



if-else Strings

© A+ Computer Science - www.apluscompsci.com

What is an if else?

© A+ Computer Science - www.apluscompsci.com

the if-else statement

**if I am tired
I go to sleep
else
I go for a run**



© A+ Computer Science - www.apluscompsci.com

If-else statements are just simple decision-making statements.

A condition is checked and something may or may not happen based on the evaluation of that condition.

the if-else statement

```
if I like the current song  
  I make it louder  
else  
  I change the song
```



© A+ Computer Science - www.apluscompsci.com

If-else statements are just simple decision-making statements.

A condition is checked and something may or may not happen based on the evaluation of that condition.

the if-else statement

```
if( boolean condition placed here )  
{  
    do something 1;  
}  
else  
{  
    do something 2;  
}
```



© A+ Computer Science - www.apluscompsci.com

If the condition is true, do something 1 will occur.

If the condition is false, do something 2 will occur.

the if-else statement

```
int num=990;  
if(num>100)  
{  
    System.out.println("> 100!");  
}  
else  
{  
    System.out.println("! > 100!");  
}
```

OUTPUT

> 100!



© A+ Computer Science - www.apluscompsci.com

If num is greater than 100, >100! is displayed.

If num is not greater than 100, !>100! is displayed.

the if-else statement

```
int num=50;  
if(num>100)  
{  
    System.out.println("> 100!");  
}  
else  
{  
    System.out.println("! > 100!");  
}
```

OUTPUT

! > 100!



© A+ Computer Science - www.apluscompsci.com

If num is greater than 100, >100! is displayed.

If num is not greater than 100, !>100! is displayed.

the if-else statement

```
int num=100;  
if(num>=100)  
{  
    System.out.println(">= 100!");  
}  
else  
{  
    System.out.println("! >= 100!");  
}
```

OUTPUT

>= 100!



© A+ Computer Science - www.apluscompsci.com

If num is greater than or equal to 100, >=100! is displayed.

If num is not greater than or equal to 100, !=100! is displayed.

the if-else statement

```
int uilScore=200;  
if(uilScore>190)  
{  
    System.out.println("team");  
}  
else  
{  
    System.out.println("bench");  
}
```

OUTPUT

team



© A+ Computer Science - www.apluscompsci.com

If uilScore is greater than 190, team is displayed.

If uilScore is not greater than 190, bench is displayed.

the if-else statement

```
String s = "one";  
if(s.equals("one"))  
{  
    System.out.println(s + " is one!");  
}  
else  
{  
    System.out.println(s + " is not one!");  
}
```

OUTPUT

one is one!



© A+ Computer Science - www.apluscompsci.com

s is a String reference. s stores the location/address of a String Object.

The equals () method compares the contents of two String Objects to see if they contain the same letters in the same order in the same case.

If s contains the letters one, one is one! is displayed.

If s does not contain the letters one, letters is not one! is displayed.

open ifelse.java

© A+ Computer Science - www.apluscompsci.com

**open
ifelsestring.java**

© A+ Computer Science - www.apluscompsci.com

nesting ifs

```
int num=1;  
if(num>2)  
{  
    if(num<10)  
        System.out.println(">2<10");  
}  
else{  
    System.out.println("<2");  
}
```

OUTPUT

<2



© A+ Computer Science - www.apluscompsci.com

Nesting occurs when one thing is placed inside of another thing.

`if (num<10)` has been nested inside of `if (num>2)`

`if (num<10)` will only be tested if `if (num>2)` is true.

The `else` is associated with `if (num>2)` . Without the braces, the `else` would be associated with `if (num<10)` as `if` and `else` are paired based on proximity.

nesting ifs

```
int num=11;  
if(num>2)  
    if(num<10)  
        System.out.println(">2<10");  
else  
    System.out.println("<2");
```

OUTPUT

<2

Always use braces with ifs to indicate which statements are related.



© A+ Computer Science - www.apluscompsci.com

Nesting occurs when one thing is placed inside of another thing.

`if (num<10)` has been nested inside of `if (num>2)`

`if (num<10)` will only be tested if `if (num>2)` is true.

