



Java[™] For RPG Programmers

Phil Coulthard: coulthar@ca.ibm.com
George Farr: farr@ca.ibm.com

IBM | 2003, 2004

© 2004 IBM Corporation
[™]Java is a trademark of Sun Microsystems Inc



Disclaimer

IBM

Acknowledgement:



- This presentation is a collaborative effort of the IBM Toronto AS/400 Application Development presentation team, including work done by:

- ▶ *Phil Coulthard, George Farr*

- This presentation is based on the books (www.mcpressonline.ibmpress)

- ▶ *Java for RPG Programmers, ISBN 1-931182-06-X*

- ▶ *Java for S/390 and AS/400 COBOL Programmers, 1-58347-011-5*

- It also contains information from the related Student Workbook (www.mcpressonline.ibmpress)

- ▶ *Java for RPG and COBOL Programmers on iSeries Student Workbook*



Disclaimer:

- The information contained in this document has not been submitted to any formal IBM test and is distributed on an as is basis without any warranty either express or implied. The use of this information or the implementation of any of these techniques is a customer responsibility and depends on the customers' ability to evaluate and integrate them into the customers' operational environment. While each item may have been reviewed by IBM for accuracy in a specific situation, there is no guarantee that the same or similar results will result elsewhere. Customers attempting to adapt these techniques to their own environment do so at their own risk.

Reproduction:

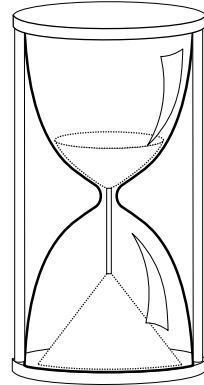
- The base presentation is the property of IBM Corporation. Permission must be obtained PRIOR to making copies of this material for any reason.



Agenda

IBM

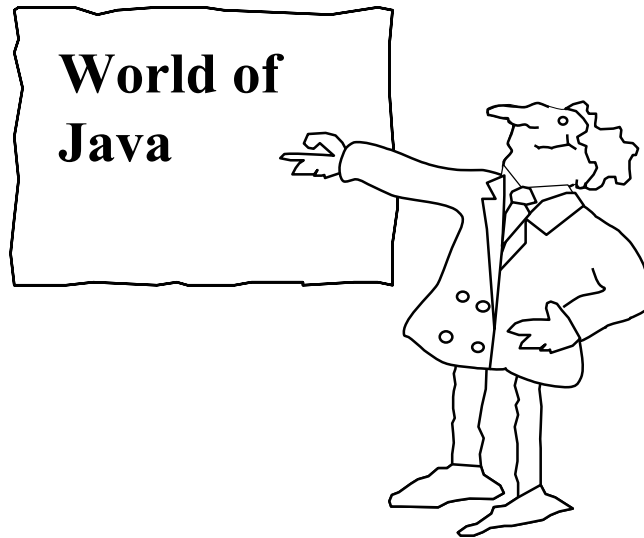
- **World of Java**
- **Java versus RPG:**
 - RPG IV and ILE Review
 - Application Anatomy
 - Syntax, Data Types, Variables
 - Operators, Statements
 - Arrays, Strings
- **OO Terminology**
- **Exception Handling**





Agenda

IBM





What is Java?

IBM

- **An OO programming language**
 - ▶ **Created by Sun Microsystems Inc, in 1995**
 - ▶ **Adopted by Netscape in 1996**
 - ▶ **Heavy investment by IBM ever since**
- **Initially for:**
 - ▶ **applets that run in Web Browsers**
 - ▶ **applications that are client-GUI or server non-GUI**
- **Now also for:**
 - ▶ **Servlets that run in a Web Server**
 - ▶ **Enterprise JavaBeans that run on a server**
 - ▶ **and much more...**



Java Mantra

IBM

• **"Write Once, Run Anywhere"**

- ▶ **Java code is interpreted**
- ▶ **Java interpreter has been ported to just about every operating system, Web browser, Web server, and hand-held device in existence today**
- ▶ **Java language comes with many pre-defined functions and services**
 - In the form of "packages"
 - Reduces dramatically the need to rely on operating system APIs

• **"Learn Once, Use Everywhere"**

- ▶ **Use Java for GUI, Web, Business Logic, Tools, Business Applications, Games, ...**



Three Flavors of Java

IBM

- **Java 2 Standard Edition (J2SE):**

- ▶ **For JavaBeans, applets, GUI/non-GUI application**

- ▶ **START HERE**

- but also use Servlet and JSP support from your Web Application Server

- **Java 2 Enterprise Edition (J2EE):**

- ▶ **For Java Servlets, JavaServer Pages**

- Although also available via Application Servers such as WebSphere

- ▶ **For Enterprise JavaBeans, Java Naming and Directory Interface, Java Messaging Service, ... and much more!**

- ▶ **GROW HERE**

- **Java 2 Micro Edition (J2ME):**

- ▶ **For small, embedded devices: chips, phone, hand-helds...**



• How is Java packaged?

▶ For developers

- In a "**Java Development Kit**" (**JDK**)

- ▶ Compiler, runtime, command line tools

▶ For runtime

- In a "**Java Virtual Machine**" (**JVM**)

- ▶ Interpreter

• How do you get Java?

▶ For developers

- **JDK** from Sun (www.java.sun.com) or IBM (www.ibm.com/java)

- **JDK** also built-in to Java tools like VisualAge for Java, WSSD

▶ For runtime

- **JVM** built-in to many Operating Systems, Web Browsers, App'n Servers, PDAs, Cell Phones, etc



JDK Contents

IBM

Command line tools

JVM /
interpreter
("java")

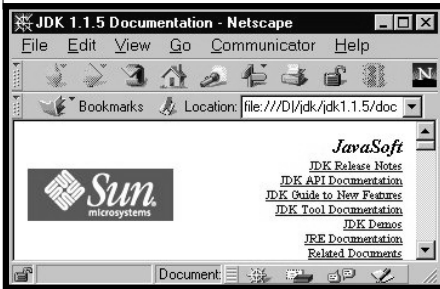
Bytecode
compiler
("javac")

Doc'n tool
("javadoc")

Compression
tool
("jar")

...

Documentation



Packages

language

database
access

math

graphics

file system

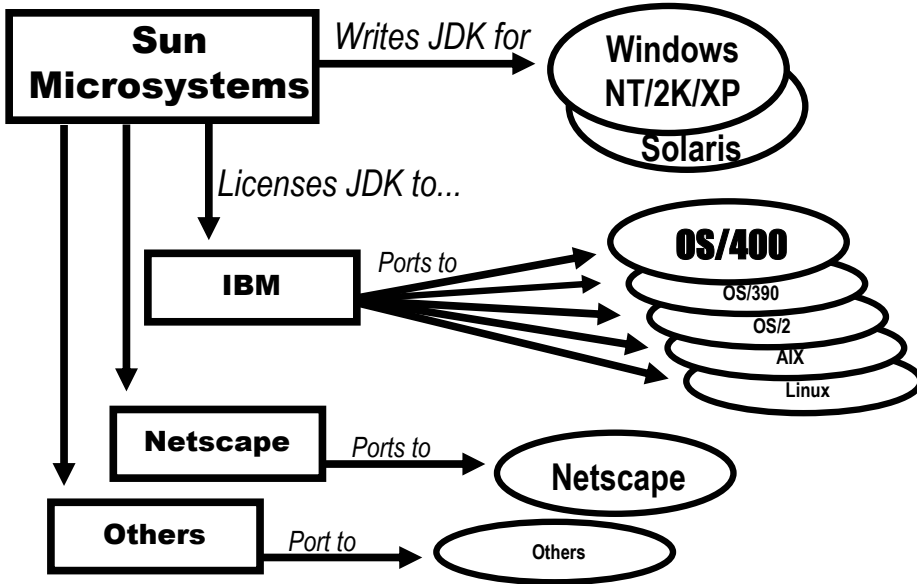
...

networking

utilities



JDK Licensing





•Classes

▶ **Compilation unit**

- no matter what you are using Java for!

▶ **All fields and executable code are inside classes**

▶ **Source files are compiled into class files**

•Bytecode

▶ **What are inside class files**

▶ **Assembler language for Java**

- what the JVM "interprets"



Java Beans

IBM

- **JavaBeans™**

- ▶ **Classes designed for fine-grained re-use**

- Java's components, like Microsoft VB's VBX
 - Not to be confused with Enterprise JavaBeans!!

- ▶ **Beans contain**

- properties (fields),
 - methods (paragraphs),
 - events (eg, button-pressed)

- ▶ **Tools can discover contents dynamically**

- ▶ And present list to use to select from or change

- **JAR™ Files (Java ARchive)**

- ▶ **Java way to group/compress class files**

- for easy distribution (uses ZIP technology)



Using Java

IBM

- **Applications**

- ▶ Java command line programs (you call)

- **Appletstm**

- ▶ Java Web Browser programs (Web Browser calls)

- **Enterprise JavaBeanstm**

- ▶ Enterprise-scale re-usable components (Application Server calls)
- ▶ Large scale (eg payroll) versus JavaBeans (eg, tax)

- **Java Servletstm**

- ▶ Java Web Server programs (Web Server calls)

- **JavaServer Pagestm**

- ▶ HTML plus embedded Java (Servlets call)



Java Tools

IBM

- **Java Tools are**

- ▶ **Optional**

- minimal requirement: JDK + editor

- ▶ **Productive**

- eg, wizards and debuggers

- ▶ **Numerous**

- From IBM, Symantec, Sun, Inprise, ...

- **IBM Java Tool story**

- ▶ **"Next generation" tools are**

- WebSphere Studio Site Developer (Java, Web, XML tooling)
 - WebSphere Studio Application Developer (+ EJB tooling)

- ▶ **For iSeries, there is**

- WebSphere Development Studio Client



Java and the Web

IBM

- **Java and Web are a good fit!**

- ▶ **Most common use of Java today for business is:**

- Glue between business logic / transactions, and Web pages

- ▶ **This is done by**

- Wrapping the business logic / transaction in a JavaBean/EJB
 - Calling the JavaBean/EJB from a **Java Servlet**
 - Creating **JavaServer Pages** (JSPs) that
 - ▶ Are simply HTML static pages with "holes" for dynamic data
 - ▶ Are called by the Java Servlet, which passes the dynamic data in the form of a simple Java Bean (think of this bean as a data structure)
 - ▶ Are resolved into straight HTML by the JSP engine and passed to the Web Browser

- **Java Servlets and JSPs...**

- ▶ **are industry standard**

- run in a Web Application Server that meets industry standard



WebSphere

IBM

- **Three flavors in 5.1**

- \$ ▶ **WebSphere Application Server Express**

- \$\$ ▶ **WebSphere Application Server**

- \$\$\$ ▶ **WebSphere Application Server Network Deployment**

- **Runtime engine for JSPs and Servlets**

- ▶ **Plugs into Web server such as**

- IBM HTTP Server "classic", Apache, IIS, Domino

- **Runtime engine for EJBs**

- ▶ **Except in WAS Express**

www.ibm.com/software/webservers

www.ibm.com/series/websphere



Web Tools

IBM

- **Web Tools are for**

- ▶ **Web site management**

- Organizing Web projects and files
 - Publishing to test and product application servers

- ▶ **HTML and other static content**

- Images, audio, video, etc

- ▶ **Java Servlets and JavaServer Pages**

- For designing JSPs, generating servlets and JSPs via wizards

- **IBM Web Tool story**

- **WebSphere Studio Site Developer** (Java, Web, XML tooling)
 - + iSeries Extensions = **WebSphere Development Studio Client**
 - **WebSphere Studio Application Developer** (+ EJB/J2EE tooling)
 - + iSeries Extns = **WebSphere Development Studio Client Adv'd**



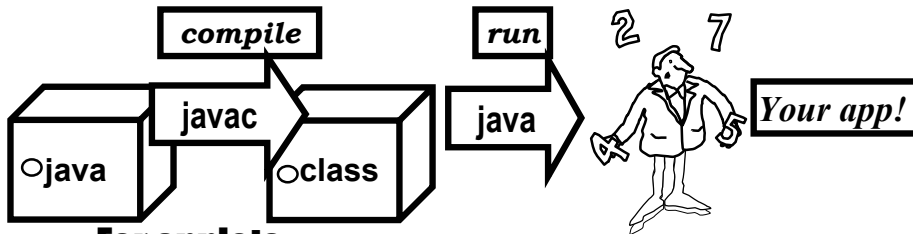
Java LifeCycle

IBM

- **Use javac to compile**

- ▶ **Use java to run from command line**

- if it is application or to unit-test this individual class



- ▶ **For applets**

- Use HTML/JSP file with <APPLET> tag pointing to the applet

- ▶ **For servlets**

- Use HTML/JSP file with <FORM> tag pointing to the servlet

- ▶ **For EJBs**

- Include in Web Application that is deployed to a J2EE Container



Compiling Java

IBM

```
/* Prototypical Hello World application */  
public class Hello  
{  
    public static void main(String args[])  
    {  
        System.lout.println("Hello World!");  
    }  
}
```



"main"
method
called by
JVM



```
E:\mycode>javac Hello.java  
Hello.java:6: No variable lout defined in class System  
    System.lout.println("Hello World!");  
           ^  
1 error
```



```
System.out.println("Hello World!");
```



```
E:\mycode>javac Hello.java
```

