**Phần I. Primitive & Object Data Type**

\* Trong Java có 2 danh mục kiểu dữ liệu: ***Primitive Data Type*** (Kiểu dữ liệu nguyên thủy) và ***Non–Primitive Data Type*** (Kiểu dữ liệu không nguyên thủy hay kiểu dữ liệu tham chiếu)

|  |  |
| --- | --- |
| **Primitive Data Type** | **Object** |
| Lưu trữ biến và giá trị bên trong Stack | Lưu trữ biến bên trong Stack và tham chiếu tới Object được lưu trong bộ nhớ Heap |
| Là dữ liệu đã được định nghĩa trước bởi Java không có các thuộc tính và phương thức | Là kiểu dữ liệu được người dùng định nghĩa và được tạo ra từ các Class và có các thuộc tính và phương thức |
| Khi một biến được tạo ra và gán giá trị và một biến mới copy biến đã tạo thì hai biến sẽ có cùng một giá trị và khi thay đổi một biến sẽ không ảnh hưởng tới biến còn lại. | Khi một biến được tạo ra và tham chiếu tới một đối tượng và một biến mới copy biến đã tạo thì hai biến sẽ cùng tham chiếu tới một đối tượng trên Heap. Khi thay đổi một biến tham chiếu sẽ làm thay đổi đối tượng được tham chiếu. |
| Kiểu dữ liệu nguyên thủy không có giá trị mặc định là Null | Kiểu Object sẽ có giá trị định là null |
| Các kiểu dữ liệu nguyên thủy được định nghĩa bởi Java:  Numberic Type: byte, short, int, long, float, double.  Non-Numberic Type: char, boolean | Các kiểu dữ liệu Object được định nghĩa bởi Java:  Array, String, Integer, Double…  Các kiểu dữ liệu Object được định nghĩa bởi người dùng: User, Product,… |
| Xử lý dữ liệu nhanh và tốn ít bộ nhớ hơn | Object linh hoạt, phong phú chức năng và phù hợp với việc tùy chỉnh dữ liệu |

\* Có thể chuyển đổi kiểu dữ liệu nguyên thủy sang kiểu dữ liệu đối tượng (Wrapper Class) qua cơ chế AutoBoxing và UnBoxing

* Autoboxing: chuyển đổi kiểu dữ liệu từ kiểu nguyên thủy sang kiểu dữ liệu đối tượng (wrapper class)
* Unboxing: chuyển đổi kiểu dữ liệu từ kiểu đối tượng (wrapper class) về kiểu dữ liệu nguyên thủy

*Ví Dụ:*

*ArrayList<Integer> list = new ArrayList<>();*

*Int x = 10; // Biến x có kiểu dữ liệu nguyên thủy int*

*List.add(x); // Autoboxing tự động chuyển đổi kiểu dữ liệu nguyên thủy int sang kiểu đối tượng Integer*

*Int y = list.get(0); // Unboxing tự động chuyển đổi kiểu đối tượng Integer sang kiểu nguyên thủy int*

- Việc chuyển kiểu giữa dữ liệu nguyên thủy và kiểu dữ liệu đối tượng (Wrapper class) giúp xử lý dữ liệu tiện lợi hơn trong một số trường hợp như:

* Làm việc với Collection hoặc Generic → Yêu cầu kiểu dữ liệu đối tượng
* Giúp truy cập vào các phương thức xử lý dữ liệu của kiểu đối tượng
* Giúp khởi tạo giá trị ***Null*** vì đối tượng cho phép ***Null***

\* Không thể so sánh trực tiếp giữa kiểu dữ liệu nguyên thủy và kiểu dữ liệu đối tượng, nhưng có thể so sánh thông qua cơ chế AutoBoxing và UnBoxing

* Khi so sánh một dữ liệu nguyên thủy với kiểu dữ liệu đối tượng thì cơ chế Unboxing đối tượng về nguyên thủy được thực thi để so sánh xử dụng toán tử ‘ == ‘.

*Ví dụ:*

*Int a = 10;*

*Integer b = 10;*

*Boolen c = a == b; // Unboxing b với kiểu đối tượng Integer về kiểu nguyên thủy int*

* Khi so sánh kiểu đối tượng với kiểu nguyên thủy cơ chế Autoboxing thực thi để chuyển kiểu nguyên thủy sang kiểu đối tượng để so sánh sử dụng phương thức ‘ equals() ‘.

*Ví dụ:*

*Integer a = 10;*

*Int b = 10;*

*Boolen c = a.equals(b); // Autoboxing b với kiểu nguyên thủy int sang kiểu đối tượng Integer*

\* Giá trị khởi tạo của dữ liệu nguyên thủy và dữ liệu đối tượng

- ***Với kiểu dữ liệu nguyên thủy***

* Giá trị khi khởi tạo biến sẽ được khởi tạo mặc định khi khai báo trong lớp và không có giá trị mặc định khi khai báo trong phương thức mà phải được định nghĩa trước giá trị.

*VD: Các giá trị nguyên thủy mặc định khi khai báo trong lớp*

*Kiểu “ byte, short, int, long ” giá trị mặc định: 0*

*Kiểu “ float, double ” giá trị mặc định: 0.0*

*Kiểu “ char ” giá trị mặc định: ký tự null*

*Kiểu “ boolen ” giá trị mặc định: false*

- ***Với kiểu dữ liệu đối tượng***

* Với kiểu dữ liệu đối tượng như các “wrapper class” được khởi tạo mặc định là “null” khi khai báo trong lớp và không có giá trị mặc định khi khai báo trong phương thức mà phải được định nghĩa trước giá trị.