

	Methods Aka: subroutines, modules, functions, procedures, etc.

	Quick Heads Up!
	<ul style="list-style-type: none">■ Assignment 1, 2, and 3 – Due today<ul style="list-style-type: none">– If you did not do the assignment in the lab, email the source code or use the online form; no later than tonight 11:59 pm■ Assignment 4 – Posted and Due 6/16■ Performance Test 1 – Tomorrow

	Event Driven Programming
	<ul style="list-style-type: none">■ What is Event-Driven programming?■ Getting started with event-driven programming in Java:<ul style="list-style-type: none">– Applet starts with <code>init()</code>, <code>start()</code> and <code>paint()</code> methods– <code>action()</code> trap users response to the object

Questions & Answer

- Assignment 2?
- Assignment 3?
- Java?

Why doesn't code work?

```
import java.applet.Applet;  
  
public class AppHW extends Applet {  
    public void paint(Graphics g) {  
        System.exit(1);  
    }  
}
```

System.exit() in an Applet

- Terminates the currently running Java Virtual Machine.
- What is the result in an applet?
 - Causes what's known as a SecurityException
- This will not work in a browser.

Constant Variables

- Constant variable names should be capitalized.
- The underscore "_" is used to separate words, ONLY USED WITH CONSTANT VARS.
- Use the keyword final in the variable declaration:

```
public static final float ADULT_TICKET_PRICE = 8.75f;
```

- A Java(TM) programming language keyword. You define an entity once and cannot change it or derive from it later.

Local Variables

- Local variables are variables that are located inside a code block and are not global in respects to the program.
- These variables are located inside a method, main(), init(), paint(), or a code block like an if/else structure.
- All local variables should begin with a lowercase letter and any subsequent words should be capitalized.

```
myName  
intChildTickets
```

Brief Review

- Java has two types of programs
 - **Applet** – runs in a web browser; *init()*, *start()*, *paint()*
 - **Application** – stand alone program; *main()*
- Each program consists of a number of statements
- Statements are the most fundamental building block

Brief Review

- Programs execute in sequential order
- Data Types
 - Primitive Data Types; int, char, short, long, boolean, byte, etc.
 - Reference Object Data Types; String s; or Graphics g

Brief Review

- Type Casting (new type)
 - int i = (int) doubleValue;
- Operator precedence; +, -, *, %, etc.
 - operations occur from left to right and by operator precedence

Brief Review

- We can determine the execution path with "Control Structures"
 - If/Else Statements
 - Switch/Case Statements
 - While Loops
 - Do/While Loops
 - For Loops
 - break and continue statements (Very Important)

Brief Review??

- Bitwise and Logical Operators
 - & - "And" - Both
 - | - "Or" - Either
 - ^ - "XOr" - One or the other, but not both
 - && - "And" - Short Circuit
 - || - "Or" - Short Circuit
 - Short circuit operators don't test both operands under certain situations

Brief Review

- `if(false && true) // Tests the left side only`
- `if(false & true) // Test both sides of the operator`
- `if(true || false) // Tests the left side only`
- `if(false | true) // Tests both sides`

Brief Review

- `if (input == 65 || input == 97)`
- `if(input != null && input == "a")`
`//input is a String`
- Please refer to the book for bitwise operators pages 1117 - 1134

Conditional Operator

- ```
if(input == 1) {
 System.out.println("1");
} else {
 System.out.println("2");
}
```
- Equivalent Conditional Operator  

```
(input == 1) ? System.out.println("1"); :
 System.out.println("2");
```

---

---

---

---

---

---

---

---

## Conditional Operator

( *conditional-statement* ) ? *statement1* :  
*statement2*

This is Java's only ternary operator (takes three operands)

*(request == HOME\_PAGE) ? forward("index.htm") : forward("other.htm");*

---

---

---

---

---

---

---

---

## Methods

- Now we move onto Java methods

---

---

---

---

---

---

---

---

## Top-Down Design

- How do you eat an elephant?
- One Bite At a Time

---

---

---

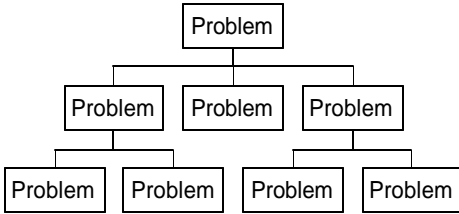
---

---

---

---

## Top-Down Design



---

---

---

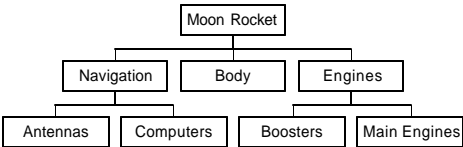
---

---

---

---

## Top-Down Design



---

---

---

---

---

---

---

## Top-Down Design

- Divide and Conquer: break big problem down to a series of smaller problems
- Why?
  - Simplify complexity
  - Division of labor (across programmers)
  - Reuse of functionality across programs

