**Thư viện xử lý chuỗi trong Java**

**1. StringBuilder**

Trong Java, StringBuilder được sử dụng để quản lý một chuỗi mà có thể thay đổi kích thước và nội dung. Sau đây tôi sẽ giới thiệu cách khởi tạo 1 StringBuilder, các phương thức chính của StringBuilder và ví dụ minh họa các phương thức này.

### **Tạo mới 1 StringBuilder**

Cú pháp khai báo StringBuilder có 3 dạng như sau:

**Cú pháp**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11 | public static void main(String[] args) {      // Dạng 1: tạo 1 StringBuilder rỗng có khả năng lưu trữ 16 ký tự.      StringBuilder stringBuilder = new StringBuilder();        // Dạng 2: tạo 1 StringBuilder rỗng có khả năng lưu trữ số ký tự = capacity      StringBuilder stringBuilder1 = new StringBuilder(int capacity);        // Dạng 3: tạo 1 StringBuilder từ 1 chuỗi str có sẵn      // khả năng lưu trữ của stringBuilder2 = 16 + chiều dài của chuỗi      StringBuilder stringBuilder2 = new StringBuilder(String str);  } |

Ví dụ dưới đây sẽ minh họa 3 cú pháp khai báo StringBuilder:

**Ví dụ**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12 | public static void main(String[] args) {      // Dạng 1: tạo 1 StringBuilder rỗng có khả năng lưu trữ 16 ký tự.      StringBuilder stringBuilder = new StringBuilder();        // Dạng 2: tạo 1 StringBuilder rỗng có khả năng lưu trữ số ký tự = 32      StringBuilder stringBuilder1 = new StringBuilder(32);        // Dạng 3: tạo 1 StringBuilder từ 1 chuỗi str có sẵn      // khả năng lưu trữ của stringBuilder2 = 16 + chiều dài của chuỗi      String str = "Hello World!";      StringBuilder stringBuilder2 = new StringBuilder(str);  } |

### **Các phương thức của StringBuilder**

Đối với StringBuilder, chúng ta có 4 loại phương thức phổ biến đó là: append(), insert(), delete() và reserve().

**Phương thức append()**

Phương thức append() là phương thức dùng để nối chuỗi thay thế cho dấu "+". Chúng ta nên sử dụng phương thức này trong trường hợp chúng ta cần nối chuỗi nhiều (đặc biệt là khi tương tác với file và dữ liệu từ Internet). Ví dụ sau sẽ minh họa điều này:

**Ví dụ**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10 | public static void main(String[] args) {      StringBuilder stringBuilder = new StringBuilder();        for (int i = 0; i < 10; i++) {          stringBuilder.append("Hello lần " + i);          stringBuilder.append("\t"); // nối "\t" vào cuối stringBuilder      }        System.out.println(stringBuilder);  } |

Kết quả sau khi minh họa chương trình:



**Phương thức insert()**

Phương thức insert() là phương thức dùng để chèn 1 ký tự/số/chuỗi vào 1 vị trí bất kỳ trong StringBuilder.

