

Mat 2170  
Week 6

**Methods**

Mat 2170  
Week 6

**Methods**

Spring 2012

# Student Responsibilities

Mat 2170  
Week 6

Methods

- Reading: Textbook, Chapter 5.1 – 5.3
- Lab
- Attendance

# Chapter Five Overview: 5.1 – 5.3

Mat 2170  
Week 6

Methods

## Methods

- 5.1 Quick overview of methods
- 5.2 Writing our own methods
- 5.3 Mechanics of the method–calling process

# Quick Overview of Methods

Mat 2170  
Week 6

Methods

- You've been working with methods ever since Lab 1 and the HelloWorld program.
- The `run()` method in every program is one example.
- Other examples are `println()`, `setColor()`, and `getHeight()`.

# Methods

Mat 2170  
Week 6

Methods

Basically a **method** is a  
**sequence of statements**  
collected together and **given a name**.

- The name makes it possible to execute the statements more easily.
- Instead of copying the entire list of statements, we can simply provide the method name.

# Useful Terms

Mat 2170  
Week 6

Methods

- **Invoking** a method (using its name) is known as **calling** that method.
- The part of a program or code that invokes a method is named the **caller** or **calling program**.
- The caller can **pass** information to a method by using **arguments** (the java expressions in the parentheses), e.g.:

`R.setFilled(true)`

`R.setFilled((row + col) % 2 == 0)`

`R.setColor(Color.green)`

- When a method completes its operation, it returns to the code which invoked it (i.e., the caller).
- A method can pass information back to the caller by **returning a result**.

# Method Calls as Messages

Mat 2170  
Week 6

Methods

- The act of **calling a method** is often described in terms of **sending a message to an object**.

- The method call: `R.setColor(Color.RED);`

is regarded metaphorically as **sending a message to** the **GRect** object **R** telling it to change its color.

# Receivers

Mat 2170  
Week 6

Methods

- The object to which a message is sent is called the **receiver**.
- The general pattern for sending a message to an object is:

```
receiver.methodName(argument_list);
```

# Information Hiding

Mat 2170  
Week 6

Methods

- **One important advantage of methods:**

They allow us to **ignore the inner workings** of complex operations.

- When a method is used, it is **more important** to know

**what** the method does  
than to understand exactly  
**how** it does it.

- The underlying details of a method are of interest only to the programmer who implements and maintains it.

# Method Interface

Mat 2170  
Week 6

Methods

- Programmers who **utilize** a method are concerned about:
  1. The **method interface**
    - 1.1 **return type**
    - 1.2 method **name**
    - 1.3 **order, number** and **type** of arguments
  2. Whether the method is **correct**.
  
- They can usually ignore the **implementation** altogether.
  
- The idea that callers should be **insulated** from the details of a method is the principle of **information hiding**, one of the cornerstones of **software engineering**.

# Methods as Tools for Programmers

Mat 2170  
Week 6

## Methods

- A method provides a **service** to a **programmer**, who is typically creating some kind of application.
- A programmer **utilizes** methods to reduce the amount of work he or she must do, and to organize the software they are writing.
- Methods like **readInt()** and **println()** are used to communicate with and obtain information **from the user**.

# Method Calls as Expressions

Mat 2170  
Week 6

Methods

- Syntactically, method calls in Java are part of the **expression** framework.
- Methods that **return a value** can be used as **terms** in an expression, just like variables and constants.
- The **Math** class in the **java.lang** package defines several methods that are useful in writing mathematical expressions.

# Math Library Method Calls

Mat 2170  
Week 6

Methods

- You must **include the name of the class**, along with the method name, for example: **Math.sqrt()**.

Methods that **belong to the entire class**  
are called **static** methods.

