



Basic Java Syntax

Originals of slides and source code for examples: <http://courses.coreservlets.com/Course-Materials/java.html>
 Also see Java 8 tutorial: <http://www.coreservlets.com/java-8-tutorial/> and many other Java EE tutorials: <http://www.coreservlets.com/>
 Customized Java training courses (onsite or at public venues): <http://courses.coreservlets.com/java-training.html>



2

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

Java 7, Java 8, JSF 2, PrimeFaces, Android, JSP, Ajax, jQuery, Spring MVC, RESTful Web Services, GWT, Hadoop
 Developed and taught by well-known author and developer. At public venues or onsite at *your* location.



**For customized Java-related training at
your organization, email hall@coreservlets.com**
Marty is also available for consulting and development support



Taught by lead author of *Core Servlets & JSP*, co-author of *Core JSF* (4th Ed), & this tutorial. Available at public venues, or customized versions can be held on-site at your organization.

- Courses developed and taught by Marty Hall
 - JSF 2.2, PrimeFaces, servlets/JSP, Ajax, JavaScript, jQuery, Android, Java 7 or 8 programming, GWT, custom mix of topics
 - Courses available in any state or country. Maryland/DC area companies can also choose afternoon/evening courses.
- Courses developed and taught by coreservlets.com experts (edited by Marty)
 - Spring MVC, Core Spring, Hibernate/JPA, Hadoop, HTML5, RESTful Web Services

Contact hall@coreservlets.com for details



Topics in This Section

- Basics
- Accessing arrays
- Looping
- Indenting code
- if statements and other conditionals
- Strings
- Building arrays
- Performing basic mathematical operations
- Reading command-line input
- Converting strings to numbers

4

© 2015 Marty Hall



Setup, Execution, and Most Basic Syntax



5

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

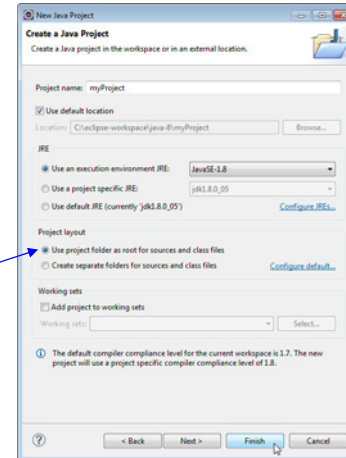
Java 7, Java 8, JSF 2, PrimeFaces, Android, JSP, Ajax, jQuery, Spring MVC, RESTful Web Services, GWT, Hadoop

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

Eclipse: Making Projects

- **Main steps**

- File → New → Project → Java → Java Project
 - Pick any name
- If you will ever use applets
 - Choose sources/classes in project folder



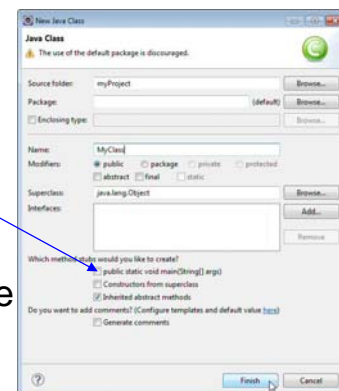
Eclipse: Creating Classes

- **Main steps**

- R-click on project → New → Class
 - You can have Eclipse make “main” when class created, but easier to use shortcut to insert it later
 - Eventually you will make package (subdirectory) first, then put class there

- **Alternative**

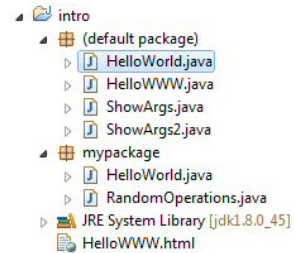
- Can also copy/rename existing class



Getting Started: Syntax

- **Example**

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



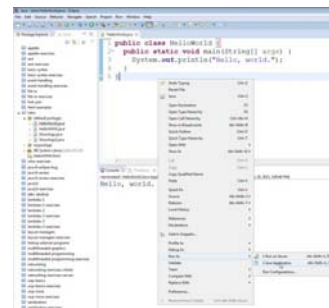

- **Details**

- 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
 - > javac HelloWorld.java

- **Executing**

- Eclipse: R-click, Run As, Java Application
 - > java HelloWorld
 - Hello, world.

