

More on Strings

String methods and equality

Produced Dr. Siobhán Drohan
by: Mr. Colm Dunphy
 Mr. Diarmuid O'Connor



Waterford Institute *of* Technology
INSTITIÚID TEICNEOLAÍOCHTA PHORT LÁIRGE

Department of Computing and Mathematics
<http://www.wit.ie/>

Topics list

1. Strings: index of characters

2. String methods:

- **charAt(int index)**
- **substring (int beginIndex, int endIndex)**
- **compareTo (String anotherString)**

3. Recap: Primitive vs Object

4. String identity vs equality

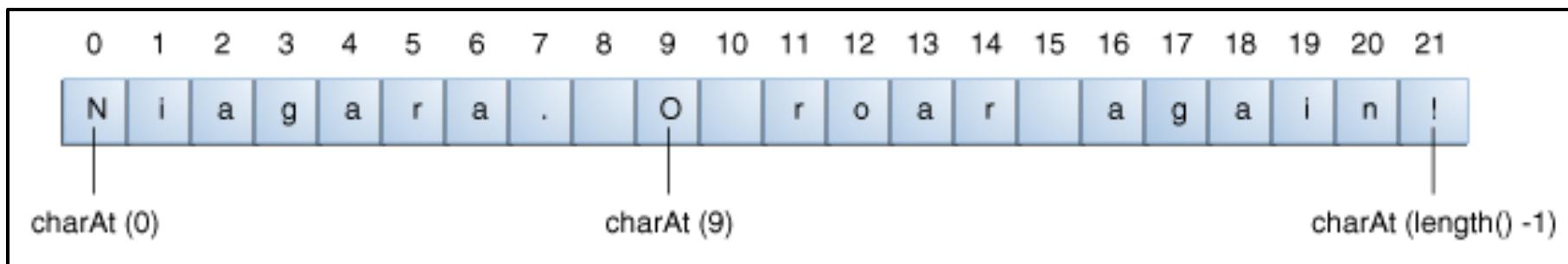
5. Common Errors with Strings

6. null

7. Escape Sequences

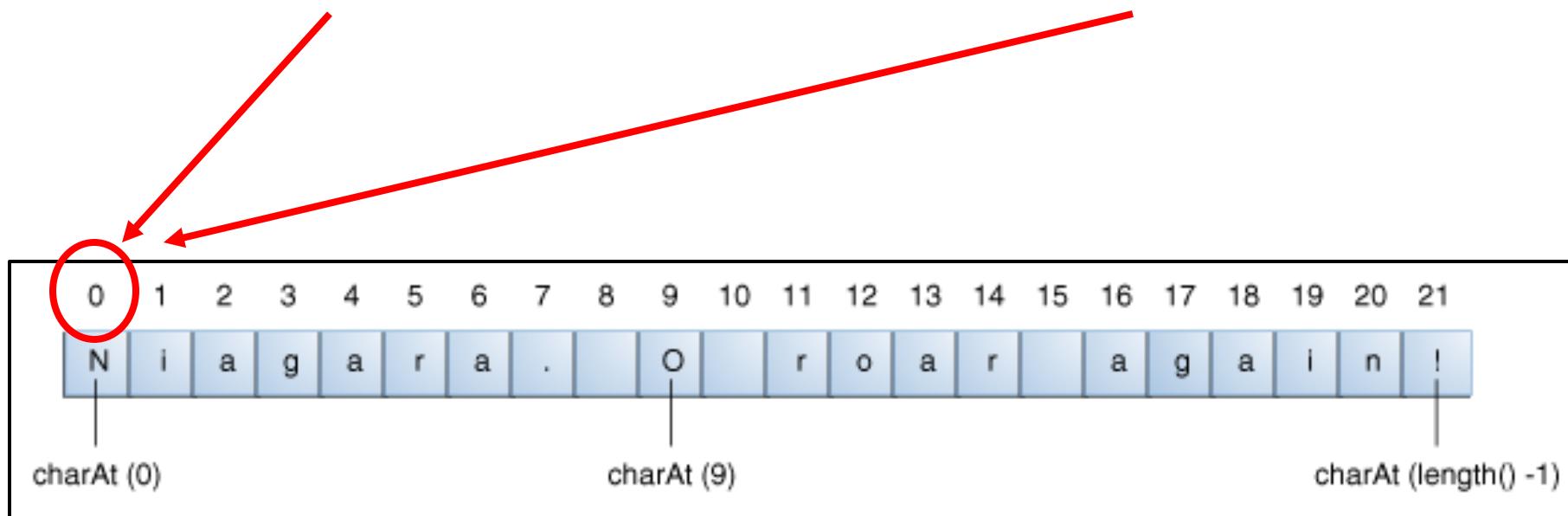
Strings: index of characters

- A String holds a sequence of characters



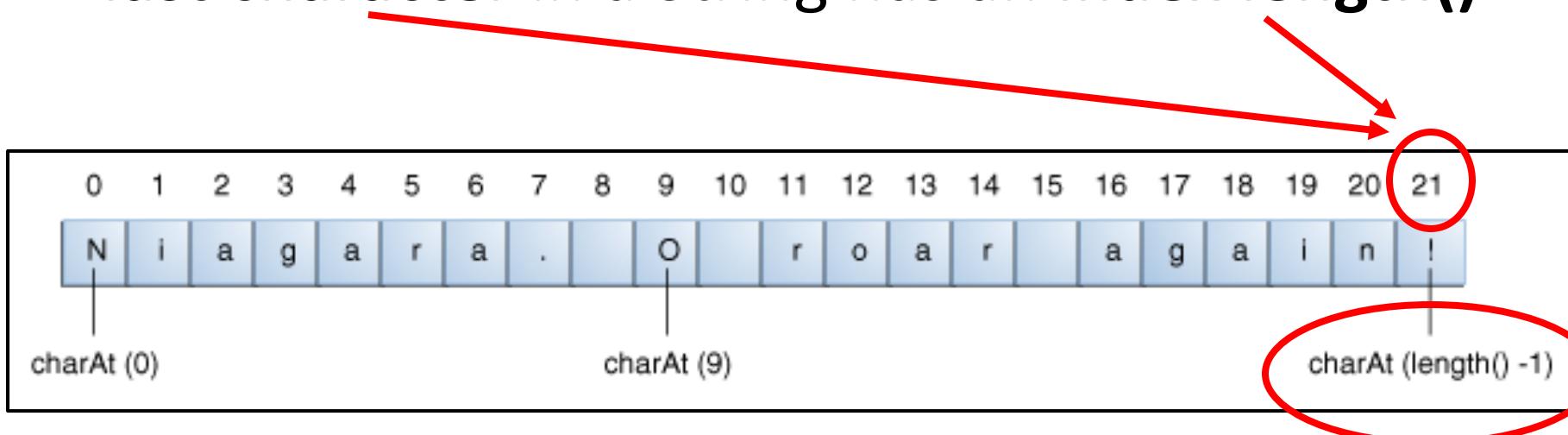
Strings: index of characters

- A String holds a sequence of characters.
- **first character** in a String has an **index 0**