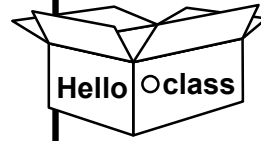




Running Java Apps

IBM

```
E:\mycode>java Hello  
Hello World!
```





Java Entry Points

IBM

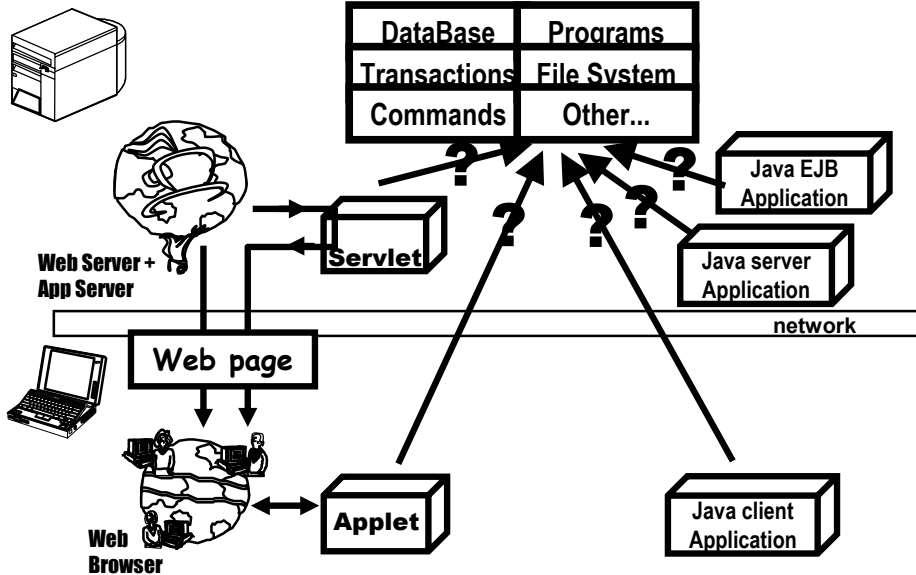
	Appl'n	Applet	Servlet
Where runs	Anywhere	Web Browser	Web Server
Who calls	User	Browser	Web Server
How invoked	java command	<APPLET> html tag	By mapping to URL
Entry point	main method	init, then paint	init, then doGet / doPost
Security restrictions	No	Yes	Optional



Non-Java Resources

IBM

► How to access non-Java resources?





Accessing Data

IBM

• Standards for Accessing Data

▶ JDBC™

- Dynamic SQL access to relational data or stored procedures
- Part of the Java standard
- Patterned after ODBC, but with OO versus C-APIs
- JDBC driver manager comes with Java
- JDBC drivers supplied by DB vendors or others
 - ▶ IBM UDB, HIT Software, Oracle, Sybase, Inprise, . . .

▶ SQL/J

- Static SQL embedded inside Java
- Created by Oracle, supported by IBM UDB

▶ Java Stored Procedures

- IBM UDB supports writing stored procedures in Java



iSeries Toolbox For Java

IBM

- **100% Java classes for:**

free!

- **JDBC access to DB2/400**
- **DDM record level access to DB2/400**
- **Data Queue access**
- **Print access**
- **Program Call, Command Call**
- **File system access**
- **Client to server connections, remote login**
- **Much more...**

Shipped
with
OS/400,
WDS c

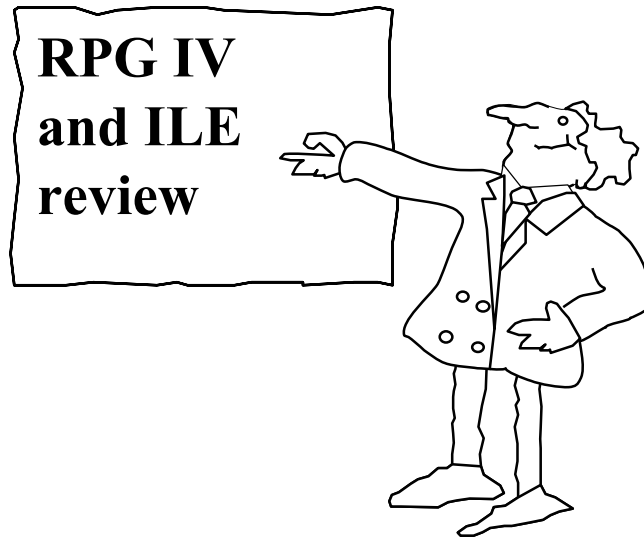
- **Runs anywhere**

- **OS/400, Windows, Linux, Unix, ...**



Agenda

IBM





The Java Language

IBM

- **We compare Java to RPG IV**
 - ▶ **closer match to Java than III**
 - ▶ **more modern constructs**
 - ▶ **easier skills transfer to Java!**

•
•
•

Continue your RPG IV journey!



RPG IV Review

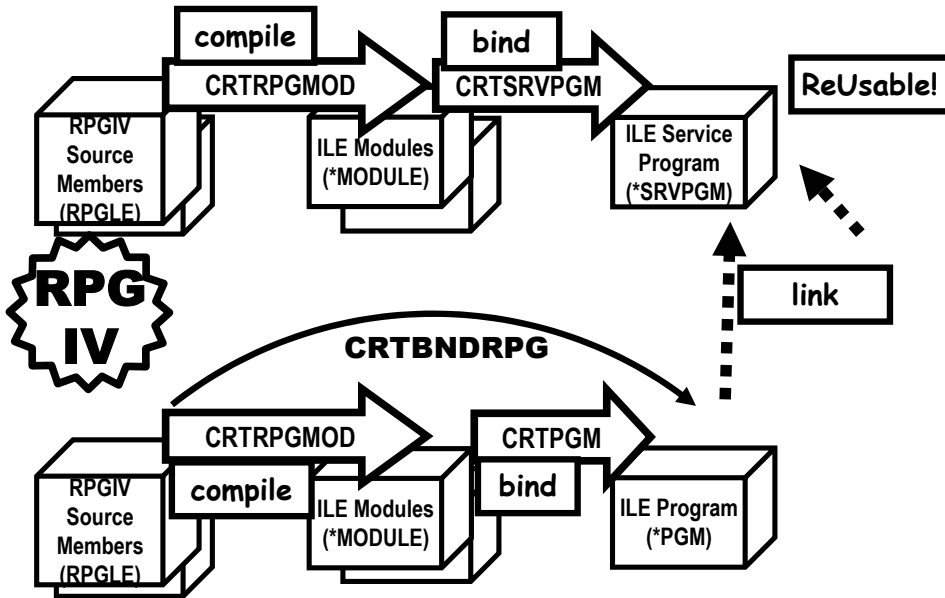
IBM

- ✓ **Longer names (10, but 4096 in V3R7)**
- ✓ **Mixed case (folded to upper by compiler)**
- ✓ **New D spec (Definition) for declares**
- ✓ **Free form expressions in factor 2 of some op-codes: EVAL, IF, DOW, DOU, WHEN**
- ✓ **New data types**
 - Date, Time, Timestamp, Integer, Float, Null, Variable-Length (V4R2), Indicator (V4R2) fields
- ✓ **Built-in functions (like %TRIML / R)**
- ✓ **Procedures ("grown up subroutines")**
 - fast intra/inter-module calls. New CALLP op-code



ILE Compiling, Binding

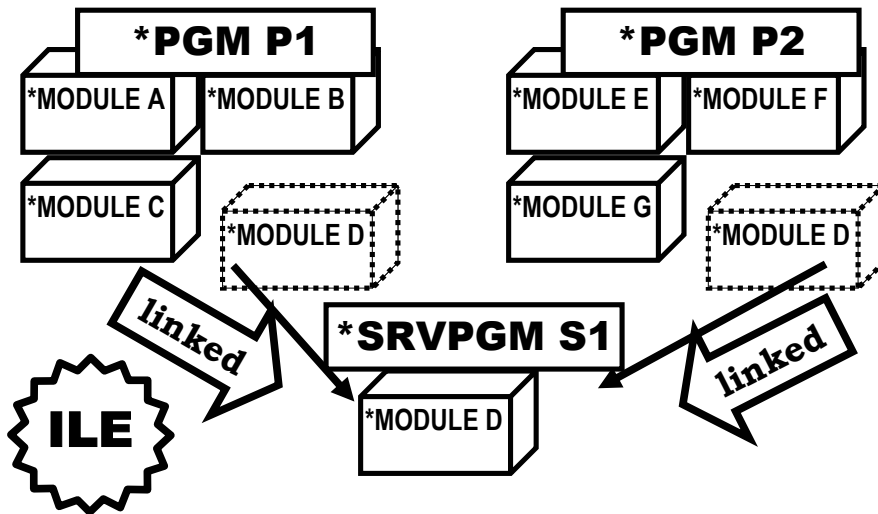
IBM





ILE Service Programs

IBM



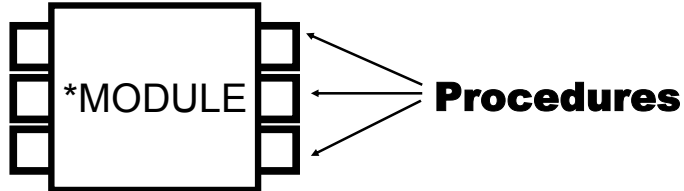
- ▶ allow you to extract out common code
- ▶ are linked, not bound, to *PGMs



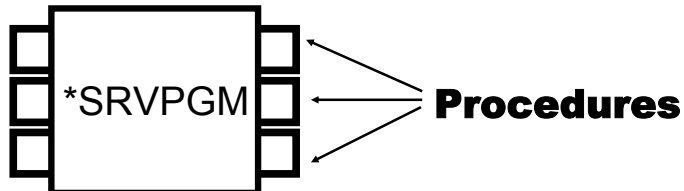
Inter-Module Calls

IBM

- **Modules *call* each other...**
 - ▶ **by calling procedures**



- ***PGMs *call* *SRVPGMs...**
 - ▶ **By calling procedures**





RPG IV Procedures

IBM

- ✓ **Local Variables**
- ✓ **Return Values**
- ✓ **PARMS: Value & Reference**
- ✓ **Recursion**
- ✓ **Exporting**
- ✓ **Prototyping**

"grown up
subroutines"

Modularity

Re-Use

Skills transfer to
Java (methods)

Anatomy of Procedures

RPG IV

P-spec

```
Pmax B Procedure Beg EXPORT  
* return value  
D max PI 5P 0 return type  
* parameters  
D parm1 parameters 5P 0 VALUE  
D parm2 5P 0 VALUE  
* local variables  
D temp S local field 5P 0  
* local code  
C IF parm1 > parm2  
C EVAL temp = parm1  
C ELSE  
C EVAL temp = parm2  
C ENDIF  
C RETURN temp return value  
* end of procedure  
Pmax E Procedure End
```

P-spec



Advertisement!

IBM

•WDS Sc has a Procedure Wizard

RPG Procedure Wizard

Create RPG procedure

Procedure type: Subprocedure

Procedure name: Max

External name (EXTPROC): Maximum

Purpose: Return maximum of two numbers

Exportable for use with other code (EXPORT)

Generate free-form calculations

Parameters created for this procedure:

Name	Type	Text
Parm1	Integer	
Parm2	Integer	

Pass the operational descriptors with the parameters

Return a value

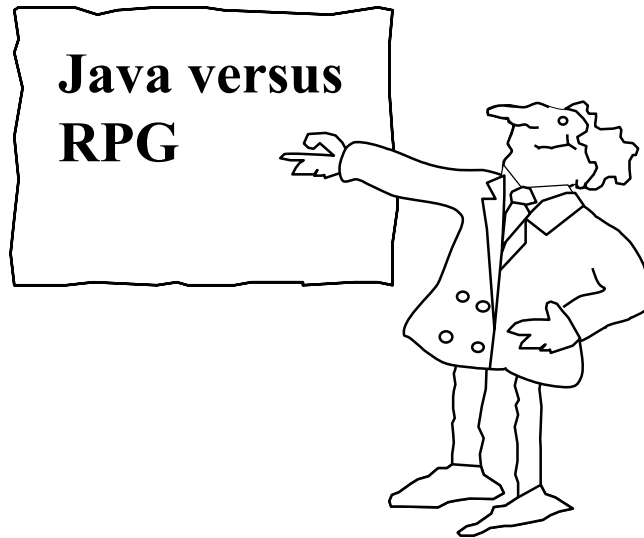
< Back Next > Finish Cancel

- ▶ generates skeleton code for you
- ▶ saves you from memorizing procedure syntax



