IT2030 - Object Oriented Programming

Lecture 06
Strings in Java

Contents

- Introduction to String
- String manipulation
- StringBuffer
- StringBuilder

Strings in Java

- `String` is a sequence of characters
- Java implements strings as objects (created by class `String`)
- `String`, `StringBuffer` and `StringBuilder` classes are defined in java.lang package. Thus, are available to a program automatically
- All `String`, `StringBuffer` and `StringBuilder` classes are final
- `String` objects are *immutable*
- `StringBuffer` and `StringBuilder` objects are mutable

Common ways of creating String objects

- `String` class have many constructors

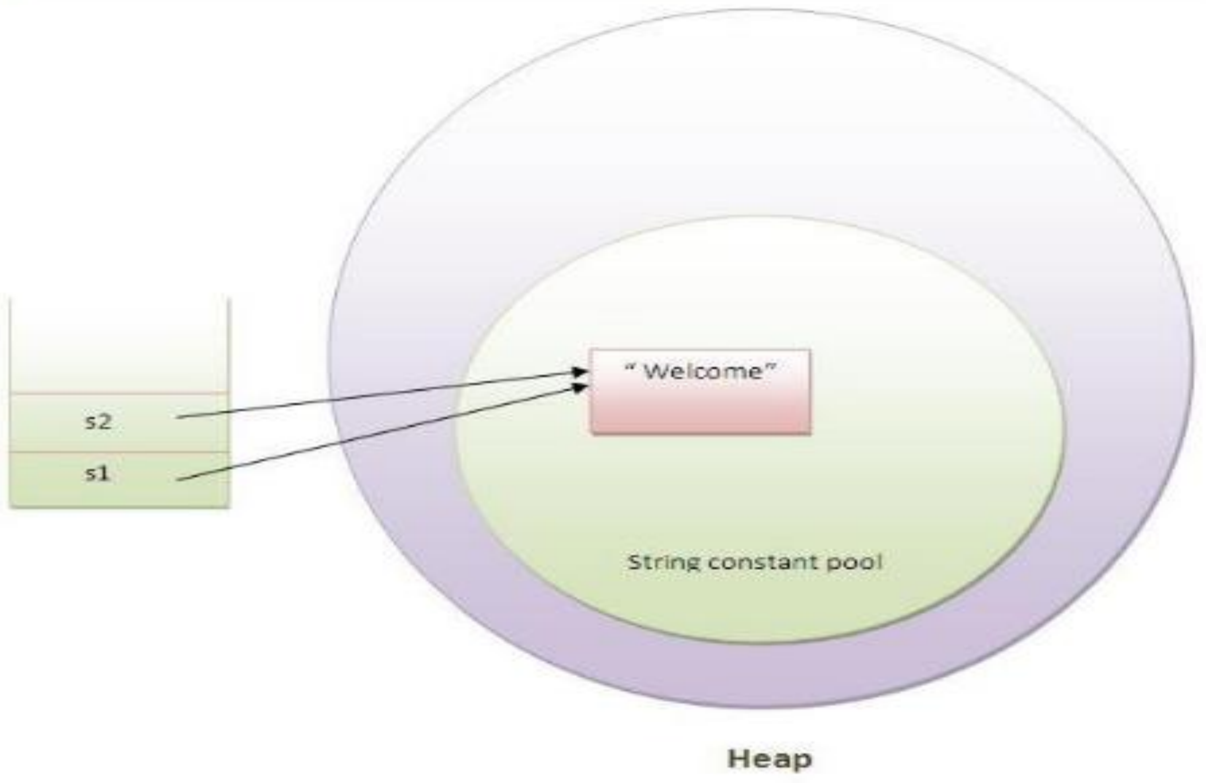
```
//method 1
char arr[] = {‘a’, ‘b’, ‘c’};
String s = new String(arr);

//method 2
String s = new String(“abc”);

//method 3
String s = “abc”;
```

Creation of String Literals

- Each time you create a string literal, the JVM checks the string constant pool first
- If the string already exists in the pool, a reference to the pooled instance is returned
- If string doesn't exist in the pool, a new string instance is created and placed in the pool

Creation of String Literals cont.

```
String s1="Welcome";
String s2="Welcome";//will not create new instance
```



