Agenda Key: 31CD
Session Number: 480063

Java™ 101: Basic Syntax and Structure

Jeff Lee

Java related sessions

Basic Java

- Mon 9:30 - 22MD A Java Introduction to Object-Oriented Programming (OOP)
- Mon 3:30 - 26MJ Debugging the New Java ***New Session***
- **Tues 8:00 - 31CD** Java 101: Basic Syntax and Structure
- Tues 11:00 - 33LA LAB: Introduction to Java ***LAB***
- Web 9:30 - 42CD The Future of Java on IBM i

Java Toolbox

- Tues 2:00 - 35CB Introducing the IBM Toolbox for Java
- Wed 8:00 - 41LA LAB: IBM Toolbox for Java ***LAB***
- Thur 9:30 - 52CC IBM Toolbox for Java: Advanced

Advanced Java related topics

- Mon 11:00 - 23MH Introduction to XML Processing with Java
- Tues 3:30 - 36MG Java Application Performance Analysis and Tuning on IBM i
- Wed 2:00 - 45CD Using the JVM Tools Interface (JVMTI)
- Thur 12:30 - 54CB Multi-Threaded Programming Using Java
- Thur 2:00 - 55MH Java Stored Procedures and Java User-Defined Functions

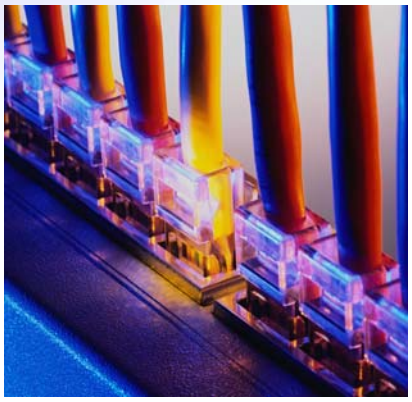
Outline

- Introduction
- Why Java?
- Object-Oriented Overview
- Java Definitions
- Elementary Java Structure
- Java Syntax and Control Flow
- Tips for Approaching Java Code
- Tools for Java Development

Introduction

- Goals
  - Introduce basic Java syntax elements, compared with RPG
  - Develop skills for reading and understanding Java source code
- Expand skills in writing Java code
  - Get you understanding Java code syntax
  - Help you find different ways of looking at code
- How to get there
  - Look at Java code
  - Help you understand it

Why Java? It's simple.



- Most popular programming language
- Distributed
- Robust
- Multithreaded
- Write once, run anywhere
- Internationalization

The New Power Equation

OO in 5 Minutes or Less

- Class
  - A pattern, template, or model for an object
- Object
  - An **instance** of a class
  - A combination of **data** and **functions (behavior)**
- Fields
  - The **data** in an object (attributes, characteristics)
- Methods
  - The **functions** in an object (procedures, subroutines)

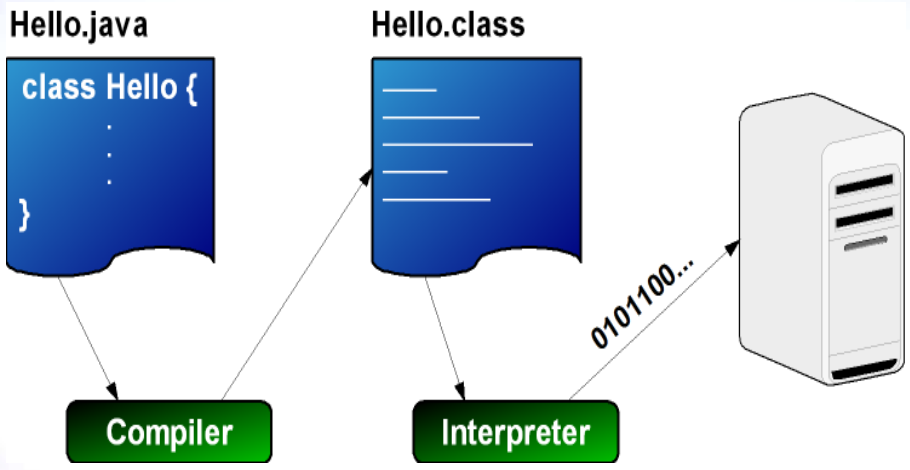
The New Power Equation

© 2009 IBM Corporation

Java Definitions

- Class file
  - Result of compiling Java **source code** into **byte codes**
- Jar File
  - Java archive; a **collection** of class files and other resources
- Java Virtual Machine (JVM)
  - Runtime environment that converts **byte codes** to **machine code**
- Classpath
  - Path that the JVM searches for classes and other resources
- Package
  - Collection of related classes providing access **protection** and namespace management

