JAVA
- STRING
- STRINGBUFFER
- STRINGBUILDER
- STRINGTOKENIZER

Day 10 month 6 year 2017

By: Hang Long Moc

PHẦN 1:STRING

String

- String nằm trong lớp java.lang.String là lớp xử lí chuỗi chính trong java
- Đối tượng String là không thể thay đổi(immutable). Có thể thay đổi bằng cách tạo một đối tượng mới có giá trị mà bạn muốn đổi sang
  - Tình huống nếu ta cần thực hiện các thao tác như add, update, delete .v..v với một đối tượng String sẽ dẫn đến hàng loạt các đối tượng string khác trong bộ nhớ => Bộ thu gom rác(Garbage Collection) càng phải làm việc nhiều hơn để dọn dẹp lại bộ nhớ: StringBuffer và StringBuilder sẽ giúp ta giải quyết vấn đề này.

- Cấp phát thông dụng:
  - String string = “hello”;
  - String s1 = “hello”;
  - String string = new String(“Hello”);
  - String s1 = new String(“hello”);
    - Tốn bộ nhớ
- Sinh ra từ phương thức trong lớp char
  - String string = ch.toString();
  - Vi dụ: char[] ch = {1,2,3,4} => String s = ch.toString(); => s =“1234”;
- Truyền vào từ bàn phím:
  - String string = new Scanner(System.in).nextLine();
  - Chú ý: Đọc số chuỗi xen kẽ sẽ dẫn đến hiện tượng «trôi lệnh». Đây là vấn đề sâu về stream mà bản chất là kí tự «\n» còn tồn tại trong bộ nhớ đệm. QUY TẮC CHUNG: Thêm sc.nextLine();
  - System.out.print(“Nhập vào sô: ”); n = sc.nextInt();
  - Scanner sc = new Scanner(System.in);

- **int compareTo(Obejct obj)**
  - So Sánh string với obj
  - Thường dùng so sánh chuỗi: s1.compareTo(s2)
    - Trả về 0 nếu s1.equals(s2) == true
    - Trả về < 0 nếu đối số là một chuỗi kí tự từ điển lớn hơn
    - Trả về > 0 nếu đối số là một chuỗi kí tự từ điển nhỏ hơn
- **int compareToIgnoreCase(String str)**
  - So sánh 2 chuỗi theo thứ tự từ điển. Bỏ qua sự phân biệt hoa thường
- **boolean equals(Object obj)**
  - So sánh chuỗi hiện tại với obj. True nếu chúng tương ứng như nhau
  - equalsIgonoreCase(String annotherString): compateTo k phân biệt hoa thường

4. Phương Thức Trả Về

- **int length()**
  - string.length(): Trả về chiều dài của chuỗi
- **char charAt(int index)**
  - S.charat(index):Trả về kí tự tại index.
- **String copyValueOf(char[ ] ch)**
  - str = str.copyValueOf(ch):Trả về chuỗi đại diện cho mảng kí tự
  - Str = str.copyValueOf(ch, int start, int stop): Trả về chuỗi đại diện cho mảng kí tự bắt đầu từ start đến stop
  - Ch = {1,2,3,4} => str = str.copyValueOf(ch, 0, 2) => str = “12”; [0, 2)
- **boolean endsWith(String string)**
  - Trả về true nếu chuỗi kết thúc bằng string
- **String subString(int beginIndex, int endIndex)**
  - Trả về chuỗi mới là chuỗi con của string được lấy ra từ [beginIndex, endIndex)
  - Ví Dụ: Nơi này có anh

```
String s = "C://music//nhạc trịnh//Nơi này có anh.mp3";
int n = s.lastIndexOf("/");
int m = s.lastIndexOf(".");
String string = s.substring(n + 1, m);
System.out.println(string);
```

3. Phương Thức Trả Về

- **int hashCode()**
  - str.hashCode(): trả về mã băm của str
  - Nếu 2 chuỗi có cùng hashCode
- **int indexOf(char ch):**
  - **str.indexOf(ch):** Trả về chỉ số của sự xuất hiện đầu tiên của đối số đầu tiên trong str
  - **str.indexOf(char ch, int fromIndex):** Trả về chỉ số của ch đầu tiên trong str. Kiểm tra từ fromIndex
  - **str.indexOf(String string):** Trả về chỉ số của sự xuất hiện đầu tiên của subString trong str
  - **str.indexOf(String string, int fromIndex):** Được kiểm tra bắt đầu từ fromIndex
  - **str.lastIndexOf(Object obj):** obj in string,ch: Trả về sự xuất hiện cuối cùng của obj trong str
    - Có thể thêm đối số: int fromIndex