Strings: index of characters

- A String holds a sequence of characters
- **first character** in a String has an **index 0**
- **last character** in a String has an **index `length()`-1**

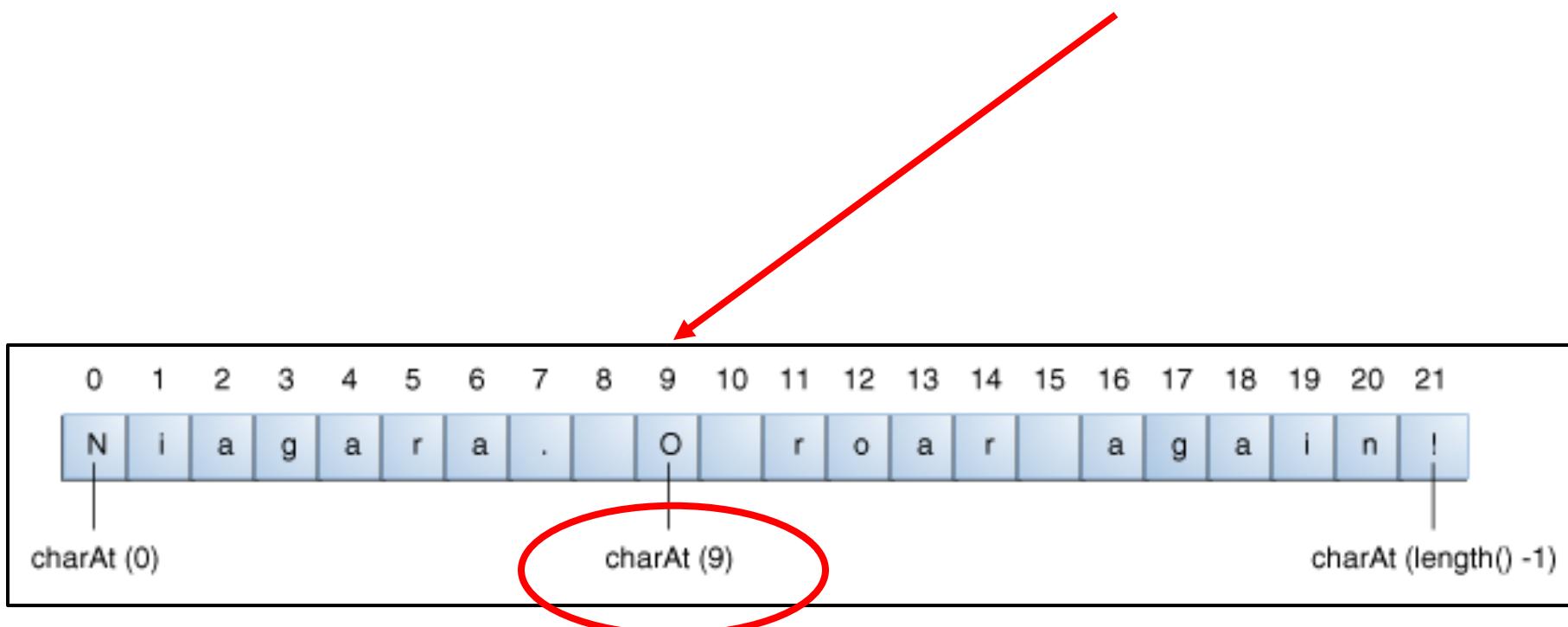


Topics list

1. Strings: index of characters
2. **String methods:**
 - – **charAt(int index)**
 - **substring (int beginIndex, int endIndex)**
 - **compareTo (String anotherString)**
3. Recap: Primitive vs Object
4. **String identity vs equality**
5. Common **Errors** with Strings
6. **null**
7. **Escape Sequences**

String methods: `charAt` (int index)

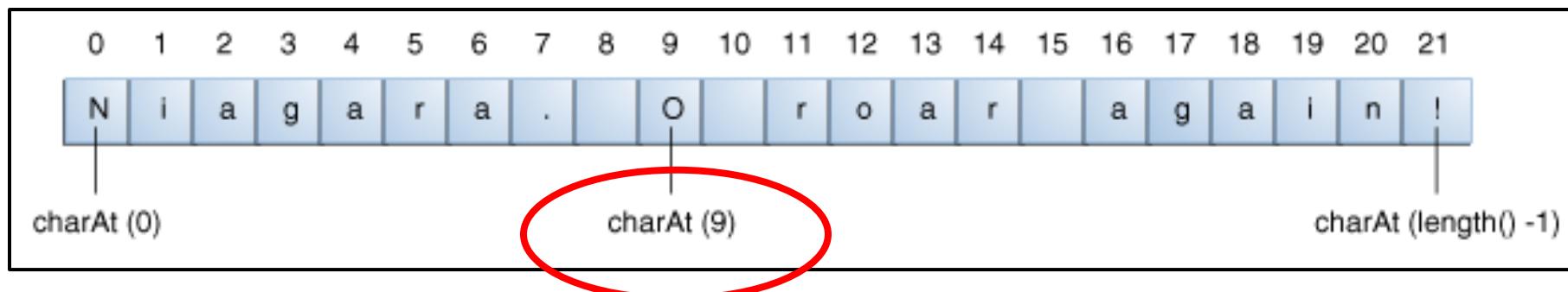
- Say we want the character at index 9 in a String:



String methods: charAt(int index)

- Say we want the character at index 9 in a String:

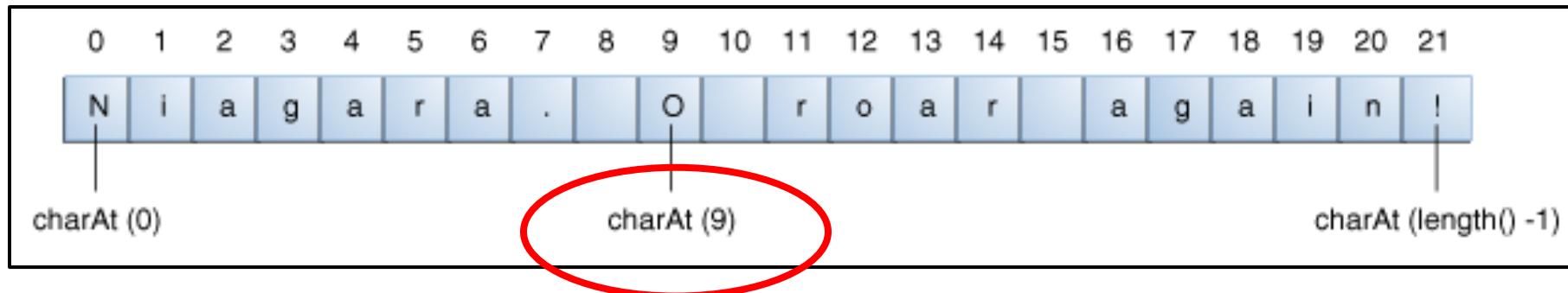
```
String anotherPalindrome = "Niagara. O roar again!";
char aChar = anotherPalindrome.charAt(9);
```



String methods: charAt(int index)

- Say we want the character at index 9 in a String:

```
String anotherPalindrome = "Niagara. O roar again!";
char aChar = anotherPalindrome.charAt(9);
```



Indices begin at 0, so the character at index 9 is 'O' i.e. the 10th character

Finding the character located at specific **position** in a String.

Example 4.1

Example_4_1

```
1 String alphabet = "abcdefghijklmnopqrstuvwxyz";
2 String errorMessage404 = "HTTP 404 Not Found Error";
3
4 println("The character at position 4 in "
5         + alphabet
6         + " is "
7         + alphabet.charAt(3));
8
9 println("The character at position 10 in "
10        + errorMessage404
11        + " is "
12        + errorMessage404.charAt(9));
```

position 4
= **index 3**
= **d**

position 10
= **index 9**
= **N**

The character at position 4 in abcdefghijklmnopqrstuvwxyz is d
The character at position 10 in HTTP 404 Not Found Error is N

Console

Errors

Example 4.1

Example_4_1

```
1 String alphabet = "abcdefghijklmnopqrstuvwxyz";
2 String errorMessage404 = "HTTP 404 Not Found Error";
3
4 println("The character at position 4 in "
5         + alphabet
6         + " is "
7         + alphabet.charAt(3));
8
9 println("The character at position 10 in "
10        + errorMessage404
11        + " is "
12        + errorMessage404.charAt(9));
```

Finding the character located at specific position in a String.

The character at position 4 in abcdefghijklmnopqrstuvwxyz is d
The character at position 10 in HTTP 404 Not Found Error is N

Console

Errors

Example 4.1

Example_4_1

```
1 String alphabet = "abcdefghijklmnopqrstuvwxyz";
2 String errorMessage404 = "HTTP 404 Not Found Error";
3
4 println("The character at position 4 in "
5         + alphabet
6         + " is "
7         + alphabet.charAt(3));
8
9 println("The character at position 10 in "
10        + errorMessage404
11        + " is "
12        + errorMessage404.charAt(9));
```

Finding the character located at specific position in a String.

The character at position 4 in abcdefghijklmnopqrstuvwxyz is d
The character at position 10 in HTTP 404 Not Found Error is N

Console

Errors

Topics list

1. Strings: index of characters
2. **String methods:**
 - `charAt(int index)`
 - – `substring (int beginIndex, int endIndex)`
 - `compareTo (String anotherString)`
3. Recap: Primitive vs Object
4. **String identity vs equality**
5. Common **Errors** with Strings
6. **null**
7. **Escape Sequences**

String methods:

substring (int beginIndex, int endIndex)

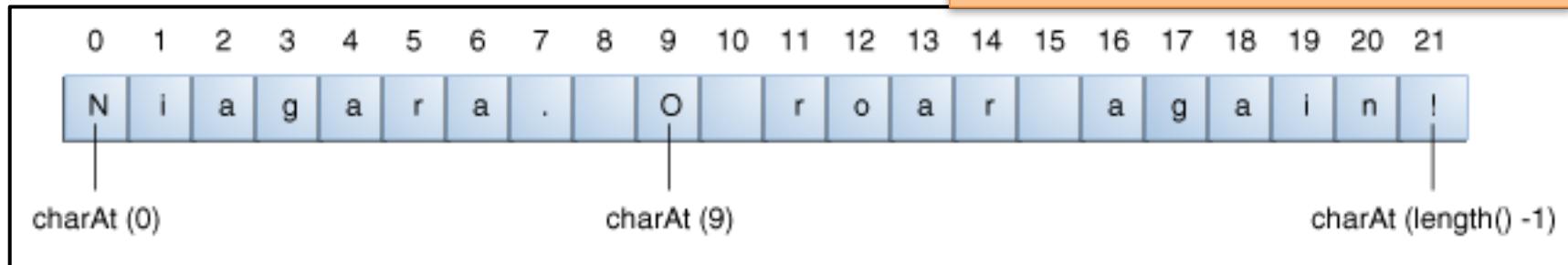
- This method returns a new String that is a substring of this String.

String methods:

substring (int beginIndex, int endIndex)

- This method returns a new String that is a substring of this String.

Given this String...

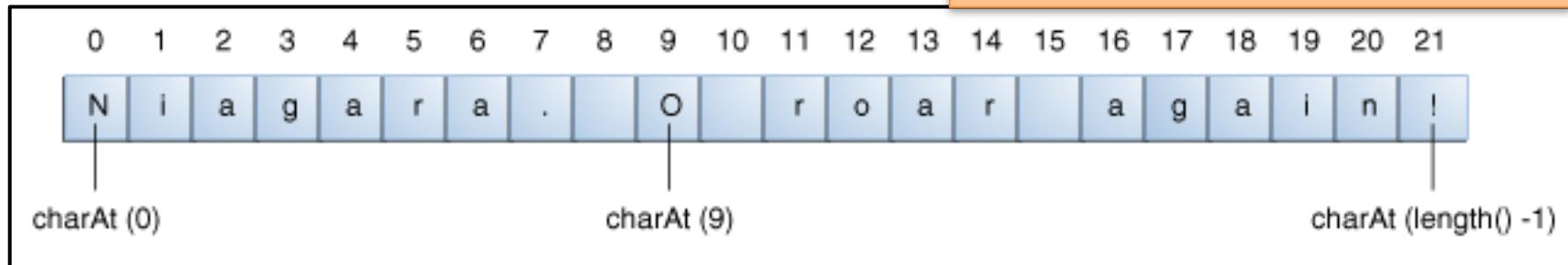


String methods:

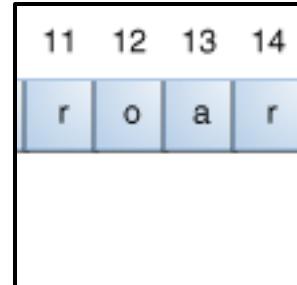
substring (int beginIndex, int endIndex)

- This method returns a new String that is a substring of this String.