Life of a Java program



Example Java program

```
import javax.swing.*;

class Hello
{
  public static void main( String [] args )
  {
    JOptionPane.showMessageDialog( new JFrame(),
        "Hello, world!",
        "Example Java Program",
        JOptionPane.PLAIN_MESSAGE);
    System.exit(0);
  }
}
```

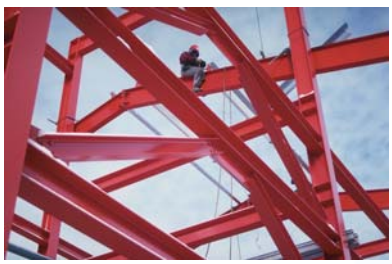
- To compile Java source: javac Hello.java
- To run Java program: java Hello

Elementary Java Structure

- Dot-delimited class name implies underlying **subdirectory structure**

```
java.awt.Button => java/awt/Button.class
com.ibm.as400.system.Hello => com/ibm/as400/system/Hello.class
```

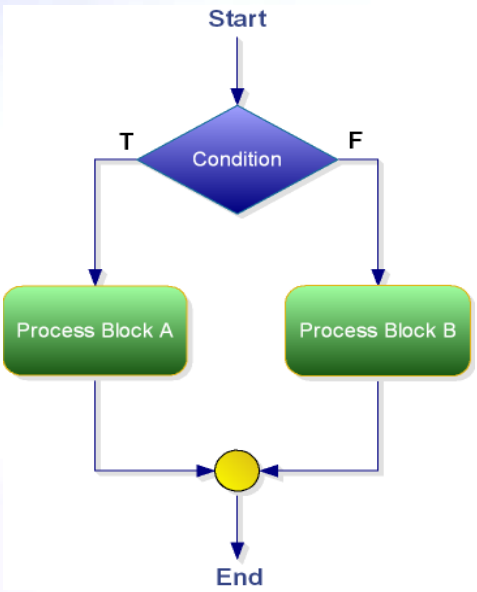
- Several common styles of writing Java code
- Compiler does NOT CARE how you indent, but YOU should!
  - Whitespace is ignored
  - Make code readable



Control Flow Constructs

- Conditional
  - Branches and loops
- Exceptional
  - try, catch, throw
- Unconditional
  - Method calls

Conditional Control Flow: if/else



- `if/else`
  - executes block if expression evaluates to “true”

Conditional Control Flow: if/else

Java Syntax

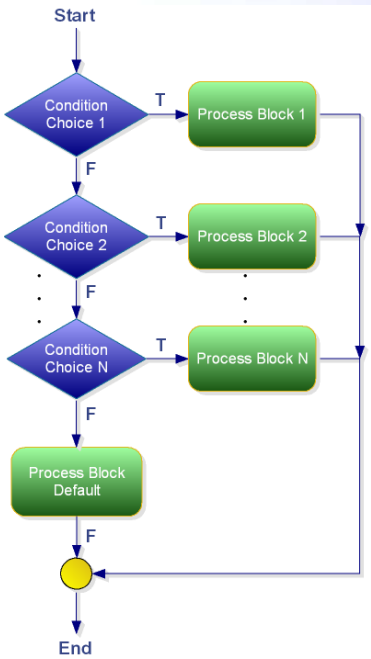
```
if (nameLength > 8) {
  truncate = true;
}
else {
  truncate = false;
}
```

RPG Syntax

```
if nameLength > 8;
   truncate = *ON;
else;
   truncate = *OFF;
endif;
```


Conditional Control Flow: switch

- **`switch`**
  - An easy-to-read collection of if statements
  - Must use the **`break`** keyword to transfer control to just after the switch statement
  - Argument to **`switch()`** must be scalar value (i.e. integer or character)



Conditional Control Flow: switch

Java Syntax

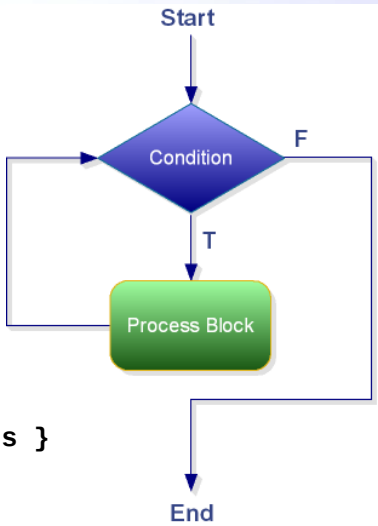
```
switch (status) {
  case 1:
    System.out.println ("Error.");
    break;
  case 2:
    System.out.println ("End of File.");
    break;
  default:
    System.out.println ("Success!");
    break;
}
```

