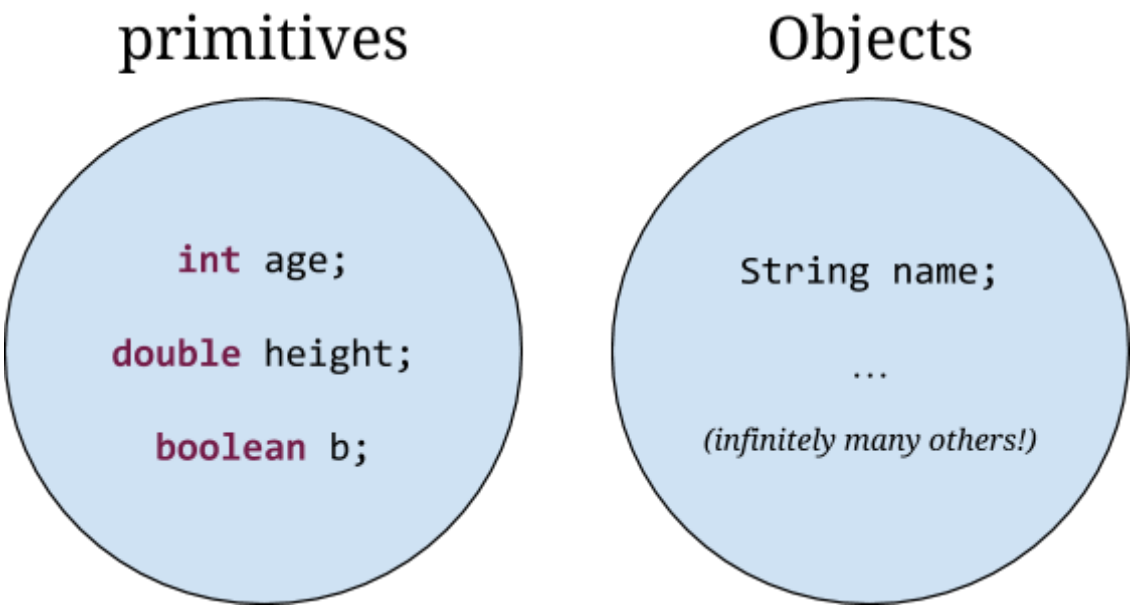
Primitives, Objects, & String Processing
Solution

We have used int, double, boolean, and String types in class so far.

It's time to reveal a new fact about Java types: they all belong one of two fundamental categories of data types, primitives and Objects:



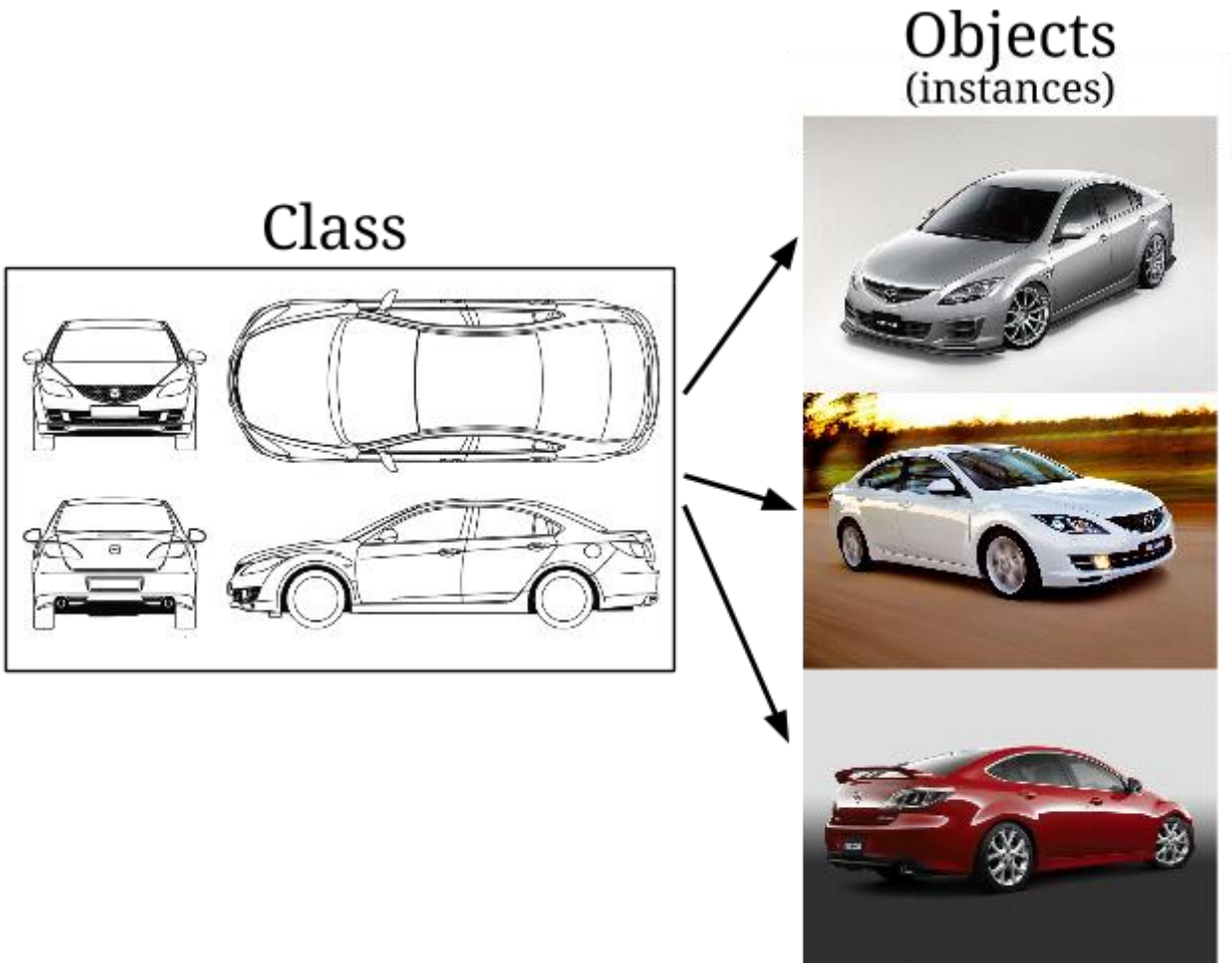
Primitives are like "atoms" from chemistry/physics - they are the simplest types that programmers use. They are composed of no other data types, and cannot be broken down any further. Objects are like molecules from chemistry: they are built up from primitives. They can become extremely complicated.

Formally, an Object is a programming entity that contains data and behavior (methods). In a moment we'll see some examples of what this means. And we will see many, many examples of Objects throughout the rest of the course -- they are an extremely important concept in Computer Science!

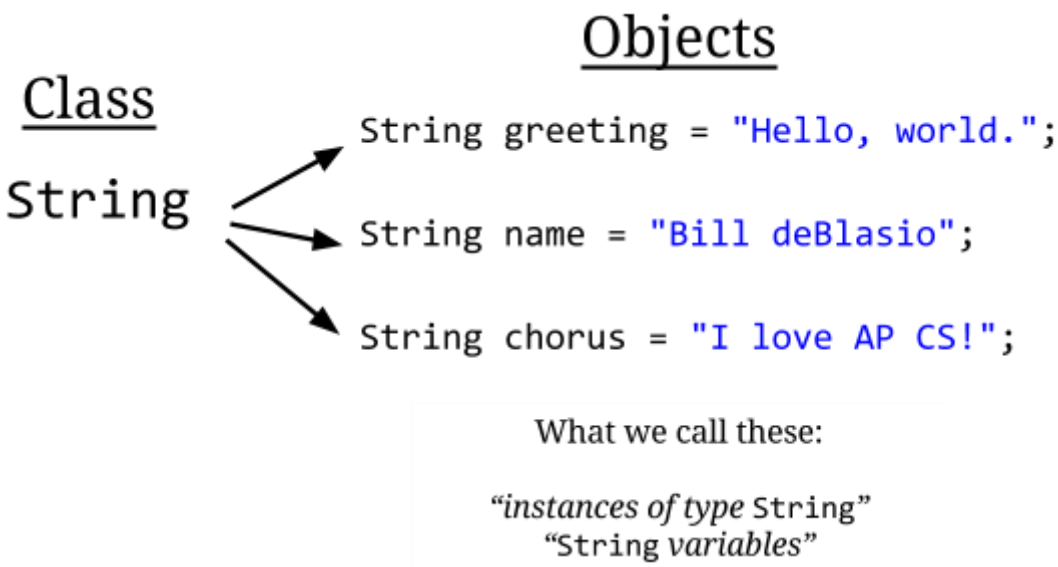
We need another concept to talk about Objects - the idea of a **Class**. You can think of a Class as a "blueprint" for making Objects: it gives the instructions for how to make an Object, what information the Object should contain, and what behavior it should have. We call an individual Object built from the Class blueprint an instance of that class (this is an important term, so remember it!)

