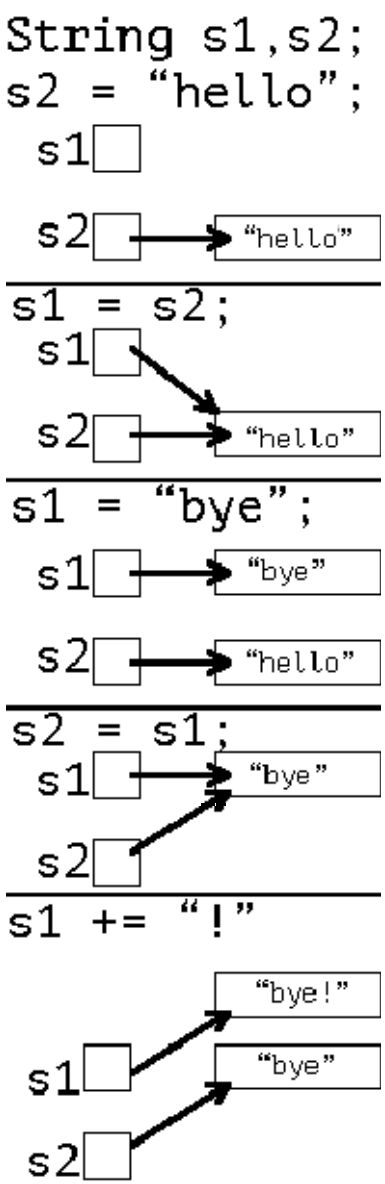
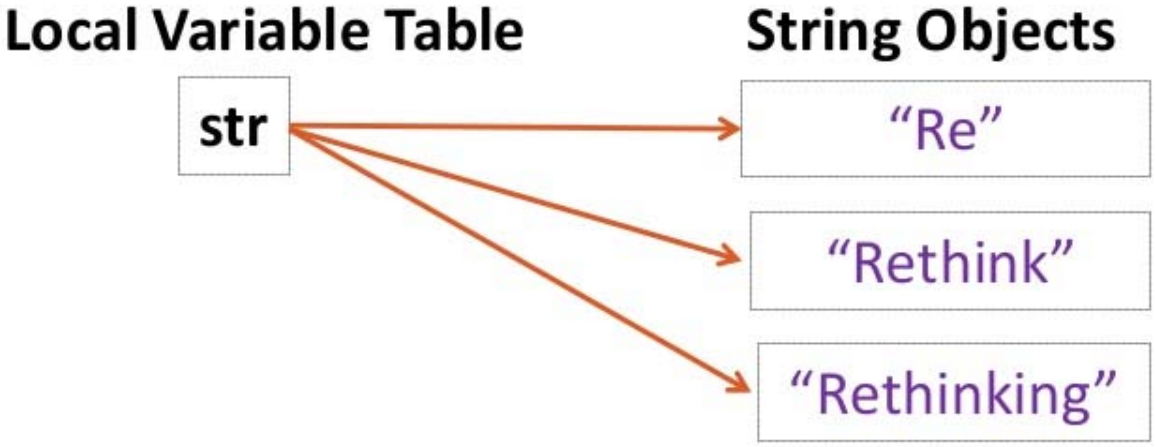
String Objects


String s1,s2;
s2 = "hello";
s1
s2
"hello"
s1 = s2;
s1
s2
"hello"
s1 = "bye";
s1
"bye"
s2
"hello"
s2 = s1;
s1
"bye"
s2
s1 += "!"
"bye!"
"bye"
s1
s2

String Immutability

```
String str;
str  = "Re";
str = str + "think"; //Rethink
str = str + "ing";   // Rethinking
```

Every concat: Create new String object
Unused objects: "Re", "Rethink" go to garb. coll.

