

# Class String



---

---

---

---

---

---

---

---

## Before starting classes

- Let's have a look at a particular class that we have already used.
- Remember?  
String s = "I am a string."  
System.out.println("Hello! " + s);  
String m = s;



---

---

---

---

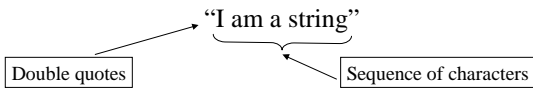
---

---

---

---

## A value



- A string is built with characters
- Similar to an array of characters  
**BUT DIFFERENT**



---

---

---

---

---

---

---

---

## How to declare a String?

- `String str = "abc";`
- `char[] data = {'a', 'b', 'c'};`  
`String str = new String(data);`
- `String str = new String();`
  - Empty sequence of characters



---

---

---

---

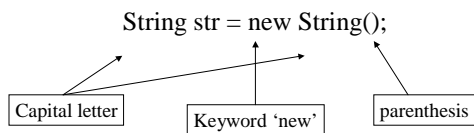
---

---

---

---

## Notice: String is a class



`str` is an object, which is an instance of the class `String`  
`str` can use all the methods of class `String`



---

---

---

---

---

---

---

---

## Length

`int length()`  
Type of the return value

- It's equal to the number of characters of the string

### Examples:

```
String str = new String();  
int a = str.length(); // a == 0  
String str = "abc";  
int b = str.length(); // b == 3
```



---

---

---

---

---

---

---

---


### Access: charAt

---

- Because it's array-like, characters can directly be accessed with an index.
- For this *charAt* method can be used:

*char charAt (int index)*

Type of the return value →      ← Type of the parameter



Department of Computer Science

---

---

---

---

---

---


---

---

### charAt

---

- Example:  
String str = "abc";  
char myChar = str.charAt(2); // myChar = 'c'
- Index is in the range [0 .. str.length()-1]



Department of Computer Science

---

---

---

---

---

---

---

---


### Access: substring

---

*String substring(int beginIndex) ;*  
*String substring(int beginIndex, int endIndex) ;*

- Use as:

```
String str = "I am a string" ;  
String sst1 = str.substring(0,4);  
String sst2 = str.substring(5,str.length()-1);  
//sst1 = "I am " and sst2 = "a string"
```



Department of Computer Science

---

---

---

---

---

---

---

---

## Comparing

- Method for comparing Strings
  - compareTo
  - equals
  - equalsIgnoreCase
  - *but not* ==
- Comparing Strings is more complicated than it might seem!



---

---

---

---

---

---

---

---

## compareTo

```
String str = ... ;  
int r = str.compareTo("Hello") ;
```

- Compares according to ordering of *characters*.
- Returns:
  - 0 if strings have exactly the same characters.
  - <0 if *str* precedes "Hello".
  - >0 if *str* follows "Hello".



---

---

---

---

---

---

---

---

## Example

```
String s = "the" ;  
System.out.println(s.compareTo("she")) ; // 1  
System.out.println(s.compareTo("the")) ; // 0  
System.out.println(s.compareTo("there")) ; // -2  
System.out.println(s.compareTo("The")) ; // 32  
System.out.println(s.compareTo("123")) ; // 67
```



---

---

---

---

---

---

---

---

## equals

- Returns true if two Strings hold exactly the same characters.
- Equivalent to:  
`(str.compareTo(s) == 0)`



---

---

---

---

---

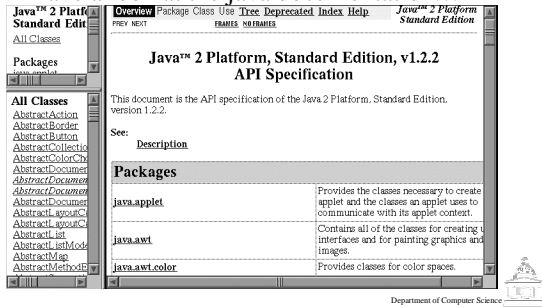
---

---

---

## More?

- Have a look at the java documentation...



---

---

---

---

---

---

---

---