Agenda

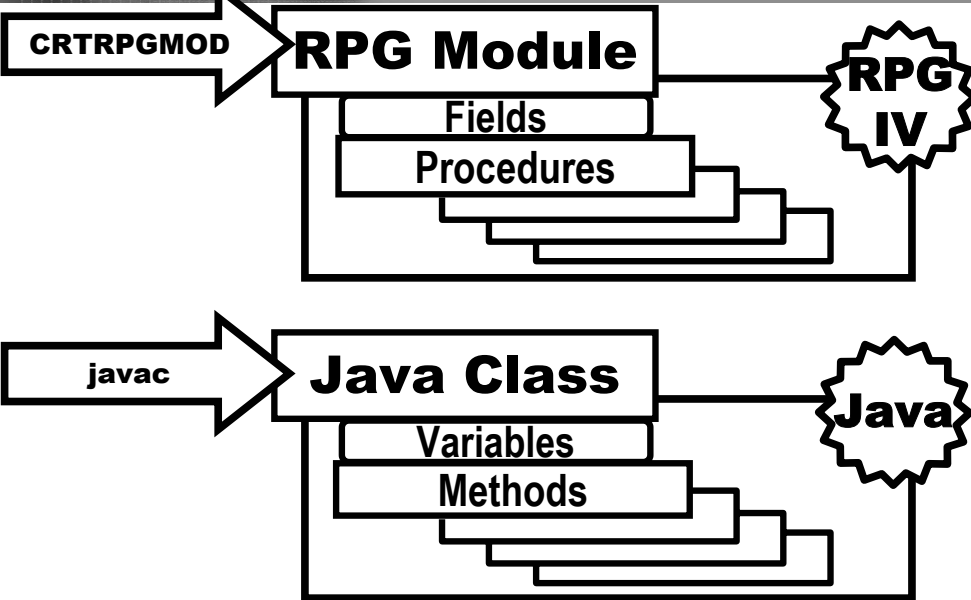
IBM



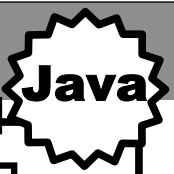


MODULE vs CLASS

IBM



Java Class Syntax



class
keyword:
what we're
defining

```
public class Customer
```

class name

public modifier:
anyone can use

```
private int custId;  
private char custCode;
```

Global
variables

field name

private modifier:
only code in this
class can access

```
public static void main(String args[])  
{  
    ...  
}
```

"main" method
(described later)

braces { } delimit
start and end of
class

```
public int setMyVariable(int newValue)  
{  
    ...  
}
```

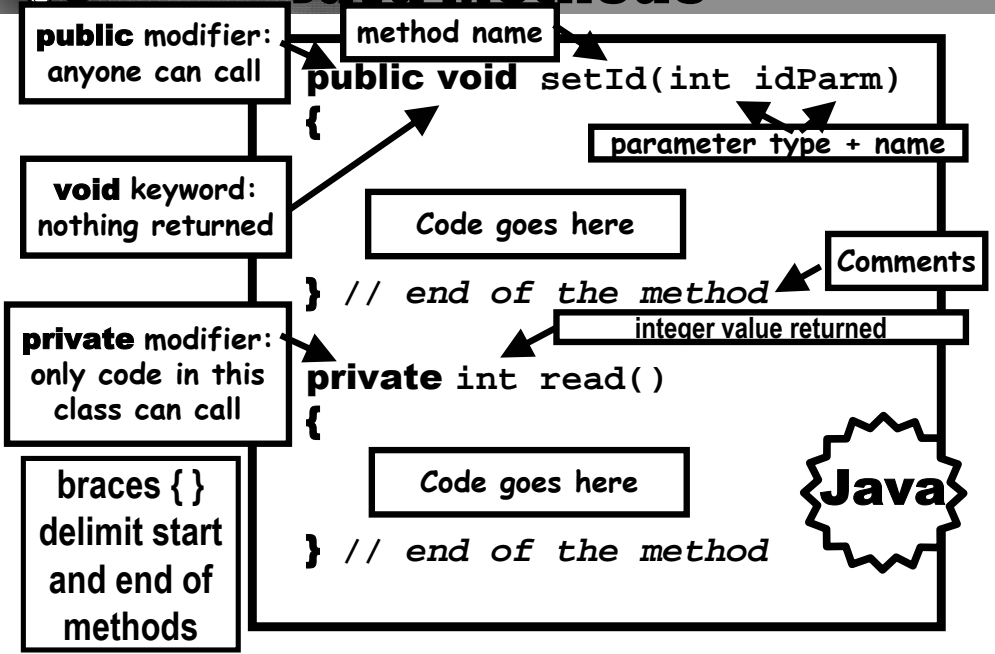
Methods
(described
next)

```
}
```



Java Methods

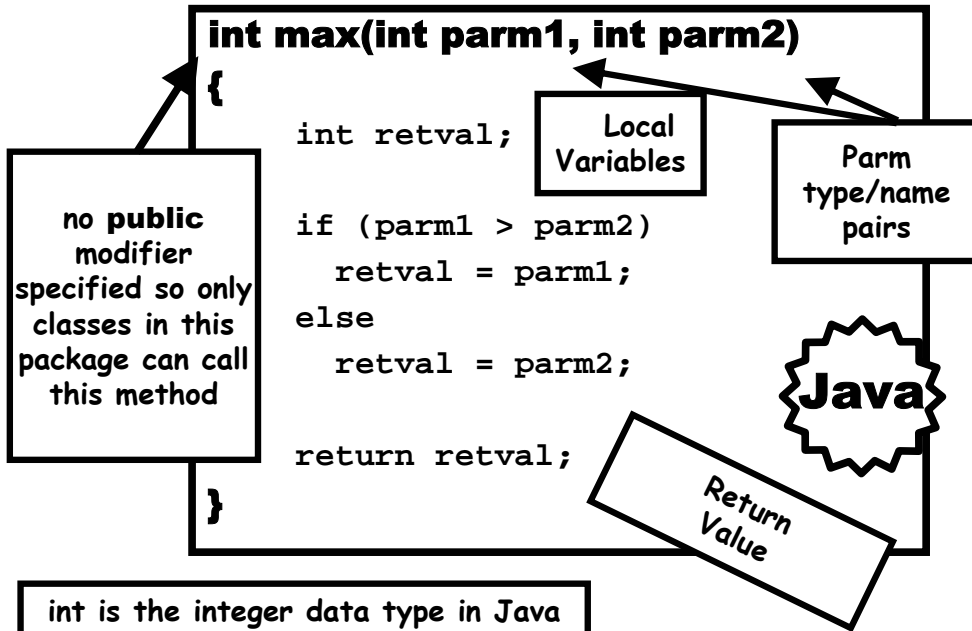
IBM





Java Method Example

IBM





Naming Conventions

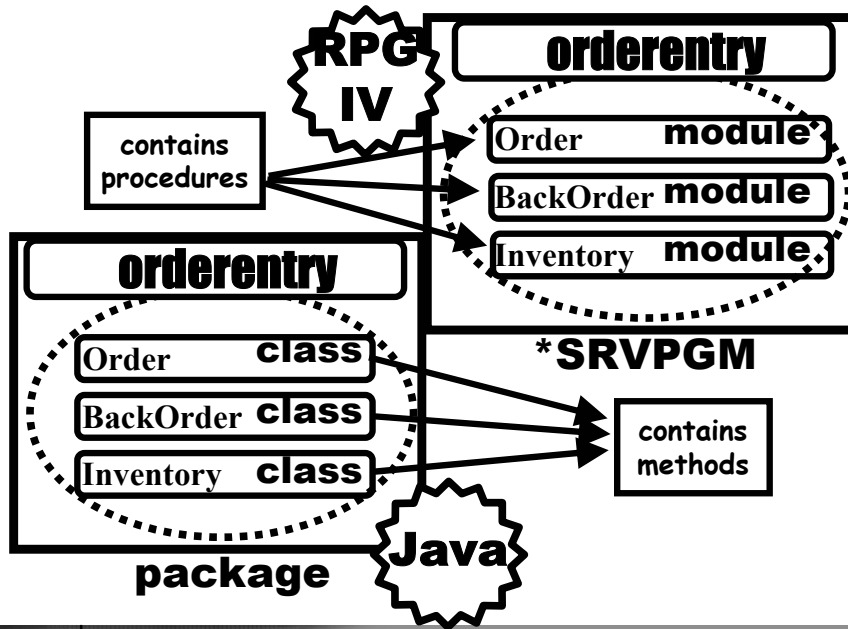
IBM

- **Class names should**
 - ▶ **be all lowercase except**
 - *first letter of each word*
 - *eg: OrderEntry*
- **Method / field names should**
 - ▶ **be all lowercase except**
 - *first letter of each word other than first*
 - *eg: processOrder*
- **Constants should**
 - ▶ **be all uppercase**
 - *eg SUNDAY*



Packages

IBM

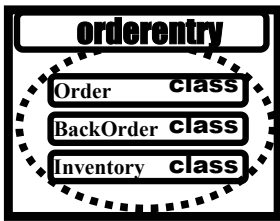




Defining Packages



"package" must be first statement in source file



"package" is like compiler directive

Order.java

```
package orderentry ;  
  
public class Order  
{  
    ...  
}
```

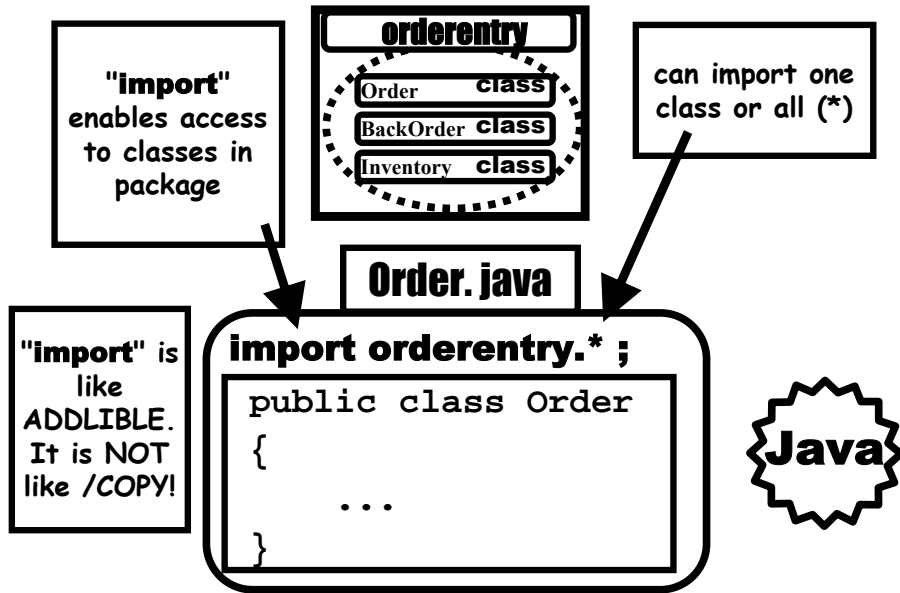
BackOrder.java

```
package orderentry ;  
  
public class BackOrder  
{  
    ...  
}
```



Using Packages

IBM





Naming Packages

IBM

- **Package names are**
 - ▶ usually all lowercase
 - ▶ usually multi-part, dot separated
- **Java-supplied packages**
 - ▶ all named `java.xxx`
 - ▶ for example: `java.awt` or `java.awt.event`
- **Your packages**
 - ▶ will start with `com.xxx`, where `xxx.com` is your company's domain name
 - ▶ eg, IBM's start with `com.ibm.xxx`

`java.lang`
always
imported
for you



Packages vs File System

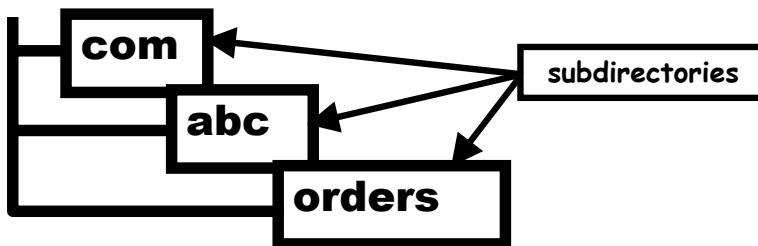
IBM

- **Packages**

- ▶ **have no file system objects!**
- ▶ **map to directories**
 - *One per dot-separated name part*

- **For example**

- ▶ **Consider package name `com.abc.orders`**





Comparing Anatomies

IBM

RPG	JAVA	COMMENTS
*PGM	Application	<i>Program object == Application</i>
*SRVPGM	Package	
Module	Class	<i>Compilation unit</i>
Fields	Variables	<i>Global variables</i>
Procedures	Methods	<i>Functions</i>
Variables	Variables	<i>Local Variable</i>
Code	Code	<i>Executable code</i>



CLASSPATH

IBM

- **How are classes found?**
 - ▶ when referred to by code in other classes
 - ▶ by the compiler (javac) and runtime (java)
- **Answer!**
 - ▶ by searching the **CLASSPATH** env variable
- **CLASSPATH**
 - ▶ is a list of semi-colon separated directories
 - *colon separated on OS/400*
 - ▶ **much like library list on OS/400!**
 - *system searches it for first match*



CLASSPATH

IBM

- **CLASSPATH entries are**
 - ▶ **directories to search for classes**

```
SET CLASSPATH = .;c:\myJava
```

search
current
directory

search
c:\myJava
directory



Classpath & Packages

IBM

- **For classes in packages**
 - ▶ name parent directory containing subdirs
- **Consider package `com.abc.orders`:**
 - ▶ if `com\abc\orders` is off of `c:\myJava ...`

```
SET CLASSPATH = .;c:\myJava
```

searches inside
`.\com\abc\orders`

searches inside
`c:\myJava\com\abc\orders`



ZIP and JAR Files

IBM

- **Two options for distribution:**
 - ▶ **ZIP files**
 - *Industry standard compression technology*
 - ▶ **JAR files (Java ARchive)**
 - *Same as ZIP by written in Java, part of JDK*
- **To compress multiple files together**
 - ▶ **Use WINZIP or PKZIP utilities on Windows or...**
 - ▶ **Use jar command that comes with JDK**

```
jar -cvf myClasses.jar *.class
```

create myClasses.jar file

put all class files in it



Classpath+Jar/Zip

IBM

- **You don't have to uncompress!**
 - ▶ **JVM can find and read classes directly from ZIP files and JAR files!!**
 - *That's cool!*
- **However, the .zip or .jar file must be on the CLASSPATH environment var**
 - ▶ **place actual file name on path, not just dir!**

```
SET CLASSPATH =  
.;c:\myJava;c:\myJava\myClasses.jar
```