Given this String...



...this is a substring →

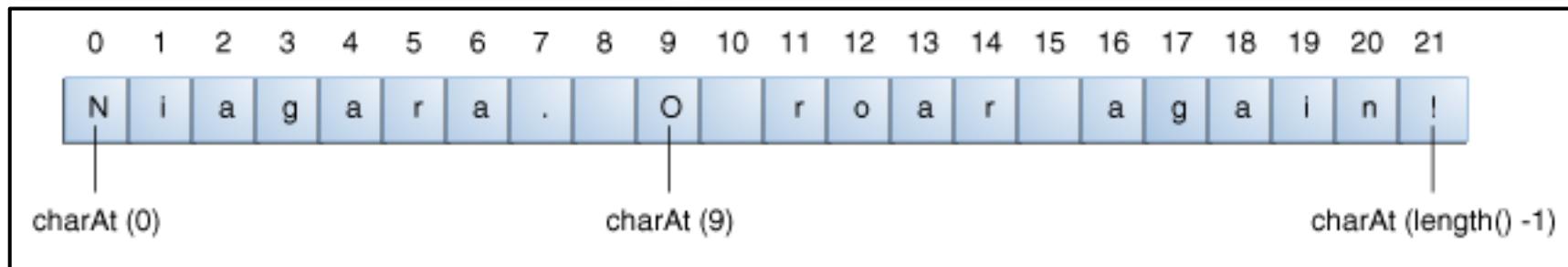


String methods:

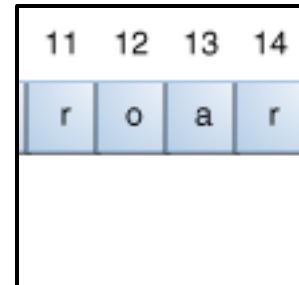
substring (int beginIndex, int endIndex)

The substring begins
at the specified
beginIndex...

...and extends to the
character at index **endIndex-1**

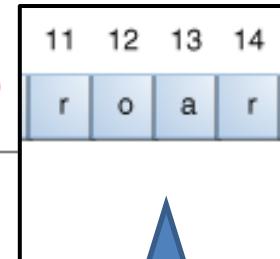


...this is a substring →



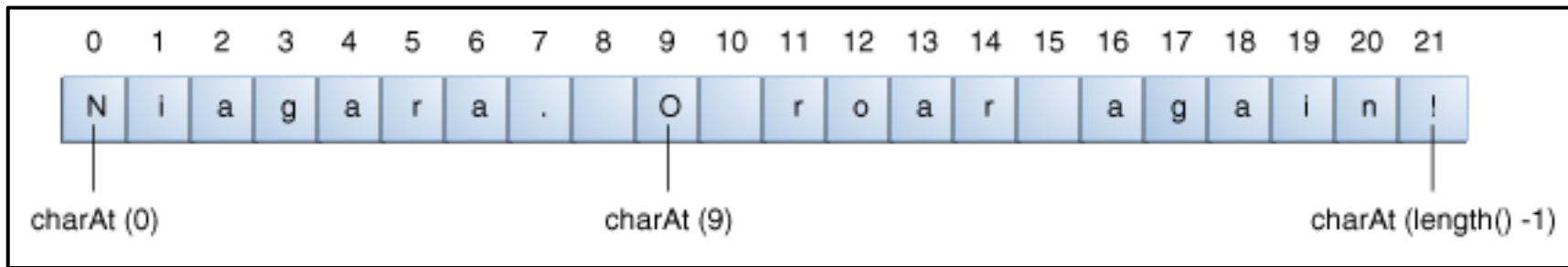
String methods:

substring (int beginIndex, int endIndex)



The substring begins
at the specified
beginIndex...

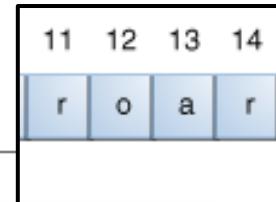
...and extends to the
character at index **endIndex-1**



```
String anotherPalindrome = "Niagara. O roar again!";
String roar = anotherPalindrome.substring(11, 15);
```

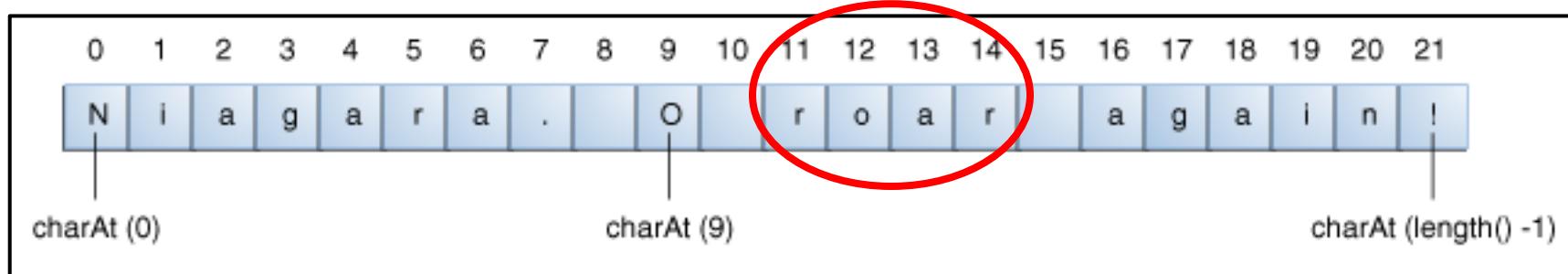
String methods:

substring (int beginIndex, int endIndex)



This code returns a substring ("roar") from anotherPalindrome.

