BKACAD®

HỌC VIỆN CÔNG NGHỆ BKACAD

Lập trình Java – String

Ths. Vũ Duy Khương

Nội dung

1. Giới thiệu Lớp String
2. Xử lý chuỗi sử dụng các lớp String
3. Ví dụ và Bài tập về String

Lớp String trong java (1)

- Lớp **String trong java** cung cấp rất nhiều các phương thức để thực hiện các thao tác với chuỗi như: **compare**(), **concat**(), **equals**(), **split**(), **length**(), **replace**(), **compareTo**(), **intern**(), **substring**(), ...
- Xây dựng chuỗi:

  String message = "Welcome to Java!"

  String message = new String("Welcome to Java!“);

  String s = new String();

Lớp String trong java (2)

- Lấy độ dài chuỗi và lấy các ký tự cụ thể trong chuỗi.
- Ghép chuỗi (concat)
- Chuỗi con (substring(index), substring(start, end))
- So sánh (equals, compareTo)
- String Conversions
- Tìm 1 ký tự hoặc chuỗi con trong 1 chuỗi
- Chuyển đổi giữa Strings và Arrays
- Chuyển đổi các ký tự và các g/t số thành chuỗi

Lớp String trong java (3)

No	Phương thức	Mô tả
1	char charAt(int index)	Trả về giá trị char cho chỉ số cụ thể.
2	int length()	Trả về độ dài chuỗi.
3	static String format(String format, Object... args)	Trả về chuỗi được format.
4	static String format(Locale l, String format, Object... args)	Trả về chuỗi được format theo vùng miền(quốc gia).
5	String substring(int beginIndex)	Trả về chuỗi con bắt đầu từ chỉ số index.
6	String substring(int beginIndex, int endIndex)	Trả về chuỗi con từ chỉ số bắt đầu đến chỉ số kết thúc.
7	boolean contains(CharSequence s)	Kiểm tra chuỗi chứa chuối không, kết quả trả về là giá trị boolean.
8	static String join(CharSequence delimiter, CharSequence... elements)	Trả về chuỗi được nối từ nhiều chuỗi.
9	boolean equals(Object another)	kiểm tra sự tương đương của chuỗi với đối tượng.
10	boolean isEmpty()	Kiểm tra chuỗi rỗng.

Lớp String trong java (4)

No	Phương thức	Mô tả
11	String concat(String str)	Nối chuỗi cụ thể.
12	String replace(char old, char new)	Thay thế tất cả giá trị char cụ thể bằng một giá trị char mới.
13	static String equalsIgnoreCase(String another)	So sanh chuỗi, không phân biệt chữ hoa hay chữ thường.
14	String[] split(String regex)	Trả về mảng các chuỗi được tách ra theo giá trị regex.
15	int indexOf(int ch)	Trả về vị trí của ký tự ch cụ thể.
16	int indexOf(String substring)	Trả về vị trí của chuỗi con substring.
17	String toLowerCase()	Trả về chuỗi chữ thường.
18	String toUpperCase()	Trả về chuỗi chữ hoa.
19	String trim()	Xóa khoảng trắng ở đầu và cuối của chuỗi.
20	static String valueOf(int value)	Chuyển đổi giá trị kiểu dữ liệu đã cho thành chuỗi.

Ví dụ lớp String trong java (1)

```
String name = "Hello java";
char ch = name.charAt(4);
String sf1 = String.format("name is %s", name);
String sf2 = String.format("value is %f", 32.33434);
String joinString1 = String.join("-", "welcome", "to", "java");
System.out.println(ch);                          // Hiển thị ký tự "o"
System.out.println(name.length());               // Hiển thị độ dài 10
System.out.println(sf1);                         // Hiển thị "name is Hello java"
System.out.println(sf2);                         // Hiển thị "value is 32.334340"
System.out.println(name.substring(3));           // Hiển thị        "lo java"
System.out.println(name.substring(3, 7));        // Hiển thị        "lo ja"
```