The `else` is associated with `if (num<10)` . If braces were present around `if (num<10)` , the `else` would be associated with `if (num>2)` as `if` and `else` are paired based on proximity.

open
ifnesting.java
danglingelse.java

© A+ Computer Science · www.apluscompsci.com

COMMON ERRORS

```
if(total >= 25)
{
}
else(total = 10)
{
}
```



© A+ Computer Science - www.apluscompsci.com

{ and ; rule

Never put a ;
before an open { brace

;{ **illegal**

}; **legal**



© A+ Computer Science - www.apluscompsci.com

Start work on the labs

© A+ Computer Science - www.apluscompsci.com

More Strings

© A+ Computer Science - www.apluscompsci.com

String Objects

String objects are immutable.

The String class does not contain any modifier methods.

```
new String("uiltcea");  
"statechamps"  
"alligator"
```

© A+ Computer Science - www.apluscompsci.com

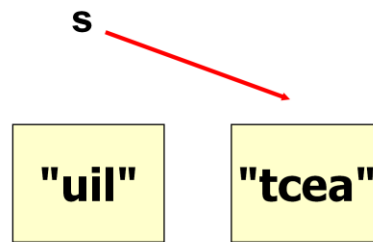
Once a String Object has been instantiated, that String Object can never be modified.

The String class does not contain any modifier methods.

String References

A String reference variable can be changed, but the String object the variable refers to cannot be changed.

```
String s = "uil";  
out.println(s);  
s = "tcea";  
out.println(s);
```



© A+ Computer Science - www.apluscompsci.com

`s` is a reference that refers to a String Object.

`s` starts out referring to String Object `uil`.

`s` stores the location/address of String Object `uil`.

`s` is then referred to the String Object `tcea`.

`s` stores the location/address of String Object `tcea`.

String References

A String reference variable can be changed, but the String object the variable refers to cannot be changed.

```
String s = "compsci ";  
out.println(s);  
s.toUpperCase();  
out.println(s);  
s=s.toUpperCase();  
out.println(s);
```

OUTPUT
compsci
compsci
COMPSCI

© A+ Computer Science - www.apluscompsci.com

s is a reference that refers to a String Object.

s starts out referring to String Object comps*c*i.

s.toUpperCase() returns a new String COMPSCI.

s is still referring to String Object comps*c*i.

s is referred to s.toUpperCase().

s.toUpperCase() returns a new String COMPSCI.

s is now referring to String Object COMPSCI.

String References

```
String one = new String("compsci");  
String two = new String("compsci");
```

```
if(one==two)  
    System.out.println("==");  
else  
    System.out.println("!=");
```

OUTPUT

!=

== compares the String references which are the memory addresses of the actual String objects.

© A+ Computer Science - www.apluscompsci.com

one is a reference that refers to a String Object.

two is a reference that refers to a String Object.

one starts out referring to a String Object `compsci`.

two starts out referring to a different String Object `compsci`.

`one==two` compares the locations/addresses stored in one and two.

one and two do not store the same location/address.

Open
toUpperCase.java
StringRef.java

© A+ Computer Science - www.apluscompsci.com

Comparing Objects

Object references can be compared with ==.

The actual object contents can be compared using equals() or compareTo()

© A+ Computer Science - www.apluscompsci.com

== can be used to compare the memory addresses of objects.
.equals and .compareTo can be used to compare the actual object contents.

Comparing Objects

```
Integer one = 90;  
Integer two = 75;
```

```
out.println(one.compareTo(two));  
out.println(two.compareTo(one));
```

```
two = 90;  
out.println(two.equals(one));  
out.println(two.compareTo(one));
```

OUTPUT

```
-1  
1  
true  
0
```

compareTo() returns a negative value when A is less than B and a positive value when A is greater than B.
0 is returned with the A and B are the same.

© A+ Computer Science - www.apluscompsci.com

The Integer compareTo() method compares the contents of the 2 Integer objects.

If a is greater than b, then a.compareTo(b) will return a positive value.