It extends from index **11** up to **15 -1**, i.e. 11,12,13,14



```
String anotherPalindrome = "Niagara. O roar again!";
String roar = anotherPalindrome.substring(11, 15);
```

Example 4.2, version 1

Example_4_2 ▾

```
1 String anotherPalindrome = "Niagara. 0 roar again!";  
2 String roar = anotherPalindrome.substring(11, 15);  
3 print(roar);
```

Printing out a substring of a String
to the console.

roar

Console

Errors

Example 4.2, version 2

```
Example_4_2
1 //Version 2 (without roar variable)
2 String anotherPalindrome = "Niagara. O roar again!";
3 print(anotherPalindrome.substring(11, 15));
4
```

Printing out a substring of a String
to the console.

roar

Console

Errors

Topics list

1. Strings: index of characters
2. **String methods:**
 - `charAt(int index)`
 - `substring (int beginIndex, int endIndex)`
 - – `compareTo (String anotherString)`
3. Recap: Primitive vs Object
4. **String identity vs equality**
5. Common **Errors** with Strings
6. **null**
7. **Escape Sequences**

String methods: `compareTo`

`int compareTo (String anotherString)`

- This method compares two strings **lexicographically**
 - i.e.
based on the Unicode value of the characters in the String.
- It returns an integer indicating whether this string is:
 - greater than (result is > 0)
 - equal to (result is $= 0$) or
 - less than (result is < 0)the argument, `anotherString`.

Examples 4.3 - 4.6

- In the next 4 examples we compare 2 strings
`str1.compareTo(str2)`
- where str2 = "Cat"
- And str1 =
 - "Dog"
 - then "cat"
 - then "Animal"
 - then "Cat"

Example 4.3 – Dog

```
String str1 = "Dog";  
String str2 = "Cat";
```

Q: What will be printed to the console?

Q: Which boolean expression evaluates to true?

```
if (str1.compareTo(str2) < 0)      { // before  
    println(str1 + " comes before " + str2 + " in the alphabet");  
}  
else if (str1.compareTo(str2) > 0)  { // after  
    println (str1 + " comes after " + str2 + " in the alphabet");  
}  
else{  
    println ("The strings are identical");  
}
```

Example 4.3

```
String str1 = "Dog";  
String str2 = "Cat";
```

A: str1.compareTo (str2)

returns a positive integer
as “Dog” (str1) comes **after** “Cat” (str2).

```
if (str1.compareTo(str2) < 0)      { // before  
    println(str1+" comes before "+str2 +" in the alphabet");  
}  
  
else if (str1.compareTo(str2) > 0)  { // after  
    println (str1 + " comes after "+str2 +" in the alphabet");  
}  
  
else{  
    println ("The strings are identical");  
}
```

Dog comes after Cat in the alphabet

Console

Errors

Example 4.4 - cat

```
String str1 = "cat";  
String str2 = "Cat";
```

Q: What will be printed to the console?

Q: Which boolean expression evaluates to true?

```
if (str1.compareTo(str2) < 0)      { // before  
    println(str1+" comes before "+str2 +" in the alphabet");  
}  
else if (str1.compareTo(str2) > 0)  { // after  
    println (str1 +" comes after "+str2 +" in the alphabet");  
}  
else{  
    println ("The strings are identical");  
}
```

Example 4.4

```
String str1 = "cat";  
String str2 = "Cat";
```

A: str1.compareTo(str2)

returns a positive integer
as “cat” (str1) comes after “Cat” (str2)
in the Unicode character map.

```
if (str1.compareTo(str2) < 0) { // before  
    println(str1 + " comes before " + str2 + " in the alphabet");  
}  
  
else if (str1.compareTo(str2) > 0) { // after  
    println (str1 + " comes after " + str2 + " in the alphabet");  
}  
  
else{  
    println ("The strings are identical");  
}
```

cat comes after Cat in the alphabet

>- Console

! Errors

Example 4.5 - Animal

```
String str1 = "Animal";  
String str2 = "Cat";
```

Q: What will be printed to the console?

Q: Which boolean expression evaluates to true?

```
if (str1.compareTo(str2) < 0)      { // before  
    println(str1+" comes before "+str2 +" in the alphabet");  
}  
else if (str1.compareTo(str2) > 0)  { // after  
    println (str1 +" comes after "+str2 +" in the alphabet");  
}  
else{  
    println ("The strings are identical");  
}
```

Example 4.5

```
String str1 = "Animal";  
String str2 = "Cat";
```

A: str1.compareTo(str2)

returns a negative integer
as **Animal(str1)** comes before **Cat (str2)**
in the Unicode character map.

```
if (str1.compareTo(str2) < 0)      { // before  
    println(str1+" comes before "+str2 +" in the alphabet");  
}  
  
else if (str1.compareTo(str2) > 0)  { // after  
    println (str1 +" comes after "+str2 +" in the alphabet");  
}  
  
else{  
    println ("The strings are identical");  
}
```

Animal comes before Cat in the alphabet

Example 4.6 - Cat

```
String str1 = "Cat";  
String str2 = "Cat";
```

Q: What will be printed to the console?

Q: Which boolean expression evaluates to true?

```
if (str1.compareTo(str2) < 0)      { // before  
    println(str1+" comes before "+str2 +" in the alphabet");  
}  
else if (str1.compareTo(str2) > 0)  { // after  
    println (str1 +" comes after "+str2 +" in the alphabet");  
}  
else{  
    println ("The strings are identical");  
}
```

Example 4.6

```
String str1 = "Dog";  
String str2 = "Cat";
```

A: str1.compareTo(str2)

returns 0
as Cat (str1) is identical to Cat (str2).

```
if (str1.compareTo(str2) < 0) { // before  
    println(str1 + " comes before " + str2 + " in the alphabet");  
}  
  
else if (str1.compareTo(str2) > 0) { // after  
    println (str1 + " comes after " + str2 + " in the alphabet");  
}  
  
else{  
    println ("The strings are identical");  
}
```

The strings are identical

Console

Errors

Topics list

1. Strings: index of characters
2. **String methods:**
 - **charAt(int index)**
 - **substring (int beginIndex, int endIndex)**
 - **compareTo (String anotherString)**
3. Recap: Primitive vs Object
4. **String identity vs equality**
5. Common **Errors** with Strings
6. **null**
7. **Escape Sequences**

Recap: Object types

e.g. **String**

- Strings
 - are a sequence of characters enclosed by double quotes ")
- **String**
 - is an object type.
- The Java API
 - provides information about the String class
 - lists methods that can be used on Strings
 - (<https://docs.oracle.com/javase/8/docs/api/java/lang/String.html>).
- The most direct way to create a String is to write:
String greeting = "Hello world!";

Primitive types vs. Object types

Primitive type

```
int i = 17;
```

Primitive types vs. Object types

Primitive type

```
int i = 17;
```

**Directly stored
in memory...**

17

Primitive types vs. Object types

Primitive type

```
int i = 17;
```

Object type

```
String hi = "Hello";
```

Directly stored
in memory...