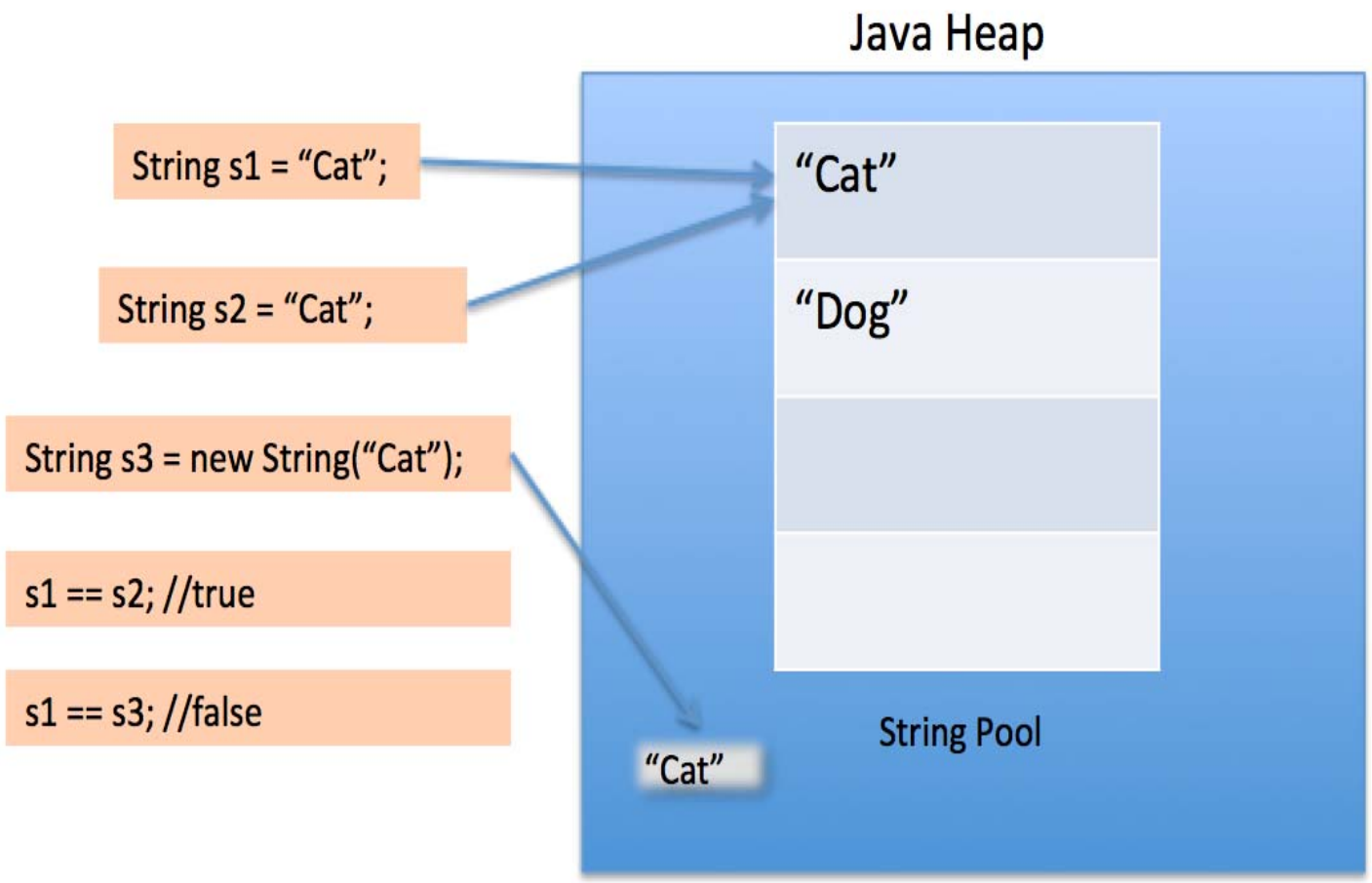


The String Methods

Method	Description	
s.length()	Returns the number of characters that s contains.	
String s = s1 + s2; **s = s1.concat(s2);**	Concatenation	
char c; **c = s.charAt(i);**	Returns the character at position i of s.	
s1.equals(s2) **s1.equalsIgnoreCase(s2)**	Returns true or false if s1 and s2 are equal.	
s1.compareTo(s2)	Value	Meaning
	< 0	s1 is less than s2
	> 0	s1 is greater than s2
	0	s1 is the same as s2

Method	Description
int index; **index = s.indexOf(str);** **index = s.indexOf(str, i);**	Returns the position of the first occurrence of str in s. Returns the position of the first occurrence of str in s starting at position i. Returns -1 if str is not found. Also works for char.
s1 = s.substring(m); **s1 = s.substring(m, n);**	Extracts the string starting at position m from s. Extracts the string starting at position m and ending at position n from s.

