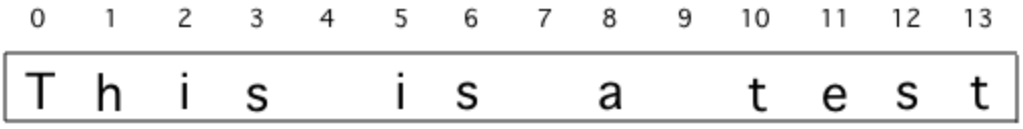
Strings

Strings are a sequence of characters, surrounded by quotations.

Strings are made up of characters, or chars. Each char is at a certain position in the String, also known as its **index**. For example, the String below has 14 chars in it, and these are the index numbers:



Notice that each char in the String has an index number, even the spaces! A space is a char as well! Indices always start at 0 and end at length -1

String Declaration

Like normal variables, we can assign String values and declare them in the same statement.

But if we don't know what a String is when we create the variable, we can also set it to the keyword `null`, which keeps the variable empty. For example:

```
public class Main {
   public static void main(String[] args) {
      String myStr = "This is a string";
      System.out.println(myStr);

      String greeting = null;
      // null doesn't need quotes because it's a keyword in Java

      System.out.println(greeting);
   }
}
```

```
This is a string
null
```

String Concatenation

Strings can be added together, or appended to each other, using + or +=, just like in math. We call this "**String concatenation**".

```
public class Main {
   public static void main(String[] args) {
      String start = "meep morp";
      String end = "robot captain engage";

      // Add 3 Strings together
      String result = start + " " + end;

      // Add onto the same String
      result += "!";

      System.out.println(result);
   }
}
```

```
meep morp robot captain engage!
```

Note that spaces are not added between strings automatically. If you want a space between two strings then add one using + " " +, or else your words will be smushed together.

```
public class Main {
   public static void main(String[] args) {
      String s1 = "xy";
      String s2 = s1;

      s1 = s1 + s2 + "z";

      System.out.println(s1);
   }
}
```

```
xyxyz
```

You can add int values to Strings. Java will automatically cast your int values to Strings if you do this. This is because int ➜ String is a widening cast (and we learned previously that his happens automatically).

```
public class Main {
   public static void main(String[] args) {
      String message1 = "12" + 3 + 4;
      String message2 = 4 + 3 + "21";

      System.out.println(message1);
      System.out.println(message2);
   }
}
```

```
1234
721
```

The order in which types are given will determine if addition or concatenation will happen.

String Methods

Because Strings are Objects, they have lots of useful methods to help you do things! As always, there are more - but here are a few to get you started!

You will need an existing String Object to call all of these methods (see example below).

`length()`

Returns the number of characters in the String.

`substring(start)`

- **`start`**: The index which the substring should begin from (inclusive)

Returns up until the end of the String.

`substring(start, end)`

- **`start`**: The index which the substring should begin from (inclusive)
- **`end:`** The index which the substring will end at (exclusive)

`chartAt(index)`

- **`index`**: The index of the desired char

`indexOf(find)`

- **`find`**: The String that will be searched for

Returns the index of the beginning of **`find`** in the current String, or returns -1 if **`find`** isn't found.

`compareTo(other)`

- **`other`**: Another String to compare the current String to

Returns a negative value if the current String is less than **`other`** alphabetically, 0 if the two Strings have the same characters in the same order, or a positive value if the current String is greater than **`other`** alphabetically.

`equals(other)`

- **`other`**: Another String to compare the current String to

Returns true if the current String and **`other`** have the same characters in the same order, and false otherwise.

```
public class Main {
   public static void main(String[] args) {
      String myStr = "blueberry";
      System.out.println(myStr.length());
      System.out.println(myStr.substring(4));
      System.out.println(myStr.substring(3,6));
      System.out.println(myStr.charAt(0));
      System.out.println(myStr.indexOf("berry"));
      System.out.println(myStr.indexOf("apple"));
      System.out.println(myStr.compareTo("apple"));
      System.out.println(myStr.compareTo("cantaloupe"));
      System.out.println(myStr.compareTo("blueberry"));
      System.out.println(myStr.equals("blueberry"));
      System.out.println(myStr.equals("BLUEBERRY"));
   }
}
```

```
9
berry
ebe
b
4
-1
1
-1
0
true
false
```