Java Syntax

IBM

- **Statements are free-format**

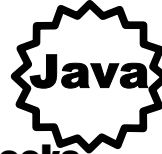
- ▶ extra blanks and lines are ignored
- ▶ statements end with semi-colon ;

C-like
syntax

- **Blocks use braces**



- ▶ start and end of classes
- ▶ start and end of methods
- ▶ start and end of conditional / loop blocks



- **All names are case-sensitive**

- ▶ abc NOT= ABC
- ▶ even source file names are case sensitive

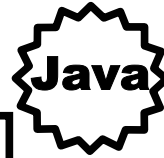


I Want To Be Free

IBM

- **Java is totally free form**

- ▶ extra blanks and lines ignored



```
void myMethod(int param1)
{
    return;
}
```

```
void myMethod(int param1) {
    return;
}
```

```
return;
```

```
return
;
```



Java Comments

IBM

● Multi-line comment:

```
/* this is a multi  
line comment */
```

```
/*-----*  
* Please read these comments *  
* as they are very important! *  
*-----*/
```

● Single line comment:

```
// This whole line is a comment  
int myVariable = 10; // Only this part is a comment
```

● JavaDoc comment:

note
double
asterisk

```
/** This is the <U>scan package</U>  
* this is the second line.  
* @author George & Phil  
* @version Feb 26,2000  
*/
```



JavaDoc Comments

IBM

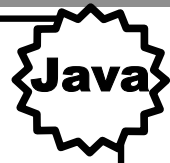
- **Can use special tags**
 - ▶ special meaning to javadoc formatter
 - ▶ can also use any HTML tags like **bold**

Tag	Description
@author	Author of this class or method
@see	References another class or method. Generates a link
@version	Version number of this class or method
@since	Release or version this class or method has existed since
@deprecated	This is an obsolete method
@return	Describes what this method returns
@param	Describes a parameter to this method



JavaDoc Example

IBM



```
/**
 * Shows a message
 * @param message The msg string to show
 * @return void
 * @see MyClass#myMethod2(String message)
 */
public void myMethod(String message)
{
```

@param

@return

@see

Generated Documentation (Untitled) - ...

File Edit View Favorites Tools Help

All Classes

MyClass

myMethod

public void **myMethod**(java.lang.String message)

Shows a message

Parameters:

message - The msg string to show

Returns:

void

See Also:

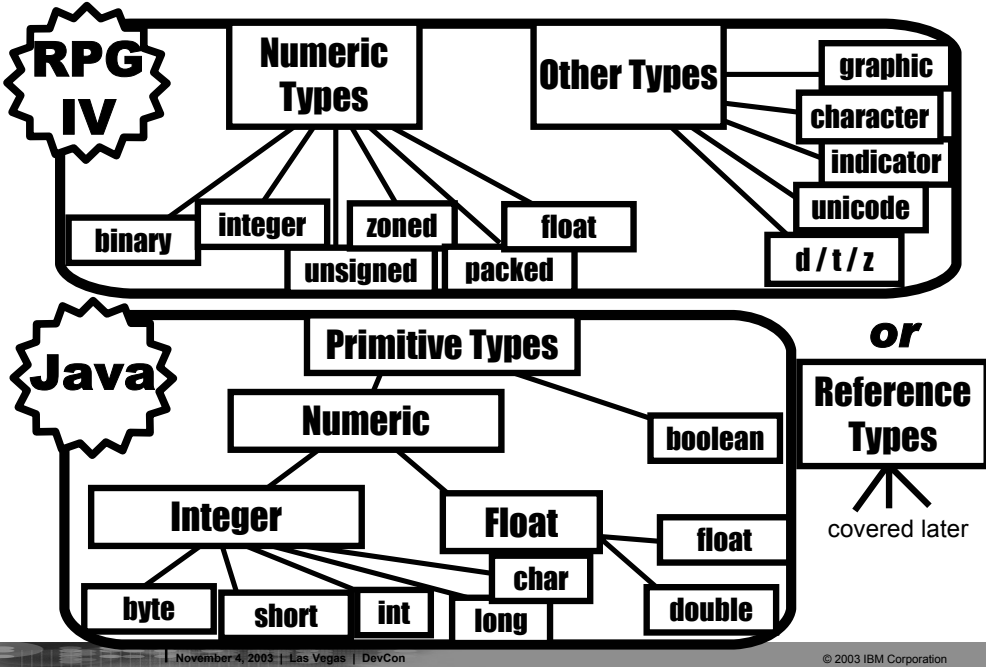
[myMethod2\(String message\)](#)

My Computer



Data Types Overview

IBM





Java Primitive Types

IBM

Type	In Use	Description
Integer	int i;	4 byte signed: about +- 2 billion
Long	long l;	8 byte signed: about +- huge #
Byte	byte b;	1 byte signed: -128 to + 127
Short	short s;	2 byte signed: -32768 to 32767
Character	char c;	2 byte unicode. 1 char only!
Boolean	boolean flag;	true or false
Float Single	float f;	32 bit
Float Double	double d;	64 bit





Data Types...

IBM

RPG	Java	Comments
numeric (no decimals)	short or int	depends on length
numeric (with decimals)	float or double, or BigDecimal class	depends on length. BigDecimal is a Java supplied class
float (length 4)	float	Both are IEEE standard
float (length 8)	double	Both are IEEE standard
character (length one)	char	single character only
character (length n)	String class	A class, not a primitive type
graphic	String class	A class, not a primitive type
unicode	String class	A class, not a primitive type
indicator	boolean	'1' = true, '0' = false
date, time, timestamp	GregorianCalendar class	A class, not a primitive type



More on Boolean

IBM

- **Can be assigned true or false:**

- ▶ `boolean myFlag = true;`

language
keywords

- **Can be assigned an expression:**

- ▶ `boolean myFlag = (rate > 10);`

- **Can be in an expression:**

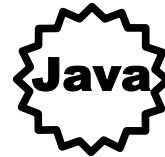
- ▶ `if (rate > 10) ... *** or ***`

- ▶ `if (myFlag)`

- **Can be negated:**

- ▶ `myFlag = !myFlag;`

- ▶ `while (!myFlag) ...`

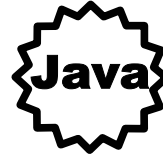




What about Packed?

IBM

- **No packed decimal data type in Java**
 - ▶ Could use float / double, but precision is a problem for "fixed decimal" numbers
- **Answer: BigDecimal class**
 - ▶ Part of **java.math** package
 - ▶ A class, not a built-in "primitive" data type
 - ▶ *Software simulation* of fixed decimal numbers
 - ▶ Unlimited **precision** (total number of digits)
 - ▶ Program control over **scale** (number of decimal digits)
 - ▶ Methods include: **add, subtract, divide, multiply, setScale**
- **See also: BigInteger class**





Declaring Fields in RPG

IBM

```

+*.. 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+...
***** Beginning of data *****

```

```

FQSYSVRT  O   F   80          PRINTER OFLIND(*INOV)
D FIRST          S              7A  INZ('George ')
D AGE            S              2B  0  INZ(25)
D*-----
C   *LIKE        DEFINE  FIRST          LAST          -3
C                   EVAL   LAST='FARR'
C                   MOVE   ' AGE WAS--->'AGETEXT      12
C                   EXCEPT RESULT
C                   MOVE   *ON              *INLR
C*-----
OQSYSVRT  E           RESULT
O          FIRST          5
O          LAST          10
O          AGETEXT       22
O          AGE           26

```



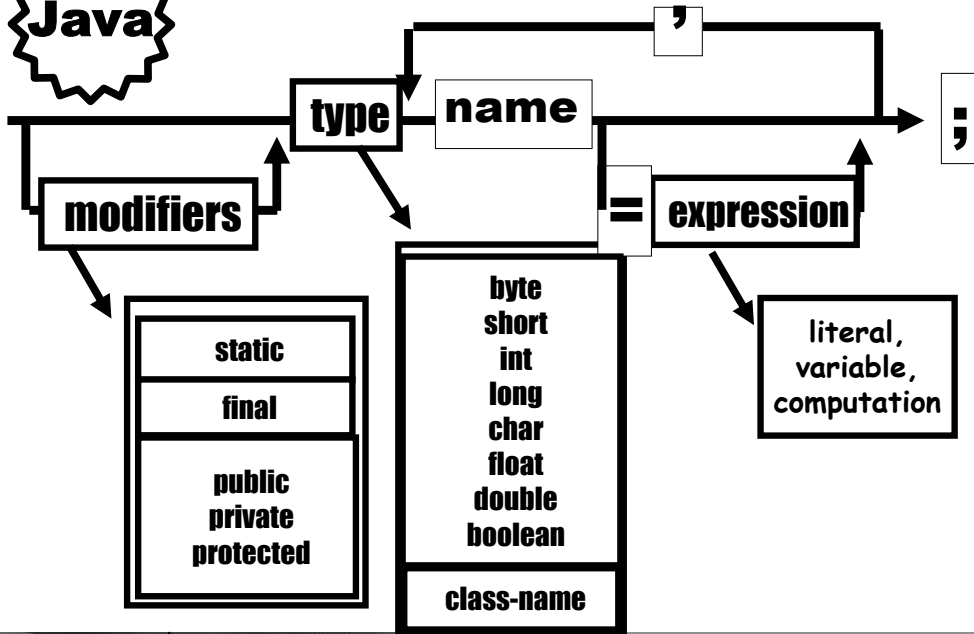
- On the C specification
- Using the Define operation code
- ✓ On the new Definition specification



Java

Declaring Variables

IBM





Declaring

IBM

**RPG
IV**

Java

<pre>D*..1....+....2....+....3....+....4.. DEmpRcd DS D number 5I 0 D type 1A D name 20A D address 50A D hired D D salary 9P 2</pre>	<pre>public class EmployeeRecord { private int number; private char type; private String name; private String address; private Date hired; private BigDecimal salary; }</pre>
---	--

DS = "Data Structure"
S = "Standalone"

access
modifiers

data
types

actual
variable
names



Where's the Length?

IBM

- **You do not specify #digits!**

- ▶ **Data Type determines # of bytes**

- *which determines how much var can hold*

- *eg: short holds -32768 to 32767*



- **Usually you will use:**

- ▶ **integer ("int") when no decimals (unless numbers > 2 billion)**

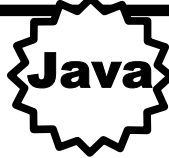
- ▶ **BigDecimal class when decimals needed**

- ▶ **String class when dealing with characters**



Declaring and Init'g

IBM



```

public class EmployeeRecord
{
    private int    number = 0;
    private char   type   = 'R';
    private String name   = "Joe Public";
    private String address = "1 Young St";
    private Date   hired  = new Date();
    private BigDecimal salary = new BigDecimal("30000.00");
}

```

Note: new operator described later



```

D*..1.....2.....3.....4.....5
DEmpRcd          DS
D number          5I 0 INZ(0)
D type            1A  INZ('R')
D name            20A INZ('Joe Public')
D address         50A INZ('1 Young St')
D hired           D  INZ(D'1999-12-31')
D salary          9P 2 INZ(30000)

```



Declaring Constants

IBM

**"static"
and "final"
keywords
define a
constant**

```
public class EmployeeRecordDefaults
{
    static final int    NUMBER = 0;
    static final char   TYPE   = 'R';
    static final String NAME   = "Joe Public";
    static final String ADDRESS = "1 Young St";
    static final Date   HIRED   = new Date();
    static final BigDecimal SALARY =
        new BigDecimal("30000.00");
}
```

Java

```
D*..1.....2.....3.....4.....5
D*EmpRcdDFT      DS
D numberDFT      C          CONST(0)
D typeDFT        C          CONST('R')
D nameDFT        C          CONST('Joe Public')
D addressDFT     C          CONST('1 Young St')
D hiredDFT       C          CONST(D'1999-12-31')
D salaryDFT      C          CONST(30000)
```

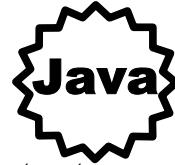
**RPG
IV**



Wrapper Classes

IBM

- **Primitive types have wrappers**
 - ▶ **classes in java.lang package**
 - *always imported for you!*
 - ▶ **sometimes you will need them**
 - *such as for Vectors as we'll see*
 - *they also have handy methods and constants*



Primitive	Wrapper
byte	Byte
short	Short
int	Integer
long	Long

Primitive	Wrapper
char	Character
boolean	Boolean
float	Float
double	Double



Casting in RPG

IBM

casting
is
always
implicit
in RPG

```

..... 1 ..... 2 ..... 3 ..... 4 ..... 5 .....
***** Beginning of data *****
FQSYSVRT  O   F   80          PRINTER OFLIND(*INOV)
D DS1                      DS
D int5                      5I 0 INZ(25)
D BIN9                      9B 0 INZ(22)
D ZONE9                    9S 0 INZ(30)
D PACK9                    9P 0 INZ(40)
D*-----
C                      MOVE      BIN9          INT5
C                      EXCEPT  RESULT
C                      MOVE      PACK9        INT5
C                      EXCEPT  RESULT
C                      MOVE      ZONE9       INT5
C                      EXCEPT  RESULT
C                      MOVE      *ON         *INLR
OQSYSVRT  E                      RESULT
O                      INT5          15
***** End of data *****

```

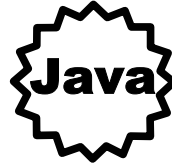




Casting in Java

IBM

```
public class TestCast
{
    public static void main(String args[])
    {
        short sValue = 10; // 2 bytes
        long lValue = 30; // 8 bytes
        lValue = sValue; // implicit
        sValue = (short)lValue; // explicit
    }
}
```



• Casting in Java

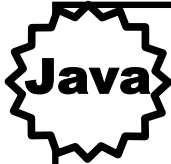
- ▶ only implicit if target *type* larger than source
- ▶ else must explicitly cast: **(target-type)source**



What About Overflow?