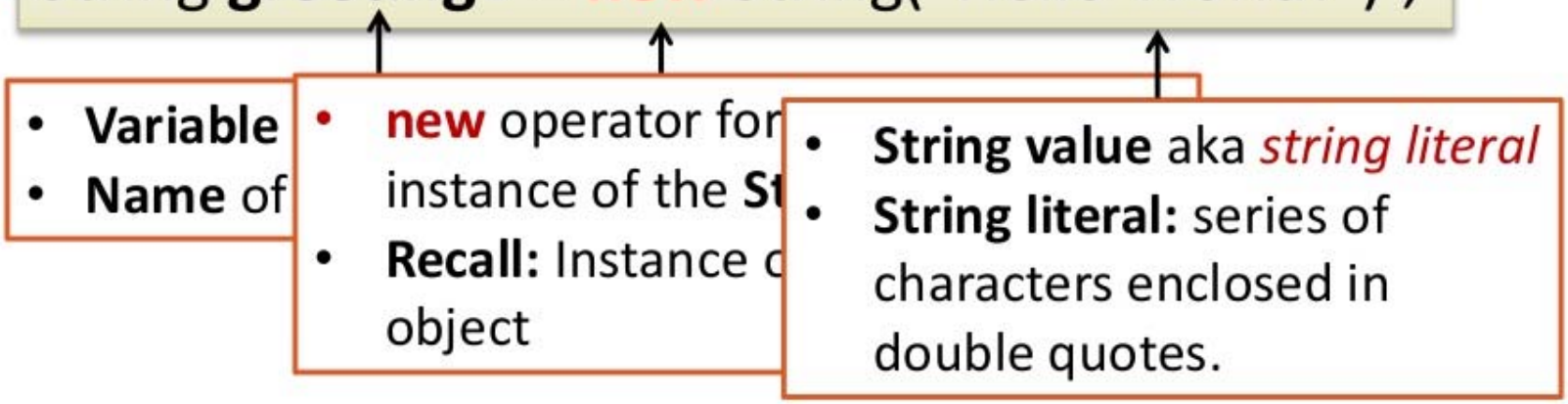
Method	Description
s1 = s.trim();	Removes all whitespace from the beginning and the end of s.
s1 = s.toLowercase(); **s1 = s.toUppercase();**	Changes the case of s.


Java Heap
String s1 = “Cat”;
String s2 = “Cat”;
String s3 = new String(“Cat”);
s1 == s2; //true
s1 == s3; //false
“Cat”
“Dog”
“Cat”
String Pool

Creating String Objects

There are 2 ways to create String objects

Method 1: String **greeting1** = **new** String("Hello World!") ;



Creating String Objects

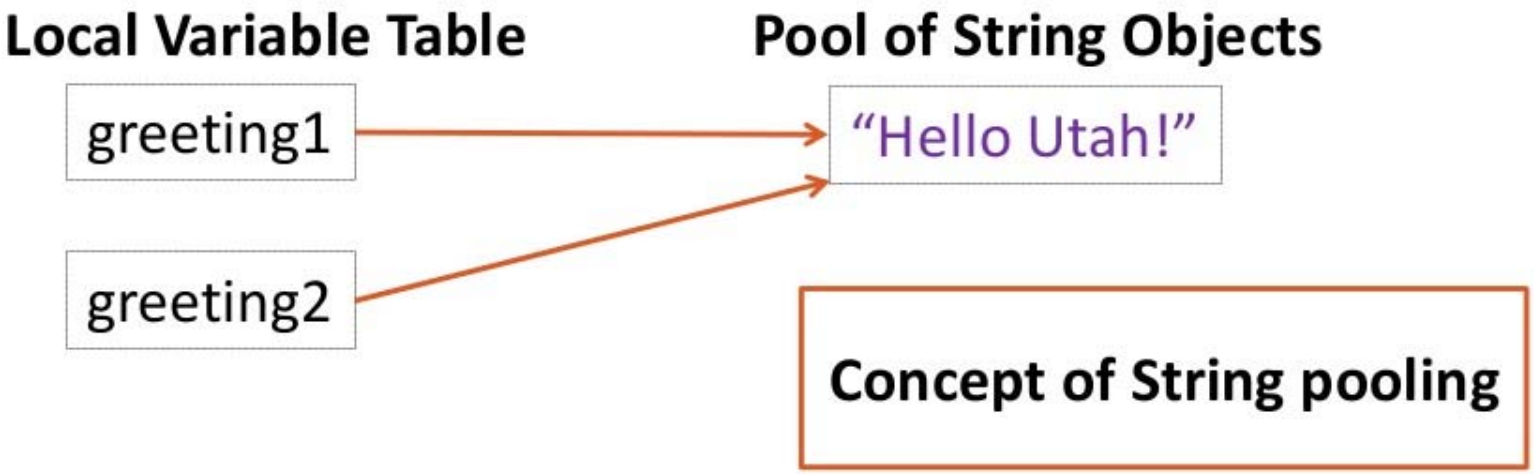
There are 2 ways to create String objects

