# Phân biệt kiểu dữ liệu nguyên thủy và kiểu dữ liệu đối tượng trong java.

1. **Kiểu dữ liệu nguyên thủy (Primitive Data Types):**

* Là những kiểu dữ liệu cơ bản có sẵn trong java và không phải object.
* Không có phương thức hoặc thuộc tính và lưu giá trị trực tiếp trong vùng nhớ (mỗi biến nguyên thủy giữ giá trị của nó và có thể truy cập trực tiếp).
* Không nhận giá trị null, không thể tạo ra các kiểu con từ chúng.
* Các loại dữ liệu nguyên thủy:
  + int: 4 bytes (lưu trữ số nguyên trong khoảng từ -2,147,483,648 đến 2,147,483,647).
  + double: 8 bytes (lưu trữ số thực).
  + float: 4 bytes (lưu trữ số thực, cần thêm ký tự “f” khi khởi tạo).
  + char: 2 bytes (lưu trữ 1 ký tự unicode).
  + boolean: byte không xác định (lưu trữ giá trị true hoặc false).
  + byte: 1 byte (lưu trữ số nguyên trong khoảng từ -128 đến 127).
  + short: 2 bytes (lưu trữ số nguyên trong khoảng từ -32,768 đến 32,767).
  + long: 8 bytes (lưu trữ số nguyên, cần thêm ký tự “L” khi khởi tạo).

1. **Kiểu dữ liệu đối tượng (Object Data Types)**

* Là các object được tạo từ các class trong Java và có thể bao gồm các thuộc tính, phương thức.
* Các object lưu trữ tham chiếu đến giá trị trong vùng nhớ chứ không lưu trực tiếp giá trị (khi tạo object từ 1 class thì biến sẽ không lưu trữ giá trị của object mà sẽ lưu trữ địa chỉ hoặc tham chiếu đến vị trị bộ nhớ object được lưu trữ).
* Có thể nhận giá trị null, có thể tạo ra các lớp con từ các lớp cha.
* Ví dụ về kiểu dữ liệu object:
* String: Lưu trữ chuỗi ký tự.
* Array: Lưu trữ 1 tập hợp các giá trị.
* Class: Tạo các đối tượng có thuộc tính và phương thức riêng.
* Interface: Định nghĩa các phương thức mà lớp thực hiện.

# Có thể chuyển đổi giữa hai kiểu dữ liệu này không?

1. **Chuyển Đổi Từ Kiểu Dữ Liệu Nguyên Thủy Sang Kiểu Dữ Liệu Đối Tượng**

* Từng kiểu dữ liệu nguyên thủy đều có các lớp bọc (wrapper classes) để có thể chuyển đổi thành object.
* byte: Byte
* short: Short
* int: Integer
* long: Long
* float: Float
* double: Double
* char: Character
* boolean: Boolean
* Ví dụ:

int num = 10; // Kiểu dữ liệu nguyên thủy

Integer numObj = Integer.valueOf(num); // Chuyển đổi sang object

1. **Chuyển Đổi Từ Kiểu Dữ Liệu Đối Tượng Sang Kiểu Dữ Liệu Nguyên Thủy**

* Có thể chuyển đổi từ object về kiểu dữ liệu nguyên thủy bằng các phương thức tương ứng của lớp bao bọc (wrapper classes).
* Ví dụ

Integer numObj = 10; // Object

int num = numObj.intValue(); // Chuyển đổi về kiểu dữ liệu nguyên thủy

1. **Tự Động Boxing và Unboxing**

* Java hỗ trợ boxing và unboxing tự động (khi gán một giá trị nguyên thủy cho một đối tượng bao bọc, Java tự động chuyển đổi giá trị nguyên thủy đó thành đối tượng (boxing). Ngược lại, khi gán một đối tượng bao bọc cho một biến kiểu dữ liệu nguyên thủy, Java tự động chuyển đổi đối tượng đó về kiểu dữ liệu nguyên thủy (unboxing)).
* Ví dụ:

Integer numObj = 10; // Boxing tự động

int num = numObj; // Unboxing tự động

1. **Lưu ý**

* Khi thực hiện unboxing từ một đối tượng bao bọc có giá trị là null, sẽ gây ra NullPointerException
* Ví dụ:

Integer numObj = null;

int num = numObj; // Gây ra NullPointerException

# Có thể so sánh hai kiểu dữ liệu này với nhau không?

* Có thể so sánh 2 kiểu dữ liệu này với nhau vì java sẽ tự động unboxing object về kiểu dữ liệu nguyên thủy trước khi so sánh
* Ví dụ:

int num = 10; // Kiểu dữ liệu nguyên thủy

Integer numObj = 10; // object

if (num == numObj) {

System.out.println("num and numObj are equal ");

} else {

System.out.println("num and numObj are not equal ");

}

Kết quả: num and numObj are equal

* Lưu ý:

Nếu đối object có giá trị là null, phép so sánh sẽ gây ra NullPointerException

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

1. **Kiểu dữ liệu nguyên thủy (Primitive Data Types):**

* boolean: false
* byte: 0
* short: 0
* int: 0
* long: 0L
* float: 0.0f
* double: 0.0
* char: \u0000 (ký tự null)

1. **Kiểu dữ liệu đối tượng (Object Data Types)**

* Giá trị khi khởi tạo biến với object tất cả đều là null