17

Primitive types vs. Object types

Primitive type

```
int i = 17;
```

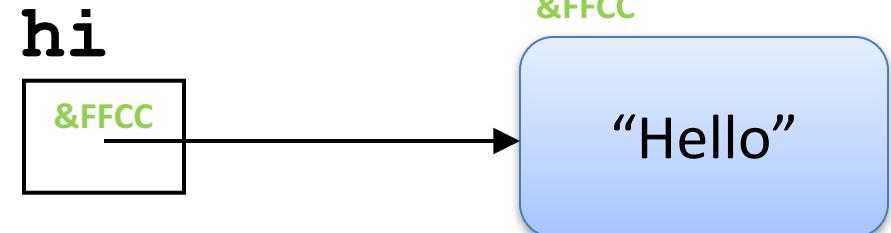
Directly stored
in memory...

17

Object type

```
String hi = "Hello";
```

hi variable
contains a **reference (address)**
to where the String is stored in
memory



Primitive types vs. Object types

Primitive type

```
int i = 17;
```

Directly stored
in memory...

17

With **primitive** type variables
(e.g. int, float, char, etc)

the **value** of the variable
is stored
in the memory location
assigned to the variable.

Primitive types vs. Object types

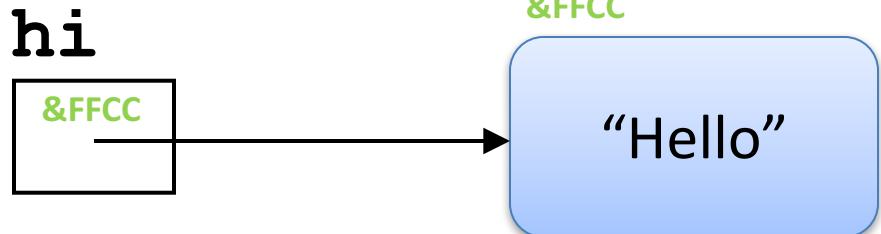
With **object** types,
the variable holds the
memory address
of where the object is
located
– **not the values**
inside the object.

This memory address
is called a **reference**
to the object.

Object type

`String hi = "Hello";`

hi variable
contains a reference (*address*)
to where the String is stored in
memory



Primitive types vs. Object types

Now that we know how primitive types and object types store data,

we will look at this statement (`b=a`)
in the context of primitive and object types.

b = a;

Primitive types vs. Object types

Primitive types

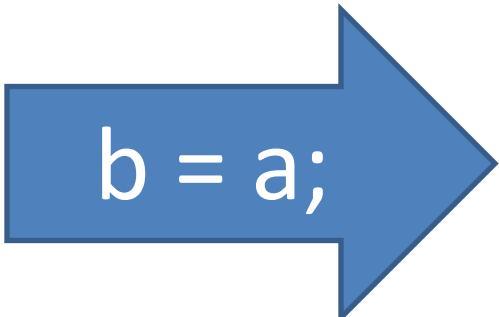
```
int a;
```

Primitive types vs. Object types

Primitive types

```
int a;
```

17



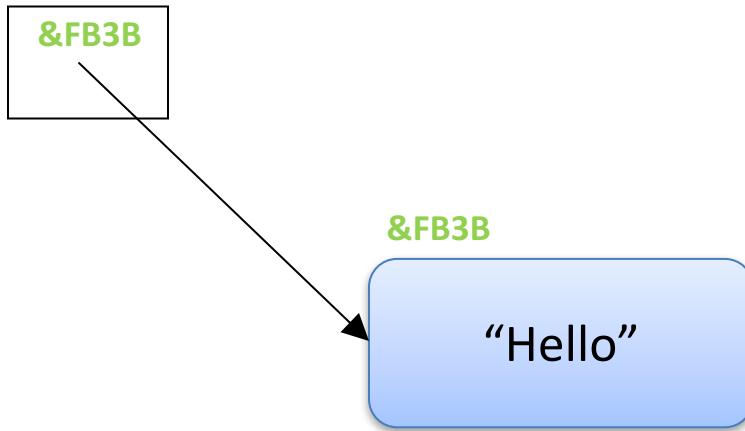
```
b = a;
```

```
int b;
```

17

Primitive types vs. Object types

```
String a;
```



```
b = a;
```

Object types

Primitive types vs. Object types

String a;

&FB3B

b = a;

String b;

&FB3B

&FB3B

“Hello”

b = a;

Object types

Primitive types vs. Object types

String a;

&FB3B

b = a;

String b;

&FB3B

&FB3B

“Hello”

b = a;

int a;

17

b = a;

int b;

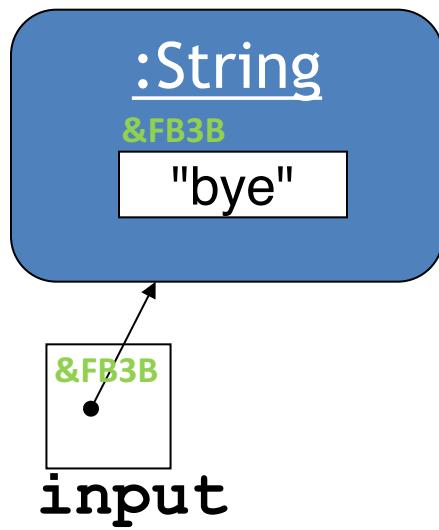
17

Topics list

1. Strings: index of characters
2. **String methods:**
 - **charAt(int index)**
 - **substring (int beginIndex, int endIndex)**
 - **compareTo (String anotherString)**
3. Recap: Primitive vs Object
4. **String identity vs equality**
5. Common **Errors** with Strings
6. **null**
7. **Escape Sequences**