Think of a Class as a blueprint for a car, and the actual cars that are built from that blueprint as the

Objects (or instances):



.



Object Methods

Objects (and sometimes Classes, too) often have methods available on them that you can call.

What if we wanted to find out the length of a String? It turns out there's a way to do that, by calling a method of an Object:

```
String s = "Hasta mañana";
s.length();
```

this will return the *length* of the String s, which is 12

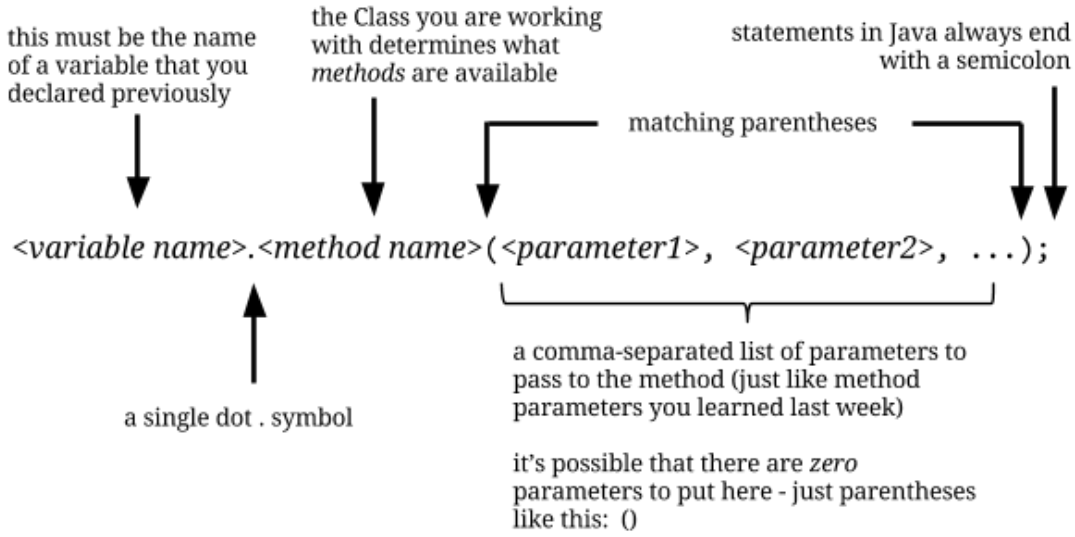
Of course, we need to do something with the return value of the method for this to be useful:

```
String s = "Hasta mañana";
System.out.println("The length of s is " +
s.length());
```

Output:

The length of s is 12

The Java syntax for method calls on objects looks like this:



There's another method for Strings that we can use, called charAt(), which picks out an individual character from the String:

```
String s = "Hasta mañana";
System.out.println(
     "The first character of s is " + s.charAt(0));
System.out.println(
     "The last character of s is " + s.charAt(11));
```

```
The first character of s is H
The last character of s is a
```

What's going on here? To understand what charAt does, we need to talk about indexing.

Let's take a look at how Java stores this String in memory:

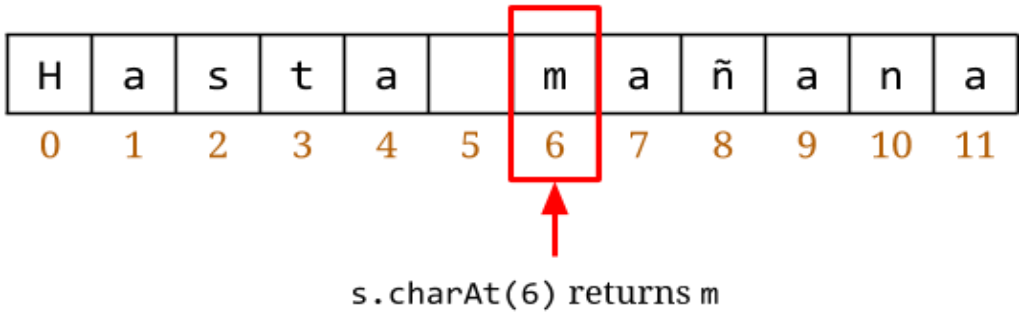
```
String s = "Hasta mañana";
```

H	a	s	t	a		m	a	ñ	a	n	a
0	1	2	3	4	5	6	7	8	9	10	11

Java keeps track of the individual characters in the String by counting starting with the integer zero. (This is very common tradition in Computer Science! Your book has more details. This is why we started numbering your Tests with 0!)