Ví dụ lớp String trong java (2)

```
String name = "Hello java";
System.out.println(name.contains("java"));                    // True
System.out.println(joinString1);                              // welcome-to-java
System.out.println(name.equals("java"));                      // True
System.out.println(name.equals("Java"));                      // False
System.out.println(name.isEmpty());                           // False
System.out.println(name.concat("string is immutable"));       // Hello java string is immutable
System.out.println(name.replace('a','o'));                    // Hello jovo
System.out.println(name.equalsIgnoreCase("JAVA"));            // True
```

Ví dụ lớp String trong java (3)

```
String name = "Hello java";
String[] words = name.split("\\s");
for (String w : words) {
        System.out.println(words);                    // Hello
}                                                     // java
System.out.println(name.indexOf("java"));                    // 6
System.out.println(name.indexOf("a",8));                     // 9
System.out.println(name.toLowerCase());                      // hello java
System.out.println(name.toUpperCase());                      // HELLO JAVA
```

Xây dựng chuỗi/string (1)

- Thông thường, string là một chuỗi các ký tự.
- Nhưng, trong java string là một đối tượng biểu diễn một nối tiếp của các ký tự. Lớp java.lang.String được sử dụng để tạo đối tượng string

- **Có 2 cách để tạo đối tượng String:**

1. Sử dụng string literal
2. Sử dụng từ khóa new

Sử dụng string literal (1)

1. Sử dụng string literal

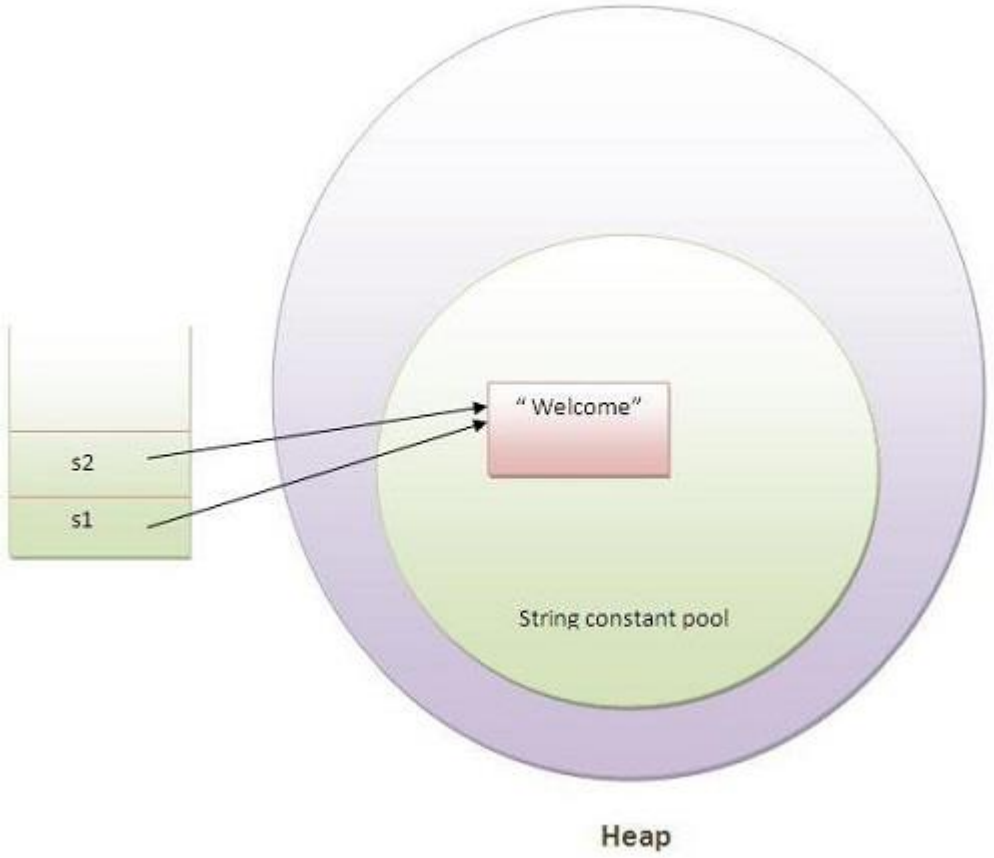
- String literal được tạo ra bằng cách sử dụng 2 dấu nháy kép.
- Ví dụ:

```
String s = "welcome";
```

Chú ý: Các đối tượng String được lưu trong một vùng nhớ đặc biệt đó là Pool hằng số chuỗi.

