



Biến và kiểu dữ liệu

Ngọc Lục

Kiểu dữ liệu (Types)

Kiểu dữ liệu



1

Các kiểu dữ liệu nguyên
thủy (Primitive Types)



2

Các kiểu dữ liệu tham
chiếu (Reference Types)



Kiểu primitive



Kiểu số nguyên

Java cung cấp 4 kiểu số nguyên khác nhau là: byte, short, int, long.

Kiểu dữ liệu	Miền giá trị	Giá trị mặc định	Kích cỡ mặc định
byte	-128 đến 127	0	1 byte
short	-32768 đến 32767	0	2 byte
int	-2^{31} đến $2^{31}-1$	0	4 byte
long	-2^{63} đến $2^{63}-1$	0L	8 byte

Kiểu số nguyên



```
byte a = 5;  
short b = 10;  
int c = 20;  
long d = 100L;
```

Kiểu số thực



Đối với kiểu dấu chấm động hay kiểu thực, java hỗ trợ hai kiểu dữ liệu là float và double. Số kiểu dấu chấm động không có giá trị nhỏ nhất cũng không có giá trị lớn nhất.

Kiểu dữ liệu	Giá trị mặc định	Kích cỡ mặc định
float	0.0f	4 byte
double	0.0d	8 byte

Kiểu số thực



```
float a = 2.5f;
```

```
double b = 2.5d;
```

```
double c = 2.5; //Vì double là kiểu mặc định cho kiểu số  
thực, nên có thể viết gọn hơn
```


Kiểu ký tự



Kiểu ký tự trong ngôn ngữ lập trình java có kích thước là 2 bytes và chỉ dùng để biểu diễn các ký tự trong bộ mã Unicode. Như vậy kiểu char trong java có thể biểu diễn tất cả $2^{16} = 65536$ ký tự khác nhau.

Giá trị mặc định cho một biến kiểu char là null

Giá trị nhỏ nhất của một biến kiểu ký tự là 0 và giá trị lớn nhất là 65535

```
char a = 'u';  
char b = '5';  
char c = 65; //c == 'A'
```

Kiểu luận lý



Kiểu boolean chỉ nhận 1 trong 2 giá trị: true hoặc false.
Trong java kiểu boolean không thể chuyển thành kiểu nguyên và ngược lại.

Giá trị mặc định của kiểu boolean là false

```
boolean a = true;  
boolean b = false;
```

Kiểu reference



Kiểu reference (Kiểu dữ liệu tham chiếu) là kiểu dữ liệu của đối tượng.

Một số kiểu dữ liệu cụ thể như mảng (Array), lớp đối tượng (Class) hay kiểu giao tiếp (Interface), kiểu String, ...

Biến (variable)

Biến là vùng nhớ dùng để lưu trữ các giá trị của chương trình. Mỗi biến gắn liền với một kiểu dữ liệu và một định danh duy nhất gọi là tên biến.



Cú pháp khai báo biến



Cú pháp khai báo biến:

```
<Kiểu dữ liệu> <Tên biến>;
```

Gán giá trị cho biến:

```
<Tên biến> = <Giá trị>;
```

Phân loại biến



Biến toàn cục

Được khai báo trong một lớp, bên ngoài constructor, phương thức, block. Được sử dụng với phạm vi truy cập

Biến cục bộ

Biến chỉ có thể truy xuất trong khối lệnh nó khai báo

Biến toàn cục (Biến instance)

```
public class Person {  
    public String name;  
    public int age;  
    public String address;
```

```
    /*
```

```
    Ba biến name, age, address là biến instance
```

```
    Trong đó với biến name và address có giá trị mặc định là null
```

```
    Biến age có giá trị mặc định là 0
```

```
    */
```

```
}
```


Biến cục bộ



```
public class Main {  
    public static void main(String[] args) {  
        int x = 10; //Biến cục bộ  
        System.out.println(x);  
    }  
}
```

Quy tắc khai báo biến



- Chỉ được bắt đầu bằng một ký tự(chữ), hoặc một dấu gạch dưới(_), hoặc một ký tự dollar(\$)
- Tên biến không được chứa khoảng trắng
- Bắt đầu từ ký tự thứ hai, có thể dùng ký tự(chữ), dấu gạch dưới(_), hoặc ký tự dollar(\$)
- Không được trùng với các từ khóa
- Có phân biệt chữ hoa và chữ thường

Lưu ý: Hãy đặt tên biến dễ nhớ, dễ đọc và có ý nghĩa

Ép kiểu đối với kiểu dữ liệu
nguyên thủy

Ép kiểu là gì?



