Trong 1 phép tính thì bên phải là tính toán, còn bên trái không tính toán gì hết

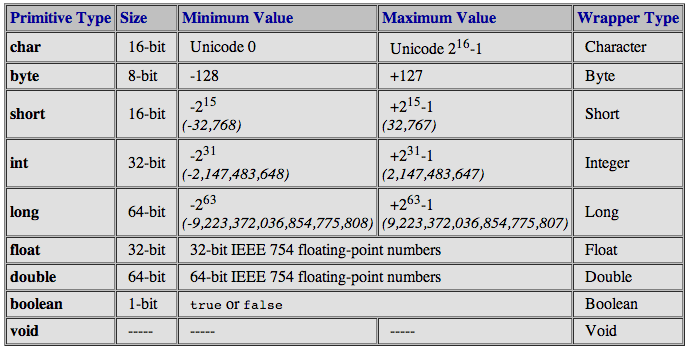
VD: b = 5 + a

1. **Datatype**

* Có 2 loại data type
  + Primitive (nguyên thủy) (đi kèm luôn có wrapper class)
  + Object (đối tượng)

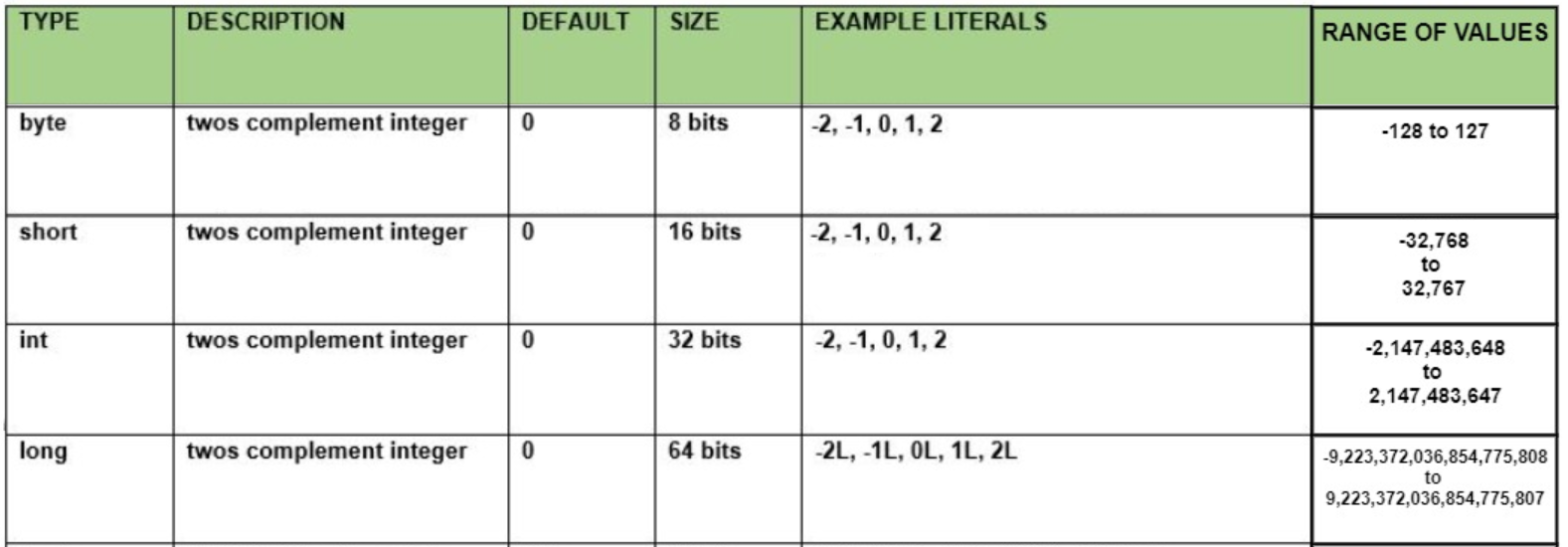
