Notes: Parameters, Return Types, and Objects

Key Words

- **scope**: the part of the program where a variable exists
- A variable declared in a for loop exists only in that loop
- A variable declared in a method exists only in that method
- **parameter**: sends information **in** from the caller of a method
- **return**: sends information **out** from a method to its caller

parameters

- Parameters are used to bring information into a method
- Parameters (formal parameters) must have the data type in front of a variable name
- Arguments (actual parameters) are what we call the information actually passed to the method
- You can have methods with the same exact name, as long as the number or order of types of the parameters is different. This is called **method overloading**.

```
public static void main(String[] args) {
   // calls the method "food" with the arguments "eggs", "salad", and "steak"
   food("eggs","salad","steak");
}

// Defines the method "food" to take three parameters
public static void food(String breakfast, String lunch, String dinner) {
   System.out.println("I had " + breakfast + " for breakfast");
   System.out.println("I had " + lunch + " for lunch");
   System.out.println("I had " + dinner + " for dinner");
}
```

return statements

- A return statement sends information **out** from a method to its caller
- There cannot be statements executed after a return statement in a method
- The return type should be provided as the third keyword in your method header
- The type of the information returned from a method must match the return type, unless the return type is `void` which means that nothing is returned from the method.

```
public static void main(String[] args) {
   hello();   // calls a void method
   three();  // calls method three() but does nothing with returned value
   int val = three(); // calls three() and saves the returned value in variab
le 'val'
   System.out.println("method three() returned: " + val); // displays the val
ue returned by three()
}

// This method has a return type of void
public static void hello() {
   System.out.println("Hello");
}

// This method has a return type of int
public static int three() {
   return 3;
}
```

Math class

- The **Math** class has a number of useful methods; check out more in the Java API
- Many of these methods return double. If you want an int, you will use casting

Math methods

Methods	Description
`Math.abs`	returns the absolute value of a number
`Math.sqrt`	returns the square root of a number
`Math.pow(base,exp)`	returns base raised to the exp
`Math.max(x,y)`	returns the larger of x and y
`Math.min(x,y)`	returns the smaller of x and y
`Math.ceil(x)`	returns x rounded up to the nearest whole number
`Math.floor(x)`	returns x rounded down to the nearest whole number
`Math.round(x)`	returns x rounded in the appropriate direction (up for 0.5 and above), down otherwise

String class

String methods

Methods	Description
`str.length()`	returns the number of characters in str
`str.charAt(index)`	returns the character at index
`str.indexOf(str2)`	returns the index of the first occurrence of str2, -1 if str2 is not present
`str.substring(start, stop)`	returns a string of the characters from start (inclusive) to stop (exclusive)
`str.toUpperCase()`	returns str in all uppercase
`str1.equals(str2)`	tests whether str1 contains the same characters as str2

Methods	Description
`str1.equalsIgnoreCase(str2)`	tests whether str1 contains the same characters as str2, ignoring case
`str1.startsWith(str2)`	tesets whether str1 starts with the characters in str2
`str1.endsWith(str2)`	tests whether str1 ends with the characters in str2
`str1.contains(str2)`	tests whether str2 is found inside of str1

String are Objects

- Strings in Java are objects
- Strings can contain the same characters but not be equal because in Java they are stored as different objects (even though they have the same characters)
- Because of this, you should use `.equals()` when comparing Strings and not `==`

```
// word1 and word2 are different objects
String word1 = "hello";
String word2 = "hello";

// do not use!
if (word1 == word2) {
    ...
}

// use this instead!
if (word1.equals(word2)) {
    ...
}

// you could also have use this; it does the same as the one directly above
if (word2.equals(word1)) {
    ...
}
```