Exercise 1

Draw the String pool for the below code

a)
```
String first = “Tooth”;
 first = “Tooth” +” Fairy”;
String second =first +5;
```

b)
```
String first = “Tooth”;
 first = “tooth” +”Fairy”;
String second =first +5;
```

String Literal & String Object

```
String str1 = "Hello World!!";
String str2 = "Hello World!!";
System.out.println(str1 == str2);// true
```

- When the `String` literal `str2` is created, the string “Hello World” is not created again. Instead, it is `str1` `String` is reused as it is already existing in the string constant pool.
- Since both `str1` and `str2` are referring to the same String in the pool, `str1 == str2` is true.

String Literal & String Object cont.

```
String str3 = new String("Hello World!!");
String str4 = new String("Hello World!!");
System.out.println(str3 == str4);  // false
```

- In this case, new `String` objects are created and separately referred by `str3` and `str4`. Thus, `str3 == str4` is `false`.

equals() versus ==

- Both `equals()` and `==` operator performs different operations
- `equals()` is a method that compares the characters in a string object
- `==` is an operator that compares two object references to see whether they refer to the same instance

Strings in Java are Immutable

- Strings are immutable. That is, once a String is constructed, its contents cannot be modified
- However, the variable declared as `String` reference can be changed to point at some other `String` instance
- It is not efficient to use `String` if you need to modify your string frequently (that would create many new Strings occupying new storage areas)

Exercise 2

Consider the below code block and state each statement return true or false

```
String s1= “Java”;
String s2= “java”;
String s3=new String(“java”);
String s4=new String(“Java”);
String s5=s4;
```

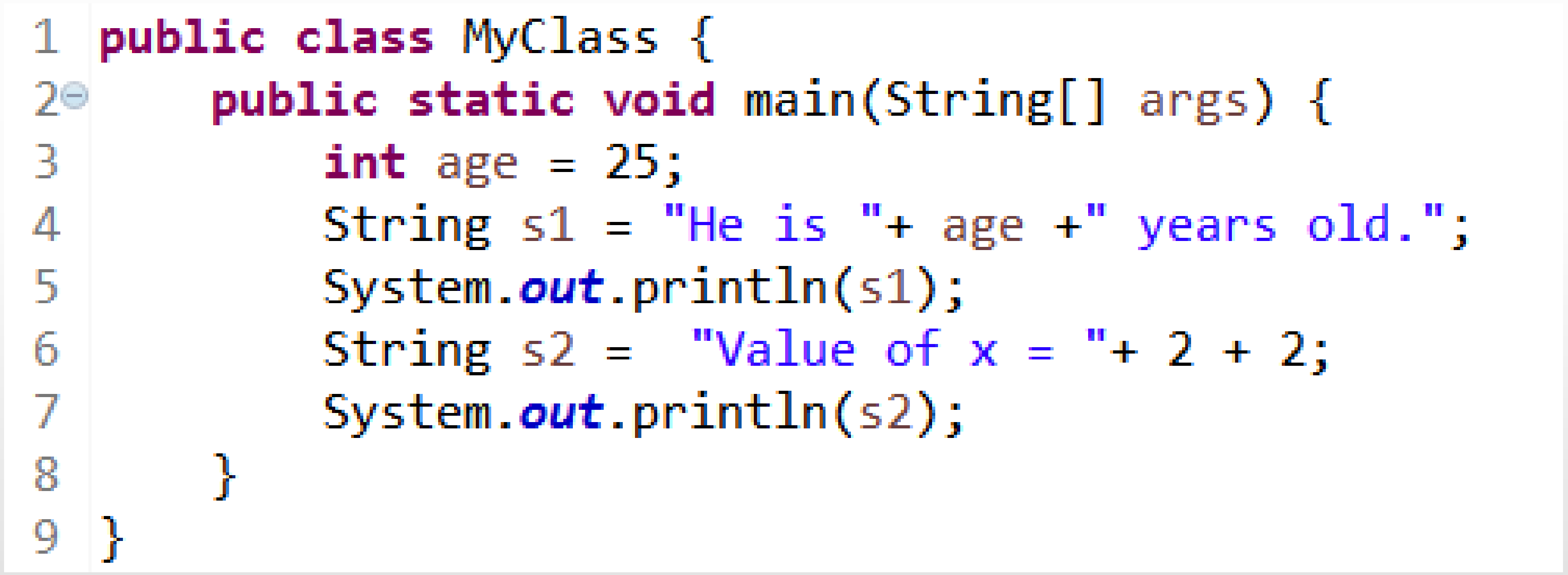
1) System.out.println (s1 == s2); true
2) System.out.println (s1.equals(s2)); false
3) System.out.println (s1.equals(“Java”)); true
4) System.out.println (s3.equals(s4)); false
5) System.out.println (s5.equals(s4)); false
6) System.out.println (s5 == s4); true