Sử dụng string literal (2)

String s1 = "welcome";
String s2 = "welcome"; // se khong tao instance moi



Tại sao java sử dụng string literal?

Để làm cho Java sử dụng bộ nhớ hiệu quả hơn

Sử dụng từ khóa new

2. Sử dụng từ khóa new

String s=new String("Welcome"); // Tạo 2 đối tượng và 1 biến tham chiếu

➔ JVM sẽ tạo ra 1 đối tượng string mới như 1 đối tượng trong bộ nhớ HEAP

Chú ý

➢ CHÚ Ý: Một đối tượng String là không thể thay đổi nội dung. Xét ví dụ sau:

```
String s = "Java";
s = "HTML";
```

Ví dụ trên: đối tượng String lưu chuỗi "Java" vẫn tồn tại nhưng không được tham chiếu bởi biến nào cả.

➢ Nếu sử dụng lệnh tắt tạo 2 đối tượng String với cùng nội dung, JVM lưu 2 đối tượng đó vào trong cùng một đối tượng để cải thiện hiệu năng và tiết kiệm bộ nhớ.

➔ Do đó, người ta thường sử dụng lệnh tắt để tạo các chuỗi.

Ví dụ

```
String s = "Welcome to Java!";
String s1 = new String("Welcome to Java!");
String s2 = s1.intern();
System.out.println("s1 == s is " + (s1 == s));
System.out.println("s2 == s is " + (s2 == s));
System.out.println("s1 == s2 is " +(s1 == s2));
```

Hiển thị:

```
s1 == s is false
s2 == s is true
s1 == s2 is false
```

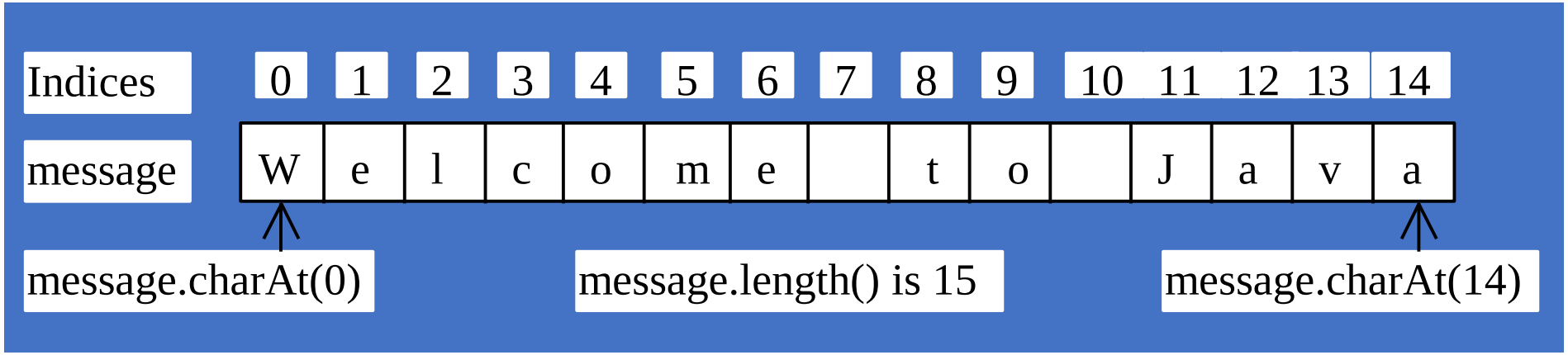
Lấy độ dài chuỗi

Sử dụng phương thức **`length()`** :

```
message = "Welcome";
message.length() (returns 7)
```

