StringBuilder And StringBuffer Class

StringBuilder Class

- **문자열의 저장 및 변경을 위한 Memory Buffer를 지니는 Class**

- StringBuilder Instance 내에는 Memory Buffer(여러개의 데이터를 저장할 수 있는 메모리공간)를 가지고 있다.
- Memory Buffer 내에는 숫자,문자,문자열..등 다양한 데이터를 넣을 수 있다.
- **Memory Buffer에 쌓인 여러 데이터들을 조합하고 toString() Method를 이용해 하나의 String Instance로 만들어 주는 담당**

```
class BuiderString{
     public static void main(String[] args) {

          StringBuilder strBuf = new StringBuilder("AB");
          // Buf : AB (문자열 AB가 저장되었다기 보다는 '문자데이터 A, B가 Buffer에 저장되어 있다'가 맞는 해석)

          strBuf.append(25); // Buf : AB25

          strBuf.append('Y').append(true); //Buf : AB25Ytrue
```

append method가 호출되고 나서, **strBuf의 instanse 참조값을 반환**하기 때문에 이러한 표현이 가능

결국 append('Y')와 append("true")는 동일한 인스턴스를 대상으로 한다.

```
          System.out.println(strBuf);
```

Buf	A	B	2	5	Y	t	r	u	e
index	0	1	2	3	4	5	6	7	8

↓ strBuf.insert(2, false);

Buf	A	B	f	a	l	s	e	2	5	Y	t	r	u	e
index	0	1	2	3	4	5	6	7	8	9	10	11	12	13

```
          strBuf.insert(strBuf.length(), 'Z'); // Buf : ABfalse25YtrueZ
          System.out.println(strBuf);
```

```
    }
}
```

- Buf : Memory Buffer
- **StringBuilder Class에 문자열을 저장한다고 해도 실제로는 문자 하나하나가 데이터로 저장된다.**

Instance의 참조값을 반환하는 Method

StrBuf.append().append();

StrBuf.append()가 반환하는 Instance의 참조값은 StrBuf의 참조값일까, 새로운 Instance의 참조값일까?
답 : append가 호출된 그 Instance의 참조값 즉, strBuf의 참조변수의 참조값이 반환

```
class SimpleAdder {
    Private int num;
    public SimpleAdder add(int num) {
        this.num += num;

        return this;
    }

    public void showResult(){
        System.out.println("add result : "+num);
    }
}

class MainClass {
    public static void main(String[] args) {
        SimpleAdder adder = new SimpleAdder();

        adder.add(1).add(3).add(5).showResult();
```

this의 반환은 this 문장을 실행하는 Instance 자신의 반환을 의미한다.
그리고 이렇게 반환되는 Instance 자신을 대상으로 연이은 Method 호출이 가능하다.

```
1. adder.add(1).add(3).add(5).showResult();
                ↓
2. adder.add(3).add(5).showResult();
                ↓
3. adder.add(5).showResult();
```

```
                ↓
3.  adder.add(5).showResult();
                ↓
4.  adder.showResult();
```

    }
}

StringBuilder의 Buffer와 문자열 조합

- 추가되는 데이터의 크기에 따라서 버퍼의 크기가 자동으로 확장된다.
- 생성자를 통해서 초기 버퍼의 크기를 지정할 수 있다.
    - `public StringBuilder(){} // 기본 16개의 데이터저장 버퍼생성`
    - `public StringBuilder(int capacity){} // capacity개의 데이터저장 버퍼생성`
    - `public StringBuilder(String str){} // str.length() + 16개의 데이터저장 버퍼생성`

- 문자열의 복잡한 조합의 과정에는 StringBuilder의 Instance가 활용된다.
  때문에 추가로 생성되는 Instance의 최대 두 개이다.

```
//String str = 1+"lemon"+2;

String str = new StringBuilder().append(1).append("lemon").append(2).toString();
             // StringBuilder Instance                                  // String Instance
```

StringBuffer Class

- **StringBuilder Class와 거의 동일한 Class**
    - Method의 수 (생성자 포함)
    - Method의 기능
    - Method의 이름과 Parameter의 Data Type

- **StringBuilder Class와 StringBuffer Class의 차이점**
    - **StringBuffer Class는 멀티쓰레드 환경에서 안전하다.**
    - **StringBuilder Class는 멀티쓰레드에 환경에서 불안전하다.**
    - 쓰레드에 관한 내용은 추후에 공부하고 다시 보면 이해가 된다.