- Suppose we need to compute the distance from the origin to the point  $(x, y)$ , which is given by the formula:

$$d = \sqrt{x^2 + y^2}$$

- We can apply the square root function by calling the **sqrt()** method from the **Math** class:

```
double distance = Math.sqrt(x * x + y * y);
```

# Useful Methods in the Math Class

Mat 2170  
Week 6

## Methods

<code>Math.abs(x)</code>	Returns the absolute value of $x$
<code>Math.min(x, y)</code>	Returns the smaller of $x$ and $y$
<code>Math.max(x, y)</code>	Returns the larger of $x$ and $y$
<code>Math.sqrt(x)</code>	Returns the square root of $x$
<code>Math.log(x)</code>	Returns the natural logarithm of $x$ ( $\log_e(x)$ )
<code>Math.exp(x)</code>	Returns the inverse logarithm of $x$ ( $e^x$ )
<code>Math.pow(x, y)</code>	Returns the value of $x$ raised to the $y$ power ( $x^y$ )

# The Math Class — Trig Functions

Mat 2170  
Week 6

Methods

<code>Math.sin(<i>theta</i>)</code>	Returns the sine of <i>theta</i> , <b>measured in radians</b>
<code>Math.cos(<i>theta</i>)</code>	Returns the cosine of <i>theta</i>
<code>Math.tan(<i>theta</i>)</code>	Returns the tangent of <i>theta</i>
<code>Math.asin(x)</code>	Returns the angle whose sine is x
<code>Math.acos(x)</code>	Returns the angle whose cosine is x
<code>Math.atan(x)</code>	Returns the angle whose tangent is x

# The Math Class — Conversion Functions

Mat 2170  
Week 6

Methods

<code>Math.toRadians(<i>degrees</i>)</code>	Converts an angle from degrees to radians
<code>Math.toDegrees(<i>radians</i>)</code>	Converts an angle from radians to degrees

# Solving Quadratic Equations

Mat 2170  
Week 6

Methods

- The standard **quadratic** equation is:

$$ax^2 + bx + c = 0$$

- The quadratic **formula**, to solve for the roots, is:

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

- Of interest to us is the **radicand**, since it determines the **number** of solutions:

$$b^2 - 4ac$$

- How many **real** solutions are there when the radicand is...
  1. positive?
  2. zero?
  3. negative?

## Also of Use in Lab 6

Mat 2170  
Week 6

Methods

More messages available for **GRect** and **G Oval** objects

<code>getX()</code>	returns the x component of the object's position
<code>getY()</code>	returns the y component of the object's position
<code>getWidth()</code>	returns the width of the object
<code>getHeight()</code>	returns the height of the object
<code>move(dx, dy)</code>	moves the object using the displacements dx and dy

# Examples

Mat 2170  
Week 6

Methods

Assume **R** is a **GRect** and **C** is a **GOval**:

<code>R.getX()</code>	<code>C.getX()</code>
<code>R.getY()</code>	<code>C.getY()</code>
<code>R.getWidth()</code>	<code>C.getWidth()</code>
<code>R.getHeight()</code>	<code>C.getHeight()</code>
<code>R.move(1.0, -1.5)</code>	<code>C.move(1.0, -1.5)</code>
<code>R.move(dx, dy)</code>	<code>C.move(dx, dy)</code>

# Problem Solving — in Class

Mat 2170  
Week 6

Methods

Design algorithms, then programs, to:

- Create a table of values for  $x$ ,  $\sqrt{x}$ , and  $x^2$  running from  $x = 0$  to  $x = 100$  **by 10s**
- Create a table of values for  $x$ ,  $\sin(x)$ , and  $\cos(x)$  as  $x$  runs from  $0$  to  $2\pi$  by  $\frac{\pi}{4}$  increments.
- Solve for the  $n^{\text{th}}$  Fibonacci number, defined by the sequence  $0, 1, 1, 2, 3, 5, 8, 13 \dots$ . The first two terms are  $0$  and  $1$ , and every subsequent term is the sum of the preceding two.
- Modify the previous program to display **all** the terms in the Fibonacci sequence that are smaller than  $1,000$ .