Packages

- **Idea**

- Packages are subdirectories used to avoid name conflicts
- Java class must have “package subdirname;” at the top
 - But Eclipse puts this in automatically when you right-click on a package and use New → Class

- **Naming conventions**

- Package names are in all lower case
- Some organizations use highly nested names
 - com.companyname.projectname.projectcomponent

- **Creating packages in Eclipse**

- R-click project, New → Package
- Then R-click package and New → Class

10

HelloWorld with Packages (in src/mypackage folder)

```
package mypackage;
```

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

Run from Eclipse in normal manner: R-click, Run As → Java Application. Running from the command line is a pain: you must go to *parent* directory and do “java mypackage.HelloWorld”. Run from Eclipse and it is simple to use packages.

11



The + Operator, Array Basics, Command Line Args



12

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

Java 7, Java 8, JSF 2, PrimeFaces, Android, JSP, Ajax, jQuery, Spring MVC, RESTful Web Services, GWT, Hadoop
Developed and taught by well-known author and developer. At public venues or onsite at *your* location.

The + Operator

- **Use + for addition**

- If both arguments are numbers, + means addition.
- Example:

```
double result = 2.3 + 4.5;
```

- **Use + for string concatenation**

- If either argument is String, + means concatenation. + is *only* overloaded operator in all of Java.
- Examples

```
String result1 = "Hello, " + "World";  
String result2 = "Number " + 5;
```

Array Basics: Accessing Elements

- **Arrays are accessed with []**
 - Array indices are zero-based

```
int[] nums = { 2, 4, 6, 8 }
```

 - `nums[0]` is 2
 - `nums[3]` is 8
 - Trying to access `nums[4]` results in error
- **Main is passed an array of strings**
 - `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

14

Array Basics: The length Field

- **The length variable tells you number of array elements**
 - Gives the number of elements in *any* array

```
String[] names = { "John", "Jane", "Juan" };
```

 - `names.length` is 3
 - But last entry ("Juan") is `names[2]`, not `names[3]`
 - For command-line arguments
 - In main, `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

15

Command-line Arguments

- **Are useful for learning and testing**
 - Command-line args are helpful for beginner's 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 (Start icon, Run... → cmd)
 - Type "java *Classname* arg1 arg2 ..."

Example: Command Line Args and the length Field

- **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 fewer 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)**
`> javac ShowTwoArgs.java`
- **Manual execution**
`> java ShowTwoArgs Hello Class`
First arg: Hello
Second arg: Class

`> java ShowTwoArgs`
[Error message]
- **Eclipse execution (cumbersome)**
 - To assign command line args: R-click, Run As, Run Configurations, click on “Arguments” tab

18

© 2015 Marty Hall



Loops



19

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

Java 7, Java 8, JSF 2, PrimeFaces, Android, JSP, Ajax, jQuery, Spring MVC, RESTful Web Services, GWT, Hadoop

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

Looping Constructs

- **for/each**

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

- **for**

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

- **while**

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

- **do**

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

20

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
```

21

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
```

22

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
```

23

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
```

24

© 2015 [Marty Hall](#)



Class Structure and Formatting



25

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

Java 7, Java 8, JSF 2, PrimeFaces, Android, JSP, Ajax, jQuery, Spring MVC, RESTful Web Services, GWT, Hadoop

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

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);  
    }  
}
```

These methods say "static" because they are called directly from "main". In the upcoming 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".

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

26

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;  
    }  
}
```

27

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;  
    }  
}
```

28

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;  
    }  
}
```

29



Conditionals



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

Java 7, Java 8, JSF 2, PrimeFaces, Android, JSP, Ajax, jQuery, Spring MVC, RESTful Web Services, GWT, Hadoop
Developed and taught by well-known author and developer. At public venues or onsite at *your* location.