IBM

- **Source won't fit in target?**
 - ▶ **Nothing happens!!**
 - *No overflow indicators in Java!!*
 - ▶ **You're job to check first before casting:**
 - Use **MIN_VALUE** and **MAX_VALUE** constants in wrapper classes



```
if ((lValue <= Short.MAX_VALUE) &&  
    (lValue >= Short.MIN_VALUE))  
    sValue = (short)lValue; // cast  
else  
    // overflow/underflow error...
```



Casting Summary Table

IBM

	byte	char	short	int	long	float	double
byte	No	Cast ¹	No	No	No	No	No
char	Cast	No	Cast ¹	No	No	No	No
short	Cast	Cast	No	No	No	No	No
int	Cast	Cast	Cast	No	No	No	No
long	Cast	Cast	Cast	Cast	No	No	No
float	Cast	Cast	Cast	Cast	Cast	No	No
double	Cast	Cast	Cast	Cast	Cast	Cast	No

Java

read left to right →

¹ Potential loss of sign



Assignment

IBM

- **RPG IV:**

- ▶ free-format EVAL op-code & equal operator '='

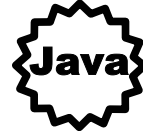
- **Java:**

- ▶ no op-code, just equal operator "="

RPG III	RPG IV	Java
C MOVE 0 X	C EVAL X = 0	X = 0;

- **Java also allows stringing:**

```
A = B = C = 25;
```





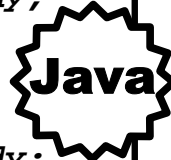
If-Else

IBM

```
C*  op-code  factor2
C    IF      expression
C*      Body
C    ELSE
C*      Body
C    ENDIF
C*      :
```



```
if (condition)
{
    //Body;
}
else
{
    //Body;
}
```



- **Similar in both languages**
- **But in Java**
 - ▶ **Body can be compound or single statement**
 - *Single statement bodies don't need braces*



IF Example

IBM

```

C  AGE      IFLE      2
C                MOVE      0  PRICE
C                ELSE
C  AGE      IFLE      10
C                MOVE      5  PRICE
C                ELSE
C                MOVE      10 PRICE
C                ENDIF
C                ENDIF
C

```

RPG III

```

if (age <= 2)
  price = 0;
else if (age <= 10)
  price = 5;
else
  price = 10;

```

Java

```

C          IF      AGE <= 2
C          EVAL    PRICE = 0
C          ELSE
C          IF      AGE <= 10
C          EVAL    PRICE = 5
C          ELSE
C          EVAL    PRICE = 10
C          ENDIF
C          ENDIF
C

```

RPG IV

note single statement in body so braces not required



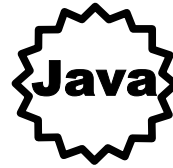
Conditional Operator

IBM

- **Conditional operator '? :'**
 - ▶ also called a *ternary* operator
- **Short form for if statement**
 - ▶ when only binary decision to make

```
result = (idx == 20) ? 30 : 35;
```

```
// same as...  
if (idx == 20)  
    result = 30;  
else  
    result = 35;
```





SELECT vs switch

IBM

RPG

```
C      SELECT
C      WHEN day = MON
C*     do something
C      WHEN day = TUE
C*     do something
C      WHEN day = WED
C*     do something
C      WHEN day = THU
C*     do something
C      OTHER
C*     do something
C      ENDSL
```

Java

```
switch (day)
{
  case MON:
    // do something
    break;
  case TUE:
    // do something
    break;
  ....
  default:
    // default code
} // end switch statement
```

Improved readability over nested IFs
Structures are similar in both languages!



Same But Different

IBM

**RPG
IV**

- Each **WHEN** expr evaluated until true
- Code executed until next **WHEN**

Java

- **switch** expression evaluated
- Result compared to each **case**
- In first match, code executed until "**break;**" or end of **switch**

RPG SELECT	Java Switch
SELECT	switch
WHEN or WHENxx	case
OTHER	default
ENDSL	end brace }

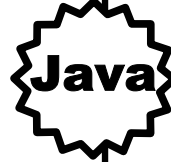


Breakless Switch

IBM

```
switch (day)
{
    case 1:
    case 2:
    case 3:
        // Mon-Wed code
        break;
    case 4:
    case 5:
        // Thur-Fri code
        break;
    default:
} // end switch statement
```

Control goes to first "case" that matches the expression, then executes until "break" is encountered, or the end brace





Looping Around

IBM

- RPG and Java, like all other languages have three main loops, they are...

RPG	JAVA
<pre>C start DO limit index C* : C ENDDO DO</pre>	<pre>for (initialization; condition; increments) { // body } FOR</pre>
<pre>C DOW expression C* : C DO-WHILE ENDDO</pre>	<pre>while (expression) { // body } WHILE</pre>
<pre>C DOU expression C* : C DO-UNTIL ENDDO</pre>	<pre>do { DO-WHILE // body } while (expression);</pre>

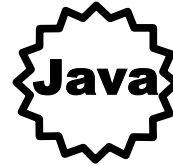


for-loop

IBM

- (A)** ● <declare> and initialize index variable
- (B)** ● loop while true
- (C)** ● increment / decrement index

```
static final int MAX = 10;
(A) (B) (C)
for (int idx=0; idx < MAX; idx++)
{
    // body;
}
```



```
C          1          DO  10          I
C*
C          ...
C          ENDDO1
```



```
C*  initial-value DO          Limit-value  index
C   1          DO          10          I
C*  ...
C*  ENDDO          Increment-value
C   ENDDO          1
```

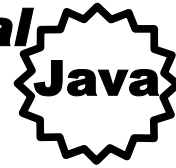




for-loop Parts

IBM

- **All three parts are *optional***
- ▶ **Only convention that:**
 - first part is for initing index variable,
 - expression is for comparing index value and
 - increment is for incr'tg/decr'tg index value
- **All three can even be empty!**



```
for ( ; ; )  
    System.out.println("looping...");
```

Never ending loop!



for-loop Parts

IBM



- Simple bodies can be done in incrementing part versus body
 - ▶ Comma-separated statements

```
for (idx = 0; Blank out entire array  
    idx < myCharArray.length;  
    myCharArray[idx] = ' ', idx++)  
;
```

all work done in increment part. No need for body

two statements, comma separated



New FOR-loop in RPG!

IBM

Example 1: n!

```

C*RN01Factor 1-----Opcode----Factor 2-----Result-Field
C*
C          EVAL          Factorial = 1
C          FOR           i = 1 to n
C          EVAL          Factorial = Factorial * i
C          ENDFOR

```



If n = 5,
 $n! = 5 * 4 * 3 * 2 * 1 = 120 \dots$

Example 2: Last non-blank character

```

C*RN01Factor 1-----Opcode----Factor 2-----Result-Field
C*
C          FOR           i = %len(SayWhat) DOWNT0 1
C          IF           %SUBST(SayWhat:i:1) <> ' '
C          LEAVE
C          ENDIF
C          ENDFOR

```

if SayWhat =
 'New For RPG4 ',
 Last non-blank = 12

**Java Skills
 Transfer!**



while-loop

IBM



- (A)** ● loop while true
- (B)** ● set variable to force end of loop
- loop iterations ≥ 0

```
boolean in30 = false;
while (!in30) (A)
{
    if (endOfFile())
        in30 = true; (B)
    else
        readLine();
}
```

```
C          *IN30          DOWNE*OFF
C*
C          ...
C          END
```



```
C          DOW          *IN30 NE *OFF
C*
C          END
```



Free Form Factor 2



do-loop

IBM

Java

- (A)** ● loop *until* true
- (B)** ● set variable to force end of loop
- loop iterations ≥ 1

```
boolean in30 = false;
do
{
    if (endOfFile()) (B)
        in30 = true;
    else
        readLine();
} while (!in30); (A)
```

```
C          *IN30      DOUNE*OFF      RPG III
C*
C          ...
C          END
```

```
C          RPG IV      DOU          *IN30 NE *OFF
C*
C          ...          Free Form Factor 2
C          END
```



continue, break

IBM

Label:

Note: continue and break can specify a labeled loop to explicitly **iterate** or **leave**

A

```

out: for (int i=0; i < 10; i++)
{
  for (int j=0; j < 10; j++)
  {
    if (intArray[i][j] == -1)
    {
      // some code
      continue out;
    }
    if (intArray[i][j] == -2)
      break;
  } // end inner for-loop
  // outside inner loop
} // end outer for-loop

```

Java

A

RPG IV

```

C      DOW      RECORDN = 2938174
C      IF       CODE='A1'
C      ITER
C      ENDIF
C      LEAVE
C      ENDDO

```



Operators: Relational

IBM

Operation	Java Operator	RPG Op-Codes	RPG IV Operator
Equal	==	EQ	=
Not Equal	!=	NE	<>
Greater Than	>	GT	>
Less Than	<	LT	<
GT or Equal	>=	GE	>=
Lt or Equal	<=	LE	<=
Or	 	ORxx	OR
And	&&	ANDxx	AND
Negation	!	NOT	NOT



Relational Example

IBM



```

if ( (age <= 2) ||
      ( (age >= 65) &&
        (currDay == SENIORS_DAY) ) )
price = 0;

```

note double equals: ==

```

C  AGE      IFLE 2
C  AGE      ORGE 65
C  CURDAY   ANDEQSENDAY
C           MOVE 0      PRICE
C           ENDIF

```



```

C  IF      (age <= 2) OR
C          ((age >= 65) AND
C           (currdays = SENIORS-DAY) )
C  EVAL   price = 0
C  ENDIF

```





Operators: Math

IBM

Operation	Java Operator	RPG Op-Codes	RPG Operator
Add	+	ADD, Z-ADD	+
Subtract	-	SUB, Z-SUB	-
Multiply	*	MULT	*
Divide	/	DIV	/
Modulus	%	DIV and MVR	n / a
Power	Use exp or pow in Math class	n / a	**



Math Examples

IBM

RPG III

```
C* A = B+C
C           B           ADD C           A           50
C* A = (B+C)/12
C           B           ADD C           A           50
C           A           DIV 12          A
```

RPG IV

```
C EVAL a = b + c
C EVAL a = (b+c)/12
```

Java

```
a = b + c;
a = (b+c)/12;
```

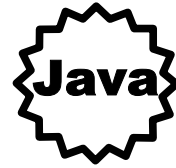


Contracted Assignment

IBM

- **What does this mean?**

```
X += 10;
```



- **Answer: short form for...**

```
X = X + 10;
```

- **All binary operators supported:**

```
X *= 10; X /= 2; Y -= 1;
```

Same as using **ADD** op-code
in *RPG* and *not* specifying
factor 1 value



Increment, Decrement

IBM

- **What does this mean?**

`x++;`

- **Answer: short form for**

`x = x + 1;`

- **Also supports decrementing:**

`x--;`

- **Can be before or after variable:**

`++x; --x;`

same
as C
and C++

more...



Increment++

IBM

● Always changes variable

```
if (X++ > 10)
```

X is
incremented

X==10?
result == false

● Prefix:

- Increment variable, use value

```
X = 10;
```

```
Y = ++X + 2;
```

x=x+1;
y=x+2;

Y == 13
X == 11

● Suffix:

- Use value, increment variable

```
X = 10;
```

```
Y = X++ + 2;
```

y=x+2;
x=x+1;

Y == 12
X == 11



Bitwise Operators

IBM

- **RPG has op-codes**
 - ▶ **TESTB, BITON, BITOFF**
- **Java has operators...**

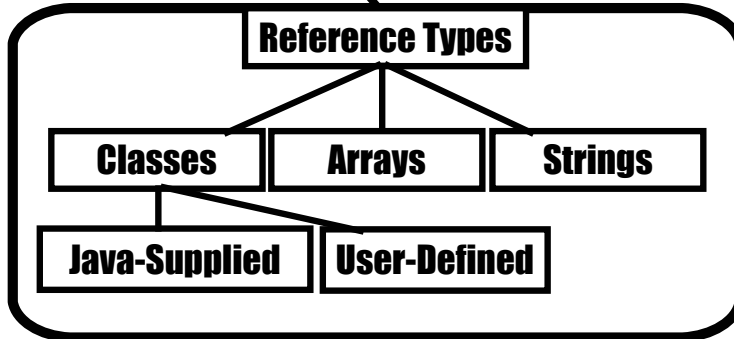
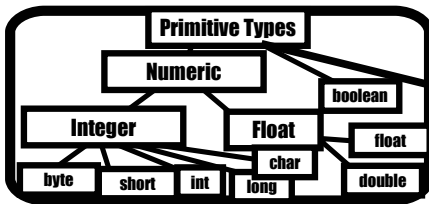
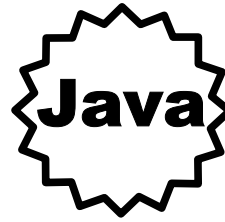
Operator	Meaning
&	Bitwise AND
	Bitwise OR
^	Bitwise Exclusive OR
~	Bitwise negation
<<	Left Shift
>>	Right Shift
>>>	Zero fill right shift

They work only on integer types!