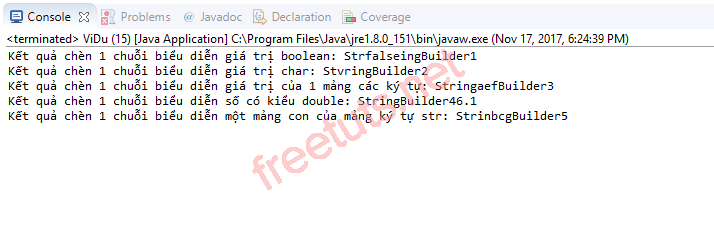
**Cú pháp**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42 | // chèn 1 chuỗi biểu diễn giá trị boolean vào StringBuilder  // bằng cách sử dụng phương thức String.valueOf() chuyển đổi giá trị boolean này thành chuỗi  // và chèn các ký tự của chuỗi này vào vị trí offset đã chỉ định  // giá trị offset phải lớn hơn 0 và <= chiều dài của StringBuilder  insert(int offset, boolean b);    // chèn 1 chuỗi biểu diễn giá trị char vào StringBuilder  // bằng cách sử dụng phương thức String.valueOf() chuyển đổi giá trị char này thành chuỗi  // và chèn các ký tự của chuỗi này vào vị trí offset đã chỉ định  insert(int offset, char c);    // chèn 1 chuỗi biểu diễn giá trị của 1 mảng các ký tự vào StringBuilder  // bằng cách sử dụng phương thức String.valueOf() chuyển đổi mảng các ký tự này thành chuỗi  // và chèn các ký tự của chuỗi này vào vị trí offset đã chỉ định  insert(int offset, char[] str);    // chèn 1 chuỗi biểu diễn số có kiểu double vào StringBuilder  // bằng cách sử dụng phương thức String.valueOf() chuyển đổi số này thành chuỗi  // và chèn các ký tự của chuỗi này vào vị trí offset đã chỉ định  insert(int offset, double d);    // chèn 1 chuỗi biểu diễn số có kiểu float vào StringBuilder  // bằng cách sử dụng phương thức String.valueOf() chuyển đổi số này thành chuỗi  // và chèn các ký tự của chuỗi này vào vị trí offset đã chỉ định  insert(int offset, float f);    // chèn 1 chuỗi biểu diễn số có kiểu int vào StringBuilder  // bằng cách sử dụng phương thức String.valueOf() chuyển đổi số này thành chuỗi  // và chèn các ký tự của chuỗi này vào vị trí offset đã chỉ định  insert(int offset, int i);    // chèn 1 chuỗi biểu diễn số có kiểu long vào StringBuilder  // bằng cách sử dụng phương thức String.valueOf() chuyển đổi số này thành chuỗi  // và chèn các ký tự của chuỗi này vào vị trí offset đã chỉ định  insert(int offset, long i);    // chèn 1 chuỗi biểu diễn một mảng con của mảng ký tự str vào StringBuilder  // index là vị trí chèn  // str là tên mảng ký tự  // offset là vị trí của ký tự đầu tiên trong mảng con được chèn  // len là số ký tự trong mảng con cần chèn  insert(int index, char[] str, int offset, int len); |

**Ví dụ**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34 | public static void main(String[] args) {      // chèn 1 chuỗi biểu diễn giá trị boolean vào vị trí số 3 trong stringBuilder1      StringBuilder stringBuilder1 = new StringBuilder("StringBuilder1");      boolean bool = false;      stringBuilder1.insert(3, bool);      System.out.println("Kết quả chèn 1 chuỗi biểu diễn giá trị boolean: " + stringBuilder1);        // chèn 1 chuỗi biểu diễn giá trị char vào vị trí số 2 trong stringBuilder2      StringBuilder stringBuilder2 = new StringBuilder("StringBuilder2");      char ch = 'v';      stringBuilder2.insert(2, ch);      System.out.println("Kết quả chèn 1 chuỗi biểu diễn giá trị char: " + stringBuilder2);        // chèn 1 chuỗi biểu diễn giá trị của 1 mảng các ký tự vào vị trí số 6 trong stringBuilder3      StringBuilder stringBuilder3 = new StringBuilder("StringBuilder3");      char[] ch1 = {'a', 'e', 'f'};      stringBuilder3.insert(6, ch1);      System.out.println("Kết quả chèn 1 chuỗi biểu diễn giá trị của 1 mảng các ký tự: " +          stringBuilder3);          // chèn 1 chuỗi biểu diễn số có kiểu double vào stringBuilder4      StringBuilder stringBuilder4 = new StringBuilder("StringBuilder4");      double doubleNumber = 6.1d;      stringBuilder4.insert(14, doubleNumber);      System.out.println("Kết quả chèn 1 chuỗi biểu diễn số có kiểu double: " + stringBuilder4);        // chèn 1 chuỗi biểu diễn một mảng con của mảng ký tự str vào stringBuilder5      StringBuilder stringBuilder5 = new StringBuilder("StringBuilder5");      char[] str1 = {'d', 'a', 'b', 'c'};      stringBuilder5.insert(5, str1, 2, 2);      System.out.println("Kết quả chèn 1 chuỗi biểu diễn một mảng con của mảng ký tự str: " +          stringBuilder5);  } |

Kết quả sau khi biên dịch chương trình:



**Phương thức delete() và deleteCharAt()**

Trong Java, chúng ta có 2 phương thức dùng để xóa ký tự đó là delete() và deleteCharAt().