String Operations

- `length()`
- `concat()`
- `toUpperCase()`
- `toLowerCase()`
- `charAt()`
- `indexOf()`
- `lastIndexOf()`
- `substring()`
- `replace()`
- `toCharArray()`
- `startsWith()`
- `endsWith()`
- `trim()`
- `split()`
- `equals()`
- `equalsIngnoreCase()`
- `join()` – new addition to JDK8

Question

What is the out put of the following code?



```java
public class MyClass {
    public static void main(String[] args) {
        int age = 25;
        String s1 = "He is "+ age +" years old.";
        System.out.println(s1);
        String s2 =  "Value of x = "+ 2 + 2;
        System.out.println(s2);
    }
}
```

StringBuffer & StringBuilder

- As strings are immutable, Java provides two other classes to support mutable strings:
  - `StringBuffer`
  - `StringBuilder`

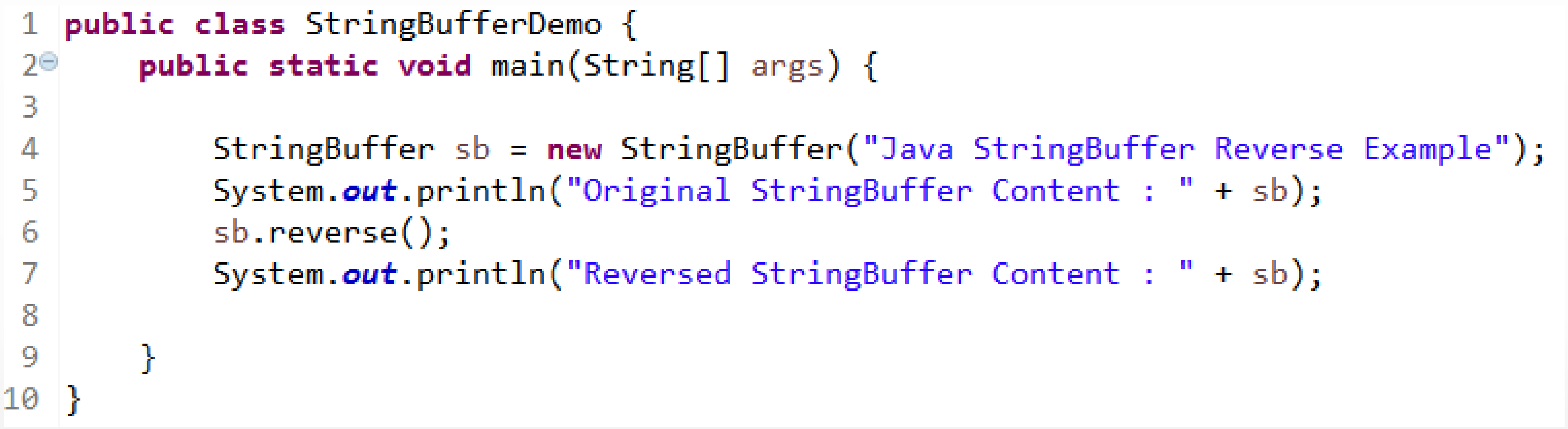
*both classes in java.lang package

- A `StringBuffer` or `StringBuilder` object is just like any ordinary object, which are stored in the heap and not shared, and therefore, can be modified without causing adverse side-effect to other objects