Người dùng chỉ tạo ra kiểu object

* 1. **Primitive**

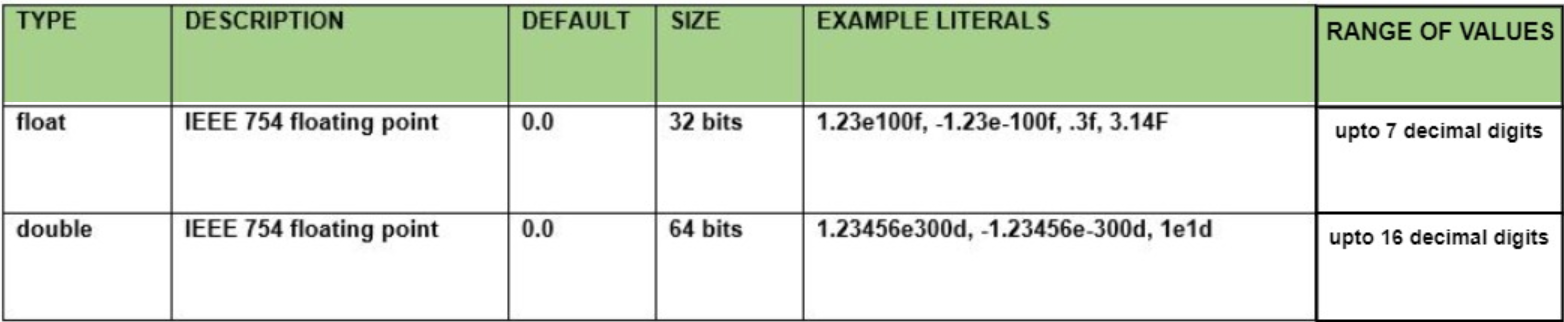


Các loại

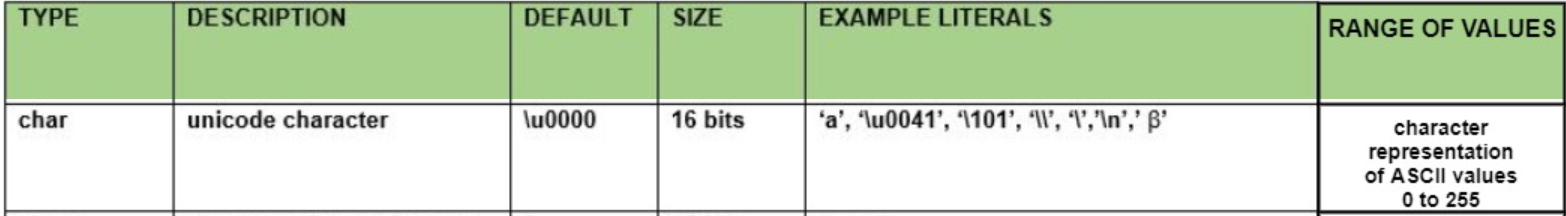
* + - **Số nguyên**
      * byte, short, int, long (Default = 0)

****

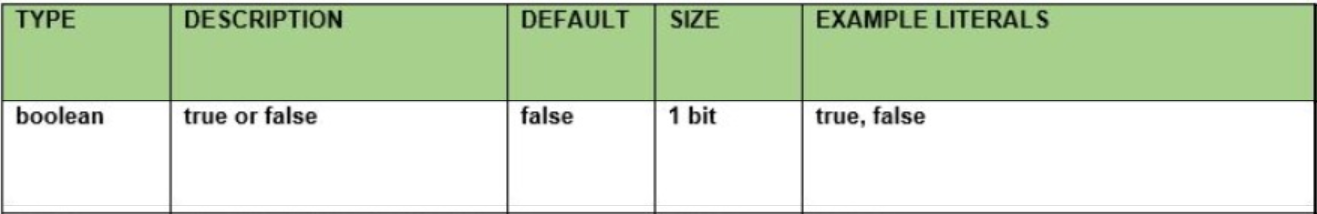
* + - **Số thực**
      * float, double (Default = 0)



