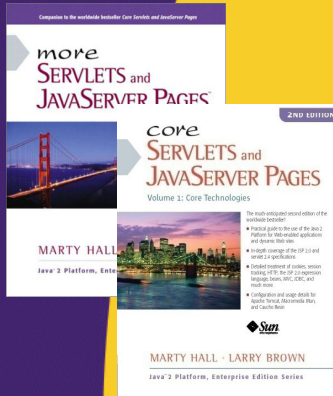




Basic Java Syntax

Originals of Slides and Source Code for Examples:
<http://courses.coreservlets.com/Course-Materials/java5.html>

Customized Java EE Training: <http://courses.coreservlets.com/>
Servlets, JSP, JSF 2.0, Struts, Ajax, GWT 2.0, Spring, Hibernate, SOAP & RESTful Web Services, Java 6.
Developed and taught by well-known author and developer. At public venues or onsite at *your* location.



**For live Java EE training, please see training courses
at <http://courses.coreservlets.com/>.**

Servlets, JSP, Struts, JSF 1.x, JSF 2.0, Ajax (with jQuery, Dojo, Prototype, Ext-JS, Google Closure, etc.), GWT 2.0 (with GXT), Java 5, Java 6, SOAP-based and RESTful Web Services, Spring, Hibernate/JPA, and customized combinations of topics.



Taught by the author of *Core Servlets and JSP*, *More Servlets and JSP*, and this tutorial. Available at public venues, or customized versions can be held on-site at your organization. Contact hall@coreservlets.com for details.

Topics in This Section

- **Basics**
 - Creating, compiling, and executing simple Java programs
- **Accessing arrays**
- **Looping**
- **Indenting Code**
- **Using if statements**
- **Comparing strings**
- **Building arrays**
 - One-step process
 - Two-step process
 - Using multidimensional arrays
- **Performing basic mathematical operations**
- **Reading command-line input**

4

© 2010 Marty Hall



Basics

Customized Java EE Training: <http://courses.coreservlets.com/>

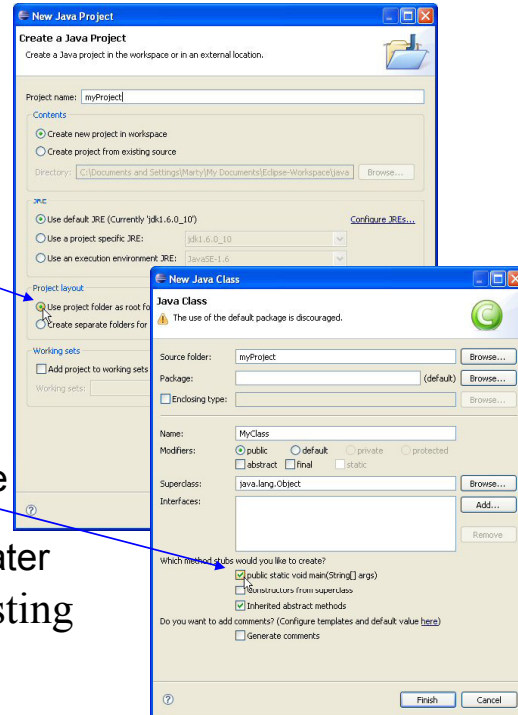
Servlets, JSP, JSF 2.0, Struts, Ajax, GWT 2.0, Spring, Hibernate, SOAP & RESTful Web Services, Java 6.

Developed and taught by well-known author and developer. At public venues or onsite at *your* location.

5

Eclipse: Making Projects

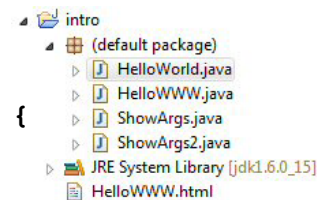
- **Creating new project**
 - File → New → Project → Java → Java Project
 - Pick any name
 - To simplify applets later, choose Sources/Classes in same folder
- **Creating new class**
 - R-click → New → Class
 - You can have Eclipse make “main” when class created, or use shortcut to insert it later
 - You can also copy/paste existing class, then give it new name