Lấy các ký tự trong chuỗi

- Không sử dụng `message[0]`
- Mà dùng `message.charAt(index)`
- Chỉ số (index) bắt đầu từ 0



Ghép chuỗi

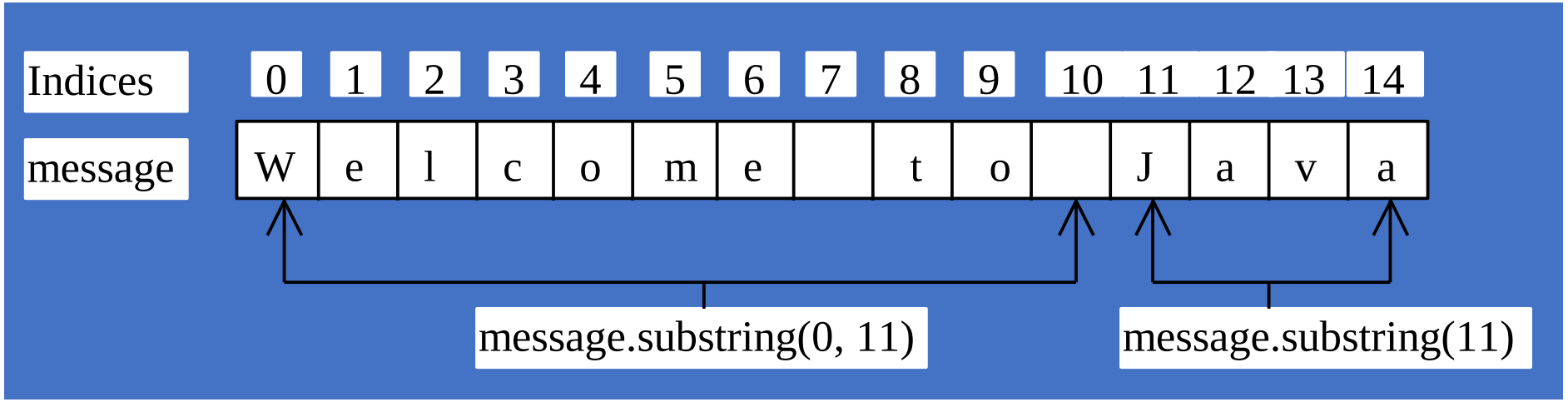
```
String s3 = s1.concat(s2);
```

String s3 = s1 + s2;

Trích chuỗi con

`String` là một lớp không thể thay đổi; giá trị của nó không bị thay đổi một cách riêng lẻ.

```
String s1 = "Welcome to Java";
String s2 = s1.substring(0, 11) + "HTML";
```



So sánh chuỗi

- **equals**

Cú pháp: *public* ***boolean equals****(Object anotherObject)*

```
String s1 = "Welcome to Java!";
String s2 = new String("Welcome to Java!”);
 if (s1.equals(s2)){
   // s1 and s2 have the same contents
 }

 if (s1 == s2) {
   // s1 and s2 have the same reference
 }
```

Chú ý: `s1.equals(s2)` là True khi và chỉ khi `s1.intern() == s2.intern()`.

So sánh chuỗi (tiếp)

- **compareTo**(Object object)

Cú pháp: *public* ***int compareTo(String*** *anotherString)*

```
String s1 = "Welcome";
String s2 = "welcome";
 if (s1.compareTo(s2) > 0) {
   // s1 is greater than s2
 }else if (s1.compareTo(s2) == 0) {
   // s1 and s2 have the same reference
 }
 else
   // s1 is less than s2
```

String Conversions

Nội dung của một chuỗi không thể thay đổi mỗi khi chuỗi được tạo. Nhưng bạn có thể convert một chuỗi thành một chuỗi mới bằng cách sử dụng các phương thức sau:

- `"Welcome".toLowerCase()`
- `"Welcome".toUpperCase()`
- `"  Welcome ".trim()`
- `"Welcome".replace('e','A')`
- `"Welcome".replaceFirst('e','A')`
- `"Welcome".replaceAll('e','A')`

