Methods & Multiple-method classes

Review of concepts we've seen before:

- Method: an encoded algorithm
    - meaning, a named set of instructions that performs a specific task
    - method collects this set of operations under a single name
    - makes code reusable, minimizes redundancy
- Method call: an instruction that causes a method to be executed (a.k.a. perform the operations in the method); includes:
    - Name of the calling object or class (unless unnecessary) followed by a dot
    - Method name
    - Set of arguments: data to be passed to the method (in parentheses)
- Method signature (or heading): specifies how to call a method; includes:
    - Visibility modifier: usually public – sometimes private or protected
    - (Optional) static designation: indicates that there is only one copy of this method and that it resides in the class, not in any instance (object)
    - Return type: designates whether or not a call to this method is an expression:
        - void means no return type – can't use method as expression
        - any data type indicates that a call to this method is an expression of the specified type
    - Name: the name of the method, to be used in the call
    - Parameter list: specifies data that must be included when method is called; includes:
        - number of arguments (syntax)
        - data type(s) of arguments (syntax)
        - order of arguments (logic and possibly syntax)

Defining a method

- Method definition consists of:
    - Heading: identical to signature, described above
    - Body: a block of code containing the instructions to be executed when the method is called
- Any time you have a chunk of code you might use more than once, you have a good candidate for a method definition

Utility class (API example: Math)

- collection of generally useful methods and constants
- members are public and static
- set of operations only: not intended to model a cohesive abstraction
    - no constructor
    - can't (or shouldn't) be instantiated

Modeling class (API example: almost all of them!)

- Blueprint or recipe for an object (cohesive abstraction) – describes:
    - knowledge or state: instance variables
    - operations: instance methods
- Instance variables:
    - declaration appears inside the class block, but outside any method block; scope is classwide
    - represent the state of an object; all instances of a class have the same set of instance variables, but their values may differ

- visibility access designator for instance variables is almost always private (occasionally protected, almost never public) – this means these values are freely accessible only by instance members of the same class, and are invisible outside the class

- Instance methods:
  - Constructors
    - provide reasonable default values for instance variables
    - have no return type other than the one specified by the method's name
    - are always public
    - all constructors are named after the class they're in, e.g. the constructor for the Scanner class is named Scanner, the constructor for the Random class is named Random, etc.
      - having multiple constructors in a class is an example of method overloading
      - compiler distinguishes between methods of the same name by differences in their parameter lists
    - default constructor: a constructor that has no parameters
      - if no constructor is explicitly defined, the automatic default constructor is called when an object is created
      - if any constructor is defined, the automatic default constructor ceases to exist, but you can define one
  - Accessors
    - provide information about an object: window for the outside world
    - return a value (usually the value of an instance variable)
    - don't change anything
  - Mutators
    - change state of object; provide new value for instance variable(s)
      - controlled access: can restrict range of values that can be assigned to preserve reasonable model
      - usually take at least one argument (value(s) to assign to instance variable(s))
    - often void; may return a value to indicate success or failure of operation
    - may throw exceptions