* + - **Kí tự**
      * char: để lưu 1 kí tự
      * String: lưu 1 chuỗi kí tự
      * Default = null

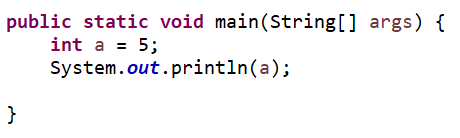


* + - **Logic**
      * boolean(1 bit, Default = false)

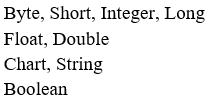


* + - **Đặc điểm**
      * Viết thường
      * Có phạm vi giá trị, có giá trị mặc định thuộc phạm vi của nó khởi tạo VD: byte thì sẽ khai báo được số từ -127 🡪 128
      * Khi khai báo int a = 5 thì java sẽ tìm trong bảng String constant xem có giá trị 5 không. Nếu có thì con trỏ a sẽ trỏ tới giá trị 5
      * So sánh thì dùng ==

VD:



* 1. **Wrapper class**

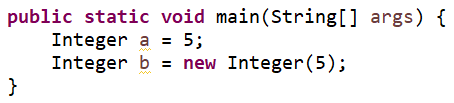


* + Là object của Java tạo và tích hợp thêm 1 vài method tiện ích

VD: các method để convert, toString, ...

* + Datatype: viết hoa chữ cái đầu
  + java sẽ tạo 1 đối tượng trong Heap Memory khi khởi tạo
  + So sánh thì có thể dùng cả equals và ==
  + Các default value giống primitive

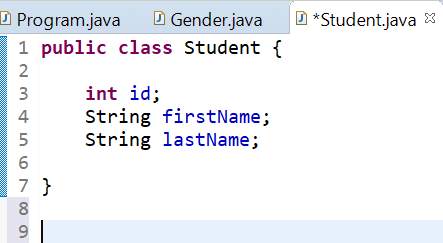
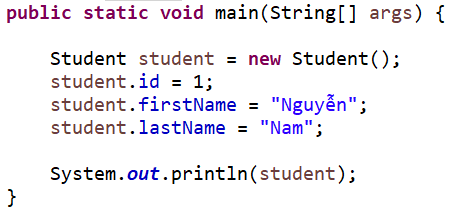
VD:

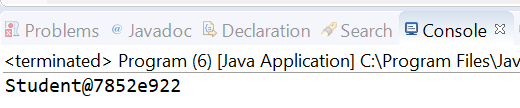


*Có thể khởi tạo bằng cả 2 cách*

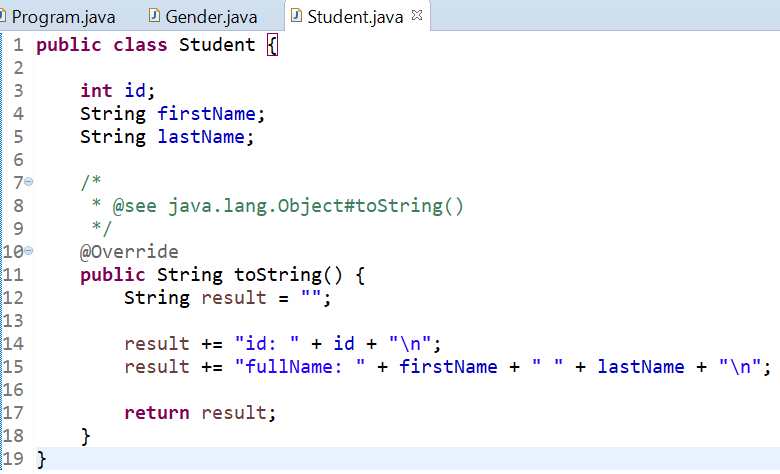
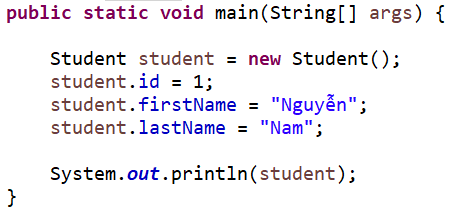
* 1. **Object**
* Là các object do người dùng tạo ra
* Mỗi object các property và object
* VD: department, account
* **Các method có sẵn trong object**
  + **toString()**