Reference Data Types

IBM





Disclaimer

IBM

- **The following information will take some time (and more reading) to absorb**
 - ▶ **RELAX!**

★ **Focus on the "how" ...**

★ **..the "why" will follow with time**



Using Classes

IBM

- **To use a class, you must do two things:**

"how"

- **Declare an *object reference variable*:**

- Declare a variable using class as the type:

```
MyClass myVariable;
```

- ***Instantiate an object using "new"***

```
myVariable = new MyClass();
```



Objects

IBM

- **Object reference variables are**

- ▶ **merely pointers, or *references*, to objects**
- ▶ **initially point to "null"**
 - a keyword in Java

- **The new operator:**

- ▶ **allocates memory for the class ("*instantiates*")**
 - Total memory needed by all global variables in class

- **Allocated memory known as**

- ▶ ***object* or**
- ▶ ***instance* of class**



```
public class Customer
{
    private int      id;
    private String   name;
    private String   address;
    private int      phone;
    private BigDecimal accountBalance;

    public void setId(int custId)
    {
        id = custId;
    }

    public boolean readInfo()
    {
        boolean readok = false;
        // read customer info from database
        return readok;
    }
    // more methods. . .
}
```

Object Example



```
Customer aCust =
    new Customer();
```

you can declare & instantiate in one step!

how to call the methods?



Dot Operator

IBM

● To call a method

- ▶ use *dot operator* on object reference variable

```
public class ProcessCustomer
{
    public static void main(string args[])
    {
        Customer aCust = new Customer();
        aCust.setId(100012);
        aCust.readInfo();
    }
}
```

Must use object reference variable,
not Class name

Can also
access non-
private
variables
with dot
operator



Why Objects?

IBM

"why"

● Why must we instantiate?

- ▶ Because you can instantiate more than one!

```
Customer cust1 = new Customer();  
cust1.setId(100011);  
  
Customer cust2 = new Customer();  
cust2.setId(100012);  
.  
.  
.
```

● Each gets their own memory

- ▶ Each hold unique values for their variables
 - Hence we call global variables "*instance variables*"



Notes on Objects

IBM

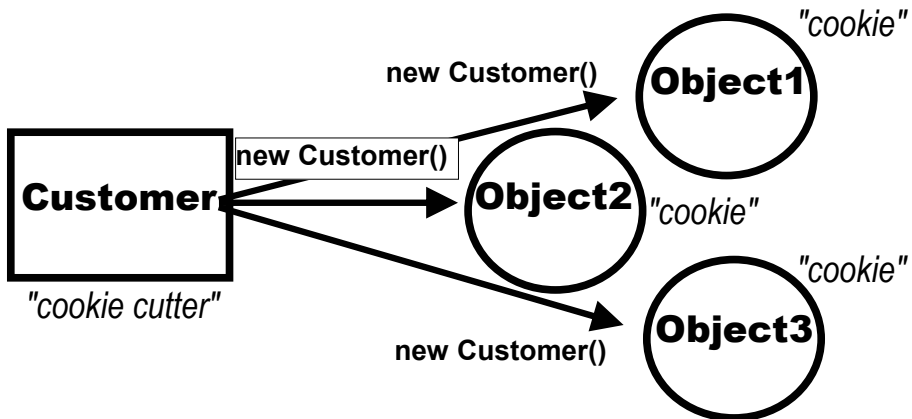
- **Classes are like templates**
 - ▶ or "cookie cutters"

- **Classes have no memory allocated**
 - ▶ **Objects have the actual memory**
 - object == "instance of class"
 - object == actual "cookies"



Class vs Objects

IBM



- ▶ **Classes like DDS source members?**
- ▶ **Objects like compiled *FILE objects?**
- ▶ **"new" operator like DDS compiler?**



Class/Object Examples

IBM

• Possible classes

- ▶ **Customer**
- ▶ **Employee**
- ▶ **StateTax**
- ▶ **CustomerId**
- ▶ **EmployeeId**
- ▶ **Payroll**
- ▶ **Order**
- ▶ **PushButton**

- Eg, in a GUI application

- Each instance might want different label text



Equating References

IBM

step 1

```
Master object1 = new Master();
```

```
Master object2 = new Master();
```

Object1

Object2

step 2

```
object1 = object2;
```

copies addresses!

*Nobody points to object1 now so it is swept up by **Garbage Collector***



Object Example 2

IBM

- **Consider a Stack class**
 - ▶ **for managing LIFO list of integers**

```
public class Stack
{
    private int list[] = new int[100];
    private int topIndex = 0;

    public void push(int topValue)
    {
        list[topIndex++] = topValue;
    }
    public int pop()
    {
        return list[--topIndex];
    }
}
```

instance
variables

Warning:
no error
checking!



Using Stacks

IBM

- **Objects allow us multiple stacks**
 - ▶ **simultaneously**

```
Stack myList = new Stack();// allocate instance of stack
Stack myList2 = new Stack();// allocate another instance

myList.push(100);           // stack contents: 100
myList.push(200);          // stack contents: 100, 200

myList2.push(1000);        // stack2 contents: 1000
myList2.push(2000);        // stack2 contents: 1000, 2000

int topValue;              // declare an integer variable
topValue = myList.pop();    // topValue:200
topValue = myList2.pop();   // topValue: 2000
```



Calling Java Methods

IBM

► Three ways to call methods:

- **Assignment statement**
 - returned result is saved in a variable
- **Expression**
 - returned result used in expression but not saved
- **Expression**
 - Runs the method and disregards return value

RPG	Java
EVAL myVar = myProc(p1 : p2)	myVar = myObject.myProc(p1,p2);
IF myProc(p1 : p2) = 10	if (myObject.myProc(p1,p2) == 10)
CALLP myProc(p1 : p2)	myObject.myProc(p1, p2);

EVAL myVar = noParms	myVar = myObject.noParms();
IF noParms = 10	if (myObject.noParms() == 10)
CALLP noParms	myObject.noParms();



Overloading

IBM

• Method Overloading

▶ Methods in same class *with same name!* But:

- Number or type of parameters are different
 - method name + nbr and type of parms == "***signature***"

▶ Official name 'method overloading'

```
public int max(int parm1, int parm2)
{
    // code to return max of two integers
}
public float max(float parm1, float parm2)
{
    // code to return max of two floats
}
```



Static Variables

IBM

- **Java variables can be *static* :**
 - ▶ **Use static modifier (like RPG's *STATIC* keyword)**
 - Cannot specify it on *local* variables in methods!
 - ▶ **Static variables are called *class variables***
 - Versus instance variables
 - ▶ **All objects share same value for static vars**
 - Qualify *with the class name* to access them

```
class RentalCar
{
    static int totalRented = 0;
    public void rentCar()
    {
        // . . .
        ++totalRented;
    }
}
```

```
if (RentalCar.totalRented > MAX_CARS)
```



Static Methods

IBM

• Methods can be static too

- ▶ called *class methods*
- ▶ Equivalent to standalone procedure

- Call by qualifying with class name, not obj ref variable

- ▶ **Cannot reference instance variables in the method**

```
class MyHelperRoutines
{
    // static method...
    static int max(int p1, int p2)
    {
        if (p1 > p2)
            return p1;
        else
            return p2;
    }
}
```

`int maxvalue = MyHelpers.max(1000,2000);`



If your method does not reference or use any instance variables, it should be static!



Constructors

IBM

● **Classes can have *constructors***

- ▶ **Special methods identified by:**
 - Same name as class
 - No return value specified (not even **void**)
- ▶ **Called by JVM when object created with new**
 - Right after allocating memory for the object
- ▶ **Your opportunity to do initialization**
 - like ***INZSR** in **RPG**

```
public class MyClass
{
    public MyClass()
    {
        // . . .
    }
}
```

constructor



Parameters to Ctors

IBM

- **Constructors can take parms**

- ▶ **Declared same as in all methods**

- On method signature

- ▶ **Passed by caller in parens after new**

- `MyClass myClass = new MyClass(10);`

- ▶ **Usually to allow caller-specified initial values**

- For the instance variables

constructor
with
parameter

```
public class MyClass
{
    private int myVariable;

    public MyClass(int parm1)
    {
        myVariable = parm1;
    } // end ctor
}
```

"ctor" is common shorthand for "constructor"

November 4, 2003 | Las

gration

Ctor is the standard shorthand for the term "constructor"



Ctor Overloading

IBM

- **Constructors can be overloaded**

- ▶ **Same as with all methods**

- Number or type of parameters must be unique

- ▶ **Compiler, Runtime determine which to call**

- By matching number, type of **new** parameters

Constructor with no parms
called *default constructor*

MyClass obj1 =
new MyClass(10);

MyClass obj1 =
new MyClass(10,20);

```
private int myVariable;  
private int myOtherVariable = 0;
```

```
public MyClass(int parm1)  
{  
    myVariable = parm1;  
}
```

```
public MyClass(int parm1,int parm2)  
{  
    myVariable = parm1;  
    myOtherVariable = parm2;  
}
```



Constructor Example

IBM

Java

use "this()" to
call another
constructor

```
class AS400
{
    private String userId;
    private String password;

    AS400() // default constructor
    {
        this("PHIL", "GREATGUY");
    }
    AS400(String userId, String password)
    {
        this.userId = userId;
        this.password = password;
    }
} // end AS400 class
```

```
AS400 host1 = new AS400(); // Call default constructor
AS400 host2 = new AS400("GEORGE","OKGUY"); // Two parm ctor
```

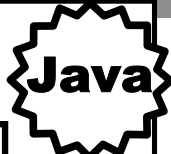


Full Example

IBM

```
/** Represents a single card in a deck */
public class Card
{
    // public constants...
    public static final int HEART = 0;
    public static final int CLUB = 1;
    public static final int SPADE = 2;
    public static final int DIAMOND=3;
    // private instance variables...
    private int    number; // value of card
    private int    suit;   // heart, spade, club, diamond
    private boolean played=false; // card been played yet?

    public Card(int number, int suit)
    {
        this.number = number;
        this.suit   = suit;
    }
    public int getNumber()
    {
        return number;
    }
    public int getSuit()
    {
        return suit;
    }
} // end of class Card
```



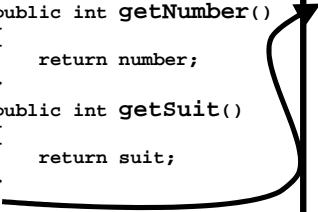
Constants

Instance variables

Constructor

public boolean isPlayed()
{
 return played;
}
public void setPlayed(boolean played)
{
 this.played = played;
}

Methods



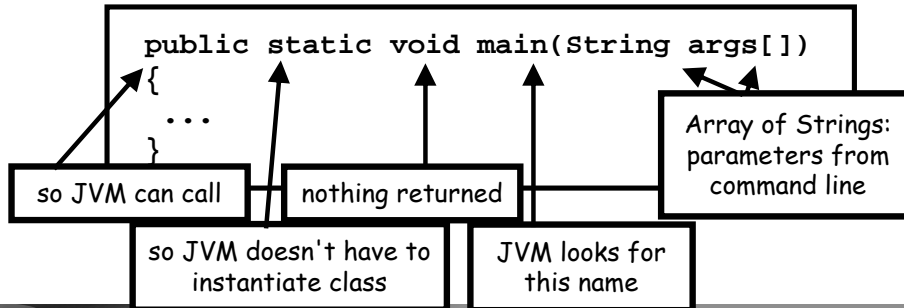


Main: RPG and Java



What	RPG IV	Java
How called	CALL command	java command
What compile unit gets control	first *MODULE without NOMAIN keyword	class identified on java command
What code gets control	first C-specs	main method

● Java's main method *must* look like:





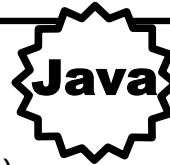
Writing To Console

IBM

```
* Prototype of this program main entry
DMAIN          PR                EXTPGM('HWORLD')
D STRING       1000A            OPTIONS(*VARSIZE)
* Definition of this program main entry
DMAIN          PI
D STRING       1000A            OPTIONS(*VARSIZE)
* Global variables
DOutString     S                52A
* Main logic
C              EVAL            OutString = 'Input: ' +
C                                     %TRIMR(%SUBST(STRING:1:45))
C      OutString  DSPLY
* End of program
C              MOVE            *ON                *INLR
```



```
// main class
public class HelloWorld
{
    // main method
    public static void main(String[] args)
    {
        // print first parameter passed
        System.out.println("Input: " + args[0]);
    }
}
```





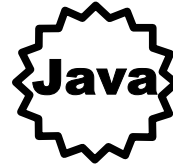
Review

IBM

```
public class Time
{
    private int hour, minute, second;

    public Time(int hour, int minute, int second)
    {
        this.hour = hour;
        this.minute = minute;
        this.second = second;
    }
    public String toString()
    {
        return "Time: " + hour + ", " +
            minute + ", " + second;
    }
    public static void main(String args[])
    {
        Time torontoTime = new Time(08,30,0);
        Time sanFranTime = new Time(05,30,0);
        System.out.println(torontoTime);
        System.out.println(sanFranTime);
    }
}
```

Instance
variables



Object reference
variables

Often, **main** is used for
testing non-initial classes



Arrays, I Need Arrays

IBM



▶ **Array Types:**

✓ One-dimension

✓ Tables

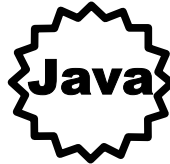
✓ Dynamic APIs

▶ **Initializing:**

✓ Compile time

✓ Pre-Runtime

✓ Runtime



▶ **Array Types:**

✓ One-dimension

✓ Multi-dimension

✓ Hashtable class

✓ Vector class

▶ **Initializing:**

✓ Compile time

✓ Runtime

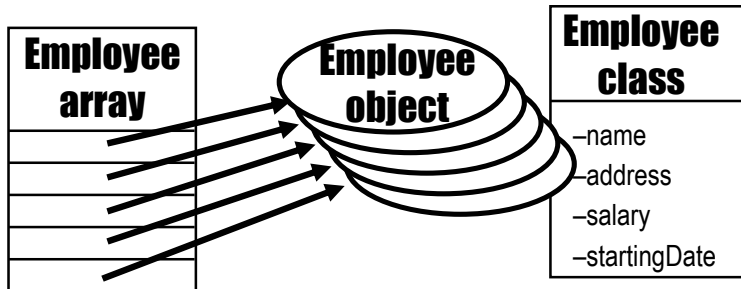


What About MODs?