If Statements: One or Two Options

- **Single option**

```
if (booleanExpression) {  
    statement1;  
    ...  
    statementN;  
}
```

The value inside parens must be strictly boolean (i.e., true or false), unlike C, C++, and JavaScript.

A widely accepted best practice is to use the braces even if there is only a single statement inside the if or else.

- **Two options**

```
if (booleanExpression) {  
    ...  
} else {  
    ...  
}
```

If Statements: More than Two Options

- **Multiple options**

```
if (booleanExpression1) {  
    ...  
} else if (booleanExpression2) {  
    ...  
} else if (booleanExpression3) {  
    ...  
} else {  
    ...  
}
```

32

Switch Statements

- **Example**

```
int month = ...;  
String monthString;  
switch(month) {  
    case 1: monthString = "January"; break;  
    case 2: monthString = "February"; break;  
    case 3: monthString = "March"; break;  
    ...  
    default: monthString = "Invalid month"; break;  
}
```

- **Syntax is mostly like C and C++**

- Types can be primitives, enums, and (Java 7+) Strings

33

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.

34

Example: If Statements

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

35



Strings



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

Java 7, Java 8, JSF 2, PrimeFaces, Android, JSP, Ajax, jQuery, Spring MVC, RESTful Web Services, GWT, Hadoop
Developed and taught by well-known author and developer. At public venues or onsite at *your* location.

36

Strings: Basics

- **Overview**

- String is a real class in Java, not an array of characters as in C++
- The String class has a shortcut method to create a new object: just use double quotes

```
String s = "Hello";
```

- Differs from normal classes, where you use **new** to build an object

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

- Never use `==` to test if two Strings have same characters!

37

Using == to Compare Strings (Wrong!)

```
public class CheckName1 {  
    public static void main(String[] args) {  
        if (args.length == 0) {  
            System.out.println("Nobody");  
        } else if (args[0] == "Marty") {  
            System.out.println("Hi, Marty");  
        } else {  
            System.out.println("Hi, stranger");  
        }  
    }  
}
```

This always prints "Hi, stranger", even if the first command line argument is "Marty".

Using equals to Compare Strings (Right!)

```
public class CheckName2 {  
    public static void main(String[] args) {  
        if (args.length == 0) {  
            System.out.println("Nobody");  
        } else if (args[0].equals("Marty")) {  
            System.out.println("Hi, Marty");  
        } else {  
            System.out.println("Hi, stranger");  
        }  
    }  
}
```

Strings: Methods

- **Methods to call *on* a String**

- contains, startsWith, endsWith, indexOf, substring, split, replace, replaceAll, toUpperCase, toLowerCase, equalsIgnoreCase, trim, isEmpty, etc.
 - For replacing, can use regular expressions, not just static strings
- Example

```
String word = "...";  
if (word.contains("q")) { ... }
```

- **Static methods in String class**

- String.format, String.join, String.valueOf, etc.
- Example

```
String numberAsString = String.valueOf(17);
```

40

© 2015 Marty Hall



More on Arrays



41

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

Java 7, Java 8, JSF 2, PrimeFaces, Android, JSP, Ajax, jQuery, Spring MVC, RESTful Web Services, GWT, Hadoop

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

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) };
```

42

Building Arrays: Two-Step Process

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

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

– E.g.:

```
int[] primes = new int[x];    // x is positive integer
```

```
String[] names = new String[someVariable]; // Positive int
```

- **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. (or use a loop)
```

```
etc. (or use a loop)
```

43

Default Array Values

- If you fail to populate an entry

- Default value is 0 for numeric arrays

```
int[] nums = new int[7];  
int value = 3 + nums[2];    // value is 3
```

- Default value is **null** for Object arrays

```
String[] words = new String[7];  
System.out.println(words[2]); // Prints null  
if (words[3].contains("q") { ... }  
    // Crashes with NullPointerException
```

44

Two-Step Process: Example 1

```
public static Circle[] makeCircles1(int numCircles) {  
    Circle[] circles = new Circle[numCircles];  
    // Empty array of proper size  
    for(int i=0; i<circles.length; i++) {  
        circles[i] = new Circle(Math.random() * 10);  
        // Populate array  
    }  
    return(circles);  
}
```

This approach is correct!