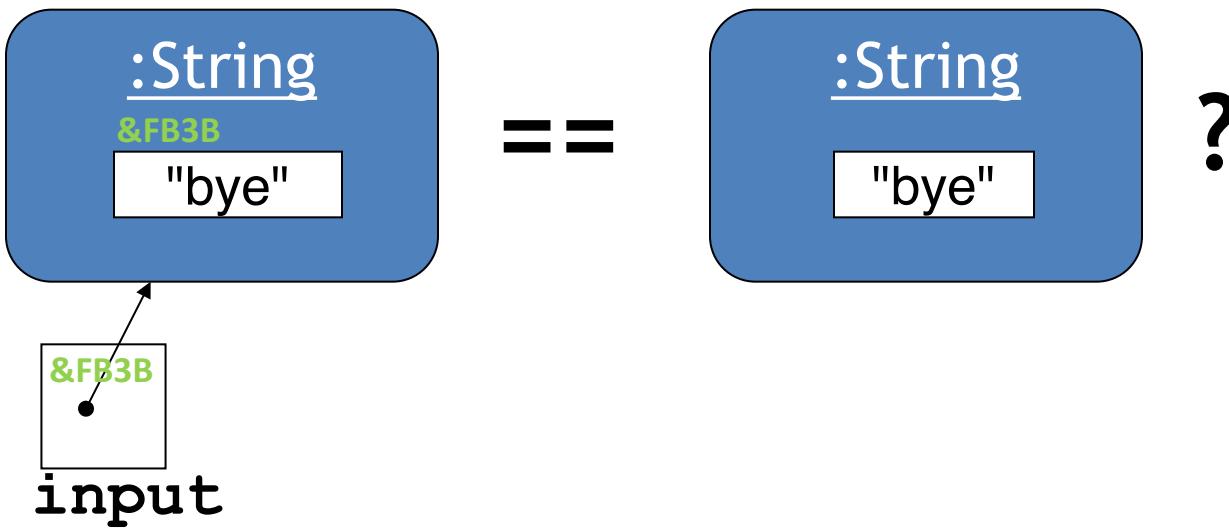
String: Identity vs Equality

```
String input = "bye";
```



String: Identity vs Equality

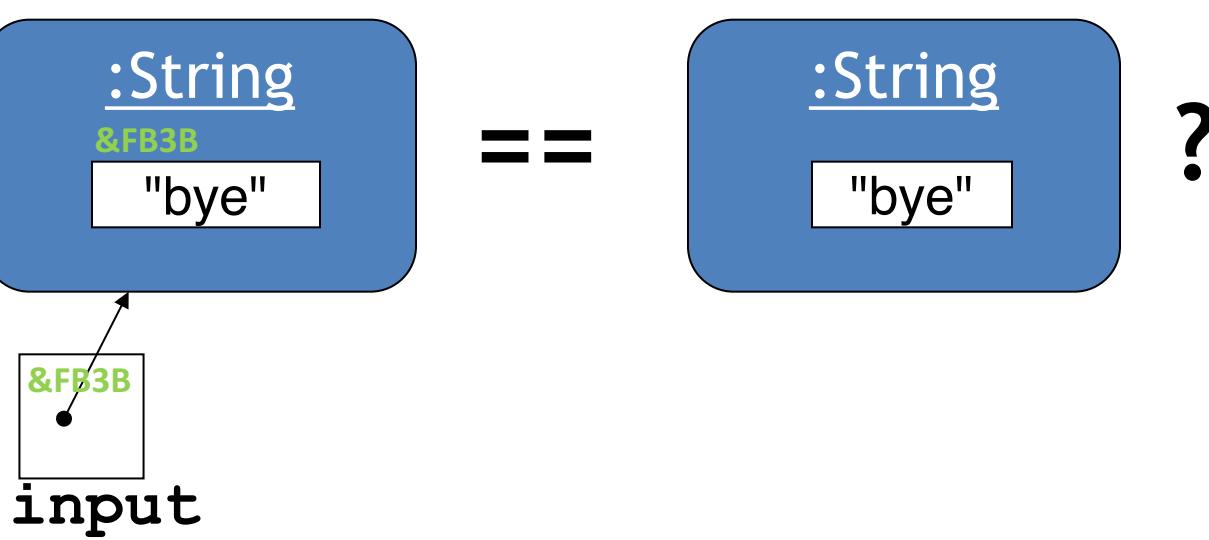
```
String input = "bye";
if(input == "bye") {
    //...
}
```



String: Identity vs Equality

```
String input = "bye";
if(input == "bye") {
    //...
}
```

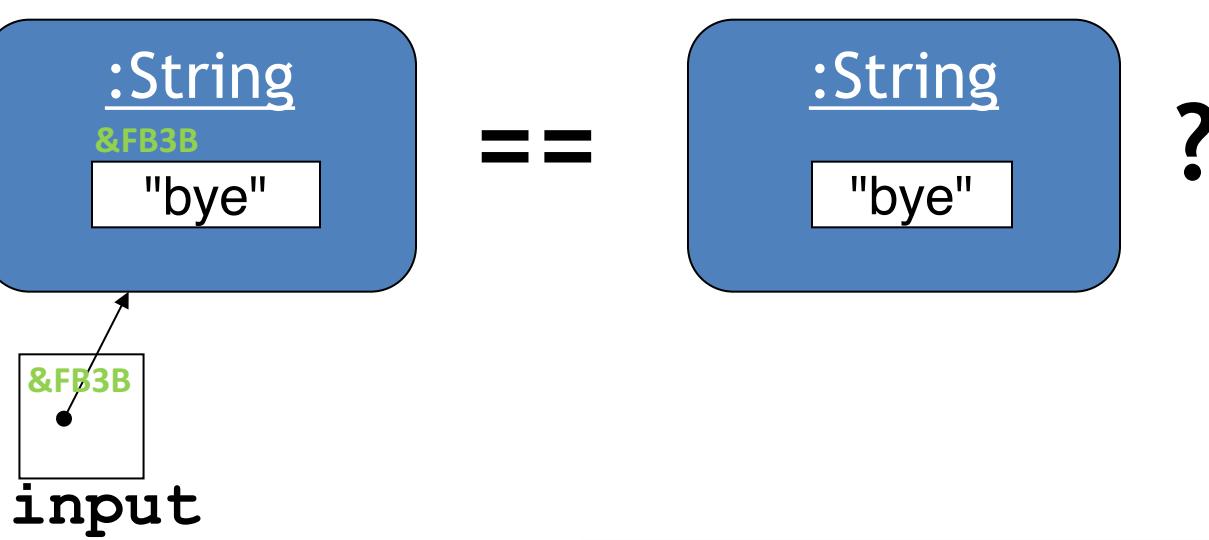
`==` tests identity



String: Identity vs Equality

```
String input = "bye";
if(input == "bye") {
    //...
}
```