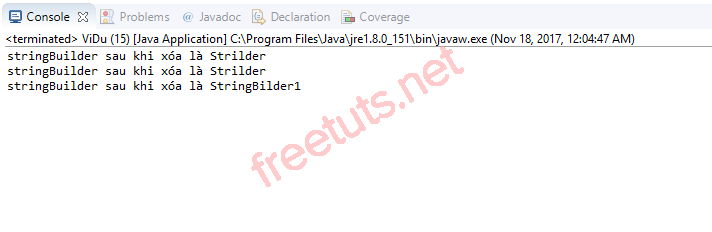
**Cú pháp**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | // xóa các ký tự trong 1 chuỗi con của StringBuilder  // sử dụng phương thức delete()  // Chuỗi con này bắt đầu tại chỉ số start và kéo dài đến ký tự có chỉ số (end - 1).  stringBuilder.delete(int start, int end);    // xóa 1 ký tự trong stringBuilder có chỉ số = index trong StringBuilder  // sử dụng phương thức deleteCharAt()  stringBuilder.deleteCharAt(7); |

**Ví dụ**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17 | public static void main(String[] args) {      StringBuilder stringBuilder = new StringBuilder("StringBuilder");        // xóa các ký tự trong 1 chuỗi con của stringBuilder      // Chuỗi con này bắt đầu tại chỉ số 3 và kéo dài đến ký tự có chỉ số 7      stringBuilder.delete(3, 8);      System.out.println("stringBuilder sau khi xóa là " + stringBuilder);        // nếu chỉ số start = end thì sẽ không có bất kỳ 1 sự thay đổi nào      stringBuilder.delete(3, 3);      System.out.println("stringBuilder sau khi xóa là " + stringBuilder);        // xóa 1 ký tự trong stringBuilder có chỉ số = 6 trong stringBuilder      StringBuilder stringBuilder1 = new StringBuilder("StringBuilder1");      stringBuilder1.deleteCharAt(7);      System.out.println("stringBuilder sau khi xóa là " + stringBuilder1);  } |

Kết quả sau khi biên dịch chương trình:



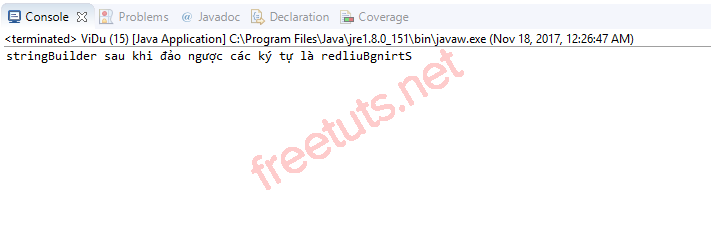
**Phương thức reserve()**

Phương thức reserve() là phương thức dùng để đảo ngược các ký tự của StringBuilder.

**Ví dụ**

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | public static void main(String[] args) {      StringBuilder stringBuilder = new StringBuilder("StringBuilder");        stringBuilder.reverse();      System.out.println("stringBuilder sau khi đảo ngược các ký tự là " + stringBuilder);  } |

Kết quả sau khi biên dịch chương trình:



**2. StringTokenizer**

Trong Java, StringTokenizer được sử dụng để tách 1 chuỗi thành các phần tử (token) nhỏ hơn (tương tự như split()). Ví dụ, mỗi từ trong một câu có thể được coi như là một token. Để tách 1 chuỗi thành các token, ta có thể tùy biến chỉ ra một tập các dấu phân cách khi khởi tạo StringTokenizer. Nếu ta không chỉ ra tập các dấu phân cách thì mặc định chuỗi sẽ được tách theo ký tự trắng (space), tab,... Ngoài ra, ta cũng có thể sử dụng các toán tử (+, -, \*, /) để phân tách 1 chuỗi. Sau đây tôi sẽ giới thiệu cách khởi tạo 1 StringTokenizer, các phương thức chính của StringTokenizer và ví dụ minh họa các phương thức này.