5. Phương Thức Thay Thế + Biến Đổi

- **String replace(char oldch, char newch)**
  - Phương thức trả về 1 chuỗi mới bằng việc thay thế tất cả sự xuất hiện của oldch bằng newch
  - replaceall(String oldStr, String newStr): Trả về chuỗi mới bằng việc thay thế **tất cả** sự xuất hiện của oldStr bằng newStr
  - replaceFirst(String oldStr, String newStr): Trả về chuỗi mới bằng việc thay thế sự xuất hiện đầu tiên của oldStr bằng newStr
- **String toLowerCase()**
  - Biến tất cả kí tự trong string thành chữ thường
- String toUpperCase()
  - Biến tất cả kí tự trong string thành chữ hoa
- String trim():
  - Cắt bỏ khoảng trắng 2 đầu
  - s = “ 124 “; s1 = s.trim(); s1 = “124”;
- *KHÔNG CÓ PHƯƠNG THỨC CHỈNH SỬA KÍ TỰ TRONG CHUỖI*

6. Phương Thức Cắt Chuỗi

- **String[ ] split(String regex)**
- **String[ ] splip(String regex, int limit)**
  - Cắt chuỗi theo regex

```
//Nếu có 1 cách chống
String s2 = "Nguyen Van Trung";
//Cắt hết
System.out.println("Cắt hết: ");
for(String str : s2.split(" "))
    System.out.println(str);
//Không cắt
System.out.println("Không cắt: ");
for(String str: s2.split(" ",1))
    System.out.println(str);
//Cắt làm hai sau cái đầu tiên
System.out.println("Cắt làm 2 sau dấu cách đầu: ")
for(String str: s2.split(" ",2))
    System.out.println(str);
```

```
Cắt hết:
Nguyen
Van
Trung
Không cắt:
Nguyen Van Trung
Cắt làm 2 sau dấu cách đầu:
Nguyen
Van Trung
```

- **String[] split(String regex)**

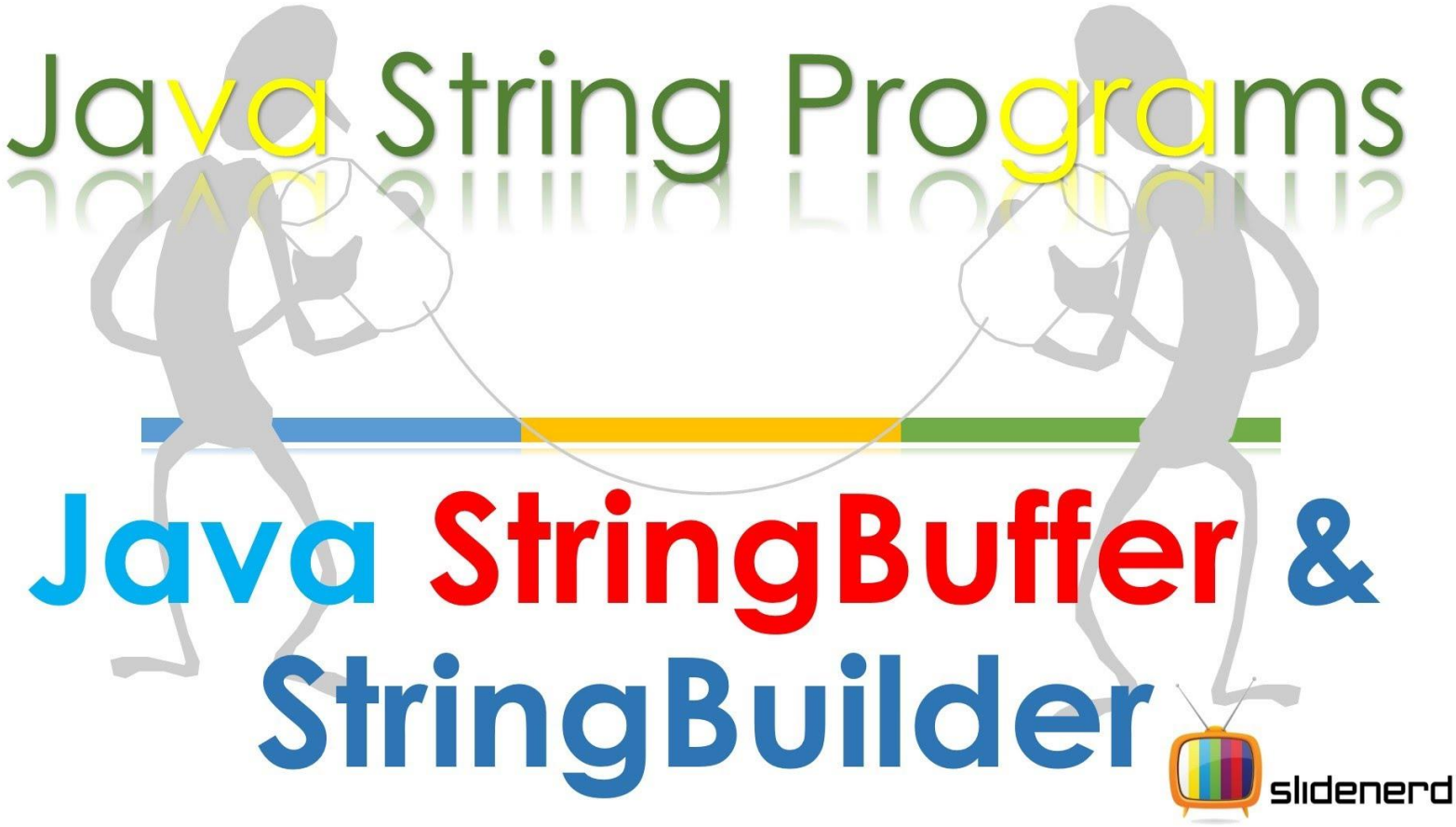
```
//Nếu có nhiều dấu cách trống quá
String s = "Nguyen    Van              Trung";
String[] arrS = s.split("\\s");

for(String str: arrS)
{
    if(str.equals("") == false)
        System.out.println(str);
}
```

