Primitive and Object data type

**a. Phân biệt kiểu dữ liệu nguyên thủy và kiểu dữ liệu object**

1. Kiểu dữ liệu nguyên thủy (Primitive Type)

Định nghĩa: Là các kiểu dữ liệu cơ bản nhất, không phải đối tượng, và được Java hỗ trợ trực tiếp. Chúng lưu trữ giá trị thực và không có phương thức hay thuộc tính đi kèm.

Các loại:

boolean (true/false)

byte (8-bit số nguyên)

char (16-bit ký tự Unicode)

short (16-bit số nguyên)

int (32-bit số nguyên)

long (64-bit số nguyên)

float (32-bit số thực)

double (64-bit số thực)

Bộ nhớ: Các biến kiểu dữ liệu nguyên thủy được lưu trữ trong stack memory và thường sử dụng ít bộ nhớ hơn.

Vd:

int a = 10;

char b = 'A';

2. Kiểu dữ liệu object (Object Type)

Định nghĩa: Là các kiểu dữ liệu tham chiếu. Thay vì lưu trữ giá trị thực, biến object chứa địa chỉ tham chiếu tới một đối tượng được lưu trữ trong heap memory. Những object này có thể chứa các thuộc tính và phương thức.

Các loại: Tất cả các lớp trong Java đều là kiểu object, ví dụ như:

String

Array

Các lớp wrapper (như Integer, Character, Boolean, v.v.)

Các lớp do người dùng tự định nghĩa.

Bộ nhớ: Object được lưu trữ trong heap memory. Khi làm việc với kiểu dữ liệu object, biến chỉ giữ địa chỉ tham chiếu tới object trong heap, chứ không phải là giá trị thực.

Vd:

String str = "Hello World";

Integer num = 20;

Vd:

public class DataTypeExample {

public static void main(String[] args) {

// Kiểu dữ liệu nguyên thủy

int x = 5;

int y = 10;

System.out.println("So sánh kiểu nguyên thủy: " + (x == y)); // false

// Kiểu dữ liệu object

Integer a = new Integer(5);

Integer b = new Integer(5);

System.out.println("So sánh kiểu object (==): " + (a == b)); // false, vì so sánh địa chỉ

System.out.println("So sánh kiểu object (equals): " + a.equals(b)); // true, so sánh giá trị

}

}

**b. Có thể chuyển đổi giữa hai kiểu dữ liệu này không ?**

Java hỗ trợ chuyển đổi giữa kiểu dữ liệu nguyên thủy và kiểu object thông qua quá trình autoboxing và unboxing.

Autoboxing: Quá trình tự động chuyển đổi từ kiểu dữ liệu nguyên thủy sang object tương ứng. Ví dụ, int có thể tự động chuyển thành Integer.

Unboxing: Quá trình tự động chuyển đổi từ kiểu object sang kiểu dữ liệu nguyên thủy tương ứng. Ví dụ, Integer có thể chuyển thành int.

Vd:

public class ConversionExample {

public static void main(String[] args) {

// Autoboxing: Chuyển từ kiểu nguyên thủy sang object

int primitiveInt = 5;

Integer objectInt = primitiveInt; // Autoboxing tự động

// Unboxing: Chuyển từ object sang kiểu nguyên thủy

Integer anotherObjectInt = 10;

int anotherPrimitiveInt = anotherObjectInt; // Unboxing tự động

// Hiển thị kết quả

System.out.println("Giá trị objectInt (Autoboxing): " + objectInt); // Kết quả: 5

System.out.println("Giá trị anotherPrimitiveInt (Unboxing): " + anotherPrimitiveInt); // Kết quả: 10

}

}

**c. Có thể so sánh hai kiểu dữ liệu này với nhau không?**

1. Có thể so sánh kiểu dữ liệu nguyên thủy

Có, kiểu dữ liệu nguyên thủy có thể so sánh với nhau bằng các toán tử so sánh như ==, !=, >, <, >=, và <=.

**Vd:**

public class PrimitiveComparison {

public static void main(String[] args) {

int x = 5;

int y = 10;

System.out.println("So sánh kiểu nguyên thủy: " + (x == y)); // Kết quả: false

}

}

2. Có thể so sánh kiểu dữ liệu object

Đối với kiểu dữ liệu object, không nên sử dụng toán tử == để so sánh vì nó sẽ so sánh địa chỉ tham chiếu của các đối tượng. Để so sánh giá trị thực của các đối tượng, cần sử dụng phương thức .equals().

Vd:

public class ObjectComparison {

public static void main(String[] args) {

Integer a = new Integer(5);

Integer b = new Integer(5);

// So sánh địa chỉ tham chiếu

System.out.println("So sánh kiểu object (==): " + (a == b)); // Kết quả: false

// So sánh giá trị thực

System.out.println("So sánh kiểu object (equals): " + a.equals(b)); // Kết quả: true

}

}

**d. Giá trị khi khởi tạo biến với hai loại kiểu dữ liệu này là gì?**

1. Kiểu dữ liệu nguyên thủy (Primitive Type)

Khi khai báo một biến kiểu dữ liệu nguyên thủy mà không khởi tạo giá trị, nó sẽ có giá trị mặc định như sau:

boolean: false

byte: 0

short: 0

int: 0

long: 0L

float: 0.0f

double: 0.0

char: '\u0000' (ký tự null)

vd:

public class PrimitiveDefaultValues {

public static void main(String[] args) {

int intValue; // Giá trị mặc định là 0

boolean boolValue; // Giá trị mặc định là false

char charValue; // Giá trị mặc định là '\u0000'

// Hiển thị giá trị mặc định

System.out.println("Giá trị mặc định của intValue: " + intValue);

System.out.println("Giá trị mặc định của boolValue: " + boolValue);

System.out.println("Giá trị mặc định của charValue: " + charValue);

}

}

2. Kiểu dữ liệu object (Object Type)

Khi khai báo một biến kiểu dữ liệu object mà không khởi tạo giá trị, nó sẽ có giá trị mặc định là null.

Vd:

public class ObjectDefaultValue {

public static void main(String[] args) {

String str; // Giá trị mặc định là null

Integer intValue; // Giá trị mặc định là null

// Hiển thị giá trị mặc định

System.out.println("Giá trị mặc định của str: " + str); // Kết quả: null

System.out.println("Giá trị mặc định của intValue: " + intValue); // Kết quả: null

}

}