kiểu dữ liệu của nó. Ví dụ nếu là kiểu int, short, byte thì giá trị mặc định là 0, kiểu double thì là 0.0d, ... Vì vậy, bạn sẽ không cần khởi tạo giá trị cho biến instance trước khi sử dụng.
* Bên trong class mà bạn khai báo biến instance, bạn có thể gọi nó trực tiếp bằng tên khi sử dụng ở khắp nới bên trong class đó.

**3.1 Ví dụ**

* Ví dụ về biến Instance trong Java

**public** **class** Sinhvien {

    // biến instance "ten" kiểu String, có giá trị mặc định là null

**public** String ten;

    // biến instance "tuoi" kiểu Integer, có giá trị mặc định là 0

**private** **int** tuoi;

    // sử dụng biến ten trong một constructor

**public** Sinhvien(String ten) {

**this**.ten = ten;

    }

    // sử dụng biến tuoi trong phương thức setTuoi

**public** **void** setTuoi(**int** tuoi) {

**this**.tuoi = tuoi;

    }

**public** **void** showStudent() {

        System.out.println("Ten  : " + ten);

        System.out.println("Tuoi : " + tuoi);

    }

**public** **static** **void** main(String args[]) {

        Sinhvien sv = **new** Sinhvien("Nguyen Van A");

        sv.setTuoi(21);

        sv.showStudent();

    }

}

Kết quả:



**4. Biến Static trong Java**

* Biến static được khai báo trong một class với từ khóa "static", phía bên ngoài các phương thức, constructor và block.
* Sẽ chỉ có duy nhất một bản sao của các biến static được tạo ra, dù bạn tạo bao nhiêu đối tượng từ lớp tương ứng.
* Biến static được lưu trữ trong bộ nhớ static riêng.
* Biến static được tạo khi chương trình bắt đầu chạy và chỉ bị phá hủy khi chương trình dừng.
* Giá trị mặc định của biến static phụ thuộc vào kiểu dữ liệu bạn khai báo tương tự biến instance.
* Biến static được truy cập thông qua tên của class chứa nó, với cú pháp: TenClass.tenBien.
* Trong class, các phương thức sử dụng biến static bằng cách gọi tên của nó khi phương thức đó cũng được khai báo với từ khóa "static".

**4.1 Ví dụ**

* Ví dụ về biến static trong Java

**public** **class** Sinhvien {

    // biến static 'ten'

**public** **static** String ten = "Nguyen Van A";

    // biến static 'tuoi'

**public** **static** **int** tuoi = 21;

**public** **static** **void** main(String args[]) {

        // Sử dụng biến static bằng cách gọi trực tiếp

        System.out.println("Ten : " + ten);

        // Sử dụng biến static bằng cách gọi thông qua tên class

        System.out.println("Ten : " + Sinhvien.tuoi);

    }

}

Kết quả:

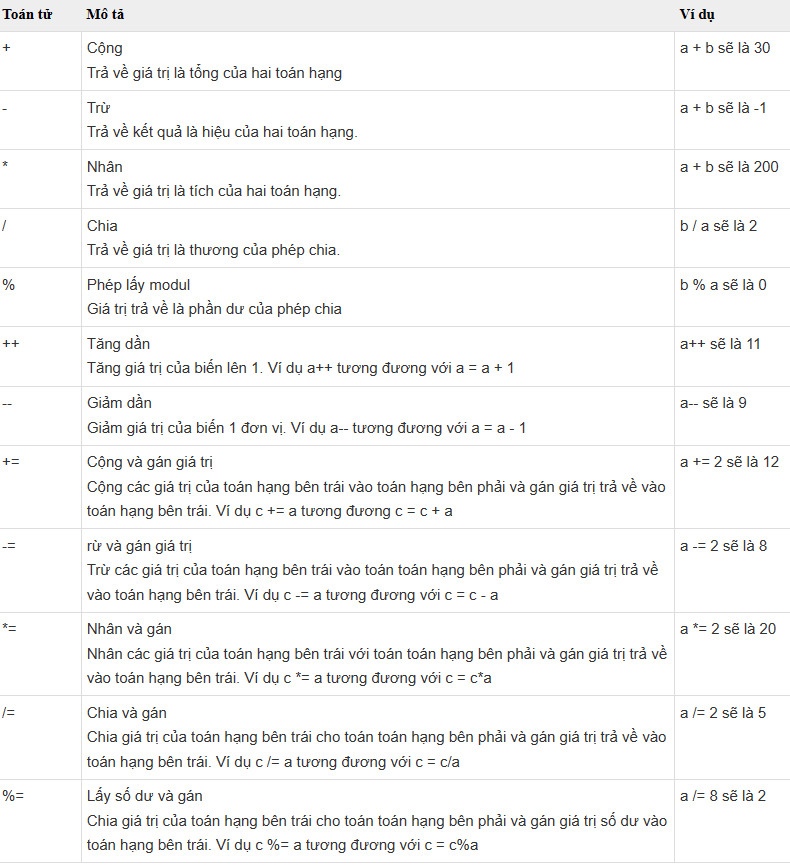


**6. Các toán tử trong Java**

**6.1 Toán tử số học**

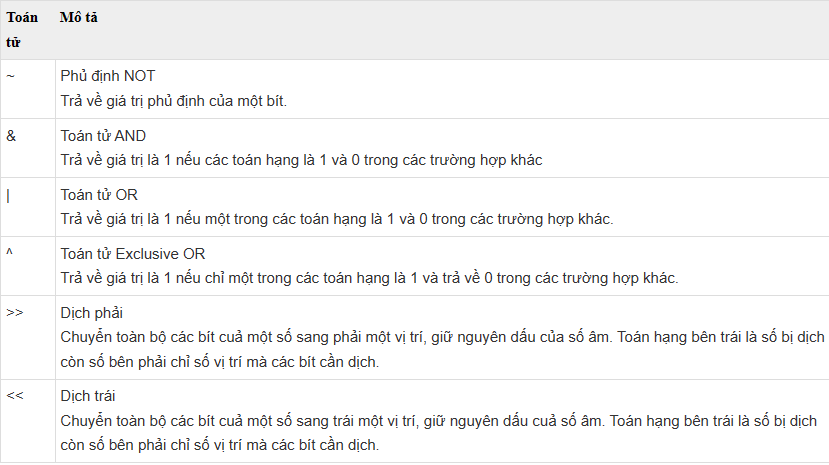
Các toán hạng của các toán tử số học phải ở dạng số. Các toán hạng kiểu boolean không sử dụng được, các toán hạng ký tự cho phép sử dụng loại toán tử này. Một vài kiểu toán tử được liệt kê trong bảng dưới đây.

Giả sử chúng ta có biến số nguyên a = 10 và b = 20.

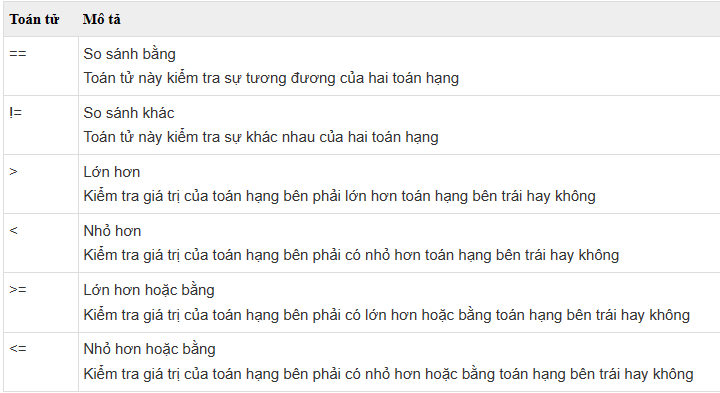


**6.2 Toán tử Bit**

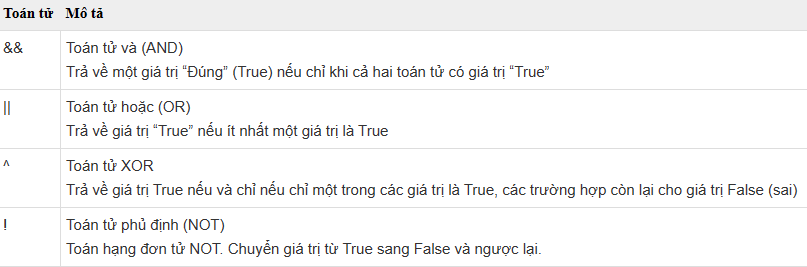
Các toán tử dạng bit cho phép chúng ta thao tác trên từng bit riêng biệt trong các kiểu dữ liệu nguyên thuỷ.



**6.3 Các toán tử quan hệ**

Các toán tử quan hệ được sử dụng kiểm tra mối quan hệ giữa hai toán hạng. Kết quả của một biểu thức có dùng các toán tử quan hệ là những giá trị Boolean (logic “true” hoặc “false”). Các toán tử quan hệ được sử dụng trong các cấu trúc điều khiển.

**6.4 Các toán tử Logic**

Các toán tử logic làm việc với các toán hạng Boolean. Các toán tử quan hệ được sử dụng trong các cấu trúc điều khiển.