Working with Strings

Jim Wilson
MOBILE SOLUTIONS DEVELOPER & ARCHITECT

@hedgehogjim   jwhh.com

Overview

String class

String equality

String methods

String conversions

StringBuilder

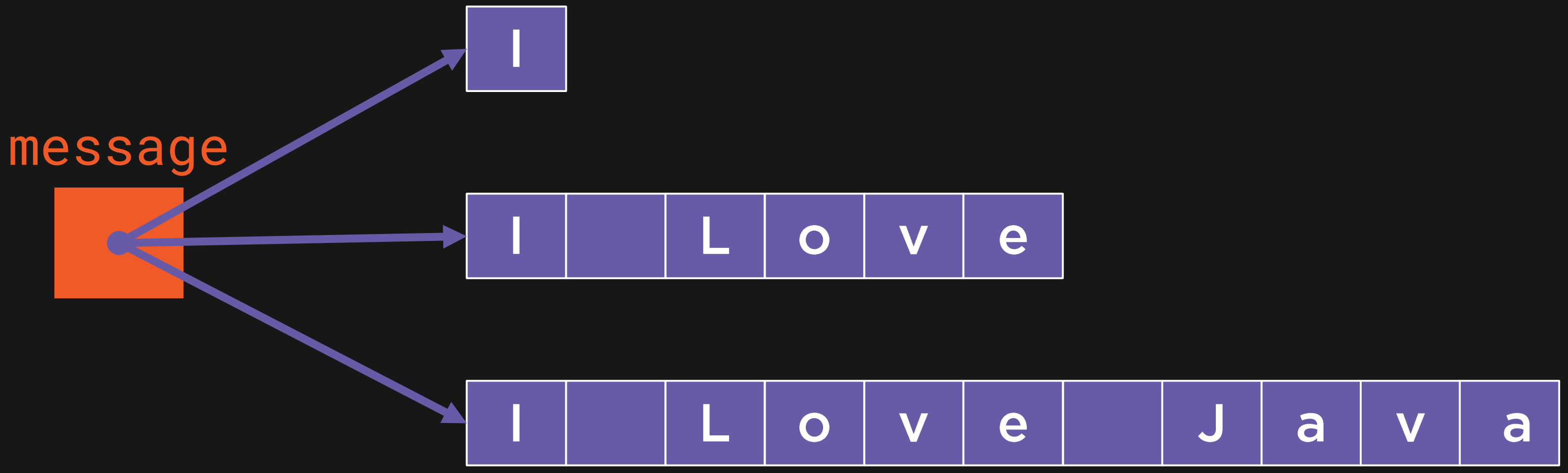
```
String name = “Jim”;

String greeting = “Hello ” + name;

System.out.println(greeting); // Hello Jim

greeting += “ good to see you!”;

System.out.println(greeting); // Hello Jim good to see you!
```

String Class

Stores a sequence of Unicode characters

- Literals are enclosed in double quotes
- Values can be concatenated using + and +=

```
String message = “I”;

message += “ Love”;

message += “ Java”;
```