IBM

● What about Multiple Occurring Data Structures?

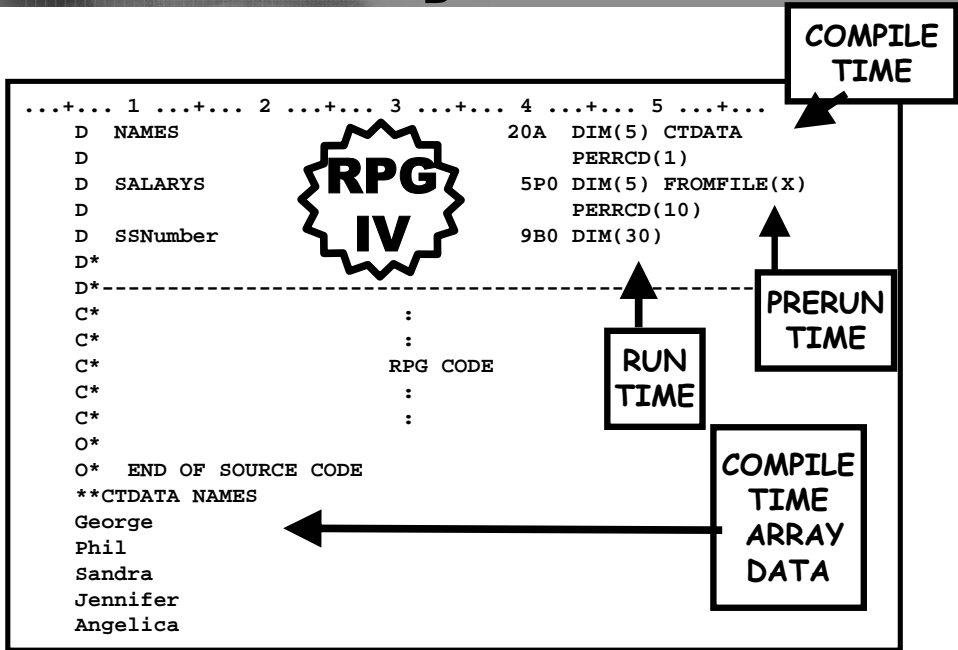
- ▶ In RPG these are arrays of structures
- ▶ In Java these are arrays of objects
 - ✓ The object's class = the DS in RPG





Arrays in RPG

IBM





Arrays in Java

IBM

• Declaration

```
int    thisArray[ ];  
long   anotherArray[ ][ ];  
char[ ] orThisOne[ ];
```

Java

• Declaration and Definition

```
int thisArray [ ]      = new int[1000];  
long anotherArray [ ][ ] = new long[10] [10];  
char[ ] orThisOne[ ]  = new char [20] [20];
```

- ▶ Spacing not important
- ▶ # bracket pairs = # dimensions
- ▶ Type is same for all elements
- ▶ Arrays are objects! Require new

#elements

[] versus ()



Run Time Init'n

IBM

```
class TestMultiArrayRT
{
    public static void main(String args[])
    {
        int rtArray[][] = new int[3][3]; // Two dim array
        int value = 1;
        // Loop through all rows...
        for (int xIdx=0; xIdx < rtArray.length; xIdx++)
        {
            // Loop through all columns...
            for (int yIdx=0; yIdx < rtArray[xIdx].length; yIdx++)
            {
                rtArray[xIdx][yIdx] = value++; // assign and incr't
                System.out.print(rtArray[xIdx][yIdx] + " ");
            } // end inner for loop
            System.out.println();
        } // end outer for loop
    } // end main method
} // end TestMultiArrayRT class
```

length is array
instance variable

arrays are
zero-based
in Java!

Use [var] to access
elements



Compile Time Init'n

IBM

- **Java allows initializing at declaration time (*compile time*):**

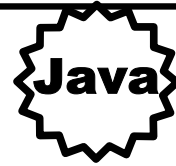
```
String employee[] = {"ABC", "DEF", "GHI", "JKL"};
```

Note: String objects are covered shortly

```
employee[0] = "ABC"  
employee[1] = "DEF"  
employee[2] = "GHI"  
employee[3] = "JKL"
```

- **Special Java syntax:**

- ▶ **Values specified between curly braces**
- ▶ **Semi-colon needed after last brace**
- ▶ **Values for each element separated by commas**
- ▶ **No need to use new operator (implied)**





Intro to Vectors

IBM

- **In both RPG and Java, once an array is created, its size is fixed**
 - ▶ **it cannot be resized**
 - ▶ **However, Java at least allows deferring creation (using new) until after size has been determined**
- **However, Java also offers *Vectors*!**
 - ▶ **Vector is a class in `java.util` package**
 - To use, need `"import java.util.*;"`
 - ▶ **Vectors are like dynamically sizable arrays**
 - do not need to specify initial size
 - size grows as needed when items are added



Using Vectors

IBM

- **To use Vectors:**

- ▶ **Create empty Vector by instantiation**
- ▶ **Add items using addElement method**

```
Vector myVector = new Vector();
String inputString = getFirstInput();
while (inputString != null)
{
    myVector.addElement(inputString);
    inputString = getNextInput();
}
```

- ▶ **Query number of elements using size method**
- ▶ **Query specific element using elementAt method**

```
for (int idx = 0 ; idx < myVector.size() ; idx++)
    System.out.println(myVector.elementAt(idx));
```



Hashtables

IBM

- **Java supplies a class for simple lookup tables**

- ▶ **Hashtable in package java.util**
- ▶ **Contains pairs of objects**
 - A key object and a value object
- ▶ **Objects can be of any class**
- ▶ **Use put to insert, get to retrieve**

```
Hashtable customers = new Hashtable();
customers.put("011002", "Phil Company");
customers.put("110034", "George Limited");
. . .
String georgeEntry = customers.get("110034");
```

insert by key,
value pair

search by key,
get value



String Basics

IBM

- **Strings are *objects* in Java**
 - ▶ of the class `String` (in `java.lang` package)
- **Language has special support:**
 - ▶ You can concatenate with the `+` operator
 - ▶ You don't *have* to use the `new` operator

```
String text1 = new String("George");  
String text2 = new String("Phil");  
String finalText = new String(text1);  
finalText = finalText.concat(" and ");  
finalText = finalText.concat(text2);
```

OR



```
String text1 = "George";  
String text2 = "Phil";  
String finalText = text1 + " and " + text2;
```



String Gotchas

IBM

•Be careful of these common mistakes:

•not assigning result of methods:

String textField = "Java";
textField.concat(" and RPG");

textField = textField.concat("and RPG");

•comparing with '==' versus equals method

if (name == "Bob")

if (name.equals("Bob"))

•copying with '=' versus clone method

String newName = oldName;

String newName = oldName.clone();

String is an *immutable* class: all methods return new objects versus changing existing



Strings: Java vs RPG

IBM

RPG o/c	RPG built-in	Description	Java Method(s)
CAT (or '+')		<i>Concatenate two strings</i>	concat(string) or '+' operator
SUBST	%SUBST	<i>Extract a substring from a string</i>	substring(int start, int end) or substring(int start)
SCAN	%SCAN	<i>Scan for a substring</i>	indexOf()
	%TRIM	<i>Trim begin, end blanks</i>	trim()
	%LEN	<i>Return length of string</i>	length()
XLATE		<i>Translate a string</i>	<i>Not Available</i>
CHECK		<i>Check for characters</i>	<i>Not Available</i>
CHECKR		<i>Check in reverse</i>	<i>Not Available</i>
	%TRIML	<i>Trim leading blanks</i>	<i>Not Available</i>
	%TRIMR	<i>Trim trailing blanks</i>	<i>Not Available</i>
	%CHAR	<i>V4R2. Converts to string</i>	valueOf(datatype value) in String class
	%REPLACE	<i>(V4R2) Allows replacement of substring</i>	<i>Not Available</i>



Some String Methods

IBM

METHOD	DESCRIPTION
compareTo	<i>Compares two Strings lexicographically</i>
endsWith, startsWith	<i>Test if String ends or starts with the specified string</i>
equals, equalsIgnoreCase	<i>Compares this String to another, ignoring case</i>
getBytes	<i>Convert this String into a byte array</i>
getChars	<i>Copies characters from this substring into the destination character array</i>
regionMatches	<i>Tests if two String regions are equal</i>
toCharArray	<i>Converts this String to a new character array</i>
toLowerCase	<i>Converts all characters in String to lower case</i>
toUpperCase	<i>Converts all characters in String to upper case</i>
valueOf	<i>Converts primitive data type value to a String (this is a static method)</i>



Java Date / Time

IBM

Class	Pack- age	Description
Date	<code>java.util</code>	Simple date/time capture. No manipulation methods
Gregorian- Calendar	<code>java.util</code>	Rich date/time functionality, including comparing, adding, subtracting, extracting
SimpleDate- Format	<code>java.text</code>	For creating "formatting objects" that will format any given Date object to the specified format pattern
TimeZone	<code>java.util</code>	For creating timezone objects representing any timezone. Apply to GregorianCalendar or SimpleDateFormat objects to get equivalent date/time in that timezone



Manipulating Dates

IBM

●GregorianCalendar knows all about dates

```
import java.util.*;

public class TestDate
{
    public static void main(String args[])
    {
        GregorianCalendar gc = new GregorianCalendar();
        System.out.println("Date before addition: " + gc.getTime());
        gc.add(Calendar.DATE,2); // add two days
        System.out.println("Date after addition: " + gc.getTime());
        gc.add(Calendar.MONTH,2); // add three months
        System.out.println("Date after addition: " + gc.getTime());
        GregorianCalendar gc2 = new GregorianCalendar(2012,0,30);
        if (gc.before(gc2))
            System.out.println("Yes, it is");
    }
}
```

Date before addition: Thu Sep 30 22:02:20 EDT 1999
Date after addition: Sat Oct 02 22:02:20 EDT 1999
Date after addition: Wed May 16 22:02:20 EDT 2001
Yes, it is

Java

Cool methods:

- add
- before / after
- isLeapYear
- get (extracting parts)

zero-based month!



Agenda

IBM





OO Terminology

IBM



**RPG IS 
NOT OO!!**



**JAVA 
IS OO !!!**



● What does "*Object Oriented*" mean?