45

Two-Step Process: Example 2

```
public static Circle[] makeCircles2(int numCircles) {  
    Circle[] circles = new Circle[numCircles];  
    // Empty array of proper size  
    for(int i=0; i<circles.length; i++) {  
        circles[i].setRadius(Math.random() * 10);  
        // NullPointerException  
    }  
    return(circles);  
}
```

This approach fails: the call to `setRadius` crashes with `NullPointerException` because `circles[i]` is null.

46

Two-Step Process: Example 3

```
public static Circle[] makeCircles3(int numCircles) {  
    Circle[] circles = new Circle[numCircles];  
    for(Circle c: circles) {  
        c = new Circle(Math.random() * 10);  
        // Fails to store c in array  
    }  
    return(circles);  
    // Array still contains only null pointers  
}
```

This approach fails: array is still empty after the loop.

47

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 } };
```

49

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();
        }
    }
}
```

50

TriangleArray: Result

```
> java TriangleArray
```

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

51

© 2015 Marty Hall



Math Routines



52

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

Java 7, Java 8, JSF 2, PrimeFaces, Android, JSP, Ajax, jQuery, Spring MVC, RESTful Web Services, GWT, Hadoop

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

Basic Mathematical Operators

- **+, -, *, /, %**
 - Addition, subtraction, multiplication, division, mod
 - mod means remainder, so `3 % 2` is 1.
- **num++, ++num**
 - Means add one to (after/before returning value)

```
int num = 3;  
num++;  
// num is now 4
```
 - Usage
 - For brevity and tradition, but no performance benefit over simple addition
- **Warning**
 - Be careful with / on **int** and **long** variables (rounds off)

53

Basic Mathematical Methods: Usage

- **Static methods in the Math class**
 - So you call `Math.cos(...)`, `Math.random()`, etc.
 - Most operate on double-precision floating point numbers
 - Examples

```
double eight = Math.pow(2, 3);  
double almostZero = Math.sin(Math.PI);  
double randomNum = Math.random();
```

54

Basic Mathematical Methods: Most Common Methods

- **Simple operations: Math.pow(), etc.**
 - pow (x^y), sqrt (\sqrt{x}), cbrt, exp (e^x), log (\log_e), 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()**
 - Math.random() returns double from 0 inclusive to 1 exclusive.
 - See Random class for more control over randomization.

55

Common Use of Math.random

- **To randomly invoke different operations**
 - Especially useful for testing

```
for(int i=0; i<10; i++) {  
    if (Math.random() < 0.5) {  
        doFirstOperation(); // 50% chance  
    } else {  
        doSecondOperation(); // 50% chance  
    }  
}
```

56

More Mathematical Options

- **Special constants**
 - Double.POSITIVE_INFINITY
 - Double.NEGATIVE_INFINITY
 - Double.NaN
 - Double.MAX_VALUE
 - Double.MIN_VALUE
- **Unlimited precision libraries**
 - BigInteger, BigDecimal
 - Contain basic math operations like add, pow, mod, etc.
 - BigInteger also has isPrime

57

© 2015 Marty Hall



Reading Input



58

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

Java 7, Java 8, JSF 2, PrimeFaces, Android, JSP, Ajax, jQuery, Spring MVC, RESTful Web Services, GWT, Hadoop

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

Reading Strings

- **Option 1: use command-line argument**
 - `String input = args[0];`
- **Option 2: use JOptionPane**
 - `String input =
JOptionPane.showInputDialog("Number:");`
- **Option 3: use Scanner**
 - `Scanner inputScanner = new Scanner(System.in);`
 - `String input = inputScanner.next();`
- **Warning**
 - In real life, you must check

