Welcome to methods

- Most programs are far more complex and involved than what we’ve been looking at.
- The best way to develop and maintain a large system is to break it down and build it from smaller pieces or modules.
- The divide and conquer technique of solving problems lends itself nicely to modular program development.
Modularity in Java

- Modularity in Java comes in three varieties: methods, classes and packages:
  - Methods modularize a program by separating tasks into self-contained units.
  - Classes group together related methods.
  - Packages group together related classes.
- Programs are pieced together from user-written methods and classes and pre-written methods and classes from the Java Application Programming Interface, or Java API.

Methods

What is a method?

- A method is a small, self-contained unit of code that accomplishes a singular, well-defined task.
- Can take multiple (or even no) inputs.
- Can return one or no results.

Java method terms / concepts

- To make a method do what it’s supposed to, you invoke it by making a method call.
- A method call specifies the method name and provides information to it as arguments.
- When a called method is done working on its task, it can send information back to the calling method via a return statement.
- You can call a method without having to worry about how the work is done or the results are calculated.
Introduction to the Math class

- Math plays a large role in computer programs.
- Because of this, there is an entire class (`Math`) that provides easy-to-use interfaces to many common mathematical methods.
- Unlike most other classes, the `Math` class is part of the `java.lang` package, which is imported automatically by the compiler when you compile a program. Therefore, you don’t need to do anything special in your program to have access to these methods.

Introduction to the Math class

- The `Math` class consists of:
  - `static` methods, which are methods that don’t depend on the contents of an object.
  - `static` fields, which are values that are usually defined to be `public`, `final` and `static`, meaning that anyone can access them outside the package. Since their values are `final`, that means that they’re constant and can’t be changed. These values are also known as constants.
### Some Math class methods

<table>
<thead>
<tr>
<th>Method Name</th>
<th>Description</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>Math.abs(x)</td>
<td>Absolute value of x</td>
<td>Math.abs(23.7) = 23.7, Math.abs(-23.7) = 23.7</td>
</tr>
<tr>
<td>Math.ceil(x)</td>
<td>Rounds x to smallest integer not less than x</td>
<td>Math.ceil(4.2) = 5.0, Math.ceil(-4.8) = -4.0</td>
</tr>
<tr>
<td>Math.exp(x)</td>
<td>Exponential method $e^x$</td>
<td>Math.exp(1.0) = 2.71828, Math.exp(2.0) = 7.38906</td>
</tr>
<tr>
<td>Math.floor(x)</td>
<td>Rounds x to the largest integer not greater than x</td>
<td>Math.floor(9.2) = 9.0, Math.floor(-9.8) = -10.0</td>
</tr>
<tr>
<td>Math.max(x, y)</td>
<td>Larger value of x and y</td>
<td>Math.max(2.3, 12.7) = 12.7, Math.max(-2.3, -12.7) = -2.3</td>
</tr>
<tr>
<td>Math.min(x, y)</td>
<td>Smaller value of x and y</td>
<td>Math.min(2.3, 12.7) = 2.3, Math.min(-2.3, -12.7) = -12.7</td>
</tr>
<tr>
<td>Math.pow(x, y)</td>
<td>$x$ raised to the power $y$</td>
<td>Math.pow(2.0, 6.0) = 64.0, Math.pow(9.0, 0.5) = 3.0</td>
</tr>
<tr>
<td>Math.sqrt(x)</td>
<td>Square root of x</td>
<td>Math.sqrt(900.0) = 30.0</td>
</tr>
</tbody>
</table>

### Math class constants

<table>
<thead>
<tr>
<th>Constant name</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Math.PI</td>
<td>The mathematical value Pi</td>
<td>3.141592653589793</td>
</tr>
<tr>
<td>Math.E</td>
<td>Base value for natural logs</td>
<td>2.718281828459045</td>
</tr>
</tbody>
</table>

### Example Math class usage

```java
import java.util.Scanner;

public class MathTest {
    public static void main(String[] args) {
        Scanner i = new Scanner(System.in);

        System.out.print("Enter number: ");
        double a = i.nextDouble();

        System.out.printf("abs: %.2f \n", Math.abs(a));
        System.out.printf("ceil: %.2f \n", Math.ceil(a));
        System.out.printf("sqrt: %.2f \n", Math.sqrt(a));
        System.out.printf("Pi: %.2f\n", Math.PI);
    }
}
```