Method 2: String **greeting2** = “Hello World Again!” ;

- **Shorthand** for String creation (most used)
- **Behind the scenes:** new instance of String class with “Hello World Again!” as the value

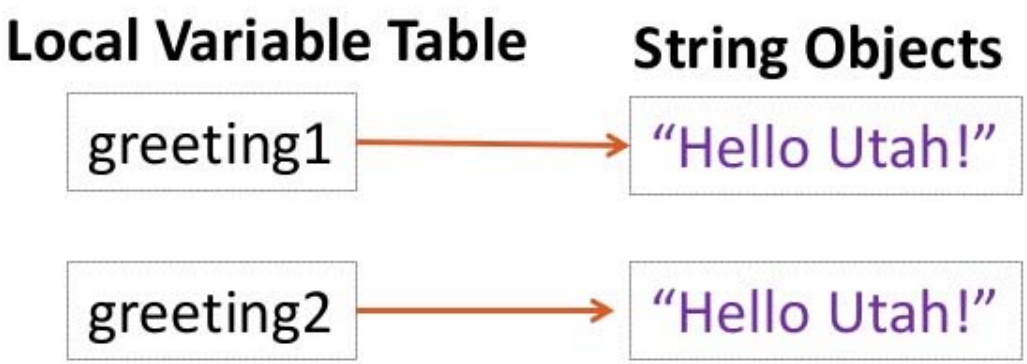
Understanding String Creation

String **greeting1** = “Hello Utah!”
String **greeting2** = “Hello Utah!”



Understanding String Creation

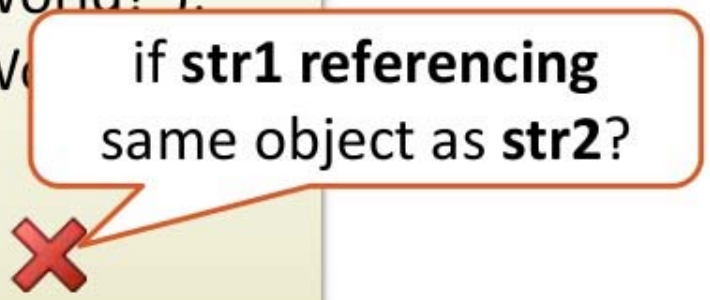
```
String greeting1 = new String ("Hello Utah!");
String greeting2 = new String ("Hello Utah!");
```