Ép kiểu là cách chuyển biến thuộc kiểu dữ liệu này thành biến thuộc kiểu dữ liệu khác

Ý nghĩa:


- Việc chuyển kiểu dữ liệu sẽ đến lúc phải cần trong quá trình xử lý chương trình
- Có thể định dạng đúng kiểu dữ liệu mình mong muốn

Các cách ép kiểu




Ép kiểu trong kiểu dữ liệu nguyên thủy được chia làm 2 loại:

01 Chuyển đổi kiểu ngầm định (implicit)



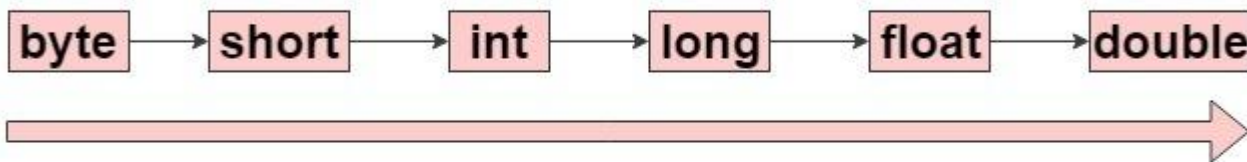
02 Chuyển đổi kiểu tường minh(explicit)



Kiểu chuyển đổi ngầm định

Việc chuyển đổi sẽ tự thực hiện bởi compiler và chúng ta không cần làm gì. Việc chuyển đổi này chỉ dành cho kiểu dữ liệu nhỏ sang kiểu dữ liệu lớn hơn. Ta có thể xem chiều từ nhỏ sang lớn như sau:

Automatic Type Conversion (Widening - implicit)



Kiểu chuyển đổi ngầm định

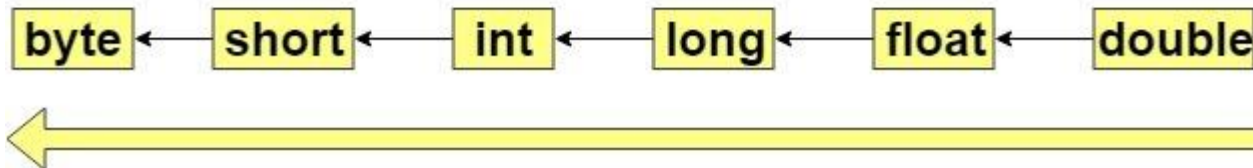


```
int a = 5;  
long b = a;  
System.out.print(b);
```

Kiểu chuyển đổi tường minh

Ngược lại với cách chuyển đổi ngầm định, việc chuyển đổi tường minh là chiều ngược lại từ kiểu dữ liệu lớn hơn sang kiểu dữ liệu nhỏ hơn (với điều kiện giá trị đó kiểu dữ liệu sẽ thay đổi có thể lưu trữ được trong kiểu dữ liệu mới).

Narrowing (explicit)



Kiểu chuyển đổi tường minh



```
long a = 6;  
int b = (int) a;  
System.out.print(a);
```

Chuỗi (String)



String

Trong java, String là một đối tượng biểu diễn một chuỗi các giá trị char

```
char[] ch = {'T', 'e', 'c', 'h', 'M', 'a', 's', 't', 'e', 'r'};  
String str = new String(ch);
```

```
String str = "TechMaster";
```



Tạo chuỗi String

Có hai cách để tạo đối tượng String:

01

Sử dụng String Literal

02

Sử dụng từ khóa new



Sử dụng String literal

String literal được tạo bằng cách sử dụng dấu nháy kép:

```
String name = "Cô giáo Ngọc";
```

Các đối tượng String được lưu trữ trong một khu vực bộ nhớ đặc biệt được gọi là String Constant Pool

Sử dụng String literal giúp cho việc sử dụng bộ nhớ hiệu quả hơn vì nếu chuỗi đã tồn tại trong Pool thì sẽ không có đối tượng mới được tạo ra



Sử dụng từ khóa new

```
String name = new String("Cô giáo Ngọc");
```

Trong trường hợp này, JVM sẽ tạo một đối tượng mới như bình thường trong bộ nhớ Heap (Không phải Pool) và hằng Techmaster sẽ được đặt trong Pool. Biến sẽ tham chiếu tới đối tượng trong Heap (Không phải Pool)



Lớp String trong Java

Lớp `java.lang.String` cung cấp rất nhiều phương thức để xử lý chuỗi. Các phương thức này giúp chúng ta thực hiện nhiều thao tác như cắt, ghép, chuyển đổi, so sánh, thay thế các chuỗi, ...



toUpperCase() và toLowerCase()

Phương thức toUpperCase() chuyển đổi chuỗi thành dạng chữ hoa và phương thức toLowerCase() chuyển đổi chuỗi thành dạng chữ thường.

```
String s="Hello Java";  
System.out.println(s.toUpperCase()); //HELLO JAVA  
System.out.println(s.toLowerCase()); //hello java  
System.out.println(s); //Hello Java
```




Phương thức trim()

Phương thức trim() được sử dụng để xóa khoảng trắng ở đầu và cuối của chuỗi

```
String s = "    Java    ";  
System.out.println(s); //    Java  
System.out.println(s.trim()); //Java
```



Phương thức length()

Phương thức length() trả độ dài của chuỗi.

```
String s="Hello Java";  
System.out.println(s.length()); //10
```



Phương thức equals()

Phương thức equals() được sử dụng để so sánh nội dung của 2 chuỗi.