Getting Started: Syntax

- **Example**

```
public class HelloWorld {
    public static void main(String[] args) {
        System.out.println("Hello, world.");
    }
}
```
- **Details**
 - Processing starts in main
 - Eclipse can create main automatically
 - When creating class: choose main as option
 - Eclipse shortcut inside class: type “main” then hit Control-space
 - Routines usually called “methods,” not “functions.”
 - Printing is done with System.out.print...
 - System.out.println, System.out.print, System.out.printf
 - Eclipse shortcut: type “sysout” then hit Control-space



Getting Started: Execution

- **File: HelloWorld.java**

```
public class HelloWorld {  
    public static void main(String[] args) {  
        System.out.println("Hello, world.");  
    }  
}
```

- **Compiling**

- Eclipse: just save file

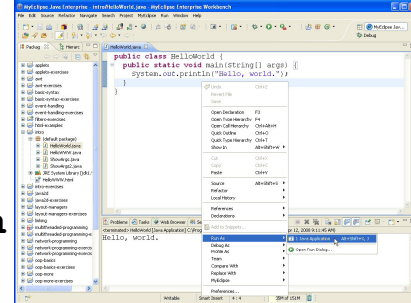
DOS> javac HelloWorld.java

- **Executing**

- Eclipse: R-click, Run As, Java Application

DOS> java HelloWorld

Hello, world.



8

More Basics

- **Use + for string concatenation**

- **Arrays are accessed with []**

- Array indices are zero-based

- The argument to **main** is an array of strings that correspond to the command line arguments

- args[0] returns first command-line argument
- args[1] returns second command-line argument, etc.
- Error if you try to access more args than were supplied

- **The length field**

- Gives the number of elements in *any* array

- Thus, `args.length` gives the number of command-line arguments
- Unlike in C/C++, the name of the program is not inserted into the command-line arguments

9

Command-line Arguments

- **Are useful for learning and testing**
 - Command-line args are helpful for practice
 - But, programs given to end users should almost never use command-line arguments
 - They should pop up a GUI to collect input.
- **Eclipse has poor support**
 - Entering command-line args via Eclipse is more trouble than it is worth
 - So, to test with command-line args:
 - Save the file in Eclipse (causing it to be compiled)
 - Navigate to folder on desktop (not within Eclipse)
 - Open command window (Run → cmd)
 - Type “java *Classname* arg1 arg2 ...”

Example

- **File: ShowTwoArgs.java (naïve version)**

```
public class ShowTwoArgs {
    public static void main(String[] args) {
        System.out.println("First arg: " +
            args[0]);
        System.out.println("Second arg: " +
            args[1]);
    }
}
```

Oops! Crashes if there are less than two command-line arguments. The code should have checked the length field, like this:

```
if (args.length > 1) {
    doThePrintStatements();
} else {
    giveAnErrorMessage();
}
```

Example (Continued)

- **Compiling (automatic on save in Eclipse)**

```
DOS> javac ShowTwoArgs.java
```

- **Executing**

```
DOS> java ShowTwoArgs Hello Class
```

```
First args Hello
```

```
Second arg: Class
```

```
DOS> java ShowTwoArgs
```

```
[Error message]
```

- **Eclipse (cumbersome)**

– To assign command line args: R-click, Run As, Run Configurations, click on “Arguments” tab

12

© 2010 Marty Hall



Loops

Customized Java EE Training: <http://courses.coreservlets.com/>

Servlets, JSP, JSF 2.0, Struts, Ajax, GWT 2.0, Spring, Hibernate, SOAP & RESTful Web Services, Java 6.

Developed and taught by well-known author and developer. At public venues or onsite at *your* location.

13

Looping Constructs

- **for/each**

```
for(variable: collection) {  
    body;  
}
```

- **for**

```
for(init; continueTest; updateOp) {  
    body;  
}
```

- **while**

```
while (continueTest) {  
    body;  
}
```

- **do**

```
do {  
    body;  
} while (continueTest);
```

14

For/Each Loops

```
public static void listEntries(String[] entries) {  
    for(String entry: entries) {  
        System.out.println(entry);  
    }  
}
```

- **Result**

```
String[] test = {"This", "is", "a", "test"};  
listEntries(test);
```

```
This  
is  
a  
test
```

15

For Loops

```
public static void listNums1(int max) {  
    for(int i=0; i<max; i++) {  
        System.out.println("Number: " + i);  
    }  
}
```