`==` tests identity

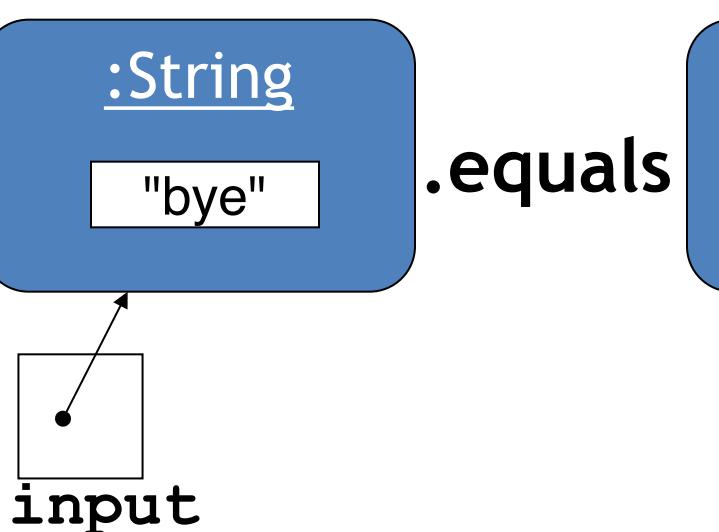


Answer: (maybe) false!

String: Identity vs Equality

```
String input = "bye";  
if(input.equals("bye")) {  
    ...  
}
```

.equals tests equality



Answer: true

"bye" equals "bye"

String: Identity vs Equality

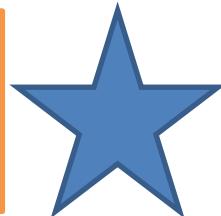
```
if(input == "bye") {  
    ...  
}
```

tests **identity**
i.e. the reference

```
if(input.equals("bye")) {  
    ...  
}
```

tests **equality**
i.e. string value

Strings should always be compared
using the **.equals** method



Topics list

1. Strings: index of characters
2. **String methods:**
 - **charAt(int index)**
 - **substring (int beginIndex, int endIndex)**
 - **compareTo (String anotherString)**
3. Recap: Primitive vs Object
4. **String identity vs equality**
5. Common **Errors** with Strings
6. **null**
7. **Escape Sequences**

Q1: What's wrong here?

```
void anyMethod()
{
    String str1 = "a";
    String str2 = "b";

    if(str1 == str2)
    {
        println(str1+" is the same as "+ str2);
    }
    else
    {
        println(str1+" is NOT same as "+ str2);
    }
}
```

A1: Strings need to use the .equals method

```
void anyMethod()
{
    String str1 = "a";
    String str2 = "b";

    if(str1 == str2)
    {
        println(str1+" is the same as "+ str2);
    }
    else
    {
        println(str1+" is NOT same as "+ str2);
    }
}
```

Q2: What's wrong here?

```
public void anyMethod()
{
    int num1 = 1;
    int num2 = 2;

    if(num1 = num2)
        println(num1+" is the same as "+ num2);
    else
        println(num1+" is NOT same as "+ num2);
}
```

A: You need two equals for equality

```
public void anyMethod()
{
    int num1 = 1;
    int num2 = 2;

    if(num1 = num2)
        println(num1+" is the same as "+ num2);
    else
        println(num1+" is NOT same as "+ num2);
}
```

Topics list

1. Strings: index of characters
2. **String methods:**
 - **charAt(int index)**
 - **substring (int beginIndex, int endIndex)**
 - **compareTo (String anotherString)**
3. Recap: Primitive vs Object
4. **String identity vs equality**
5. Common **Errors** with Strings
6. **null**
7. **Escape Sequences**

null

- `null` is a special value in Java.
- All object variables are initialised to `null`.

null

- null means that the object variable does not have a reference

e.g.

- str1 below has a reference to the string “Hello World!”
- str2 below does not have a reference. It is null.

String str1;

&FB3B

&FB3B

“Hello World!”

String str2;

null

null

You can assign and test for **null**:

```
String hours;  
  
if(hours == null)  
{  
    // ...  
}  
hours = null;
```

Topics list

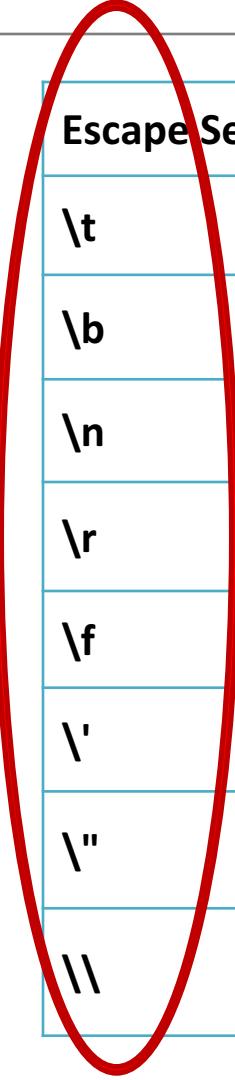
1. Strings: index of characters
2. **String methods:**
 - **charAt(int index)**
 - **substring (int beginIndex, int endIndex)**
 - **compareTo (String anotherString)**
3. Recap: Primitive vs Object
4. **String identity vs equality**
5. Common **Errors** with Strings
6. **null**
7. **Escape Sequences**

Escape sequences

When a String is printed,
certain **single characters that follow a backslash (\)**
have special meaning...

...and the compiler interprets them accordingly.

Java escape sequences



Escape Sequence	Description
\t	Insert a tab in the text at this point.
\b	Insert a backspace in the text at this point.
\n	Insert a newline in the text at this point.
\r	Insert a carriage return in the text at this point.
\f	Insert a formfeed in the text at this point.
\'	Insert a single quote character in the text at this point.
\\"	Insert a double quote character in the text at this point.
\\	Insert a backslash character in the text at this point.

Examples of escape sequences

```
print("Java\n");
```

is the exact same as:

```
println("Java");
```

```
println("    Java");
```

is similar to:

```
println("\tJava");
```

Summary

1. Strings: index of characters
2. **String methods:**
 - **charAt(int index)**
 - **substring (int beginIndex, int endIndex)**
 - **compareTo (String anotherString)**
3. Recap: Primitive vs Object
4. **String identity vs equality**
5. Common **Errors** with Strings
6. **null**
7. **Escape Sequences**

Questions?