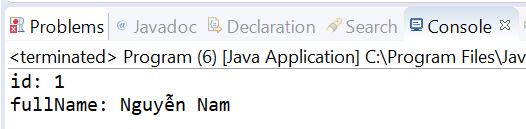
Để định nghĩa object in ra thông tin của mình



*Mặc định sẽ in ra địa chỉ con trỏ*

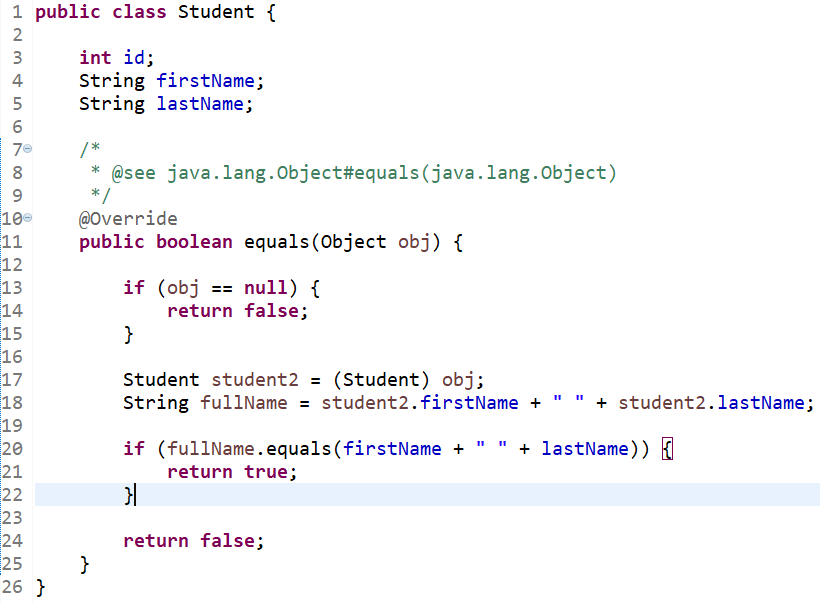
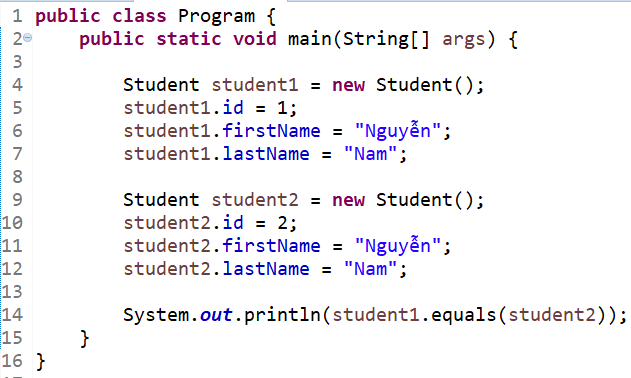
** **

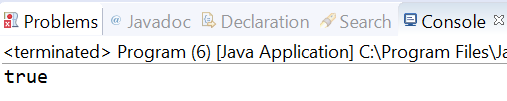
**

* + **equals()**

So sánh 2 object có bằng nhau hay không

VD:

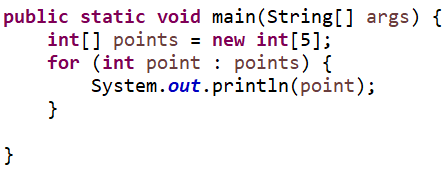
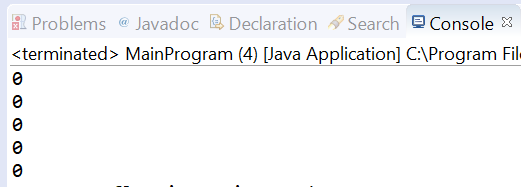
 