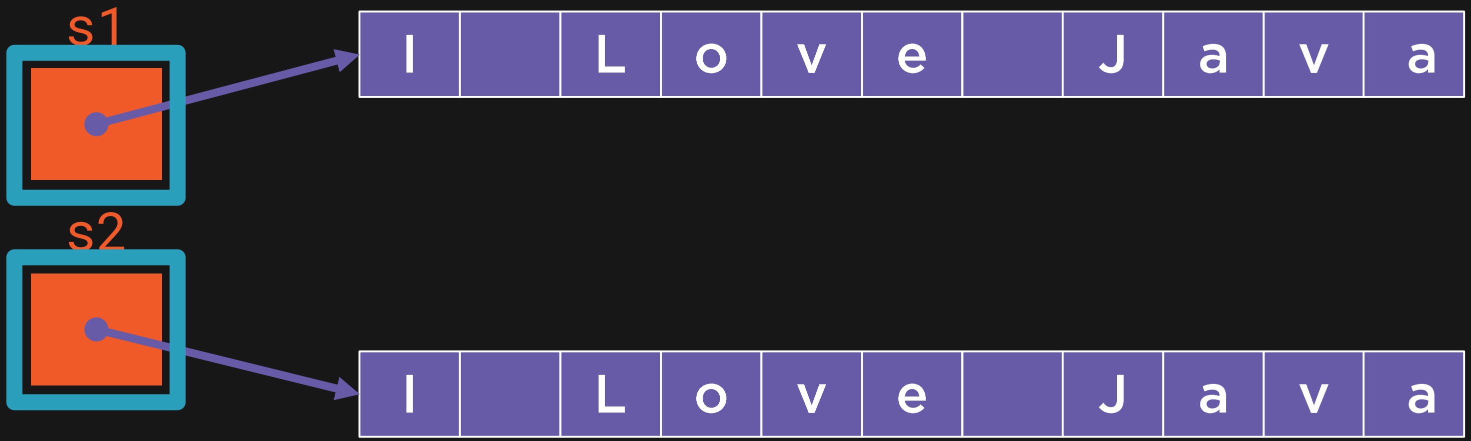


Strings Are Immutable

String variables do not directly hold the string value

- Hold a reference to the instance of string
- Changes in the value create a new instance of the string

```
String s1 = "I love";
s1 += " Java";
String s2 = "I";
s2 += " love Java";
if(s1 == s2) // false
    // do something
```



String Equality

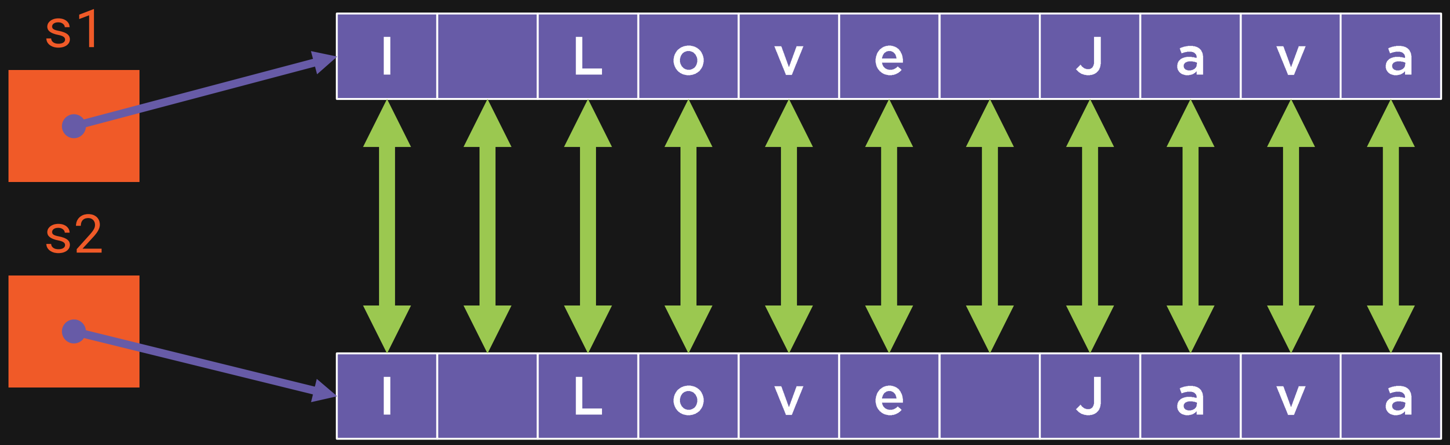
Comparing strings with the equality operator (==)

- Checks to see if both string variables reference the same string instance

Comparing strings with the equals method

- Performs a character-by-character comparison

```java
String s1 = "I love";

s1 += " Java";

String s2 = "I";

s2 += " love Java";

if(s1 == s2) // false

    // do something

if(s1.equals(s2)) // true

    // do something
```



