**Primitive Data Types và Wrapper Class**

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

* Java định nghĩa 8 kiểu dữ liệu nguyên thủy: byte, short, int, long,double, float, boolean và char**.** Tất cả các biến khác trong Java đều là kiểu đốitượng tham chiếu (**reference types**)
* Primitive data types trong Java là kiểu dữ liệu có giá trị cố định hoặc không đổi (**literals**)
* Primitive data types bắt buộc phải khai báo kiểu dữ liệu cho biến trước khi gán, nếu không sẽ xảy ra lỗi biên dịch (**compile error**)

VD:

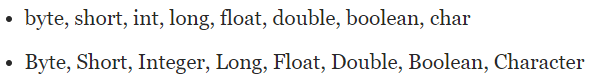
num = 10; -> Sai

int num = 10 -> Đúng

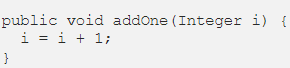
* Primitive data types có giá trị mặc định nếu không được khởi tạo (0, null, true, ….)

**2, Wrapper Class**

* Wrapper class là một đối tượng được đóng gói từ kiểu dữ liệu nguyên thủy. Tương ứng như sau:



* Wrapper class có thể thực thi những phương thức của lớp đối tượng (Object Class) như là hashCode(), equals(Object obj),….
* Wrapper class là kiểu dữ liệu tham chiếu nên giá trị mặc định sẽ là null
* Wrapper class là bất biến (immutable). Nếu wrapper class bị sửa đổi giá trị, trình biên dịch sẽ tạo 1 đối tượng mới rồi gán lại cho đối tượng ban đầu

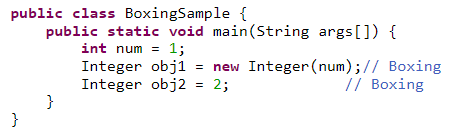


*Trong Java, các tham số có kiểu dữ liệu nguyên thủy đều là truyền tham trị. Các tham số có kiểu dữ liệu không phải là kiểu nguyên thủy đều là truyền tham chiếu*

**3, Autoboxing**

Autoboxing là quá trình chuyển đổi từ kiểu dữ liệu nguyên thủy (Primitive data types) về đối tượng tương ứng với lớp (Wrapper class) của kiểu dữ liệu đó (kiểu dữ liệu tham trị -> kiểu dữ liệu tham chiếu)

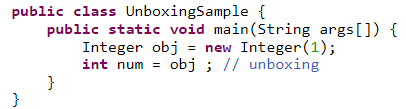
Ví dụ:



**4, Unboxing**

Unboxing là quá trình chuyển đổi từ các wrapper class sang kiểu dữ liệu nguyên thủy tương ứng (primitive data types)

Ví dụ:

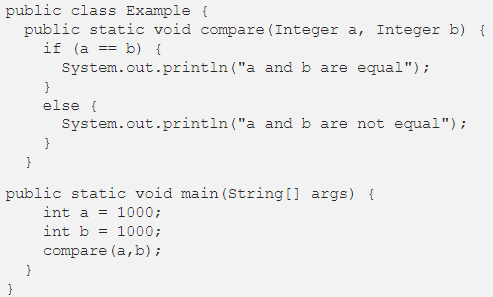


*Autoboxing và Unboxing tạo các Object trung gian khiến cho Garbage collections phải làm việc nhiều hơn có thể gây ảnh hưởng đến hiệu suất*

**5, Vấn đề của Autoboxing và Unboxing khi làm phép so sánh**

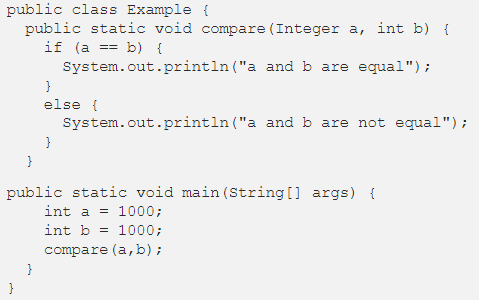
* Autoboxing và Unboxing không phải lúc nào cũng xảy ra với toán tử “==”
* Toán tử “==” để so sánh 2 đối tượng -> unboxing không xảy ra

-> Toán tử “==” sẽ so sánh đối tượng tham chiếu trong bộ nhớ



**-> a và b không bằng nhau**

* Toán tử “==” để so sánh 1 đối tượng và 1 kiểu dữ liệu nguyên thủy -> unboxing xảy ra -> Toán tử “==” sẽ so sánh giá trị



->  **a và b bằng nhau** (=1000)