The charAt method returns the character at the index that you supply as the argument:

```
String s = "Hasta mañana";
```



Exercise 1

Write down the output of the following code examples. Assume that these are contained in a valid Java file, inside the body of a valid public static void main() method.

Problem 1a

```
String s = "Hasta mañana";
for (int i = 0; i < s.length(); i++) {
    System.out.println(i + ": " + s.charAt(i));
}
```

Solution

```
6: m
7: a
8: ñ
9: a
10: n
11: a
```

Problem 1b

```
String s = "Hasta mañana";
for (int i = 0; i < s.length(); i += 2) {
    System.out.print(s.charAt(i));
}
```

Solution

```
Hsamñn
```

Problem 1c

```
String s = "Hasta mañana";
for (int i = (s.length() - 1); i >= 0; i--) {
     System.out.print(s.charAt(i));
}
```

Solution

```
anañam atsaH
```

Problem 1d

```
String s = "Hasta mañana";
for (int i = 0; i < s.length(); i++) {
     System.out.print(s.charAt(i % 2));
}
```

Solution

```
HaHaHaHaHaHa
```

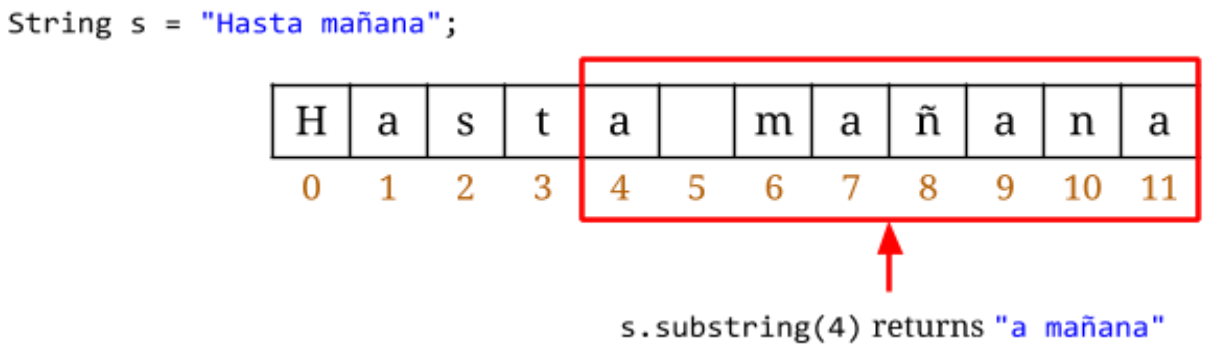
Other Important String Methods

substring

There are a bunch of other useful methods that we can call on Strings. A method call:

```
s.substring(from);
```

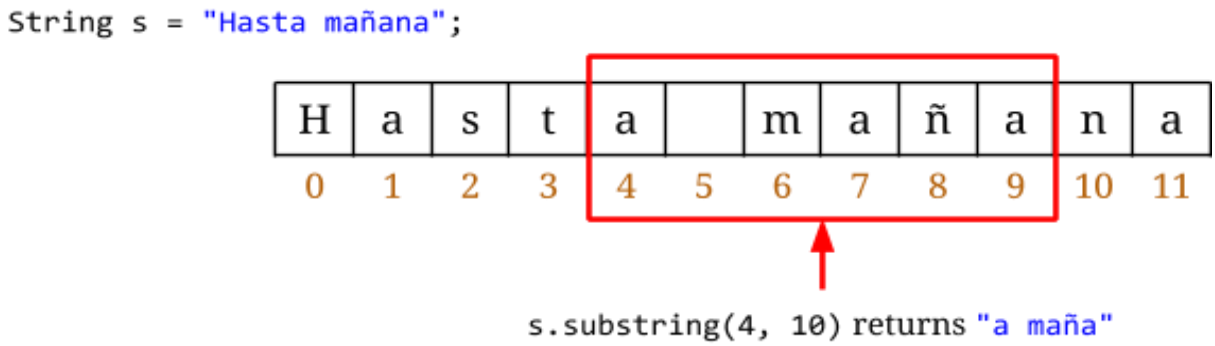
returns a new String consisting of the characters starting at the index 'from' and going to the end of the String:



A similar method for Strings lets you pick out a shorter substring:

```
s.substring(from, to);
```

returns a new String consisting of the characters starting at the index 'from' and going to one less than the index 'to':



Exercise 2

Write down the output of the following code examples. Assume that these are contained in a valid Java file, inside the body of a valid public static void main() method.