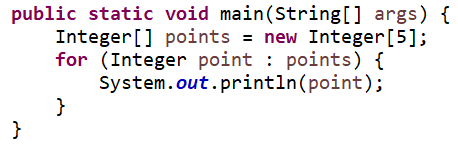
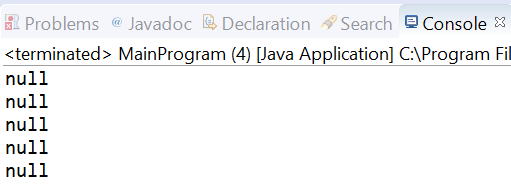


*Định nghĩa 2 Student bằng nhau khi 2 fullName bằng nhau*

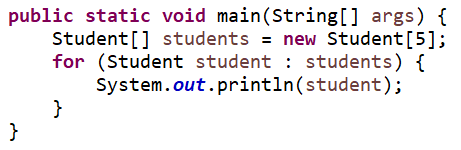
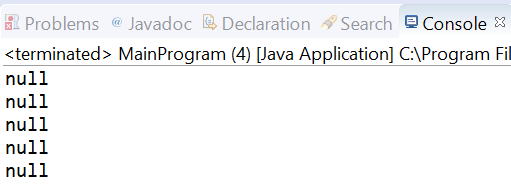
* + **Comparable & Comparator**
    - So sánh 2 object: lớn hơn, nhỏ hơn hoặc bằng
    - Có thể dùng để sắp xếp 1 list các object
    - Xem thêm trong file
  1. **Array**

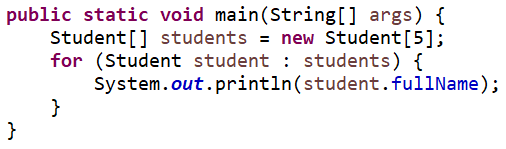
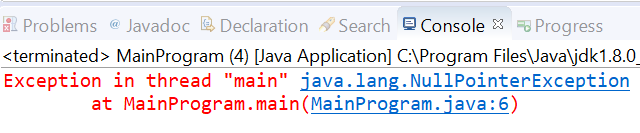
Khai báo con trỏ

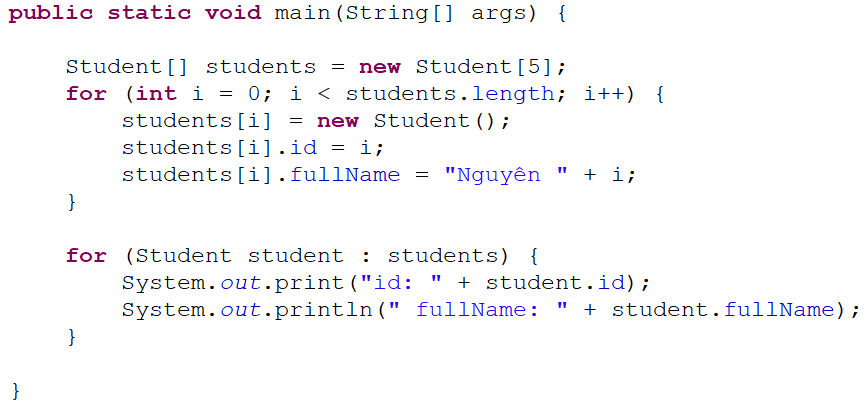
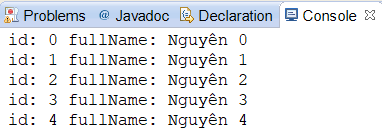
 