- **String s3 = string1.concat(string2)**
  - Nối chuỗi s2 vào cuối s1.
  - Không làm thay đổi s1 và s2.
  - Có thể: s1 + s2 :D
- **String s = String.format(String arg0, Object…)**

```
String s = String.format("Double: %f \t Integer: %d
\t String: %s",1.2,3,"format");
/* Double: 1.2        Integer: 3              String: format*/
```

PHẦN 2:STRINGBUILDER-STRINGBUILDER



- Đối tượng thuộc StringBuffer và StringBuilder là mutable(Có thể thay đổi)
- Hai lớp này được thiết kế giống nhau, đó là thao tác hiệu quả với chuỗi, Chúng có các phương thức giống nhau để làm việc với chuỗi.
- Khác biệt đáng lưu ý:
  - StringBuffer thuộc loại synchronized => Methods đều là " Thread safe " (Thích hợp với xử lí đa luồng), trong khi StringBuilder lại ngược lại không synchronized. Với đặc tính "Thread safe" các methods của StringBuffer chạy chậm hơn StringBuilder.

PHẦN 2.1:STRINGBUFFER

Lớp **StringBuffe**r là an toàn luồng (Thread-safe)
Ví dụ: nhiều Thread không thể truy cập nó đồng thời. Vì thế nó là an toàn và sẽ cho kết quả trong một thứ tự.

StringBuffer

1. Các constructor

- StringBuffer():
  - Tạo một bộ đệm chuỗi trống với dung lượng capacity ban đầu là 16
  - Sb = “s”; capacity = 16, size =1: cap = (cap*2) + 2
- StringBuffer(String s):
  - Tạo một bộ đệm chuỗi với chuỗi đã xác định
- StringBuffer(int capacity):
  - Tạo một bộ đệm chuỗi trống với capacity đã cho

- StringBuffer append(Object o)
  - append(boolean arg0): **StringBuffer** - StringBuffer
  - append(char arg0):**StringBuffer** - StringBuffer
  - append(char[] arg0): **StringBuffer** - StringBuffer
  - append(double arg0):**StringBuffer** - StringBuffer
  - append(float arg0): **StringBuffer** - StringBuffer
  - append(int arg0):**StringBuffer** - StringBuffer
  - append(long arg0): **StringBuffer** - StringBuffer
  - append(Object arg0):**StringBuffer** - StringBuffer
  - append(String arg0): **StringBuffer** - StringBuffer
  - append(StringBuffer arg0):**StringBuffer** - StringBuffer
  - append(char[ ] arg0, int arg1, int arg2): **StringBuffer** - StringBuffer
  - append(CharSequence[ ] arg0, int arg1, int arg2):**StringBuffer** - StringBuffer
  - **Cách dùng - Ý nghĩa**:sb.append(obj): sb = sb + "obj";

```
StringBuffer sb = new StringBuffer("Nguyen Van“);
StringBuffer sb1 = new StringBuffer("Trung");
sb.append(sb1); /*sb = " Nguyen VanTrung"*/
```