- ❑ **public boolean equals(Object another)**
- ❑ **public boolean equalsIgnoreCase(String another)**

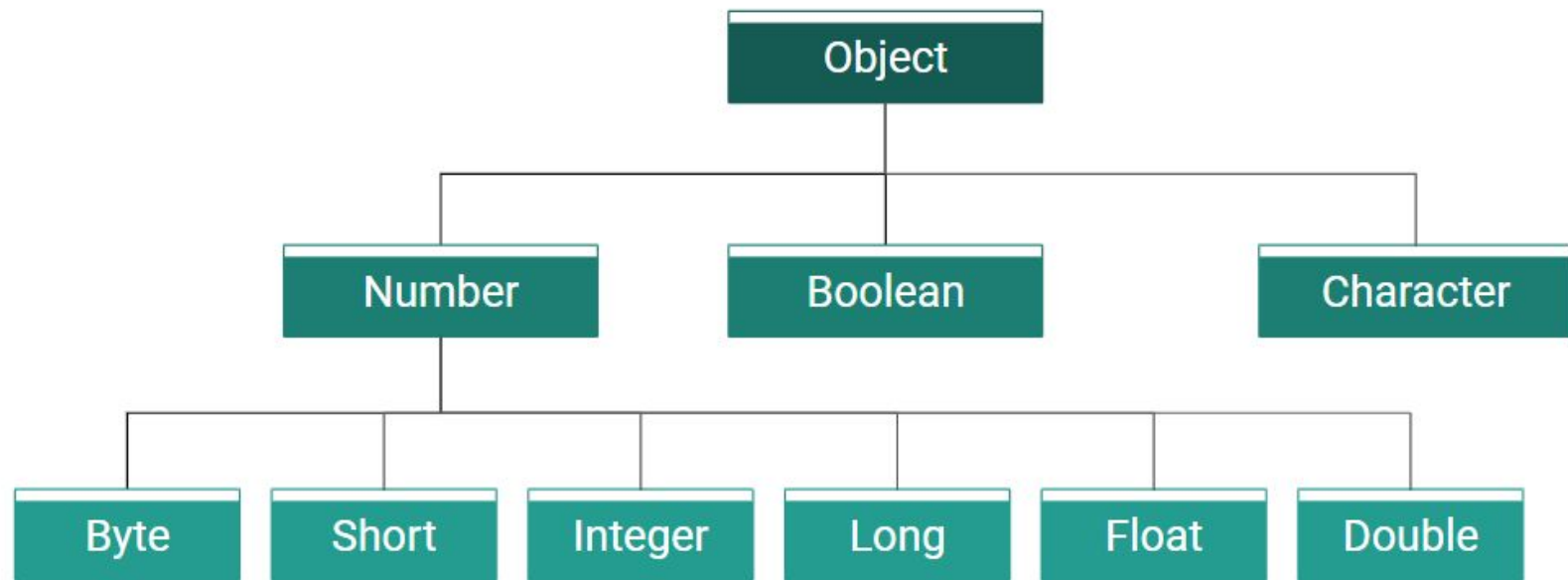
```
String s1 = "Hello";  
String s2 = "HELLO";  
String s3 = "Hello";  
  
System.out.println(s1.equals(s2)); //false  
System.out.println(s1.equals(s3)); //true  
System.out.println(s1.equalsIgnoreCase(s2)); //true
```

Lớp Wrapper



Lớp Wrapper là gì?

Lớp Wrapper thực chất chỉ là cái tên cho rất nhiều lớp khác nhau, trong Java có 8 kiểu dữ liệu nguyên thủy, nên cũng sẽ có 8 lớp wrapper cho từng kiểu dữ liệu nguyên thủy này, các lớp đều nằm trong package **java.lang**





Tác dụng của lớp Wrapper

- Lớp wrapper giúp chuyển kiểu dữ liệu qua lại giữa kiểu dữ liệu nguyên thủy và kiểu dữ liệu tham chiếu
- Cung cấp cơ chế **bọc** hoặc ràng buộc các giá trị của kiểu dữ liệu nguyên thủy vào một đối tượng
- Có thể cung cấp các giá trị null



Một số phương thức của lớp wrapper

Method	Description
<code>typeValue()</code>	Chuyển đổi một giá trị của lớp Wrapper nào đó về kiểu dữ liệu nguyên thủy
<code>compareTo()</code>	So sánh hai giá trị của hai lớp Wrapper (có cùng kiểu dữ liệu) với nhau và trả về giá trị là số nguyên
<code>equals()</code>	So sánh các giá trị của các lớp Wrapper và trả về một kiểu boolean
<code>compare()</code>	Tương tự phương thức <code>compareTo()</code> , tuy nhiên nó là phương thức static
<code>toString()</code>	Trả về là một chuỗi tương ứng với giá trị truyền vào
<code>parseType()</code>	Trả về giá trị nguyên thủy tương ứng với chuỗi truyền vào, Type tương ứng với kiểu dữ liệu