Lets Code!

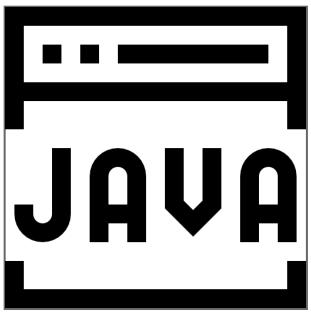
Fundamentals: Strings

Instructor: Tariq Hook
You can find me on github @code-rhino

Key Terms

- Strings Methods
- Concatenation
- String Equality
- Operations
- String Formatting
- Unicode

Strings



What is a String?

```
String words = "apple is red"
```

- An object
- An array of characters
    - It can be a character, word, phrase, or paragraphs
- Starts and ends with a double quote
- Immutable (cannot be changed)
    - Any changes will create a new string

Examples

```
String empty = new String();
String empty = "";

String oneCharacter = "a";

String word = "apple";

String phrase = "I'm sorry for what \"I
said\" when I was hungry!";

String paragraph = "It was the best of
times, it was the worst of times, it was
the age of wisdom, it was the age of
foolishness, it was the epoch of belief,
it was the epoch of incredulity, it was
the season of Light, it was the season of
Darkness, it was the spring of hope, it
was the winter of despair, we had
everything before us, we had nothing
before us, we were all going direct to
Heaven, we were all going direct the other
way – in short, the period was so far like
the present period, that some of its
noisiest authorities insisted on its being
received, for good or for evil, in the
superlative degree of comparison only.";
```

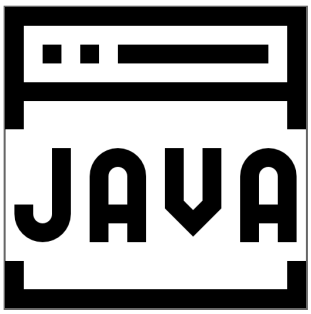
Converting other data type to strings

```
String number = Integer.toString(55); //
"55"
String number2 = String.valueOf(55); //
"55"

String decimal = Double.toString(3.14); //
"3.14"
String decimal2 = String.valueOf(3.14); //
"3.14"

String character =
Character.toString('c'); // "c"
String character2 = String.valueOf('c');
// "c"
```

Strings Methods



String Methods

.length()

```
String fruit = "apple is red";
fruit.length(); // 12
```

|a|p|p|l|e| |i|s| |r|e|d|

String Methods

.length()

```
String empty = "";
empty.length(); // 0

String space = " ";
space.length(); // 1
```

String Methods

.charAt(int index)

```
String fruit = "apple is red";
```

```
char firstCharacter =  fruit.charAt(0); //
'a'
char sixthCharacter = fruit.charAt(5); //
' '
char lastCharacter =
fruit.charAt(fruit.length() - 1); // 'd'
char secondToLastChar =
fruit.charAt(fruit.length() - 2); // 'e'
```

String Methods

.substring(int beginIndex)

```
String fruit = "apple is red";
```

```
String color = fruit.substring(9);  //
"red"
```

String Methods

.substring(int beginIndex, int endIndex)

```
String fruit = "apple is red";
```

```
String noun = fruit.substring(0, 5); //
"apple"
String verb = fruit.substring(6, 8); //
"is"
String adjective = fruit.substring(9,
fruit.length()); // "red"
```

String Methods

.trim()

```
String words = "  apple is   ";
String trimmedWord = words.trim(); //
"apple is"
```

String Methods

.compareTo(String otherString)

```
"b".compareTo("a"); // 1
"b".compareTo("b"); // 0
"b".compareTo("c"); // -1

"10".compareTo("2"); // -1
```

String concatenation

Combining strings

```
String name = "Moana";
String greeting = "Hello, " + name + "!";
// "Hello, Moana!"

String greeting2 = "Hello,
".concat(name).concat("!");

String greeting3 = String.join("
","Hello,", name, "!");
```

String concatenation

String and number/decimal

```
double pi = 3.14;
String message = "Pi is " + pi; // "Pi is
3.14"
String message2 = pi + " is pi"; // "3.14
is pi"
```

Equality

```
String fruit1 = "apple";
String fruit2 = "apple";
String fruit3 = "Apple";

fruit1.equals(fruit2);

fruit1.equals(fruit3);
fruit1.equalsIgnoreCase(fruit3);
```

Equality

Don't use == to test for equality

```
String num = "3";
String num2 = Integer.toString(3);

num == num2;

num.equals(num2);
```

Empty vs Null

```
String name = null;
System.out.println("My name is " + name);
// "My name is null"
name.length(); // ERROR!!!!!


String name2 = "";
System.out.println("My name is " + name2);
// "My name is "
name2.length(); // 0
```