RPG Syntax

```
select;
 when status = 1;
   dsply 'Error.';
 when status = 2;
   dsply 'End of File.';
 other;
   dsply 'Success!';
endsl;
```


Conditional Control Flow: Loops

- **while**
  - while (condition) { statements }
- **do … while**
  - do { statements } while (condition) ;
- **for**
  - for (init; condition; incr) { statements }



Conditional Control Flow: Loops

Java Syntax

```
while (!eof(file)) {
  processRecord();
}

do {
  processRecord();
} while(x > array.length);

for (i = start; i < end;
    i += inc) {
  processRecord();
}
```

RPG Syntax

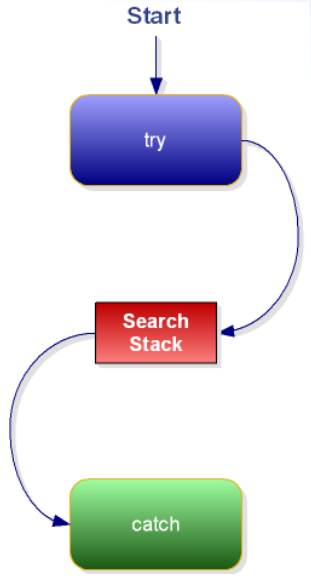
```
dow not %eof(file);
  processRecord();
enddo;

dou x > %elem(Array);
  processRecord();
enddo;

for i = start by inc
    to %elem(array);
  processRecord();
endfor;
```


Exceptional Control Flow

- **`try/catch`**
  - Errors are propagated up the stack
  - Hint: Always list catch blocks from most specific to most general
  - The **finally** statement is always executed, even if errors
  - Must re-throw exception using the **throw** keyword if not handled



Exceptional Control Flow

Java Syntax

```
try {
   // code that might 'throw'
} catch(FileException e1) {
   // handle file error
} catch(Exception e2) {
   // handle all other errors
} finally {
   // ALWAYS do…
}
```

RPG Syntax

```
MONITOR
   // code that might 'throw'
ON-ERROR *FILE
   // handle file error
ON-ERROR
   // handle all other errors
ENDMOD
```


Unconditional Control Flow: Method Calls

```
import javax.swing.*;

class MyApplication {
  public static void main( String [] args ) {
     String question = "why \"i\"?";
     String answer = showDialog("Question!", question);
     System.out.println( question + " " + answer );
     System.exit(0);
  }

  static String showDialog(String title, String message) {
    String out = JOptionPane.showInputDialog(
         new JFrame(),
         message,
         title,
         JOptionPane.PLAIN_MESSAGE);
    return out;
  }
}
```

Tips for Approaching Java Code

- Check packaging
  - Jar or class
  - Does it include the source?
- API or Application
  - If it is an API, evaluate the interface
  - If it is an Application, look for main and run it
- Look at documentation
  - Is there a javadoc?

Comment Styles: Javadoc

```
package java.lang;

/**
 * Class <code>Object</code> is the root of the class hierarchy.
 * Every class has <code>Object</code> as a superclass. All objects,
 * including arrays, implement the methods of this class.
 *
 * @author  unascribed
 * @version 1.68, 04/08/04
 * @see     java.lang.Class
 * @since   JDK1.0
 **/

public class Object {
    . . .
```

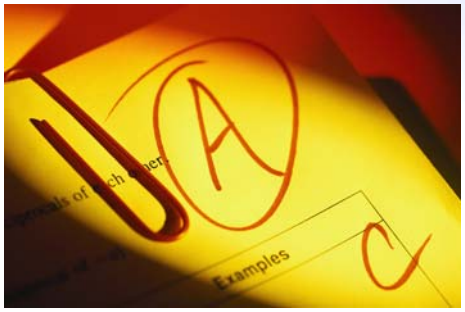
Javadoc

Tools for Java Development

- Development Environments and Editors
  - Eclipse
  - WebSphere Studio Application Developer
  - Rational Application Developer (RAD)
  - jEdit
  - Notepad++
- Modeling
  - Unified Modeling Language (UML)
  - Rational Rose XDE
- Decompilers
  - DJ Java Decompiler
  - jShrink

Summary

- Scratching the surface…
  - Why Java?
  - Terminology
  - Java Structure and Syntax
  - Development Tools and Tips
- Java Resources
  - Sun's Java Website (java.sun.com)
  - IBM developerWorks
  - IBM Toolbox for Java