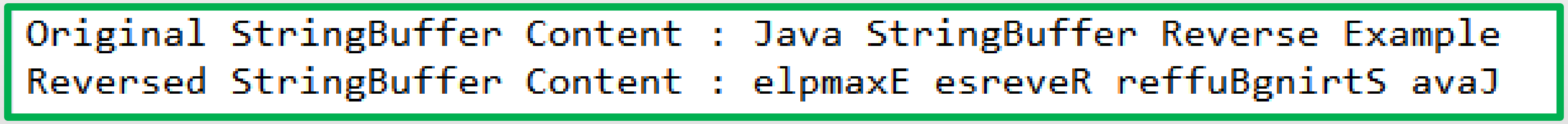
StringBuffer, String Builder Operations

- `length()`
- `indexOf()`
- `lastIndexOf()`
- `charAt()`
- `replace()`
- `substring()`
- `getChars()`
- `append()`
- `insert()`
- `reverse()`
- `deleteCharAt()`

StringBuffer



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

        StringBuffer sb = new StringBuffer("Java StringBuffer Reverse Example");
        System.out.println("Original StringBuffer Content : " + sb);
        sb.reverse();
        System.out.println("Reversed StringBuffer Content : " + sb);

    }
}
```



```
Original StringBuffer Content : Java StringBuffer Reverse Example
Reversed StringBuffer Content : elpmaxE esreveR reffuBgnirtS avaJ
```

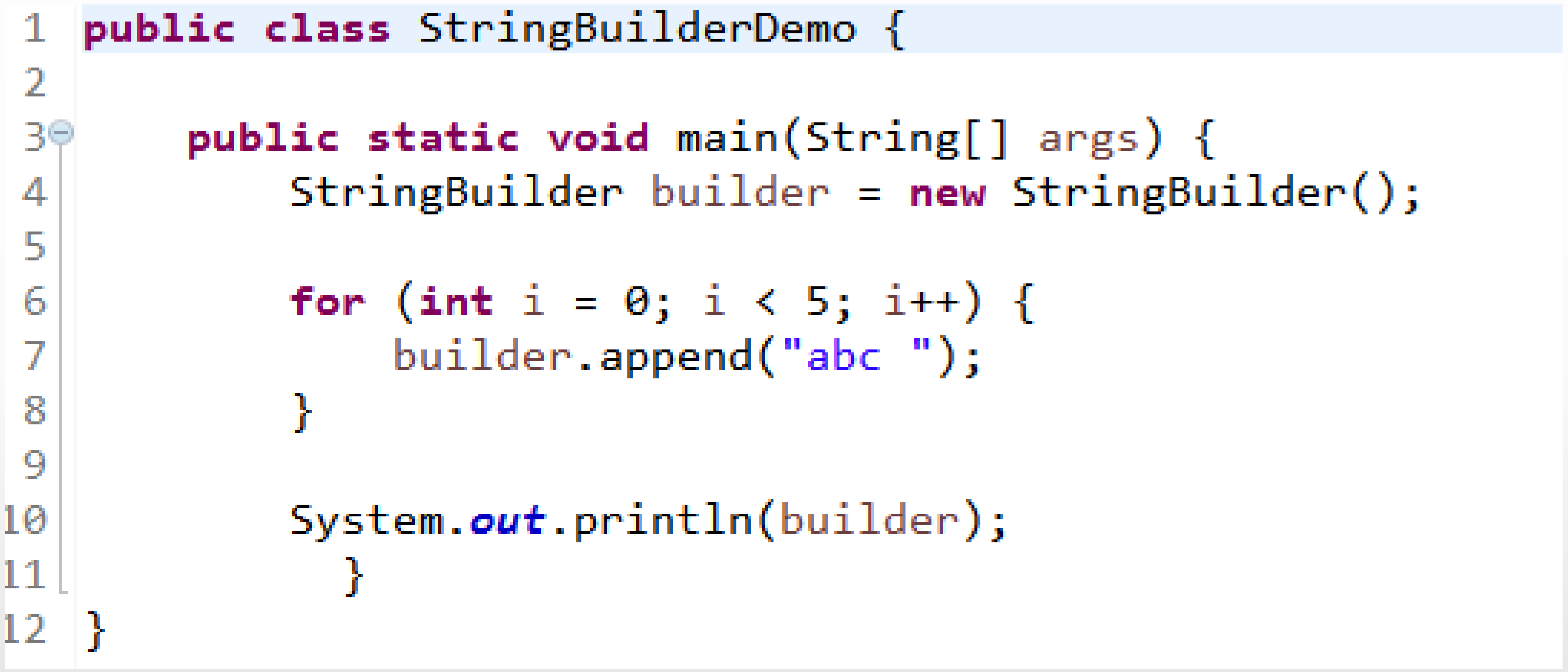
StringBufferDemo0.java

StringBuilder

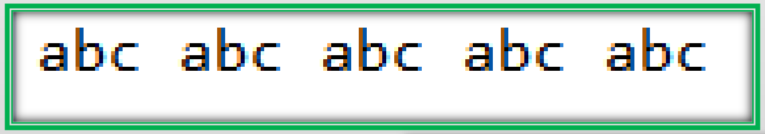
- Introduced in JDK5
- `StringBuilder` is similar to `StringBuffer` except for one difference that it is not synchronized (not thread-safe)