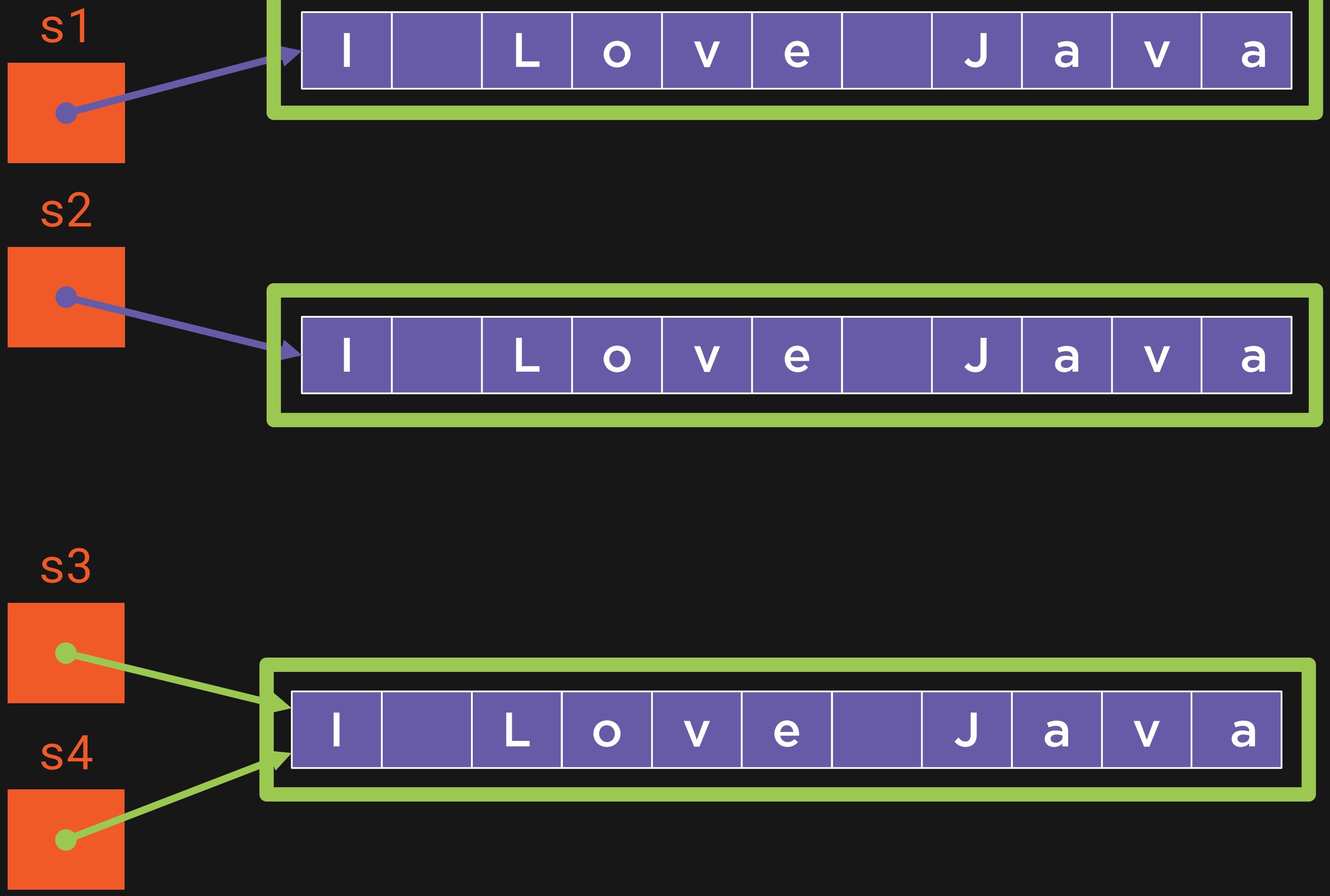
Checking string equality

- The equals method is the best choice in most cases

Interning a string

- Provides a canonicalized value
- Enables reliable == operator comparison
- Improves performance of frequently compared strings

```
String s1 = "I love";
s1 += " Java";
String s2 = "I";
s2 += " love Java";
if(s1 == s2) // false
    // do something
String s3 = s1.intern();
String s4 = s2.intern();
if(s3 == s4) // true
    // do something
```



Select String Class Methods

Operation	Methods
Length	length
Create new string(s) from existing	concat, replace, toLowerCase, toUpperCase, trim, split
Extract substring	charAt, substring
Test substring	contains, endsWith, startsWith, indexOf, lastIndexOf
Comparison	equals, equalsIgnoreCase, isEmpty, compareTo, compareToIgnoreCase
Formatting	format
String for non-string	valueOf

```
int iVal = 100;
String sVal = String.valueOf(iVal);


int i = 2, j = 3;
int result = i * j;
String output = i + “ * ” + j + “ = ” + result; // “2 * 3 = 6”
```

Converting Non-string Types to String

Virtually all data types can be converted into a String

- Can use String.valueOf
- Conversion often happens implicitly

StringBuilder

Provides mutable string buffer

- Efficiently constructs string values
- Add new content to end with append
- Add new content within with insert

Extract content to a string

- Use toString

StringBuilder

```
I flew to Florida on Flight #175
```

```
String location = "Florida";
int flightNumber = 175;
StringBuilder sb = new StringBuilder(   );
sb.append("I flew to ");
sb.append(location);
sb.append(" on Flight #");
sb.append(flightNumber);
String message = sb.toString();
```

StringBuilder

```
I flew to Florida at 9:00 on Flight #175
```

```
String time = "9:00";
int pos = sb.indexOf(" on");
sb.insert(pos, " at ");
sb.insert(pos + 4, time);
message = sb.toString();
```

Summary

String

- Sequence of Unicode characters

String variables

- Do not directly store string instance
- Hold a reference to string instance

Strings are immutable

- Changes in the value create a new string instance

Summary

String equality

- Prefer the equals method

String interning

- Provides a canonicalized value
- Enables reliable use of == operator
- Improves performance of frequently compared strings

Summary

StringBuilder

- Provides mutable string buffer
- Efficiently constructs string values
- Use toString to extract string content