Strings

Ways to create Strings

"String literal"

Concat + enation

- The `+` operator is overloaded for strings

StringBuilder()

- Slightly more efficient than concatenation when looping

new String({'H', 'e', 'l', 'l', 'o'})

Constructor can take:

- `String`
- `StringBuilder`
- `StringBuffer`
- `char[]`
- `byte[]`

Strings are immutable

- Once created, they cannot be changed
- Methods that transform a string actually return a new string (unless nothing was changed)

Strings from Objects

---

toString method

- `Object` has a `toString()` method
- used to generate a string representation of the object
- automatically called in concatenation

Overriding toString

- Almost every object should override `toString()`
- `toString` should produce a reasonable text representation of the object
- Default `Object.toString()` prints the run time class name, `@` symbol, and the hexadecimal hash of the object

---

toString infinite recursion

When overriding `toString()` if you want to print the result of the original `toString()` method, concatenating `this` or `this.toString()` will cause an infinite loop. Instead, use `super.toString()`

String operations

- `+` - String concatenation
- `length()`
- `charAt()`
- `toCharArray()`
- `equals()`, `equalsIgnoreCase()`
- `compareTo()`

- `contains()`, `startsWith()`, `endsWith()`
- `indexOf()`, `lastIndexOf()`
- `toLowerCase()`, `toUpperCase()`
- `replace()`, `replaceAll()`, `replaceFirst()`
- `split()`, `trim()`, `join()`

Formatting Strings

Ways to format

- Formatter()
- String.format()
- System.out.format()

Format parameters

- first argument: The format string with format specifiers
- all other arguments: values to fill into the format string

Format specifier syntax

`%[argument_index$][flags][width][.precision]conversion`

Conversion Characters	
d	Integral (as decimal)
c	Unicode character
b	Boolean value
s	String
f	Floating point (as decimal)
e	Floating point (in scientific notation)
x	Integral (as hex)
h	Hash code (as hex)
%	Literal "%"

Pitfalls

Some specifiers may behave in unexpected ways. eg: `%b` will produce "true" for non-null values of any other data type

---

Useful Resources

- Apache Commons StringUtils

- Java Strings tutorial

A peek at Regular Expressions

Basic symbols

- `a` , `b` , `c` - match "a", "b", "c" respectively (all numbers & letters)
- `.` - matches any one character
- `*` - match 0 or more occurrences of the last symbol
- `+` - match 1 or more occurrences of the last symbol
- `?` - match 0 or 1 occurrences of the last symbol

Examples

- `S117` - matches only the string "S117"
- `dogs?` - matches the strings "dog" or "dogs"
- `fuzzy*` - matches the strings "fuzz", "fuzzy", "fuzzyyyyy" etc.
- `wh?ee+!` - matches "whee!", "weee!", "wheeeee!" and so on

Regex sites

- regexr - test regular expressions