Problem 2a

```
String s = "Hello, world";
String t = s.substring(7);
System.out.println(t);
```

Solution

```
world
```

Problem 2b

```
String s = "Hello, world";
for (int i = 0; i < s.length(); i += 2) {
    String t = s.substring(i);
    System.out.println(t);
}
```

Solution

```
Hello, world
llo, world
o, world
 world
orld
ld
```

Problem 2c

```
String s = "Hello, world";
for (int i = (s.length() - 1); i >= 0; i -= 2) {
    System.out.println(s.substring(i));
}
```

Solution

```
d
rld
world
, world
lo, world
ello, world
```

Problem 2d

```
String s = "Hello, world";
```

```
for (int i = 0; i < s.length() - 1; i++) {
     System.out.println(s.substring(i, i+2));
}
```

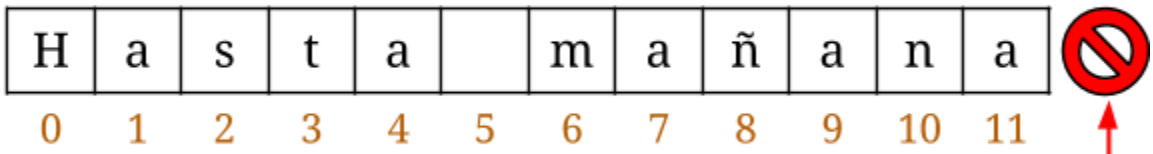
Solution

```
He
el
ll
lo
o,
,
 w
wo
or
rl
ld
```

Method calls that cause errors

It's possible to make a method call with invalid parameters:

```
String s = "Hasta mañana";
```

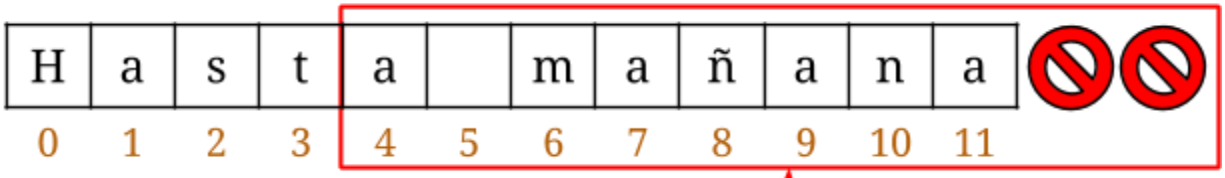



```
s.charAt(12)
```

results in:

```
java.lang.StringIndexOutOfBoundsException:
String index out of range: 12
```

```
String s = "Hasta mañana";
```




```
s.substring(4, 13)
```

results in:

```
java.lang.StringIndexOutOfBoundsException:
String index out of range: 13
```

If you see the StringIndexOutOfBoundsException message when you run your program,

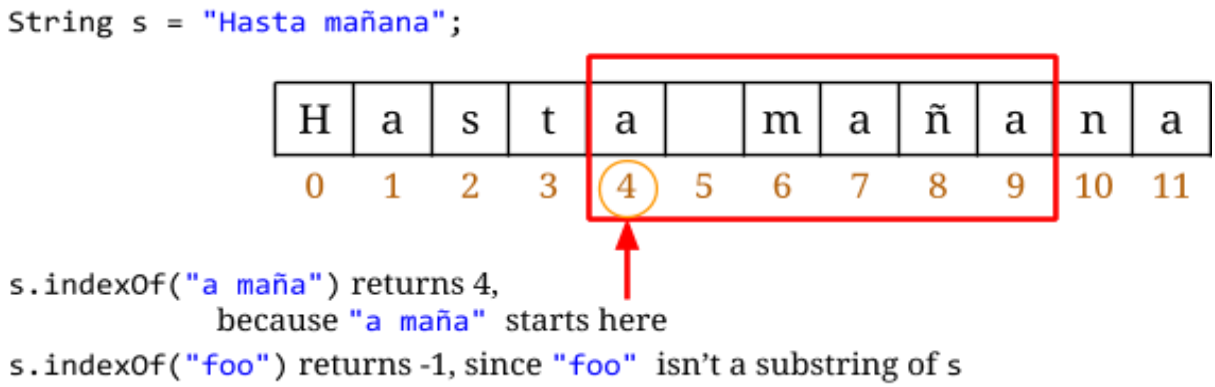
you know that you are trying to access an invalid location in the String.

indexOf

There is a method called indexOf that returns the index where a substring occurs, or -1 if that substring does not occur:

```
s.indexOf(substring);
```

For example:



Exercise 3

Write down the output of the following code examples. Assume that these are contained in a valid Java file, inside the body of a valid public static void main() method.