---

---

---

---

---

---

---

---

## Top-Down Design

- Most programming languages added constructs to break a program into smaller sub-programs or sub-routines (functions in C/C++, methods in Java)
- The name functions in C/C++ (and FORTRAN) are derived from their mathematical equivalents:

$$y = f(x)$$

bonus = sqrt(sales);

Value returned from the function

Argument passed to the function

---

---

---

---

---

---

---

---

## Predefined Methods

- Using Predefined Methods
  - The Java API (Application Programming Interface) provides you with *libraries* of predefined functions
  - Syntax:  
*Object.methodName(Argument\_List)*
  - Example: (note *import* needed)

---

---

---

---

---

---

---

---

## Sample methods in Math package

| Method             | Description                                                                                                                      | Example                                                              |
|--------------------|----------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------|
| <b>max( x, y )</b> | larger value of <i>x</i> and <i>y</i><br>(this method also has versions for <b>float</b> , <b>int</b> , and <b>long</b> values)  | <b>max( 2.3, 12.7 )</b> is 12.7<br><b>max( -2.3, -12.7 )</b> is -2.3 |
| <b>min( x, y )</b> | smaller value of <i>x</i> and <i>y</i><br>(this method also has versions for <b>float</b> , <b>int</b> , and <b>long</b> values) | <b>min( 2.3, 12.7 )</b> is 2.3<br><b>min( -2.3, -12.7 )</b> is -12.7 |
| <b>pow( x, y )</b> | <i>x</i> raised to power <i>y</i> ( <i>x</i> <sup><i>y</i></sup> )                                                               | <b>pow( 2, 7 )</b> is 128<br><b>pow( 9, .5 )</b> is 3                |
| <b>sin( x )</b>    | trigonometric sine of <i>x</i><br>( <i>x</i> in radians)                                                                         | <b>sin( 0.0 )</b> is 0                                               |
| <b>sqrt( x )</b>   | square root of <i>x</i>                                                                                                          | <b>sqrt( 900.0 )</b> is 30<br><b>sqrt( 9.0 )</b> is 3                |
| <b>tan( x )</b>    | trigonometric tangent of <i>x</i><br>( <i>x</i> in radians)                                                                      | <b>tan( 0.0 )</b> is 0                                               |

---

---

---

---

---

---

---

---

---

---

## Packages in the API

| Java API package             | Explanation                                                                                                                                                                                                                                                                                                                            |
|------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>java.applet</b>           | <i>The Java Applet Package.</i><br>This package contains the Applet class and several interfaces that enable the creation of applets, interaction of applets with the browser, and playing audio clips.                                                                                                                                |
| <b>java.awt</b>              | <i>The Java Abstract Windowing Toolkit Package.</i><br>This package contains all the classes and interfaces required to create and manipulate graphical user interfaces (these classes are discussed in detail in Chapter 10, Basic Graphical User Interface Components and Chapter 11, Advanced Graphical User Interface Components). |
| <b>java.awt.datatransfer</b> | <i>The Java Data Transfer Package.</i><br>This package contains classes and interfaces that enable transfer of data between a Java program and the computer's clipboard (a temporary storage area for data).                                                                                                                           |
| <b>java.awt.event</b>        | <i>The Java Abstract Windowing Toolkit Event Package.</i><br>This package contains classes and interfaces that enable event handling for GUI components.                                                                                                                                                                               |

---

---

---

---

---

---

---

---

---

---

## Packages in the API

| Java API package      | Explanation                                                                                                                                                                                                                                                                                                                                                                                                                    |
|-----------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>java.awt.image</b> | <i>The Java Abstract Windowing Toolkit Image Package.</i><br>This package contains classes and interfaces that enable storing and manipulation of images in a program.                                                                                                                                                                                                                                                         |
| <b>java.awt.peer</b>  | <i>The Java Abstract Windowing Toolkit Peer Package.</i><br>This package contains interfaces that enable Java's graphical user interface components to interact with their platform-specific versions (i.e., a button is implemented differently on each computer platform so its peer is used to actually display and manipulate the button in a platform-specific manner). Programmers should not use this package directly. |
| <b>java.beans</b>     | <i>The Java Beans Package.</i><br>This package contains classes and interfaces that enable the programmer to create reusable software components. Java beans can interact with non-Java and Java software components.                                                                                                                                                                                                          |
| <b>java.io</b>        | <i>The Java Input/Output Package.</i><br>This package contains classes that enable programs to input and output data (see Chapter 15, Files and Streams).                                                                                                                                                                                                                                                                      |