🡺 có thể sẽ gặp lỗi NullPointerException

🡺 Cách fix: phải khởi tạo mỗi phần tử trước khi dùng

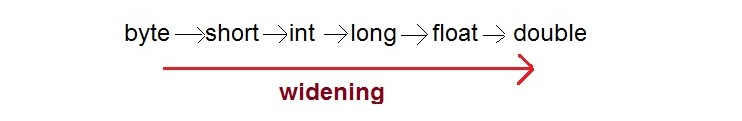
1. **Datatype Casting**

**Điều kiện:** 2 data type tương thích và có thể chuyển đổi qua nhau

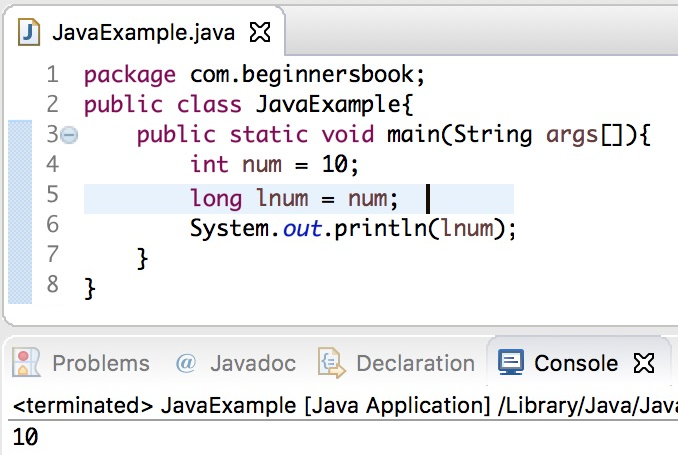
* **Widening** (implicit, up casting)
  + Bé 🡪 to 🡺 gây tốn bộ nhớ
  + Tự động ép kiểu, không cần phải khai báo rõ ràng
  + Đối với object: Super class 🡪 Sub Class (theo bộ nhớ)

VD1: int i = 1000;

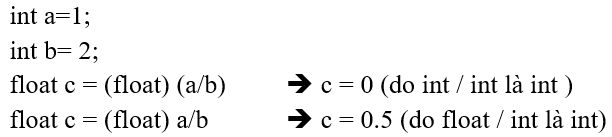
double j = i;

****

VD2:

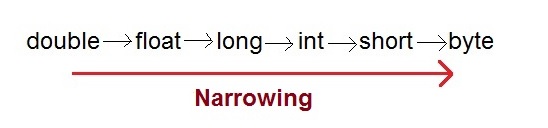


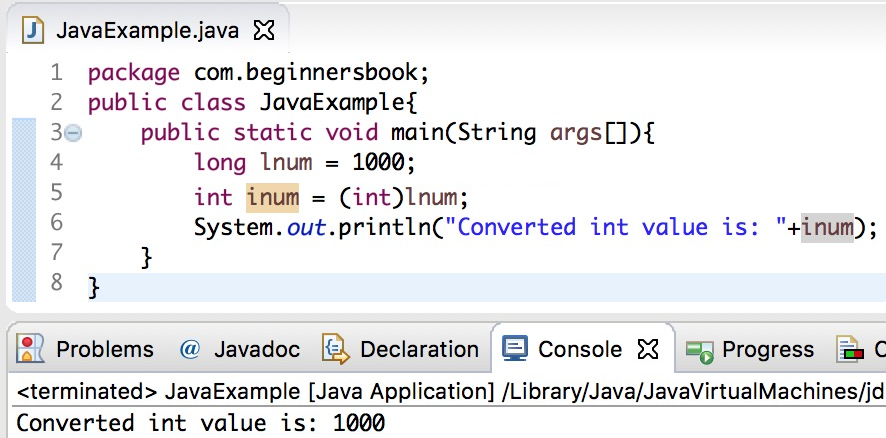
VD3:



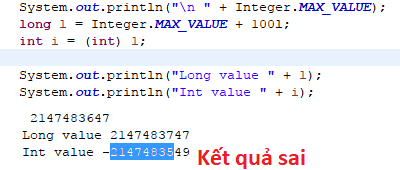
* **Narrowing** (explicit, down casting)
  + To 🡪 bé thì gây mất mát dữ liệu
  + Nếu datatype bé vượt quá giá trị có thể lưu thì kết quả sẽ sai
  + Đối với object: Sub Class 🡪 Super Class (theo bộ nhớ)