Problem 3a

```
String s = "Hello, world";
int index = s.indexOf("world");
System.out.println("world starts at index " + index);
```

Solution

```
world starts at index 7
```

Problem 3b

```
String s = "Hello, world";
String t = "world";
int index = s.indexOf(t);
System.out.println(t + " starts at index " + index);
```

Solution

```
world starts at index 7
```

Problem 3c

```
String s = "Stand clear of the closing ";
String t = "doors";
String altogether = s + t;
int index = altogether.indexOf(t);
System.out.println(t + " starts at index " + index);
```

Solution

```
doors starts at index 27
```

Problem 3d

```
String s = "Stand clear of the closing ";
String t = "doors";
int index = s.indexOf(t);
System.out.println("I got index: " + index);
```

Solution

```
I got index: -1
```

Problem 3e

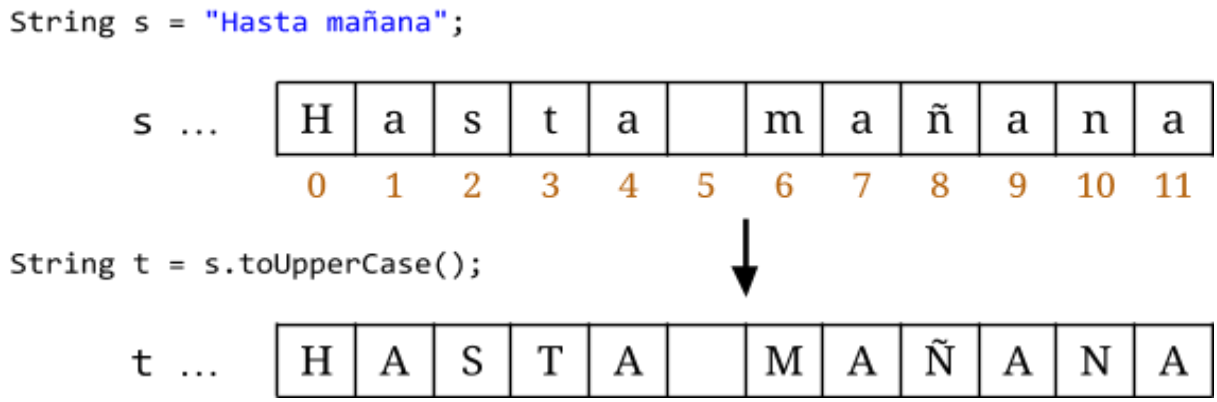
```
String s = "Stand clear of the closing doors";
int index = s.indexOf("doors");
System.out.println(s.charAt(index));
System.out.println(s.charAt(index + 1));
System.out.println(s.charAt(index + 2));
System.out.println(s.charAt(index + 3));
```

```
System.out.println(s.charAt(index + 4));
```

Solution

d
o
o
r
s

toUpperCase/toLowerCase



equals

A method called equals lets us know if one string is equal to another:

```
oneString.equals(anotherString);
```

This returns a boolean (true or false) that tells us whether or not the two strings are the same.

Exercise 4

Write down the output of the following code examples (it will be either true or false). Assume that these are contained in a valid Java file, inside the body of a valid public static void main() method.

Problem 4a

```
String s = "hi there";
String t = "HI THERE";
System.out.println(s.equals(t));
```

Solution

```
false
```

Problem 4b

```
String s = "hi there";
String t = "HI THERE";
System.out.println(s.toUpperCase().equals(t));
```

Solution

```
true
```

Problem 4c

```
String s = "hi there";
String t = "HI THERE";
System.out.println(s.equals(t.toLowerCase()));
```

Solution

```
true
```

Problem 4d

```
String s = "hi there";
String t = "HI THERE";
System.out.println(t.equals(s.toLowerCase()));
```

Solution

```
false
```