- **Result**

```
listNums1(4);
```

```
Number: 0
```

```
Number: 1
```

```
Number: 2
```

```
Number: 3
```

16

While Loops

```
public static void listNums2(int max) {  
    int i = 0;  
    while (i < max) {  
        System.out.println("Number: " + i);  
        i++; // "++" means "add one"  
    }  
}
```

- **Result**

```
listNums2(5);
```

```
Number: 0
```

```
Number: 1
```

```
Number: 2
```

```
Number: 3
```

```
Number: 4
```

17

Do Loops

```
public static void listNums3(int max) {
    int i = 0;
    do {
        System.out.println("Number: " + i);
        i++;
    } while (i < max);
        // ^ Don't forget semicolon
}
```

- **Result**

```
listNums3(3);
Number: 0
Number: 1
Number: 2
```

18

© 2010 Marty Hall



Class Structure and Formatting

Customized Java EE Training: <http://courses.coreservlets.com/>

Servlets, JSP, JSF 2.0, Struts, Ajax, GWT 2.0, Spring, Hibernate, SOAP & RESTful Web Services, Java 6.

Developed and taught by well-known author and developer. At public venues or onsite at *your* location.

19

Defining Multiple Methods in Single Class

```
public class LoopTest {
    public static void main(String[] args) {
        String[] test =
            { "This", "is", "a", "test"};
        listEntries(test);
        listNums1(5);
        listNums2(6);
        listNums3(7);
    }

    public static void listEntries(String[] entries) {...}
    public static void listNums1(int max) {...}
    public static void listNums2(int max) {...}
    public static void listNums3(int max) {...}
}
```

These methods say "static" because they are called directly from "main". In the next two sections on OOP, we will explain what "static" means and why most regular methods do not use "static". But for now, just note that methods that are directly called by "main" must say "static".

20

Indentation: blocks that are nested more should be indented more

- Yes

```
blah;
blah;
for(...) {
    blah;
    blah;
    for(...) {
        blah;
        blah;
    }
}
```

- No

```
blah;
blah;
for(...) {
    blah;
    blah;
    for(...) {
        blah;
        blah;
    }
}
```

21

Indentation: blocks that are nested the same should be indented the same

- Yes

```
blah;  
blah;  
for(...) {  
    blah;  
    blah;  
    for(...) {  
        blah;  
        blah;  
    }  
}
```

- No

```
blah;  
    blah;  
for(...) {  
    blah;  
    blah;  
    for(...) {  
        blah;  
        blah;  
    }  
}
```

22

Indentation: Number of spaces and placement of braces is a matter of taste

- OK

```
blah;  
blah;  
for(...) {  
    blah;  
    blah;  
    for(...) {  
        blah;  
        blah;  
    }  
}
```

- OK

```
blah;  
blah;  
for(...) {  
    blah;  
    blah;  
    for(...) {  
        blah;  
        blah;  
    }  
}
```

- OK

```
blah;  
blah;  
for(...)  
{  
    blah;  
    blah;  
    for(...)  
    {  
        blah;  
        blah;  
    }  
}
```

23



Conditionals and Strings

Customized Java EE Training: <http://courses.coreservlets.com/>
Servlets, JSP, JSF 2.0, Struts, Ajax, GWT 2.0, Spring, Hibernate, SOAP & RESTful Web Services, Java 6.
Developed and taught by well-known author and developer. At public venues or onsite at *your* location.

24

If Statements

- **Single Option**

```
if (boolean-expression) {  
    statement;  
}
```

- **Multiple Options**

```
if (boolean-expression) {  
    statement1;  
} else {  
    statement2;  
}
```

25

Boolean Operators

- **==, !=**
 - Equality, inequality. In addition to comparing primitive types, == tests if two objects are identical (the same object), not just if they appear equal (have the same fields). More details when we introduce objects.
- **<, <=, >, >=**
 - Numeric less than, less than or equal to, greater than, greater than or equal to.
- **&&, ||**
 - Logical AND, OR. Both use short-circuit evaluation to more efficiently compute the results of complicated expressions.
- **!**
 - Logical negation.