Tìm ký tự và chuỗi con

Sử dụng các phương thức sau:

```
public int indexOf(int ch)
public int lastIndexOf(int ch)
public int indexOf(int ch, int fromIndex)
public int lastIndexOf(int ch, int endIndex)
public int indexOf(String str)
public int lastIndexOf(String str)
public int indexOf(String ch, int fromIndex)
public int lastIndexOf(String str, int endIndex)
```

Ví dụ

```
"Welcome to Java!".indexOf('W') returns 0.

"Welcome to Java!".indexOf('x') returns -1.

"Welcome to Java!".indexOf('o', 5) returns 9.

"Welcome to Java!".indexOf("come") returns 3.

"Welcome to Java!".indexOf("Java", 5) returns 11.

"Welcome to Java!".indexOf("java", 5) returns -1.
```

Chuyển đổi ký tự và số thành chuỗi

- `char[] chars = "Java".toCharArray();`
- `String str = new String(new char[] {'J','a','v','a'});`
- Lớp String cung cấp một số phương thức static valueOf để chuyển đổi một ký tự, mảng ký tự, và các giá trị số thành chuỗi.
- Những phương thức này có cùng tên valueOf với tham số khác nhau có kiểu char, char[], double, long, int, và float.

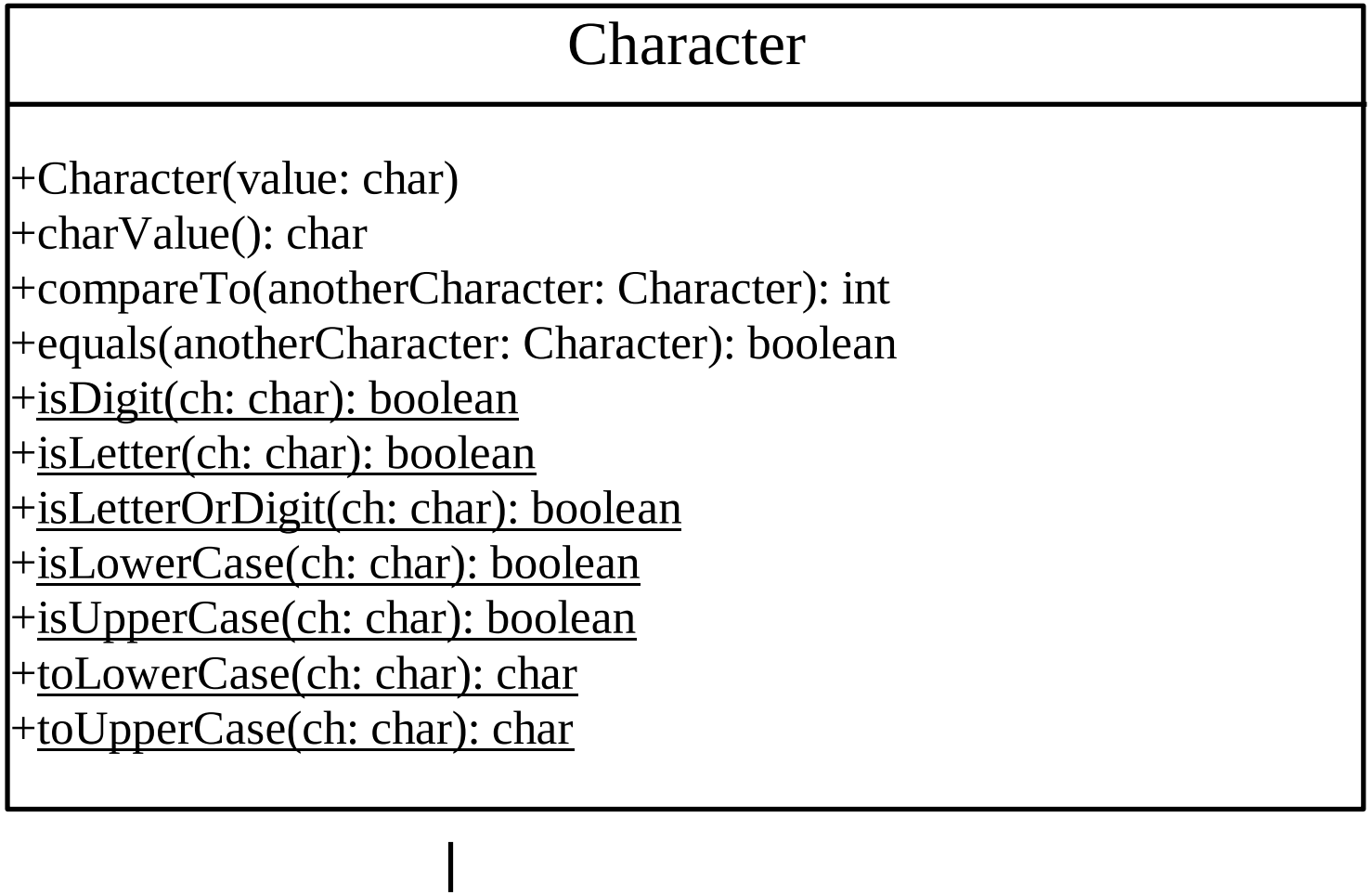
```
String str = String.valueOf(5.44);
String str = String.valueOf(new char[] {'J','a','v','a'});
```