Three attributes:



Encapsulation



Inheritance



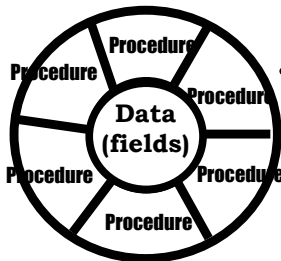
Polymorphism



OO: Encapsulation

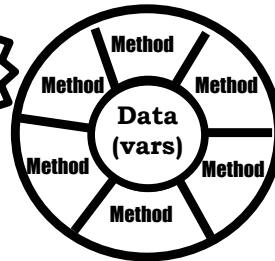
IBM

- Hide data from direct public programmer access
- Force access only via procedures or methods



**RPG
IV**

Java



- ReUse with Modules and Service Programs of modules
- "Expose" certain procedures or data with EXPORT keyword
- ReUse with Classes and Packages of classes
- "Expose" certain methods or variables with public modifier keyword



OO: Inheritance

IBM

```
// class Employee
public class Employee
{
    string name;
    int salary;
    public Employee(string id, int sal
    {
        name = id;
        salary = sal;
    }
    public void printEmployee()
    {
        System.out.print("My name is"+name);
    }
}
```

```
// class SubEmployee
public class SubEmployee extends Employee
{
}
```

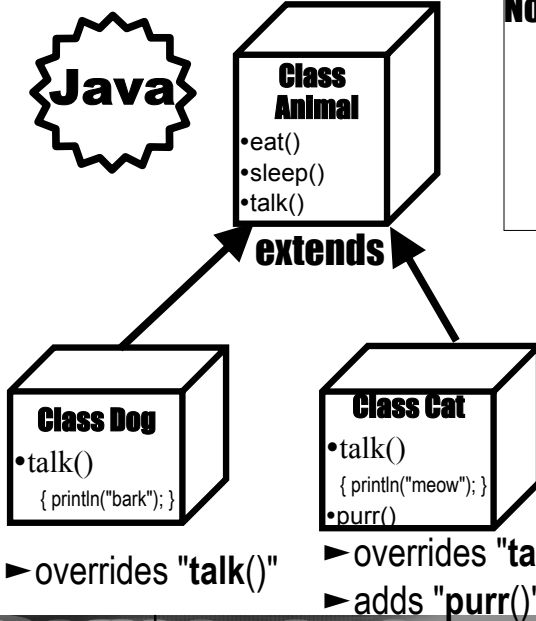
- ▶ Child class **extends** parent class
 - ▶ inherits methods, variables
- ▶ Child can also:
 - ▶ Add new methods, variables
 - ▶ Override methods

Java



OO: Inheritance

IBM



Notes:

- Can only extend *one* class
- Extended class called *parent*
- Extending class called *child*
- Signature important when overriding methods

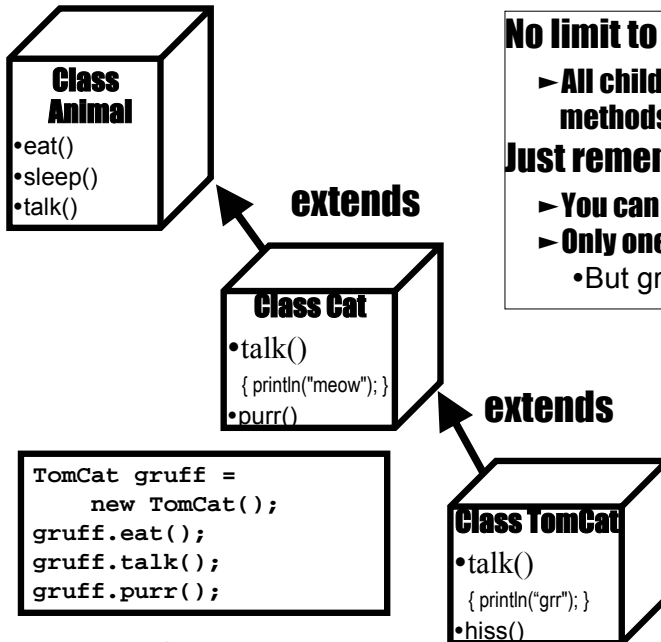
```
Cat fluff = new Cat();
fluff.eat();
fluff.talk();
fluff.purr();
```

can call inherited methods as though locally defined



OO: Inheritance

IBM

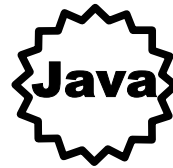


No limit to inheritance tree

- ▶ All child classes inherit methods of all parents

Just remember

- ▶ You can only extend one class
- ▶ Only one immediate parent
 - But grandparents allowed





OO: Polymorphism

IBM

• When **ClassChild** *extends* **ClassParent**:

"how"

▶ You can assign **ClassChild** objects to **ClassParent** object reference variables

- either direct child or indirect child (eg: grandchild)

```
ClassParent obj = new ClassChild();
```

▶ You can then call any method in the **ClassChild** object

- as long as it is defined in the parent class too
- if not, compile will fail (it searches declared class type)

```
obj.commonMethod();
```



OO: Polymorphism

IBM

"why"

• Why is this important?:

- ▶ You can write generic code that calls parent's "base" methods

- But actually calls child's methods at runtime, if they are overridden in the child class

```
Class ParentClass
{
    public void doSomething()
    {
        // ...
    }
}
```

```
ParentClass obj1 =
    new ParentClass();
ParentClass obj2 =
    new ChildClass();

obj1.doSomething();
obj2.doSomething();
```

```
Class ChildClass extends ParentClass
{
    public void doSomething()
    {
        // different algo
    }
}
```

no "if logic" needed!

"Poly"...."morphism"
"Many".."faces"



Polymorphism Example

IBM

```
public class Employee
{
    protected int      id;
    protected String   name;
    protected BigDecimal wage;
    protected BigDecimal hoursWorked;

    public Employee(int iD)
    {
        id = iD;
        // not shown: reading info from database
    }
    public BigDecimal calculatePay()
    {
        return wage * hoursWorked;
    }
    // other methods: getName, setName, etc
}
```

**Consider this
"parent" class**

"protected" modifier allows
only this class and child
classes access



What about
salaried
employees?



Polymorphism Example

IBM

```
public class SalaryEmp extends Employee
{
    public SalaryEmp(int iD)
    {
        id = iD;
        // not shown: reading info from database
    }
    public BigDecimal calculatePay()
    {
        return wage / 26;
    }
}
```

Constructors are not inherited!

Could call parent ctor with super(...);

This method overridden from parent

```
public class ContractorEmp extends Employee
```

```
public class PartTimeEmp extends Employee
```

```
public class xxxEmp extends Employee
```




Polymorphism Example

IBM

```
public class Payroll
{
    public static void main(String args[])
    {
        Employee allEmps[] = new Employee[100];
        // populate with Employee, SalaryEmp,
        // and PartTimeEmp objects (not shown)
        for (int idx=0; idx < allEmps.length; idx++)
        {
            BigDecimal pay =
                allEmps[idx].calculatePay();
            // not shown: rest of code
        }
    }
}
```

Elegant!

Calls appropriate method based on object type

Change-resistant!
New child class can be added anytime



OO References

IBM

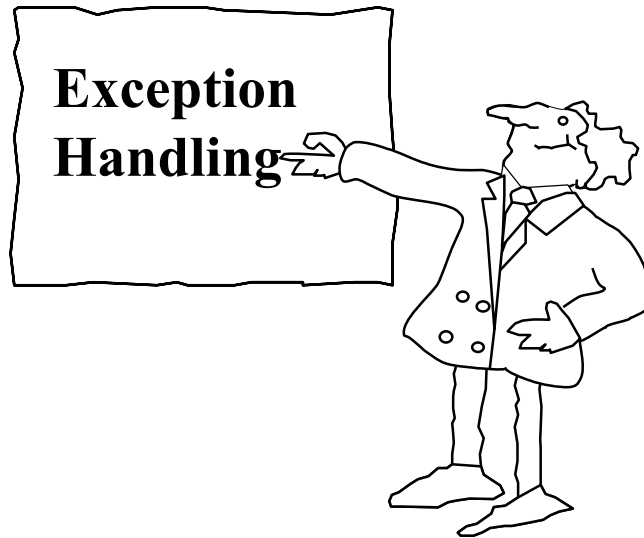
- **Recommended reading:**
 - ▶ **OBJECT-ORIENTED DESIGN IN JAVA**
 - Stephen Gilbert and Bill McCarty
 - WAITE GROUP PRESS
 - ▶ Any book on UML
 - ▶ Any book on OOA and D
 - ▶ Any book on Design Patterns

- **Look at any online bookstore**
 - ▶ www.amazon.com
 - ▶ www.chapters.com
 - ▶ etc



Agenda

IBM





Exceptions in Java

IBM

● Java has "exceptions"

- ▶ objects of classes that extend Exception
- ▶ Java supplies many existing Exception classes
- ▶ You can create your own Exception classes

```
public class MyException extends Exception
```

● Methods can *throw* exceptions

- ▶ by using throw operator with exception object

```
if (inputParameter < 100)  
    throw new IOException();
```

- ▶ Usually done when error situation detected

- preferred over returning special return codes



Throwing Exceptions

IBM

- **If a method throws an exception:**

- ▶ **It must declare which exceptions it throws using the *throws* clause on method declaration**

```
public void MyMethod()  
    throws MyException, OtherException  
{ ... }
```

- ▶ **Many Java-supplied methods throw exceptions**

- **To call such a method:**

- ▶ **calling code must be inside a try block**

```
try {  
    myObj.myMethod();  
}
```



Catching Exceptions

IBM

- **If any method call throws an exc:**

- ▶ **The appropriate catch block is executed**

```
try {  
    myObj.myMethod();  
}  
catch (MyException exc)  
{  
    System.out.println(  
        exc.getMessage());  
    exc.printStackTrace();  
}  
catch (OtherException exc)  
{  
    // do something  
}
```

Comes here is exception of class MyException or its children is thrown

All exceptions support these methods

Must try to catch all exceptions listed in throws clause: else no compile!



Java vs OS/400

IBM

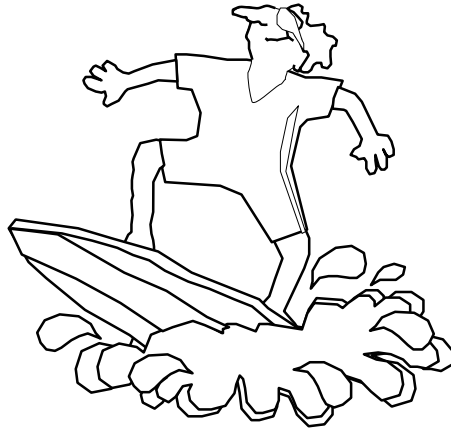
- **Java exceptions similar to OS/400 exceptions!**
 - ▶ **exception objects like OS/400 messages**
 - ▶ **throw like SNDPGMMSG**
 - ▶ **try/catch like MONMSG**
- **catch with a parent class is like**
 - ▶ **using MONMSG with message ID ending in 00**
 - Catches any exception in this family (or range)
- **catch with Exception class like**
 - ▶ **using MONMSG with CPF9999**
 - Catches any exception!



Thanks

IBM

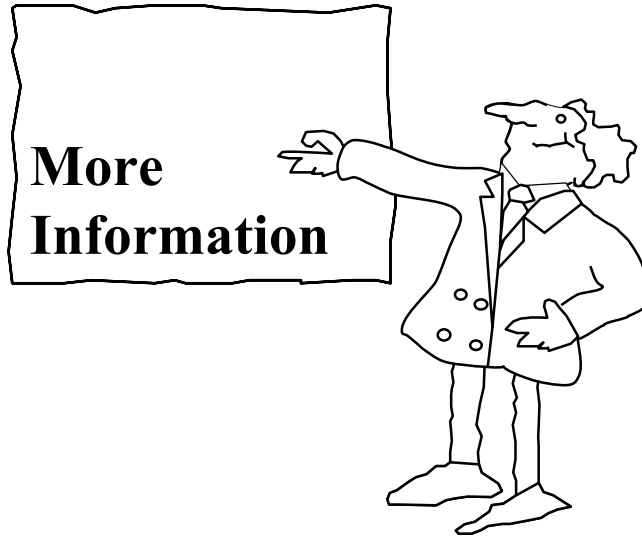
**Thanks
for
coming!!**





Agenda

IBM

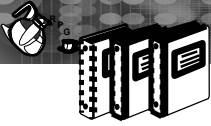




Websites

IBM

Website URL	Description
www.ibm.com/software/awdtools/wds400	IBM WebSphere Development Studio for iSeries
www.ibm.com/websphere/developer	WebSphere Developer Domain
www.ibm.com/java www.ibm.com/series/java	IBM Java
www.ibm.com/websphere www.ibm.com/series/websphere	IBM WebSphere IBM iSeries WebSphere
www.java.sun.com/products	Sun Java
www.ibm.com/series/infocenter.html www.ibm.com/rochester/as400bks	IBM iSeries online books and help
www.ibm.com/redbooks	IBM Redbooks



Books

IBM

Book, URL	By, ISBN
-Java for RPG Programmers -Java for COBOL Programmers -Student WorkBook for Java for RPG and COBOL Programmers www.mcpressonline.com	Phil Coulthard, George Farr. IBM Press.
JAVA and the AS/400 www.29thStreetPress.com	Daniel Darnell. ISBN 1-58304-033-1
Java Application Strategies for iSeries and AS/400 2nd Edition www.mc-store.com/mc-store/	Don Denoncourt. ISBN 1-58347-025-5
Core Java series www.amazon.com	Now a series of 3 books. Horstmann, Cornell
Teach Yourself Java in 21 Days www.amazon.com	Rogers Cadenhead, Laura Lamay. ISBN 1575213907



Trademarks & Disclaimers



© IBM Corporation 1994-2003. All rights reserved.

References in this document to IBM products or services do not imply that IBM intends to make them available in every country.

The following terms are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both:

AS/400	IBM(logo)		
AS/400e	iSeries		
e (logo) business	OS/400		
IBM			

Lotus, Freelance Graphics, and Word Pro are registered trademarks of Lotus Development Corporation and/or IBM Corporation. Domino is a trademark of Lotus Development Corporation and/or IBM Corporation.

C-bus is a trademark of Corollary, Inc. in the United States, other countries, or both.

Java and all Java-based trademarks are trademarks of Sun Microsystems, Inc. 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.

ActionMedia, LANdesk, MMX, Pentium and ProShare are trademarks of Intel Corporation in the United States, other countries, or both.

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

SET and the SET Logo are trademarks owned by SET Secure Electronic Transaction LLC.

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

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

All 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 in this presentation 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. Contact your local IBM office or IBM authorized reseller for the full text of the specific Statement of Direction.

Some information in this presentation 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.

Photographs shown are of engineering prototypes. Changes may be incorporated in production models.