Chapter 2 - A simple program (p. 39)

- comments
  - file comments:
    - who wrote the class
    - when it was written
    - what program it was written for
    - if applicable, what lesson or project
    - *all information about the **file***
  - class comments:
    - what the class (or object) does
    - known bugs in the class
    - any special information about how the class works or should be used
    - *all information about the **class***

- javabook
  - a special set of classes created by the author
  - `MainWindow`
    - the constructor optionally takes a window title as an argument
    - the `setVisible(boolean)` method must be called with an argument of `true` before the window will be shown
    - can hide the window by using the `setVisible(boolean)` method with an argument of `false`
  - `MessageBox`
    - the constructor *must* include a `MainWindow` as an argument
    - the most useful method is `show(String)`, which will display the `String` argument as a message in its own pop-up window

  - import statement
    - tells compiler where to look for classes whose files are not present in the current directory
    - can use `*` to indicate all classes in a certain package (directory)

- object (variable) declaration
  - only assigns a name to a *potential* object
  - name can be reused, so long as it always points to the same *type* of object
  - several names can be declared in a single line of code

- object instantiation
  - always uses the `new` keyword
  - creates the object itself, and assigns a previously-declared name to it

- memory state vs object diagrams
  - object diagrams show one name per object
  - memory state diagrams are more accurate, as an object can have multiple names, or no names (just before it gets garbage-collected)
  - message sending
- class declaration
  - modifiers, `class`, identifier (name)
  - all class information (data and method) must be enclosed within a curly brace pair
- method declaration
  - modifiers, return type, identifier, arguments

Edit-Compile-Run cycle:

- edit: writing the source code
- compile: converting source code into simplified machine code (`.class` file — byte code)
- run: using a program (like Code Warrior, Netscape, or the Java Runtime Environment) to *interpret* the byte code — that is, to convert the simplified machine code into actual machine code specific to the computer running the java program