VD1:

****

****

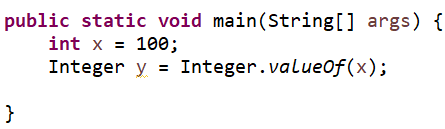
VD2:



* Note: Convert từ string 🡪 int thì sẽ theo bảng mã ASCII
* **Boxing**

Convert từ **Primitive 🡪 Object**

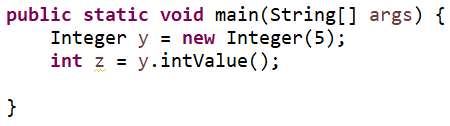
VD1:

****

* **Unboxing**

Convert từ **Object 🡪 Primitive**

VD1:



* **Primitive, Object 🡪 String**

VD:

Byte x = 100;

String s = Byte.toString(x);

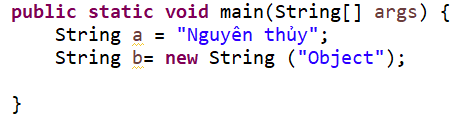
* **String 🡪 primitive, Object**

String x = "5";

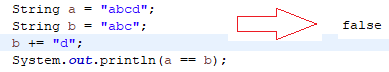
byte y =Byte.parseByte(x);

1. **String**

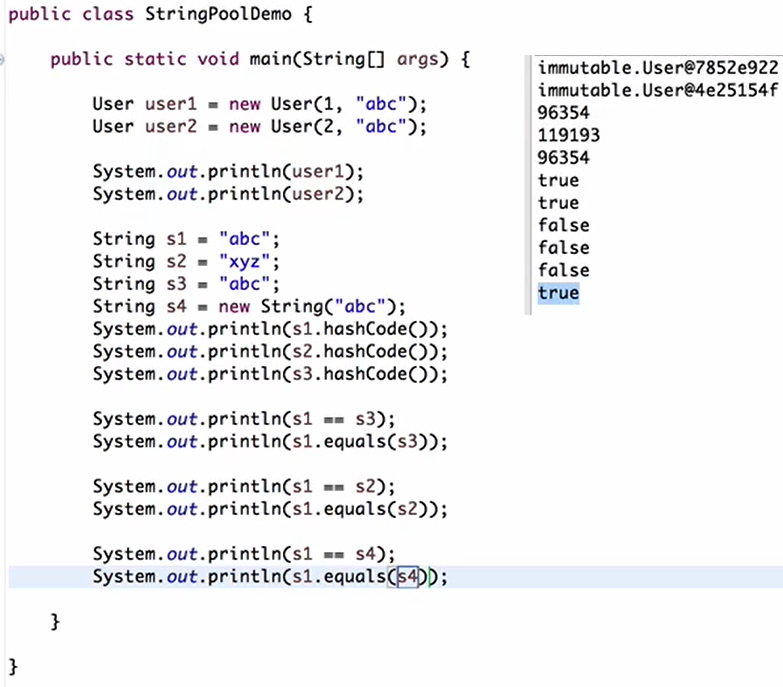
* Là 1 dãy char
* String vừa là primitive & wrapper class



* Khi khởi tạo thì Java sẽ lấy trong Spring pool, nhưng khi thực hiện phép tính thì Java sẽ tách riêng ra

* String & method hashCode()



[*https://www.journaldev.com/797/what-is-java-string-pool*](https://www.journaldev.com/797/what-is-java-string-pool)

* Khi so sánh thì nên sử dụng method equals()

Đối tượng sẽ so sánh giá trị dựa trên bảng mã ASCII của từng kí tự bên trong String

* Các method hay dùng: <https://vietjack.com/java/phuong_thuc_cua_lop_string_trong_java.jsp>
* **Reference**