26

Example: If Statements

```
public static int max(int n1, int n2) {  
    if (n1 >= n2) {  
        return(n1);  
    } else {  
        return(n2);  
    }  
}
```

27

Strings

- **Basics**

- String is a real class in Java, not an array of characters as in C and C++.
- The String class has a shortcut method to create a new object: just use double quotes
 - This differs from normal objects, where you use the `new` construct to build an object

- **Use `equals` to compare strings**

- **Never use `==`**

- **Many useful builtin methods**

- `contains`, `startsWith`, `endsWith`, `indexOf`, `substring`, `split`, `replace`, `replaceAll`
 - Note: can use regular expressions, not just static strings
- `toUpperCase`, `toLowerCase`, `equalsIgnoreCase`

28

Common String Error: Comparing with `==`

```
public static void main(String[] args) {
    String match = "Test";
    if (args.length == 0) {
        System.out.println("No args");
    } else if (args[0] == match) {
        System.out.println("Match");
    } else {
        System.out.println("No match");
    }
}
```

- **Prints "No match" for *all* inputs**

- Fix:
`if (args[0].equals(match))`

29



Arrays

Customized Java EE Training: <http://courses.coreservlets.com/>
Servlets, JSP, JSF 2.0, Struts, Ajax, GWT 2.0, Spring, Hibernate, SOAP & RESTful Web Services, Java 6.
Developed and taught by well-known author and developer. At public venues or onsite at *your* location.

30

Building Arrays: One-Step Process

- Declare and allocate array in one fell swoop

```
type[] var = { val1, val2, ... , valN };
```

- Examples:

```
int[] values = { 10, 100, 1000 };  
String[] names = {"Joe", "Jane", "Juan"};  
Point[] points = { new Point(0, 0),  
                   new Point(1, 2),  
                   new Point(3, 4) };
```

31

Building Arrays: Two-Step Process

- **Step 1: allocate an array of references:**

`type[] var = new type[size];`

– E.g.:

`int[] primes = new int[7];`

`String[] names = new String[someArray.length];`

- **Step 2: populate the array**

`primes[0] = 2;`

`names[0] = "Joe";`

`primes[1] = 3;`

`names[1] = "Jane";`

`primes[2] = 5;`

`names[2] = "Juan";`

`primes[3] = 7;`

`names[3] = "John";`

etc.

- **If you fail to populate an entry**

– Default value is 0 for numeric arrays

– Default value is **nu11** for object arrays

32

Array Performance Problems

- **For very large arrays, undue paging can occur**

– Array of references (pointers) allocated first

– Individual objects allocated next

– Thus, for very large arrays of objects, reference and object can be on different pages, resulting in swapping for each array reference

– Example

```
String[] names = new String[10000000];
```

```
for(int i=0; i<names.length; i++) {
```

```
    names[i] = getNameFromSomewhere();
```

```
}
```

- **Problem does not occur with arrays of primitives**

– I.e., with arrays of **int**, **double**, and other types that start with lowercase letter

– Because system stores values directly in arrays, rather than storing references (pointers) to the objects

33

Multidimensional Arrays

- **Multidimensional arrays**

- Implemented as arrays of arrays

```
int[][] twoD = new int[64][32];

String[][] cats = {{ "Caesar", "blue-point" },
                  { "Heather", "seal-point" },
                  { "Ted",      "red-point"  }};
```

- **Note:**

- Number of elements in each row need not be equal

```
int[][] irregular = { { 1 },
                     { 2, 3, 4 },
                     { 5 },
                     { 6, 7 } };
```

34

TriangleArray: Example

```
public class TriangleArray {
    public static void main(String[] args) {

        int[][] triangle = new int[10][];

        for(int i=0; i<triangle.length; i++) {
            triangle[i] = new int[i+1];
        }

        for (int i=0; i<triangle.length; i++) {
            for(int j=0; j<triangle[i].length; j++) {
                System.out.print(triangle[i][j]);
            }
            System.out.println();
        }
    }
}
```

35

TriangleArray: Result

```
> java TriangleArray
```

```
0  
00  
000  
0000  
00000  
000000  
0000000  
00000000  
000000000  
0000000000
```

36

© 2010 Marty Hall



Math and Input

Customized Java EE Training: <http://courses.coreservlets.com/>

Servlets, JSP, JSF 2.0, Struts, Ajax, GWT 2.0, Spring, Hibernate, SOAP & RESTful Web Services, Java 6.

