Chapter 4
Methods: Functional Abstraction

- Main topics
  - Structured Programming
  - Top-down design – also called stepwise refinement
- Methods
  - How to write
  - Parameters
  - Variable scope
  - Overloading
- Recursion
  - Self-calling methods

Functional Abstraction

- Abstraction – taking away or removing characteristics from something until only the essential ones remain
- Functional abstraction
  - Define methods that perform functions for us
  - We need to know what they will do, but we don't need to know the details of how they will do it.
  - For example, System.out.println() prints data for use, but we don't need to know how it does it.

Structured Programming

- Programming methodology that includes two guidelines
  - The program flow of control should be as simple as possible
    - Programs should have a logical structure that makes them easy to understand and modify
  - Program construction should follow top-down design

Top-Down Design

- Powerful technique for decomposing large, hard problems into small, simple ones
- Start with your hard problem
  - Break it up into smaller pieces
  - Break the pieces up into yet smaller pieces
    - Until you know how to solve them
  - Put all the small solutions together to solve the original hard problem

Top-Down Design: Example

- Take a trip to New York City
- How can we decompose this into smaller tasks?

Top-Down Design

1. Get a guidebook
2. Figure out when to go
3. Find a cheap airline ticket
4. Make a hotel reservation
5. Go
Top-Down Design

3. Find a cheap airline ticket
   1. Decide which airports to use
   2. While (prices are too high)
      1. Search the web
      2. Call the airlines
      3. Ask a travel agent
   3. Buy ticket

Methods

• In Java
  – Each subtask or sub-problem corresponds to a method
  – When we want a Java program to perform a subtask, we call the method
  – Methods in turn may call other methods to perform other subtasks

Method Invocation

• A simple Java program contains one or more methods
  – including main(), which is where the program starts
• When program control encounters a method name followed by (), it is called or invoked
  – Program control passes to the called method
  – When the called method is finished, program control returns to the called method, where program execution continues

// Message.java: Simple method use
class Message {
    public static void main(String [] args) {
        System.out.println("Hello, class!");
        printMessage(); // method call
        System.out.println("Goodbye.");
    }

    // definition of method printMessage
    static void printMessage() {
        System.out.println("A message for you:");
        System.out.println("Have a nice day!
");
    }
}