59

Converting Strings to Numbers

- **To int: Integer.parseInt**

```
String input = ...;  
int num = Integer.parseInt(input);
```
- **To double: Double.parseDouble**

```
String input = ...;  
double num = Double.parseDouble(input);
```
- **With Scanner**
 - Use `scanner.nextInt()`, `scanner.nextDouble()`
- **Warning**
 - In real life, you must handle the case where the input is not a legal number.
 - Idea shown without explanation in `Input1Alt` class
 - Try/catch blocks and exception handling covered in detail in the section on applets and basic drawing

60

Command-Line Args

```
public class Input1 {  
    public static void main(String[] args) {  
        if (args.length > 1) {  
            int num = Integer.parseInt(args[0]);  
            System.out.println("Your number is " + num);  
        } else {  
            System.out.println("No command-line args");  
        }  
    }  
}
```

Open command window and navigate to folder containing class

```
> java Input1 7  
Your number is 7
```

61

Error Checking (Explained in Applets Section)

```
public class Input1Alt {  
    public static void main(String[] args) {  
        if (args.length > 1) {  
            try {  
                int num = Integer.parseInt(args[0]);  
                System.out.println("Your number is " + num);  
            } catch (NumberFormatException e) {  
                System.out.println("Input is not a number");  
            }  
        } else {  
            System.out.println("No command-line arguments");  
        }  
    }  
}
```

Open command window and navigate to folder containing class

```
> java Input1Alt seven  
Input is not a number  
> java Input1Alt 7  
Your number is 7
```

62

JOptionPane

```
import javax.swing.*;

public class Input2 {
    public static void main(String[] args) {
        String input =
            JOptionPane.showInputDialog("Number:");
        int num = Integer.parseInt(input);
        System.out.println("Your number is " + num);
    }
}
```

Run from Eclipse (R-click, Run As → Java Application),
enter 8 in popup window
Result in Eclipse Console:
Your number is 8

63

Scanner

```
import java.util.*;

public class Input3 {
    public static void main(String[] args) {
        System.out.print("Number: ");
        Scanner inputScanner = new Scanner(System.in);
        int num = inputScanner.nextInt();
        System.out.println("Your number is " + num);
    }
}
```

Run from Eclipse (R-click, Run As → Java Application),
enter 9 after "Number:" prompt in Eclipse Console. Next line:
Your number is 9

64

In Real Applications, use GUI

- **For practice, use approaches just shown**
 - Command line args, JOptionPane, Scanner
- **Desktop and phone apps**
 - Collect input within Java app using textfields, sliders, dropdown menus, etc.
 - Convert to numeric types with Integer.parseInt, Double.parseDouble, etc.
- **Web apps (JSF2 and PrimeFaces)**
 - Collect input on browser with textfields, sliders, popup calendars, etc.
 - Java will convert automatically for simple types
 - You can set up converters for complex types
 - Details: <http://www.coreservlets.com/JSF-Tutorial/jsf2/>

65

© 2015 Marty Hall



Wrap-Up



66

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

Java 7, Java 8, JSF 2, PrimeFaces, Android, JSP, Ajax, jQuery, Spring MVC, RESTful Web Services, GWT, Hadoop

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

Summary

- **Basics**
 - Loops, conditional statements, and array access is similar to C/C++
 - But additional “for each” 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**
 - `Math.random`, `Math.sin`, `Math.cos`, `Math.pow`, etc.
- **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

67

© 2015 Marty Hall



Questions?

More info:

<http://courses.coreservlets.com/Course-Materials/java.html> – General Java programming tutorial

<http://www.coreservlets.com/java-8-tutorial/> – Java 8 tutorial

<http://courses.coreservlets.com/java-training.html> – Customized Java training courses, at public venues or onsite at your organization

<http://coreservlets.com/> – JSF 2, PrimeFaces, Java 7 or 8, Ajax, jQuery, Hadoop, RESTful Web Services, Android, HTML5, Spring, Hibernate, Servlets, JSP, GWT, and other Java EE training

Many additional free tutorials at coreservlets.com (JSF, Android, Ajax, Hadoop, and lots more)



68

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

Java 7, Java 8, JSF 2, PrimeFaces, Android, JSP, Ajax, jQuery, Spring MVC, RESTful Web Services, GWT, Hadoop

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