Checking if there is a value

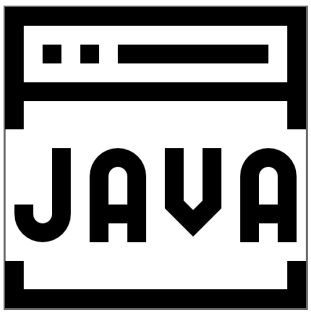
```
public static boolean hasValue(String
value) {
    if (value != null &&
value.trim().length() > 0) {
        return true;
    } else {
        return false;
    }
}

System.out.println(hasValue(null));
//false
System.out.println(hasValue("")); //false
System.out.println(hasValue("   "));
//false
System.out.println(hasValue("a")); //true
```

String operations

- `+` - String concatenation
- `length()`
- `charAt()`
- `toCharArray()`
- `equals()`, `equalsIgnoreCase()`
- `compareTo()`
- `contains()`, `startsWith()`, `endsWith()`
- `indexOf()`, `lastIndexOf()`
- `toLowerCase()`, `toUpperCase()`
- `replace()`, `replaceAll()`, `replaceFirst()`
- `split()`, `trim()`, `join()`

Strings Formatting



String Formatting

```
double total = 10.00 - 9.33;
System.out.println(total);
//0.6699999999999999
```

String Formatting

```
double total = 10.00 - 9.33;

String formattedTotal =
String.format("Total is %,15.2f", total);
System.out.println(formattedTotal);
//Total is            0.67
```

String Formatting

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

The only field that is required is conversion

- **`argument_index`** - an integer indicating the position of the argument. The first argument is referenced by "1$", the second by "2$", etc.
- **`flags`** - modifies output like left justify, 0 padded
- **`width`** - the minimum number of characters
- **`conversion`** - indicates how the argument should be formatted

String Formatting

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

Specifier	Applies to	Output
%b	Any type	`true` if non-null, `false` if null
%c	character	character
%s	any type	String value

String Formatting

```
"%[argument number] [flags][width].
[precision][type]"
```

Specifier	Applies to	Output
%d	integer (incl. byte, short, int, long, bigint)	Decimal Integer
%e	floating point	decimal number in scientific notation
%f	floating point	decimal number

String Formatting

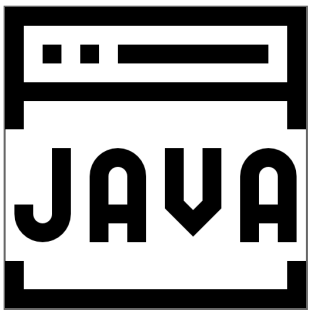
```
String format = "Total is %,15.2f"
```

Pads with space, group with comma, with minimum with of 15 and precision of 2. Conversion type is float.

Ways to format

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

Codepoint



Codepoint

- Once upon a time, there was ASCII. It assigns codes between 0 and 127 to all English letters, the decimal digits, and many symbols.
- But what about Russian, Japanese, and Chinese characters?
- Unicode to the rescue!

Codepoint

- It assigns each character in all of the writing systems ever devised a unique 16-bit code between 0 and 65535.
- When you see this it will usually be in hexidecimal from 0x0000 to 0xFFFF but Java supports encoding up to 0x10FFFF

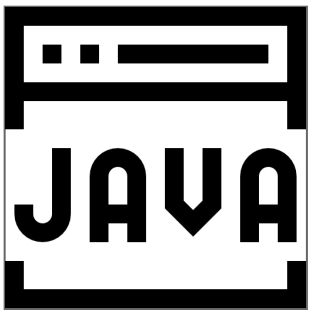
Codepoint

Unicode in Java

```
String symbol = "\ud835\udd46"; // 𝕆
symbol.length(); // 2
symbol.codePointCount(0, symbol.length());
// 1
```

List of Unicode character

Building Strings



Building Strings

Using the StringBuilder class allows you to build a string from many small pieces.

```
StringBuilder builder = new
StringBuilder("ap");

builder.append('p'); // appends a single
character
builder.append("le"); // appends a string

String completedString =
builder.toString(); // "apple"
```

StringBuilder API

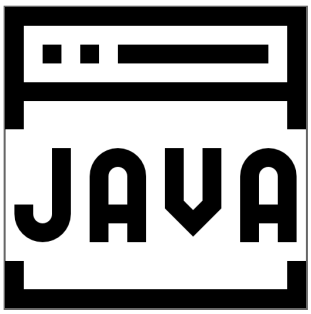
Building Strings

- Use + to concatenate a small set of string.
- Use StringBuilder for large or unknown number of string
    - Ex: for loop, or reading a file

Wrap Up

- Strings Methods
- Concatenation
- String Equality
- Operations
- String Formatting
- Unicode

Keep Coding !!!



Clean Code is Happy Code