



# Java Core

Ngọc Bản Quyền

---

# StringBuilder

## Lớp StringBuilder



Lớp StringBuilder được sử dụng để tạo chuỗi có thể thay đổi.  
Lớp StringBuilder làm cho chuỗi trở nên linh động hơn. Nó có một loạt các methods để tác động lên chuỗi.

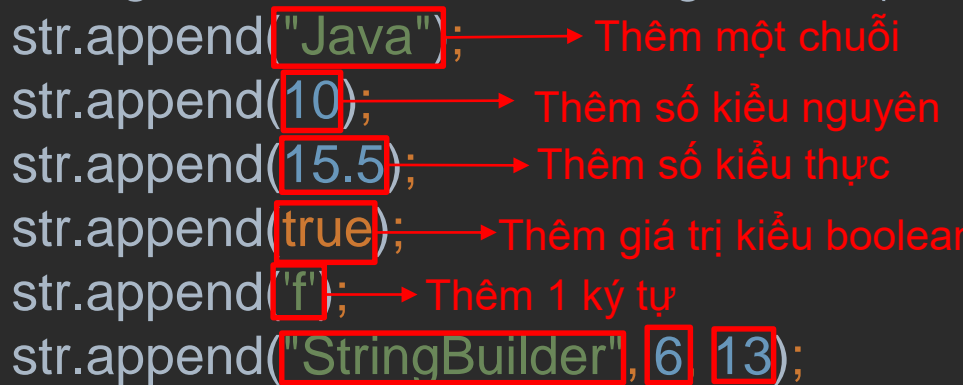
# Constructor trong lớp `StringBuilder`

- **`StringBuilder()`**: Tạo một string builder không có ký tự nào và dung lượng ban đầu là 16 ký tự
- **`StringBuilder(int capacity)`**: Tạo một string builder không có ký tự nào và có dung lượng ban đầu được chỉ định bởi đối số `capacity`
- **`StringBuilder(CharSequence seq)`** : Tạo một string builder có chứa các ký tự giống như `CharSequence` được chỉ định
- **`StringBuilder(String str)`**: Tạo ra một string builder với chuỗi cụ thể

# Một số method của lớp **StringBuilder**

**public StringBuilder append(T t)** : Được sử dụng để nối thêm các chuỗi được chỉ định với chuỗi này. T là kiểu dữ liệu như int, float, double, String,...

```
StringBuilder str = new StringBuilder("Hello");  
str.append("Java");  
str.append(10);  
str.append(15.5);  
str.append(true);  
str.append('f');  
str.append("StringBuilder", 6, 13);
```



- Thêm một chuỗi
- Thêm số kiểu nguyên
- Thêm số kiểu thực
- Thêm giá trị kiểu boolean
- Thêm 1 ký tự

**public StringBuilder appendCodePoint(int codePoint)** : Được sử dụng để nối thêm ký tự dưới dạng mã code vào cuối chuỗi

```
StringBuilder str = new StringBuilder("Hello Java");  
str.appendCodePoint(46);
```

**public StringBuilder insert(int offset, T t)** : Được sử dụng để chèn chuỗi và vị trí offset, T là kiểu dữ liệu như int, float, double, String,...

```
StringBuilder str = new StringBuilder("Hello Java");  
str.insert(2,5); //Thêm một số nguyên vào vị trí 2  
str.insert(3, 'a'); //Thêm một ký tự vào vị trí 3  
str.insert(4, 5.5); //Thêm một số thực vào vị trí 4  
str.insert(5, true); //Thêm giá trị boolean vào vị trí 5  
str.insert(6, "Maven"); //Thêm một chuỗi vào vị trí 6
```

**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

```
StringBuilder str = new StringBuilder("Hello Java");  
str.replace(6, 10, "Maven");
```



startIndex



endIndex



**public StringBuilder delete(int startIndex, int endIndex) :** Được sử dụng để xóa chuỗi từ vị trí startIndex đến endIndex

```
StringBuilder str = new StringBuilder("Hello Java");  
str.delete(6, 10);
```



Diagram illustrating the parameters of the `delete` method:

Two red arrows originate from the boxed values `6` and `10` in the code snippet above. The first arrow points from the value `6` to the text `startIndex`. The second arrow points from the value `10` to the text `endIndex`.

startIndex

endIndex

**public StringBuilder reverse()** : Được sử dụng để đảo ngược chuỗi

```
StringBuilder str = new StringBuilder("Hello Java");  
str.reverse();
```

**public int indexOf(String str)** : Trả về vị trí xuất hiện đầu tiên của chuỗi str

**public int lastIndexOf(String str)** : Trả về vị trí xuất hiện cuối cùng của chuỗi str

```
StringBuilder str = new StringBuilder("Hello Java");  
int first = str.indexOf("a");  
int last = str.lastIndexOf("a");
```

**public int capacity()** : Trả về dung lượng của chuỗi hiện tại (Dung lượng mặc định khi chuỗi rỗng là 16)

```
StringBuilder str = new StringBuilder("Hello Java");  
int capacity = str.capacity();
```

**public String substring(int beginIndex)** : Được sử dụng để trả về chuỗi con bắt đầu từ vị trí được chỉ định

**public String substring(int beginIndex, int endIndex)** : Được sử dụng để trả về chuỗi con với vị trí bắt đầu và vị trí kết thúc

```
StringBuilder str2 = new StringBuilder("Hello Java");  
System.out.println(str2.substring(6));  
System.out.println(str2.substring(0,5));
```

