Road Map

• Introduction to Methods
  – Divide and Conquer
  – Abstraction

• Using Pre-Packaged Modules
  – Understanding Method signatures
  – Understanding java.lang.Math methods

• Creating your own methods

• Reading:
  – Liang: Chapter 5: Sections 5.1 - 5.3; 5.9.1- 5.9.4
Introduction to Methods
6.1 Introduction

• Modules
  – Small pieces of a problem
    • e.g., divide and conquer
  – Facilitate design, implementation, operation and maintenance of large programs
6.2 Program Modules in Java

• Modules in Java
  – Methods (procedural programming)
  – Classes (object oriented programming)

• Java API provides several modules

• Programmers can also create modules
  – e.g., programmer-defined methods

• Methods
  – Invoked by a method call
  – Returns a result to calling method (caller)
  – Similar to a boss (caller) asking a worker (called method) to complete a task.
    • The boss method delegates certain jobs to specific methods.
Fig. 6.1  Hierarchical boss-method/worker-method relationship.
Examples of methods

• A method that determines the maximum of two numbers.
• A method that sorts a list of names
• A method that opens a file from the file system.
• A method that reads from the open file.
• A method that opens a new socket to the internet.
• A method that reads from that socket.
• A method that parses an integer value from a String.
• A method that gets a String from the user.
Important concept #1

• **Divide and Conquer**: Break large programs into a series of smaller modules
  
  – Helps manage complexity
  – Makes it easier to build large programs
  – Makes it easier to debug programs
Important concept #2

• **Abstraction**: Most of the time, you need to know what a method does, but not how it actually does it.
  
  – Also helps manage complexity
  – You use other people’s code without knowing how it does it’s job.
Using Static methods in the Java API
6.3 Math-Class Methods

• Class java.lang.Math
  – Provides common mathematical calculations
  – Calculate the square root of 900.0:
    • Math.sqrt( 900.0 )
      – Method sqrt belongs to class Math
        • Dot (.) allows access to method sqrt
      – The argument 900.0 is located inside parentheses
<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
</table>
| abs( x ) | absolute value of x (this method also has float and long versions)          | abs(23.7) is 23.7  
                        |                                                               | abs(0.0) is 0.0  
                        |                                                               | abs(-23.7) is 23.7  |
| ceil( x )| rounds x to the smallest integer not less than x                             | ceil(9.2) is 10.0  
                        |                                                               | ceil(-9.8) is -9.0  |
| cos( x ) | trigonometric cosine of x (x is in radians)                                  | cos(0.0) is 1.0  |
| exp( x ) | exponential method e^x                                                      | exp(1.0) is 2.71828  
                        |                                                               | exp(2.0) is 7.38906  |
| floor( x )| rounds x to the largest integer not greater than x                           | floor(9.2) is 9.0  
                        |                                                               | floor(9.8) is -10.0  |
| log( x ) | natural logarithm of x (base e)                                             | log(Math.E) is 1.0  
                        |                                                               | log(Math.E*Math.E) is 2.0  |
| max( x, y ) | larger value of x and y (this method also has float and long versions)      | max(2.3, 12.7) is 12.7  
                        |                                                               | max(-2.3, -12.7) is -2.3  |
| min( x, y ) | smaller value of x and y (this method also has float and long versions)     | min(2.3, 12.7) is 2.3  
                        |                                                               | min(-2.3, -12.7) is -12.7  |
| pow( x, y ) | x raised to the power y (xy)                                                | pow(2.0, 7.0) is 128.0  
                        |                                                               | pow(9.0, 0.5) is 3.0  |
| sin( x ) | trigonometric sine of x (x is in radians)                                    | sin(0.0) is 0.0  |
| sqrt( x ) | square root of x                                                            | sqrt(900.0) is 30.0  
                        |                                                               | sqrt(9.0) is 3.0  |
| tan( x ) | trigonometric tangent of x (x is in radians)                                | tan(0.0) is 0.0  |

**Fig. 6.2** Math-class methods.
6.4 Methods Declarations

• Methods
  – Allow programmers to modularize programs
    • Makes program development more manageable
    • Software reusability
    • Avoid repeating code
  – Local variables
    • Declared in method declaration
  – Parameters
    • Communicates information between methods via method calls
6.4 Method Declarations (Cont.)

• Programmers can write customized methods
```
1 // Fig. 6.3: SquareIntegers.java
2 // Creating and using a programmer-defined method.
3 public class SquareIntegers {

4    public static void main (String args[]) {
5        int result;   // store result of call to method square
6
7        // loop 10 times
8        for ( int counter = 1; counter <= 10; counter++ ) {
9            result = square( counter );  // method call
10
11                // print the result of one call to the method
12                System.out.println ("The square of " + counter + " is " +
13                    result );
14        } // end for
15
16    } // end method main()

17 // square method declaration
18 public static int square( int y ) {
19    return y * y;  // return square of y
20} // end method square
21} // end class SquareIntegers
```
6.4 Method Declarations (cont.)

• General format of method declaration:

\[
\text{modifiers return-value-type method-name( parameter1, ..., parameterN )}
\]
\[
\{
    \text{declarations and statements}
\}
\]

• Method can also return values:

\[
\text{return expression;}
\]