Functions/Methods!!!

we've already been using lots of functions:

- `System.out.println()`
- `scanner.nextLine()`
- `scanner.nextInt()`

Examples:

System → class
out → object (instance of a class) (inside System class)
println() → method (part of 'out' object)

scanner → object (instance of a class)
nextLine() → method (part of scanner object)
nextInt() → method (part of scanner object)
Functions/Methods!!!

Function – series of stored statements to execute when called.

Method – a function that specified in a class/object

- All methods are functions.
Functions!!

• Why functions? Functions provide good way to design systems (modularly)!
• And we can define our own functions...
/**
 * @param number
 * @return true if the <code>number</code> is even;
 * otherwise false
 */

private static boolean isEven(int number) {
    return (number % 2 == 0);
}
Coding Design!

- DRY principle - Don't Repeat Yourself!
  - Loops help with this too
  - But functions (procedures/methods) help even more!
- Modular design
  - Functions/procedures/modules help with this!
Functions!!!

• comprised of:
  • name – use names that describe what the function does
  • parameters (optional) - variables passed to the function
  • code body - statements that run on given parameters
  • return value (optional) - result returned by the function
Variable Scope

- Scope is the context in which a variable is accessible (what code is the variable accessible in)

- Two main types of variables and their scope:
  - Class/Object variables
  - Local variables
Class/Object Variables

- Class variables - variables declared in the class, but outside of all the function.
  - accessible globally and modifiable from any function in that class
Class/Object Variables

The Bad:

- variables take up space (memory) while in existence and class variables exist as long as that class exists
- makes functions dependent on outside data (less modular)
Local Variables

- Local variables - variables that are defined inside a specific code context and are not accessible outside of that context.
  - In functions we sometimes call these 'function variables'
  - Scope: only exists inside current code block:
    ```java
    for (int i = 0; i < 10; i++) {
        ...
    }
    System.out.println(i);  // problem?
    ```
Local Variables

- Local variables - variables that are defined inside a specific code context and are not accessible outside of that context.
  - In functions we sometimes call these 'function variables'
  - Scope: only exists inside current code block:
    ```java
    for (int i = 0; i < 10; i++) {
        ...
    }
    System.out.println(i);  // "i" doesn't exist anymore!
    ```
Local Variables

The Good:

- Deleted after current code block is done running (saves memory)
- Using local variables instead of global variables ensures good modular design
  - in general, a function shouldn't be dependent on code outside of itself, it should be self-contained.
Parameters (a.k.a. arguments)

Parameters are data that are passed to a function.

- Java is 'pass-by-value'
  - Changing the contents of a parameter does not change the contents of a original variable passed to that function.
- Create local copies of the value passed to it
- But we must go deeper...
Variables

Variables give us access to data that is stored in memory.

Two types of variables:

- Primitives (char/int/float/boolean/etc.)
- Classes/Objects/Arrays (Strings/Integer/Float/etc.)
Variables

- Primitives store the actual value
- Classes/Objects/Arrays store a reference to the location in memory where the data is stored
Return Values

- Functions have a way to pass data in: parameters
- Functions have a way to pass data back: return values
Return Values

- Return values: pass either 0 (void) or 1 item (any type, but must be specified):

```java
void printNums(int num) {
    for (int i = 0; i < num; i++)
        System.out.println(i);
    return; // done implicitly if not specified
}

int sum(int num) {
    int total = 0;
    for (int i = 0; i < num; i++)
        total += i;
    return total;
}
```
Libraries

- A library is a set of utilities that provide functionality that apply to specific topics (using functions).
  - system
  - networking
  - graphics
  - crypto
  - I/O
  - etc.