### **2.1. Tạo mới 1 StringTokenizer**

Để sử dụng StringTokenizer, chúng ta cần import gói thư viện java.util.StringTokenizer của Java. Cú pháp import và khai báo StringTokenizer như sau:

**Cú pháp**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24 | // Để sử dụng StringTokenizer   chúng ta cần import gói thư viện java.util.StringTokenizer của Java  import java.util.StringTokenizer;    public class TênClass {      public static void main(String[] args) {          // Dạng 1: Tạo 1 StringTokenizer mới          // với str là chuỗi mà chúng ta cần tách ra thành các token          // ký tự phân tách mặc định là khoảng trắng, tab, các ký tự xuống dòng          StringTokenizer stringTokenizer = new StringTokenizer(String str);            // Dạng 2: Tạo 1 StringTokenizer mới          // với str là chuỗi mà chúng ta cần tách ra thành các token          // và delim là tập các ký tự phân tách          StringTokenizer stringTokenizer = new StringTokenizer(String str, String delim);            // Dạng 3: Tạo 1 StringTokenizer mới          // với str là chuỗi mà chúng ta cần tách ra thành các token          // delim là tập các ký tự phân tách          // returnDelims có giá trị là true hoặc false.          // Nếu là true thì ký tự phân cách cũng được trả về như các token, ngược lại thì không          StringTokenizer stringTokenizer = new StringTokenizer(String str, String delim, boolean returnDelims);      }  } |

Ví dụ dưới đây sẽ minh họa 3 cú pháp khai báo StringTokenizer:

**Ví dụ**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19 | public static void main(String[] args) {      // Dạng 1: Tạo 1 StringTokenizer mới      // với str là chuỗi mà chúng ta cần tách ra thành các token      String str = "StringTokenizer dạng 1";      StringTokenizer stringTokenizer1 = new StringTokenizer(str);        // Dạng 2: Tạo 1 StringTokenizer mới      // với str1 là chuỗi mà chúng ta cần tách ra thành các token      // và ký tự phân tách là "\n"      String str1 = "StringTokenizer dạng 2";      StringTokenizer stringTokenizer2 = new StringTokenizer(str1, "\n");        // Dạng 3: Tạo 1 StringTokenizer mới      // với str2 là chuỗi mà chúng ta cần tách ra thành các token      // ký tự phân cách là "\t"      // returnDelims có giá trị là true.      String str2 = "StringTokenizer dạng 3";      StringTokenizer stringTokenizer3 = new StringTokenizer(str2, "\t", true);  } |

### **2.2. Các phương thức của StringTokenizer**

Đối với StringTokenizer, chúng ta có 5 phương thức phổ biến đó là: countTokens(), hasMoreTokens(), nextToken(), hasMoreElements() và nextElement().

| **Tên phương thức** | **Công dụng** |
| --- | --- |
| countTokens() | Trả về số các token còn lại. |
| hasMoreTokens() | Trả về true nếu còn có token trong StringTokenizer và ngược lại trả về false (tương tự như hasMoreElements()). |
| nextToken() | Trả về token kế tiếp trong StringTokenizer (tương tự như nextElement()). |
| hasMoreElements() | Trả về true nếu còn có token trong StringTokenizer và ngược lại trả về false (tương tự như hasMoreTokens()). |
| nextElement() | Trả về token kế tiếp trong StringTokenizer (tương tự với nextToken()). |

Ví dụ sau sẽ minh họa cách sử dụng các phương thức này:

**Ví dụ**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24 | public static void main(String[] args) {      String str = "Học, học nữa, học mãi!";        // ký tự phân tách mặc định ở đây là khoảng trắng      StringTokenizer stringTokenizer = new StringTokenizer(str);        System.out.println("Kết quả sau khi phân tách theo khoảng trắng: ");      // kiểm tra xem có còn token kế tiếp trong chuỗi hay không      // sử dụng phương thức hasMoreTokens()      while (stringTokenizer.hasMoreTokens()) {          // nextToken(): lấy token hiện tại ra và di chuyển tới token kế tiếp          System.out.println(stringTokenizer.nextToken());      }        // phân tách stringTokenizer1 vừa theo dấu phẩy và vừa theo khoảng trắng      StringTokenizer stringTokenizer1 = new StringTokenizer(str, ", ");        System.out.println("\nKết quả sau khi phân tách vừa theo dấu phẩy và vừa theo khoảng trắng: ");      // kiểm tra xem có còn token kế tiếp trong chuỗi hay không      // sử dụng phương thức hasMoreElements()      while (stringTokenizer1.hasMoreElements()) {          System.out.println(stringTokenizer1.nextElement());      }  } |

Kết quả sau khi biên dịch chương trình:

