

- `StringBuilder()`: Tạo ra một Builder chuỗi với dung lượng ban đầu là 16.
- `StringBuilder(String str)`: Tạo ra một Builder chuỗi với chuỗi cụ thể.
- `StringBuilder(int capacity)`: Tạo ra một Builder chuỗi với dung lượng được chỉ định như độ dài chuỗi.
- `StringBuilder(CharSequence cs)`: Tạo ra một Builder chuỗi với chuỗi cụ thể.

### 3. Các phương thức của lớp **StringBuilder**

- **public StringBuilder append(String s):** được sử dụng để nối thêm các chuỗi được chỉ định với chuỗi này. Các phương thức **append()** được nạp chồng như **append(char)**, **append(boolean)**, **append(int)**, **append(float)**, **append(double)**, ...
- **public StringBuilder insert(int offset, String s):** được sử dụng để chèn chuỗi chỉ định với chuỗi này tại vị trí quy định. Các phương thức **insert()** được nạp chồng như **insert(int, char)**, **insert(int, boolean)**, **insert(int, int)**, **insert(int, float)**, **insert(int, double)**, ...
- **public StringBuilder replace(int startIndex, int endIndex, String str):** được sử dụng để thay thế chuỗi từ vị trí **startIndex** đến **endIndex** bằng chuỗi **str**.
- **public StringBuilder delete(int startIndex, int endIndex):** được sử dụng để xóa chuỗi từ vị trí **startIndex** đến **endIndex**.
- **public StringBuilder reverse():** được sử dụng để đảo ngược chuỗi.
- **public int capacity():** được sử dụng để trả về dung lượng hiện tại.
- **public void ensureCapacity(int minimumCapacity):** được sử dụng để đảm bảo dung lượng ít nhất bằng mức tối thiểu nhất định.
- **public char charAt(int index):** được sử dụng trả về ký tự tại vị trí quy định.
- **public int length():** được sử dụng trả về chiều dài của chuỗi nghĩa là tổng số ký tự.
- **public String substring(int beginIndex):** được sử dụng trả về chuỗi con bắt đầu từ vị trí được chỉ định.
- **public String substring(int beginIndex, int endIndex):** được sử dụng trả về chuỗi con với vị trí bắt đầu và vị trí kết thúc được chỉ định.

#### + **append()**

Phương thức **append()** của lớp **StringBuilder** nối thêm tham số vào cuối chuỗi.

```
public class StringBuilderExam1 {  
  
    public static void main(String args[]) {  
        StringBuilder sb = new StringBuilder("Hello ");  
        sb.append("Java");  
        System.out.println(sb);  
    }  
}
```

#### + **insert()**

Phương thức **insert()** của lớp **StringBuilder** chèn chuỗi vào chuỗi này từ vị trí quy định.

```
public class StringBuilderExam2 {  
  
    public static void main(String args[]) {  
        StringBuilder sb = new StringBuilder("Hello ");
```

```
        sb.insert(1, "Java");
        System.out.println(sb);
    }
}
```

### + **replace()**

Phương thức `replace()` của lớp `StringBuilder` thay thế chuỗi bằng chuỗi khác từ vị trí bắt đầu và kết thúc được quy định.

```
public class StringBuilderExam3 {

    public static void main(String args[]) {
        StringBuilder sb = new StringBuilder("Hello");
        sb.replace(1, 3, "Java");
        System.out.println(sb);
    }
}
```

### + **delete()**

`StringBuilder delete()` được sử dụng để xóa chuỗi từ vị trí `startIndex` đến `endIndex`

```
StringBuilder sb = new StringBuilder("JournalABC");
sb.delete(7,14);
System.out.println(sb); // prints Journal
```

### + **reverse()**

Phương thức `reverse()` được sử dụng để đảo ngược chuỗi.

```
StringBuilder sb = new StringBuilder("lived");
sb.reverse();
System.out.println(sb); // prints devil
```

### + **capacity()**

Phương thức `capacity()` của lớp `StringBuilder` trả về dung lượng của bộ nhớ đệm. Dung lượng mặc định của bộ nhớ đệm là 16. Nếu số lượng ký tự của chuỗi tăng lên thì dung lượng được tính theo công thức  $(\text{dung lượng cũ} * 2) + 2$ . Ví dụ: Nếu dung lượng hiện tại là 16, nó sẽ tăng lên  $(16 * 2) + 2 = 34$ .

```
StringBuilder sb=new StringBuilder();
System.out.println(sb.capacity()); // default value 16
```

```
sb.append(" Java");  
System.out.println(sb.capacity()); // still 16  
sb.append("Hello StringBuilder Class!");  
System.out.println(sb.capacity()); // (16*2)+2
```

### + ensureCapacity()

Phương thức ensureCapacity() của lớp StringBuilder đảm bảo rằng dung lượng đã cho là tối thiểu với dung lượng hiện tại. Nếu nó lớn hơn dung lượng hiện tại, dung lượng hiện tại được tăng theo công thức (dung lượng cũ\*2)+2. Ví dụ, dung lượng hiện tại là 16, nó sẽ tăng lên là  $(16*2)+2=34$ .

```
public class StringBuilderExample {  
  
    public static void main(String[] args) {  
  
        StringBuilder sbObj = new StringBuilder();  
        System.out.println(sbObj.capacity()); //default 16  
  
        sbObj.append("Java StringBuilder Class!");  
        System.out.println(sbObj.capacity()); // capacity 34  
  
        sbObj.ensureCapacity(12); // no change  
        System.out.println(sbObj.capacity()); //still 34  
  
        sbObj.ensureCapacity(60); // (34*2)+2 = 70  
        System.out.println(sbObj.capacity()); //70  
    }  
}
```

---

Revision #5

Created 19 October 2019 19:15:50 by Laptrinh.vn

Updated 26 December 2020 15:17:16 by Laptrinh.vn