*In cases in which a mutable string is accessed by multiple threads, and no external synchronization is employed, you must use `StringBuffer` rather than `StringBuilder`

StringBuilder cont.



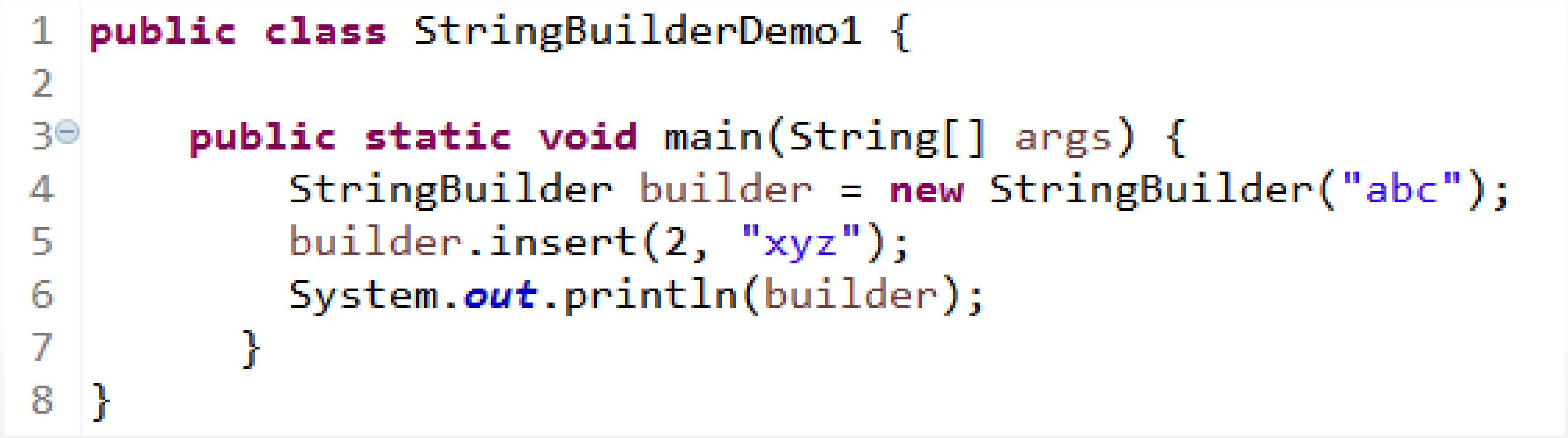
```java
public class StringBuilderDemo {

    public static void main(String[] args) {
        StringBuilder builder = new StringBuilder();

        for (int i = 0; i < 5; i++) {
            builder.append("abc ");
        }

        System.out.println(builder);
          }
}
```



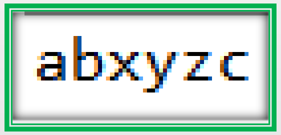
```
abc abc abc abc abc
```

StringBuilderDemo0.java

StringBuilder cont.



```java
public class StringBuilderDemo1 {

    public static void main(String[] args) {
        StringBuilder builder = new StringBuilder("abc");
        builder.insert(2, "xyz");
        System.out.println(builder);
    }
}
```



```
abxyzc
```

StringBuilderDemo1.java

Thank you!