1. Phân biệt kiểu dữ liệu nguyên thủy và kiểu dữ liệu object.

- Kiểu dữ liệu nguyên thủy (primitive): kiểu dữ liệu được cung cấp sẵn, nó dùng để lưu trữ các giá trị đơn giản, được lưu trữ trực tiếp trong **stack memory**.

- Kiểu dữ liệu object: Là các biến tham chiếu trỏ tới một đối tượng trong vùng nhớ heap. Thay vì lưu giá trị trực tiếp, nó lưu địa chỉ tham chiếu (reference). Bao gồm:

* Lớp (Class) do người dùng định nghĩa.
* Kiểu Wrapper (lớp bao bọc primitive): Integer, Double, Boolean, …
* Mảng (Array)
* Chuỗi (String)
* Các đối tượng từ API của Java (ArrayList, HashMap, …).
* Có thể có fields và methods đi kèm.

|  |  |  |
| --- | --- | --- |
| **Đặc điểm** | **Kiểu nguyên thủy** | **Kiểu object** |
| **Nguồn gốc** | Các kiểu dữ liệu được định nghĩa sẵn | Các kiểu dữ liệu do người dùng tự định nghĩa |
| **Nơi lưu trữ** | Lưu trực tiếp trong stack | Stack (lưu reference), Heap (dữ liệu thật) |
| **Khi sao chép** | Tạo ra hai biến hoàn toàn khác nhau chỉ có giá trị giống nhau | Tạo ra hai biến tham chiếu nhưng cả hai cùng trỏ tới một đối tượng trong heap |
| **Giá trị mặc định** | Không phải là null (mặc định là 0, false, … tùy kiểu dữ liệu) | Giá trị mặc định là null |
| **Kích thước** | Cố định | Không cố định (tùy thuộc đối tượng) |
| **Hiệu năng** | Nhanh, ít tốn bộ nhớ | Chậm hơn, tốn bộ nhớ hơn |
| **Ví dụ** | byte, int, long, float, double, char, boolean  int a = 5; | Array, String class  Integer b = 10; String s = "abcd"; |

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

- Có thể chuyển đổi giữa hai kiểu dữ liệu primitive và object thông qua việc dùng Wrapper Class cùng với autoboxing và unboxing.

- Ví dụ:

***Autoboxing:***

 int num = 1;

 Integer obj1 = new Integer(num);// Boxing

 Integer obj2 = 2; // Boxing

***Unboxing:***

 Integer obj = 20;

 int x1 = obj.intValue(); // cách thủ công

 int x2 = obj; // unboxing

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

- Có thể so sánh giữa primitive và object trong Java thông qua việc dùng autoboxing/unboxing.

- Nếu so sánh primitive với object bằng == hoặc != thì sẽ tự động unbox object về  primitive trước khi so sánh. Ví dụ:

int a = 10;

Integer b = 10;

System.out.println(a == b); // true

- Nếu dùng equals(), thì primitive phải được boxing thành object trước. Ví dụ:

int a = 10;

Integer b = 10;

System.out.println(b.equals(a)); // true

- Trường hợp nếu object là null thì khi unboxing có thể gây ra lỗi.

int a = 10;

Integer b = null;

System.out.println(a == b); // Lỗi NullPointerException

System.out.println(b.equals(a)); // Lỗi NullPointerException

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

***- Primitive:***

* Nếu là biến instance (thuộc class) hoặc biến static thì sẽ được gán giá trị mặc định.
* Nếu là biến local (trong method) thì bắt buộc phải khởi tạo trước khi dùng, nếu không sẽ báo lỗi compile error.

|  |  |
| --- | --- |
| **Kiểu dữ liệu** | **Giá trị mặc định** |
| byte | 0 |
| short | 0 |
| int | 0 |
| long | 0L |
| float | 0.0f |
| double | 0.0d |
| char | ký tự rỗng |
| boolean | false |

**- *Object:***

* Đối với biến tham chiếu (Object, String, Array, Wrapper class, …): Instance field (thuộc class) hoặc static field mặc định là **null**.
* Local variable phải khởi tạo trước khi dùng nếu không cũng sẽ báo lỗi giống như primitive.