Developed and taught by well-known author and developer. At public venues or onsite at *your* location.

37

Basic Mathematical Routines

- **Very simplest routines use builtin operators**
 - +, -, *, /, ^, %
 - Be careful with / on `int` and `long` variables
- **Static methods in the Math class**
 - So you call `Math.cos(...)`, `Math.random()`, etc.
 - Most operate on double precision floating point numbers
 - Simple operations: `Math.pow()`, etc.
 - `pow(xy)`, `sqrt(√x)`, `cbt`, `exp(ex)`, `log(loge)`, `log10`
 - Trig functions: `Math.sin()`, etc.
 - `sin`, `cos`, `tan`, `asin`, `acos`, `atan`
 - Args are in radians, not degrees, (see `toDegrees` and `toRadians`)
 - Rounding and comparison: `Math.round()`, etc.
 - `round/rint`, `floor`, `ceiling`, `abs`, `min`, `max`
 - Random numbers: `Math.random()`
 - `random` (`Math.random()` returns from 0 inclusive to 1 exclusive).
 - See `Random` class for more control over randomization.

38

More Mathematical Routines

- **Special constants**
 - `Double.POSITIVE_INFINITY`
 - `Double.NEGATIVE_INFINITY`
 - `Double.NaN`
 - `Double.MAX_VALUE`
 - `Double.MIN_VALUE`
- **Unlimited precision libraries**
 - `BigInteger`, `BigDecimal`
 - Contain the basic operations, plus `BigInteger` has `isPrime`

39

Reading Simple Input

- **For simple testing, use standard input**
 - If you want strings, just use `args[0]`, `args[1]`, as before
 - To avoid errors, check `args.length` first
 - Convert if you want numbers. Two main options:
 - Use Scanner class
 - Note that you need import statement. See next slide!
`Scanner inputScanner = new Scanner(System.in);`
`int i = inputScanner.nextInt();`
`double d = inputScanner.nextDouble();`
 - Convert explicitly (`Integer.parseInt`, `Double.parseDouble`)
`String seven = "7";`
`int i = Integer.parseInt(seven);`
- **In real applications, use a GUI**
 - Collect input with textfields, sliders, combo boxes, etc.
 - Convert to numeric types with `Integer.parseInt`, `Double.parseDouble`, etc.

40

Example: Printing Random Numbers

```
import java.util.*;

public class RandomNums {
    public static void main(String[] args) {
        System.out.print("How many random nums? ");
        Scanner inputScanner = new Scanner(System.in);
        int n = inputScanner.nextInt();
        for(int i=0; i<n; i++) {
            System.out.println("Random num " + i +
                               " is " + Math.random());
        }
    }
}
```

```
How many random nums? 5
Random num 0 is 0.22686369670835704
Random num 1 is 0.0783768527137797
Random num 2 is 0.17918121951887145
Random num 3 is 0.3441924454634313
Random num 4 is 0.6131053203170818
```

41



Wrap-Up

Customized Java EE Training: <http://courses.coreservlets.com/>
 Servlets, JSP, JSF 2.0, Struts, Ajax, GWT 2.0, Spring, Hibernate, SOAP & RESTful Web Services, Java 6.
 Developed and taught by well-known author and developer. At public venues or onsite at *your* location.

42

Summary

- **Basics**
 - Loops, conditional statements, and array access is similar to C and C++
 - But new for loop: `for(String s: someStrings) { ... }`
 - Indent your code for readability
 - `String` is a real class in Java
 - Use `equals`, not `==`, to compare strings
- **Allocate arrays in one step or in two steps**
 - If two steps, loop down array and supply values
- **Use `Math.blah()` for simple math operations**
- **Simple input from command window**
 - Use command line for strings supplied at program startup
 - Use `Scanner` to read values after prompts
 - Neither is very important for most real-life applications

43



Questions?

Customized Java EE Training: <http://courses.coreservlets.com/>

Servlets, JSP, JSF 2.0, Struts, Ajax, GWT 2.0, Spring, Hibernate, SOAP & RESTful Web Services, Java 6.
Developed and taught by well-known author and developer. At public venues or onsite at *your* location.