---

---

---

---

---

---

---

---

---

---

# Packages in the API

| Java API package                                                                             | Explanation                                                                                                                                                                                                                                                                                                           |
|----------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>java.lang</b>                                                                             | <i>The Java Language Package.</i><br>This package is automatically imported by the compiler into all programs. The package contains basic classes and interfaces required by many Java programs (these classes are discussed throughout the text).                                                                    |
| <b>java.lang.reflect</b>                                                                     | <i>The Java Core Reflection Package.</i><br>This package contains classes and interfaces that enable a program to discover the accessible variables and methods of a class dynamically during the execution of a program.                                                                                             |
| <b>java.net</b>                                                                              | <i>The Java Networking Package.</i><br>This package contains classes that enable programs to communicate via the Internet or corporate intranets (see Chapter 16, Networking).                                                                                                                                        |
| <b>java.rmi</b><br><b>java.rmi.dgc</b><br><b>java.rmi.registry</b><br><b>java.rmi.server</b> | <i>The Java Remote Method Invocation Packages.</i><br>These packages contain classes and interfaces that enable the programmer to create distributed Java programs. Using remote method invocation, a program can call a method of a separate program on the same computer or on a computer anywhere on the Internet. |

---

---

---

---

---

---

---

---

---

---

# Packages in the API

| Java API package                                                                    | Explanation                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
|-------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>java.security</b><br><b>java.security.acl</b><br><b>java.security.interfaces</b> | <i>The Java Security Packages.</i><br>These packages contains classes and interfaces that enable a Java program to encrypt data and control the access privileges provided to a Java program for security purposes.                                                                                                                                                                                                                                               |
| <b>java.sql</b>                                                                     | <i>The Java Database Connectivity Package.</i><br>This package contain classes and interfaces that enable a Java program to interact with a database.                                                                                                                                                                                                                                                                                                             |
| <b>java.text</b>                                                                    | <i>The Java Text Package.</i><br>This package contains classes and interfaces that enable a Java program to manipulate numbers, dates, characters and strings. This package provides many of Java's internationalization capabilities. Internationalization enables a Java program to be customized to a specific locale. For example, an applet may display strings in different languages based on the World Wide Web browser in which the applet is executing. |

Fig. 4.6 The Java API packages (part 4 of 5).

---

---

---

---

---

---

---

---

---

---

# Packages in the API

| Java API package     | Explanation                                                                                                                                                                                                                                                                                                                                                                                                  |
|----------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>java.util</b>     | <i>The Java Utilities Package.</i><br>This package contains utility classes and interfaces such as: date and time manipulations, random number processing capabilities ( <b>Random</b> ), storing and processing large amounts of data, breaking strings into smaller pieces called tokens ( <b>StringTokenizer</b> ), and other capabilities (see Chapter 13, Java Utilities Package and Bit Manipulation). |
| <b>java.util.zip</b> | <i>The Java Utilities Zip Package.</i><br>This package contains utility classes and interfaces that enable a Java program to combine Java <b>.class</b> files and other resource files (such as images and audio) into a single compressed file called a <i>Java archive (JAR)</i> file. This package also enables a Java program to read JAR files.                                                         |

Fig. 4.6 The Java API packages (part 5 of 5).

---

---

---

---

---

---

---

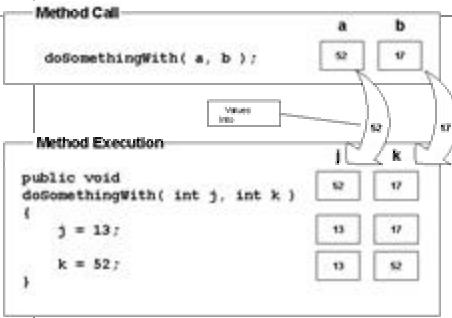
---

---

---

Figure 4-11

## Programmer-Defined Functions



---

---

---

---

---

---

---

---

## Method Definition

```
return-value-type method-name (parameter-list)
{
 declarations and statements
}
```

All variables defined in a method are called “local variables” and have a lifespan that only exists within the method block of code.

---

---

---

---

---

---

---

---

## Simple Code Example

- Method definition:  
Syntax, sample sum function
- Call it,
- Navigating the API tree, call some other predefined routines
- No return value = void
- return something; vs. return;

---

---

---

---

---

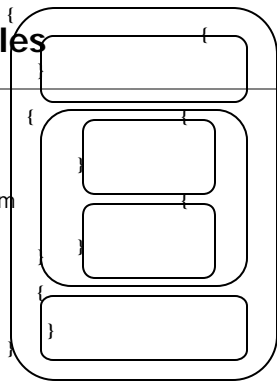
---

---

---

## Scoping Rules

- Variables exist in the scope it was created
- Visibility starts from the inside out
- Class scope vs. Block scope



---

---

---

---

---

---

---

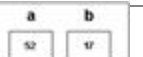
---

## Parameter Passing by Value or Reference

Figure 4-8 Passing by value

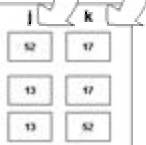
### Method Call

```
doSomethingWith(a, b);
```



### Method Execution

```
public void
doSomethingWith(int j, int k)
{
 j = 13;
 k = 52;
}
```



---

---

---

---

---

---

---

---

## More Code Example

- Illustration of
  - Scoping rules, name hiding
  - Call-by-value parameter passing

---

---

---

---

---

---

---

---

## Procedural Abstraction

- The Small Black Box Analogy:
  - Each routine should do one thing and do it well
  - Method name: selection & convention
  - Use Pseudocode outline
- Call-by-Value Formal Parameters Are Local Variables

---

---

---

---

---

---

---

---

## Understand “static”

- Key concept to remember: “Load time!”
- Static data (System-side): Global Constants and Global Variables
  - Keyword “final” used to declare constant
  - “Blank” final
- Static code (System-wide): “utility” methods
- Reference to static data and code must include the class name, NOT the variable
  - `System.out.println()` but `easyInVar.readChar()`

---

---

---

---

---

---

---

---

## More Code Example

- Re-implement KW with static utility methods

---

---

---

---

---

---

---

---

## Method Overloading

- Introduction to Overloading
- Method Signatures, return-type doesn't count
- Automatic Type Conversion

---

---

---

---

---

---

---

## Type Promotion

| Type    | Valid promotions                                               |
|---------|----------------------------------------------------------------|
| double  | None                                                           |
| float   | double                                                         |
| long    | float or double                                                |
| int     | long, float or double                                          |
| char    | int, long, float or double                                     |
| short   | int, long, float or double                                     |
| byte    | short, int, long, float or double                              |
| boolean | None (boolean values are not considered to be numbers in Java) |

**Fig. 6.5** Allowed promotions for primitive types.

---

---

---

---

---

---

---

## Recursion

- Method that calls itself
  - Disadvantages
  - Advantages

---

---

---

---

---

---

---

# Methods in Applet

| Method                           | When the method is called and its purpose                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
|----------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>public void init()</code>  | This method is called once by the Appletviewer or browser when an applet is loaded for execution. It performs initialization of an applet. Typical actions performed here are initialization of instance variables and GUI components of the Applet, loading of sounds to play or images to display (Chapter 14, Multimedia), and creation of threads (Chapter 13, Multithreading).                                                                                                                                                                                                        |
| <code>public void start()</code> | This method is called after the <code>init</code> method completes execution and every time the user of the browser returns to the HTML page on which the applet resides (after browsing another HTML page). This method performs any tasks that must be completed when the applet is loaded for the first time into the browser and that must be performed every time the HTML page on which the applet resides is revisited. Typical actions performed here include starting an animation (Chapter 14, Multimedia) and starting other threads of execution (Chapter 13, Multithreading). |

Fig. 4.18 Applet methods called automatically during an applet's execution. (part 1 of 2)

---

---

---

---

---

---

---

---

---

---

# Methods in Applet

| Method                                       | When the method is called and its purpose                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
|----------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>public void paint( Graphics g )</code> | This method is called after the <code>init</code> method completes execution and the <code>start</code> method has started executing to draw on the applet. It is also called automatically every time the applet needs to be repainted. For example, if the user of the browser covers the applet with another open window on the screen then uncovers the applet, the <code>paint</code> method is called. Typical actions performed here involve drawing with the <code>Graphics</code> object <code>g</code> that is automatically passed to the <code>paint</code> method for you. |
| <code>public void stop()</code>              | This method is called when the applet should stop executing—normally when the user of the browser leaves the HTML page on which the applet resides. This method performs any tasks that are required to suspend the applet's execution. Typical actions performed here are to stop execution of animations and threads.                                                                                                                                                                                                                                                                 |
| <code>public void destroy()</code>           | This method is called when the applet is being removed from memory—normally when the user of the browser exits the browsing session. This method performs any tasks that are required to destroy resources allocated to the applet. Typical actions performed here include terminating threads (Chapter 13, Multithreading).                                                                                                                                                                                                                                                            |

---

---

---

---

---

---

---

---

---

---

# Package

- package statement must be the first line
- Package name will be expanded into directory structure
- Package will be search from CLASSPATH
- import statement must be use to set access path to the package
- See sample code...

---

---

---

---

---

---

---

---

---

---