Ví dụ 1: Tìm chuỗi đối xứng

- Mục tiêu: Kiểm tra xem một chuỗi có đối xứng hay không.
- Ví dụ: Chuỗi 1234321 ➔ Là chuỗi đối xứng

  Chuỗi 12345321 ➔ Không đối xứng

Lớp Character



Ví dụ 2

```
Character charObj = new Character('b');

charObj.compareTo(new Character('a')) returns 1

charObj.compareTo(new Character('b')) returns 0

charObj.compareTo(new Character('c')) returns -1

charObj.compareTo(new Character('d') returns -2

charObj.equals(new Character('b')) returns true

charObj.equals(new Character('d')) returns false
```

Ví dụ 3: Đếm mỗi ký tự trong chuỗi

Mục tiêu: đếm tần số xuất hiện của mỗi ký tự trong một chuỗi. Giả sử không phân biệt ký tự hoa, thường.

Lớp StringBuffer (1)

➢ Lớp `StringBuffer` là một lựa chọn khác của lớp `String`. Nói chung, một string buffer có thể được sử dụng thay thế cho một string.

➢ `StringBuffer` linh hoạt hơn `String`. Bạn có thể add, insert, hoặc append nội dung mới vào một string buffer. Với một string thì không thể thay đổi nội dung nó được tạo.

Lớp StringBuffer (2)

StringBuffer
+append(data: char[]): StringBuffer +append(data: char[], offset: int, len: int): StringBuffer +append(v: *aPrimitiveType*): StringBuffer +append(str: String): StringBuffer +capacity(): int +charAt(index: int): char +delete(startIndex: int, endIndex: int): StringBuffer +deleteCharAt(int index): StringBuffer +insert(index: int, data: char[], offset: int, len: int): StringBuffer +insert(offset: int, data: char[]): StringBuffer +insert(offset: int, b: *aPrimitiveType*): StringBuffer +insert(offset: int, str: String): StringBuffer +length(): int +replace(int startIndex, int endIndex, String str): StringBuffer +reverse(): StringBuffer +setCharAt(index: int, ch: char): void +setLength(newLength: int): void +substring(start: int): StringBuffer +substring(start: int, end: int): StringBuffer

StringBuffer Constructors

- **`public StringBuffer()`**

  Xây dựng một string buffer rỗng có dung lượng = 16.

- **`public StringBuffer(int length)`**

  Xây dựng một string buffer rỗng có dung lượng = length.

- **`public StringBuffer(String str)`**

  Xây dựng một string buffer với nội dung là chuỗi `str`. Dung lượng khởi tạo bằng `16 + str.length()`.