2. Các Methods

- reverse()
  - Cách dùng - ý nghĩa: sb.reverse().
  - Ý nghĩa: Đảo ngược sb.

```
StringBuffer sb = new StringBuffer("Trung");
sb.reserve(sb); /*sb = “gnurT"*/
```

- delete(int start, int end)
  - Cách dùng :sb.delete(st, end).
  - Ý nghĩa: Xóa đi subString trong sb trên [start, end).
  - The same: sb.deleteCharAt(int index).

```
StringBuffer sb = new StringBuffer(“0123456");
sb.delete(0,4); /*sb = “3456"*/
```

- insert(int offset, Object obj)
  - Insert(int arg0, char arg1): StringBuffer – StringBuffer
  - Insert(int arg0, char[] arg1):StringBuffer – StringBuffer
  - Insert(int arg0, boolean arg1): StringBuffer – StringBuffer
  - Insert(int arg0, CharSequence arg1):StringBuffer - StringBuffer
  - Insert(int arg0, double arg1): StringBuffer – StringBuffer
  - Insert(int arg0, float arg1):StringBuffer - StringBuffer
  - Insert(int arg0, int arg1): StringBuffer – StringBuffer
  - Insert(int arg0, long arg1):StringBuffer – StringBuffer
  - Insert(int arg0, Object arg1): StringBuffer – StringBuffer
  - Insert(int arg0, String arg1):StringBuffer – StringBuffer
  - Insert(int arg0, char[] arg1,int arg2, int arg3): StringBuffer – StringBuffer
  - Insert(int arg0, charSequence arg1,int arg2, int arg3):StringBuffer-StringBuffer
  - **Cách dùng - Ý nghĩa**: Thêm obj vào vị trí chỉ định

```
StringBuffer sb = new StringBuffer(“0123456");
sb.insert(sb.length(), 789); /*sb = “0123456789"*/
sb.insert(sb.length(), true); /*sb = “0123456789true”*/
```

StringBuffer

2. Các Methods

- replace(int start, int end, String s)
  - **Cách dùng**: sb.replace(arg0, arg1, s)
  - **Công dụng**: Hình thành chuỗi mới bằng việc thay subString [arg0, arg1) bằng s.

```
StringBuffer sb = new StringBuffer(“0123456");
sb.replace(0, 2,“replace”);/*sb = “replace23456789"*/
```

- char charAt(int index)
  - **Cách dùng**: sb.charat(index)
  - **Công dụng**: Trả về kí tự tại chỉ mục

```
StringBuffer sb = new StringBuffer(“0123456");
char ch = sb.charat(0);/*ch = ‘0’*/
```

2. Các Methods

- setCharAt(int index, char ch)
  - **Cách dùng**: sb.setCharat(index, ch)
  - **Công dụng**: Sửa kí tự tại index bằng việc thay thế ch vào

```
StringBuffer sb = new StringBuffer(“0123456");
sb.setCharat(0, ’s’);/*sb = “s123456789"*/
```

- int lastIndexOf(String s)
  - **Cách dùng**: int index = sb.lastIndexOf(s)
  - **Công dụng**: Trả về nơi mà s xuất hiện cuối cùng

```
StringBuffer sb = new StringBuffer(“06123456");
int index = sb.lastIndexOf(“6”);/*index = 7*/
```

2. Các Methods

- String toString()
  - **Cách dùng**: String s = sb.toString()
  - **Công dụng**: Biến kiểu sb -> s.

```
StringBuffer sb = new StringBuffer(“0123456");
String s = sb.toString();/*s = “0123456"*/
```

- String subString(int start, int end)
  - **Cách dùng**: String s = sb.subString(start, end)
  - **Công dụng**: Trả về subString từ chỉ mục [start,end)

```
StringBuffer sb = new StringBuffer(“0123456");
String s = sb.subString(0, 2);/*s = “01”*/
```

PHẦN 2.2:STRINGBUILDER

StringBuilder mới được giới thiệu từ phiên bản java 1.5

Có thể nói các method của StringBulider và StringBuffer là tương tự nhau
Tuy nhiên:

- Với đặc tính "Thread safe" các method của StringBuffer chạy chậm hơn methods của StringBuilder

StringBuffer

PHẦN 3: STRINGTOKENIZER