References

- All things Java: java.sun.com
- developerWorks: www.ibm.com/developerworks
- IBM Toolbox for Java: www.ibm.com/systems/i/software/toolbox
- Eclipse: www.eclipse.org
- WebSphere: www.ibm.com/software/websphere
- jEdit: www.jedit.org
- Notepad++: sourceforge.net/projects/notepad-plus
- FindBugs: findbugs.sourceforge.net
- UML: www.uml.org
- Rational software: www.ibm.com/software/rational
- DJ Java Decompiler: members.fortunecity.com/neshkov/dj.html
- Jshrink: www.e-t.com/jshrink.html

Questions

Java related sessions

Basic Java

Mon 9:30 - 22MD A Java Introduction to Object-Oriented Programming (OOP)

Mon 3:30 - 26MJ Debugging the New Java ***New Session***

Tues 8:00 - 31CD Java 101: Basic Syntax and Structure

Tues 11:00 - 33LA LAB: Introduction to Java ***LAB***

Web 9:30 - 42CD The Future of Java on IBM i

Java Toolbox

Tues 2:00 - 35CB Introducing the IBM Toolbox for Java

Wed 8:00 - 41LA LAB: IBM Toolbox for Java ***LAB***

Thur 9:30 - 52CC IBM Toolbox for Java: Advanced

Advanced Java related topics

Mon 11:00 - 23MH Introduction to XML Processing with Java

Tues 3:30 - 36MG Java Application Performance Analysis and Tuning on IBM i

Wed 2:00 - 45CD Using the JVM Tools Interface (JVMTI)

Thur 12:30 - 54CB Multi-Threaded Programming Using Java

Thur 2:00 - 55MH Java Stored Procedures and Java User-Defined Functions

Backup Slides

Programming Languages Popularity

Position	Programming Language	Ratings
1	Java	19.797%
2	C	15.862%
3	C++	10.357%
4	PHP	9.485%
5	(Visual) Basic	8.285%
6	Python	5.185%
7	C#	4.321%
8	JavaScript	3.607%
9	Perl	3.419%
10	Delphi	2.710%

Source: TIOBE Programming Community Index for March 2009 - http://www.tiobe.com/

Trademarks and Disclaimers

References in this document to IBM products or services do not imply that IBM intends to make them available in every country.
Trademarks of International Business Machines Corporation in the United States, other countries, or both can be found on the World Wide Web at http://www.ibm.com/legal/copytrade.shtml.

Adobe, Acrobat, PostScript and all Adobe-based trademarks are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States, other countries, or both.

Intel, Intel logo, Intel Inside, Intel Inside logo, Intel Centrino, Intel Centrino logo, Celeron, Intel Xeon, Intel SpeedStep, Itanium, and Pentium are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

IT Infrastructure Library is a registered trademark of the Central Computer and Telecommunications Agency which is now part of the Office of Government Commerce.

ITIL is a registered trademark, and a registered community trademark of the Office of Government Commerce, and is registered in the U.S. Patent and Trademark Office.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Cell Broadband Engine and Cell/B.E. are trademarks of Sony Computer Entertainment, Inc., in the United States, other countries, or both and are used under license therefrom.

Java and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Other company, product, or service names may be trademarks or service marks of others.

Information is provided "AS IS" without warranty of any kind.

The customer examples described are presented as illustrations of how those customers have used IBM products and the results they may have achieved. Actual environmental costs and performance characteristics may vary by customer.

Information concerning non-IBM products was obtained from a supplier of these products, published announcement material, or other publicly available sources and does not constitute an endorsement of such products by IBM. Sources for non-IBM list prices and performance numbers are taken from publicly available information, including vendor announcements and vendor worldwide homepages. IBM has not tested these products and cannot confirm the accuracy of performance, capability, or any other claims related to non-IBM products. Questions on the capability of non-IBM products should be addressed to the supplier of those products.

All statements regarding IBM future direction and intent are subject to change or withdrawal without notice, and represent goals and objectives only.

Some information addresses anticipated future capabilities. Such information is not intended as a definitive statement of a commitment to specific levels of performance, function or delivery schedules with respect to any future products. Such commitments are only made in IBM product announcements. The information is presented here to communicate IBM's current investment and development activities as a good faith effort to help with our customers' future planning.

Performance is based on measurements and projections using standard IBM benchmarks in a controlled environment. The actual throughput or performance that any user will experience will vary depending upon considerations such as the amount of multiprogramming in the user's job stream, the I/O configuration, the storage configuration, and the workload processed. Therefore, no assurance can be given that an individual user will achieve throughput or performance improvements equivalent to the ratios stated here.

Prices are suggested U.S. list prices and are subject to change without notice. Contact your IBM representative or Business Partner for the most current pricing in your geography.