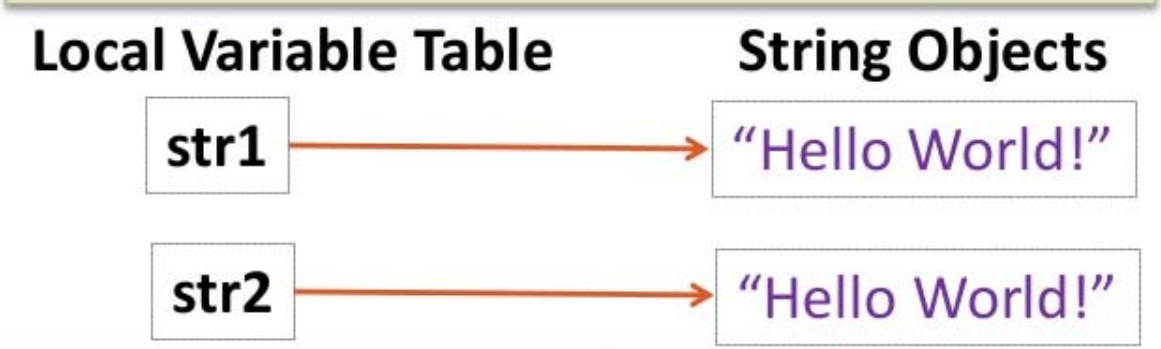


Testing String Equality

- How to check if two Strings **contain same value**?

```
String str1=new String("Hello World!");
String str2=new String("Hello W
if(str1==str2) { //eval to false
  System.out.println("same");
}
```





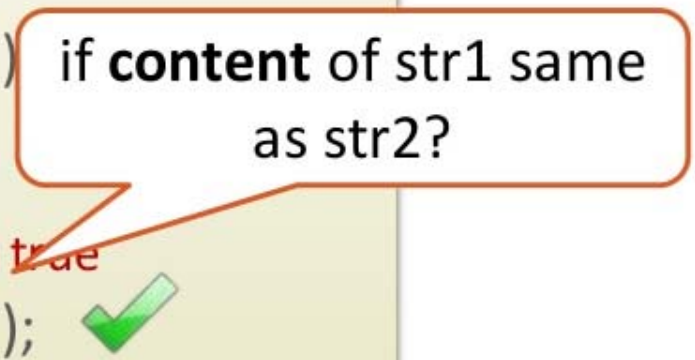
Testing String Equality

- How to check if two Strings **contain same value**?

```
String str1=new String("Hello World!");
String str2=new String("Hello World!");

if(str1==str2) { //eval to false
  System.out.println("same");
}

if(str1.equals(str2)) { //eval to true
  System.out.println("same");
}
```



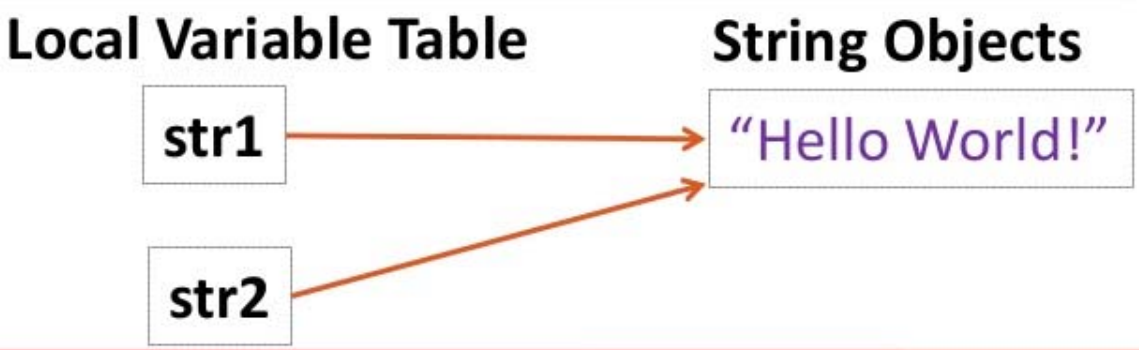
Testing String Equality

- What if "new" operator not used?

```
String str1 = "Hello World!";
String str2 = "Hello World!";

if(str1==str2) { //eval to true
  System.out.println("same");
}
```

if **str1 referencing** same object as **str2**?



Testing String Equality

- What if "new" operator not used?

```
String str1 = "Hello World!";
String str2 = "Hello World!";

if(str1==str2) { //eval to true
  System.out.println("same");
}

if(str1.equals(str2)) { //eval to true
  System.out.println("same");
}
```

Testing String Equality

- **Point to note:** String variables are references to String objects (i.e. memory addresses)
- **"str1==str2"** on String objects compares memory addresses, not the contents
- Always use **"str1.equals(str2)"** to compare contents