Sửa đổi String Buffer

Giả sử `strBuf` chứa chuỗi "Welcome to Java!" trước mỗi lời gọi phương thức sau:

```
strBuf.insert(11,"HTML and ");

strBuf.delete(8,11);

strBuf.deleteCharAt(8);

strBuf.reverse();

str.replace(11,15,"HTML");

strBuf.setCharAt(0,'w');
```

Ví dụ 4: Kiểm tra chuỗi đối xứng, bỏ qua ký tự khác chữ và số

Về nhà làm

Các Constructor của lớp StringTokenizer

Chuỗi có thể được bẻ thành các mảnh gọi là token bởi các delimiter.

- **`StringTokenizer(String s, String delim, boolean returnTokens)`**
- **`StringTokenizer(String s, String delim)`**
- **`StringTokenizer(String s)`**

Ví dụ: `String s = "Java is cool."`

```
StringTokenizer tkz = new StringTokenizer(s);
StringTokenizer tkz = new StringTokenizer(s,"ac");
StringTokenizer tkz = new StringTokenizer(s,"ac",true);
```

Các phương thức của lớp StringTokenizer

- **boolean hasMoreTokens()**
- **String nextToken()**
- **String nextToken(String delim)**

Ví dụ: `String s = "Java is cool."`

```
StringTokenizer tkz = new StringTokenizer(s);
System.out.println("Tong so token = " + tkz.countTokens());
while (tkz.hasMoreTokens())
  System.out.println(tkz.nextToken());
```

Lớp Scanner (1)

- Có từ JDK 1.5 (gói java.util.Scanner)
- Có thể dùng lớp Scanner để:
  - ấn định 1 từ làm Delimiter
  - nhập dữ liệu thuộc các kiểu khác nhau
- Ấn định 1 từ làm Delimiter:

```
String s = "Java is fun! Java is cool!";
Scanner scr = new Scanner(s);
scr.useDelimiter("Java");
while (scr.hasNext())
 System.out.println(scr.next());
```

Lớp Scanner (2)

- Nhập dữ liệu:
  - Tạo đối tượng Scanner để đọc dữ liệu vào từ System.in

```
Scanner scanner = new Scanner(System.in)
```

  - Sử dụng các phương thức **`next()`**, **`nextByte()`**, **`nextShort()`**, **`nextInt()`**, **`nextLong()`**, **`nextFloat()`**, **`nextDouble()`**, hoặc **`nextBoolean()`** để lấy một chuỗi, byte, short, int, long, float, double, hoặc boolean.
  - Nên tạo 1 lớp (ví dụ MyInput) gồm các phương thức đọc dữ liệu kiểu string và các kiểu cơ bản, đặt cùng thư mục Chương Trình.

Các tham số dòng lệnh

- Có thể truyền chuỗi cho phương thức main từ dòng lệnh khi chạy chương trình.
- Khi phương thức main được gọi, trình biên dịch Java tạo 1 mảng chứa các tham số dòng lệnh và truyền tham chiếu mảng cho args.

```
class TestMain {
  public static void main(String[] args) {
  ...
  }
}
```

```
java TestMain arg0 arg1 arg2 ... argn
```

Xử lý các tham số dòng lệnh

- Các chuỗi được truyền cho chương trình chính được chứa trong args - là một mảng các chuỗi. Các phần tử: `args[0]`, `args[1]`, `...`, `args[n]`, tương ứng với `arg0`, `arg1`, `...`, `argn` trong dòng lệnh.
- `args.length` là số chuỗi được truyền.

Ví dụ 4: Sử dụng các tham số dòng lệnh

- Mục tiêu: Viết chương trình thực hiện các phép toán 2 ngôi trên số nguyên. Chương trình nhận vào 3 tham số: 1 toán tử và 2 số nguyên.
- Ví dụ:

```
java –jar Calculator 10 + 5
java –jar Calculator 10 - 5
Java -jar Calculator 10 / 5
java –jar Calculator 10 "*" 5
```

Q&A

BKACAD®

HỌC VIỆN CÔNG NGHỆ BKACAD

THANK YOU