If a is less than b, then a.compareTo(b) will return a negative value.

If a is equal to b, then a.compareTo(b) will return 0.

Comparing Strings

String references can be compared with `==`.

The actual String contents can be compared using `equals()` or `compareTo()`

© A+ Computer Science - www.apluscompsci.com

`==` can be used to compare the memory addresses of objects.
`.equals` and `.compareTo` can be used to compare the actual object contents.

String frequently used methods	
Name	Use
<code>equals(s)</code>	checks if this string has same chars as s
<code>compareTo(s)</code>	compares this string and s for >, <, and ==
<code>trim()</code>	removes leading and trailing whitespace
<code>replaceAll(x,y)</code>	returns a new String with all x changed to y
<code>toUpperCase()</code>	returns a new String with uppercase chars
<code>toLowerCase()</code>	returns a new String with lowercase chars

© A+ Computer Science - www.apluscompsci.com

The chart above lists some very common and very useful String class methods.

`equals()` and `compareTo()` are used quite often.

`trim()` and `replaceAll()` are very useful, but that widely used.

`toUpperCase()` and `toLowerCase()` can be very useful in certain situations.

equals

```
String one = new String("compsci");  
String two = new String("compsci");
```

```
if(one.equals(two))  
    System.out.println("equal");  
else  
    System.out.println("!equal");
```

OUTPUT

equal

equals() compares the values stored in the actual String objects.

© A+ Computer Science - www.apluscompsci.com

one is a reference that refers to a String Object.

two is a reference that refers to a String Object.

one starts out referring to a String Object `compsci`.

two starts out referring to a different String Object `compsci`.

`one.equals(two)` compares the contents of the String Objects referred to by `one` and `two`.

`one` and `two` both refer to String Objects `compsci`.

`one.equals(two)` is true.

compareTo

```
String one = "region";  
String two = "uilstate";
```

```
out.println(one.compareTo(two));  
out.println(two.compareTo(one));
```

```
two = "region";  
out.println(two.compareTo(one));
```

OUTPUT

```
-3  
3  
0
```

compareTo() returns the difference in ASCII value when comparing Strings.

© A+ Computer Science - www.apluscompsci.com

The String `compareTo()` method compares the letters stored in two String Objects.

The difference in ASCII of the first two letters that do not match is returned.

**Open
equals.java
compareto.java**

© A+ Computer Science · www.apluscompsci.com

trim

```
String s = "  100  ";  
String trimmed = s.trim();  
out.println(trimmed);
```

OUTPUT

```
100  
900
```

```
out.println(Integer.parseInt(trimmed)*9);
```

trim() returns a new String with all leading and trailing white space removed.

© A+ Computer Science - www.apluscompsci.com

The `trim()` method is useful to remove leading and trailing spaces.

toUpperCase() toLowerCase()

```
String s = "compsci";  
out.println(s.toUpperCase());  
out.println(s);  
out.println(s.toLowerCase());
```

OUTPUT
COMPSCI
compsci
compsci

toUpperCase() and toLowerCase() return new Strings with the changes requested.

© A+ Computer Science - www.apluscompsci.com

toLowerCase() and toUpperCase() both return new String Objects. The new String Object contains the same letters as the original String in all uppercase or all lowercase.

toLowerCase() and toUpperCase() do not change the original String Object. Both return a new String with the changes requested.

replaceAll

```
String s = "abcdef1xyzabf1";  
s = s.replaceAll("1", "#");  
out.println( s );
```

OUTPUT
abcdef#xyzabf#

replaceAll() returns a new String with all number 1s changed to # signs.

© A+ Computer Science - www.apluscompsci.com

replaceAll () returns a new String with the changes requested.
replaceAll () does not change the original String.

replaceAll () returns a new String with the specified letters replaced with the provided letters.

Open
replaceall.java
touppercase.java

© A+ Computer Science - www.apluscompsci.com

Open
trim.java
stringtonums.java

© A+ Computer Science - www.apluscompsci.com

Continue work on the labs

© A+ Computer Science - www.apluscompsci.com