Batch Java for COBOL Programmers

Description

This series of courses is designed to teach IBM Batch Java to experienced COBOL programmers. It builds on the knowledge and experience of COBOL programmers, and highlights differences that may be confusing. (As a simple example, the verb "CONTINUE" has a different meaning in Java versus COBOL).

Additional, optional topics include web development, which can replace CICS front-end processing. For learning "batch" Java, the emphasis will be on file I/O rather than interacting with the user.

IBM's most recent update for Java, as of November 27, 2019, is Java SDK8 – version 8.0.6.0. This IBM release is fully compatible with Oracle's Java version 8, although Oracle has delivered many more releases after its Java 8.

Because this course is for experienced COBOL programmers, participants will have lots of practice converting COBOL statements into Java code, and COBOL data into Java data.

Suggested Durations

Each course in this series has a suggested duration. Durations can be lengthened by adding extensive case studies (generally 4 to 5 day each, and often ending with a presentation to the students' team leads and other management). Durations can be shortened, by eliminating topics or by presenting topics with less in-depth coverage. Eliminating workshops is not recommended.

Format

Lecture with extensive hands-on workshops to reinforce the lecture material.

Suggested Courses

- IDz for Stand-alone development (1 day). IDz practice is integrated into each course
- Java for COBOL Programmers part 1 (5 days)
- JUnit Testing and TDD (2 days)
- Debugging Java in IDz (1 day)
- Java for COBOL Programmers part 2 (5 days)
- Java for COBOL Programmers part 3 (5 days)
- Java for COBOL Programmers Special Topics (1 to 10 days)
- Running Batch Java in USS (5 days)
- Deployment with Endevor (length depends on client requirements)

Optional Additional Courses

- HTML/HTML5
- CSS/CSS3
- JavaScript
- Servlets
- JSPs and JSTL
- XML
- IDz Advanced Topics (2 to 8 days)

IDz (IBM Developer for z Systems) for stand-alone Development (1 day)

Tool Overview (brief)
- Supported environments and architectures
- Integrated into ADFz
- IDz/ADFz - Supported z System Software Languages and Data Sources
- Installation requirements/ Configuration (overview)

Editing Java Programs (with HelloWorld.java)

Hot keys

Typing Shortcuts

Building/running Java Programs (with HelloWorld.class)

Program analytics and search
- Execution Control Flow Diagram
- Logic/Branch Flow Filter View
- Data Flow Diagram and Elements

Code Coverage
- Measure Testing Quality and Application Coverage
- Integrates with and feeds IBM's ADDI test quality dashboard (Optional topic)

Code Review - Validates Code Quality, Consistency, Conformance

Edit searching
- Regex
- Hyperlinks to results

Code Reuse
- Templates
- Skeletons
- Snippits
- External copy

Enforcing client coding standards

Java for COBOL Programmers part 1 (5 days)

Every java statement and piece of code is matched to corresponding COBOL code.
Workshops include converting COBOL to Java.

Object Oriented Programming
- OOP concepts
- classes
- methods (vs functions)
- arguments (vs parmeters)
- Advantages (polymorphism, inheritance, encapsulation, abstraction, etc, are taught further into in these courses)

Documentation
- IBM Manuals
- Javadoc documentation

The Java Environment
- Portability/Security

Java Basics
- All with COBOL comparison
- Java is Object-Oriented
- The Three Pillars of OO
- Comments
- Primitive Data Types vs class Data Types
- casting
- Variables
- Literals
- Expressions
- Floating Point Operations
- Method Signatures
- Invoking methods
- Simple Statements
- Control statements (conditional, looping, branching, break, return, exit)
- Instantiating a Class
- Class Variables
- Scope
- Encapsulation
- GC – Garbage Collection
- Stack vs Heap

JavaBeans in batch
- attributes
- Getters (accessors)
- Setters (mutators)
- toString()
- equals()
- hashCode()
- Deep vs shallow compare
- compare()
- compareTo()
- interfaces

Developing Java Applications
- javac.exe vs java.exe
- JVM
- JDK
- JRE
- Build
- The javadoc and jdb Tools
- The package Statement
- The import Statement
- Packages, Classes, Files, Directories

package-info.java

Typical Operations
- Handling nulls
- Class Definitions
- Aggregation/Composition
- Method Overloading
- Constructors
- Access Specifiers
- Comparing Objects (equals)
- Class Variables and methods
- final Variables
- Finalization

Arrays and Strings
- Java Arrays
- Copying Array Elements
- String Objects
- String Methods
- String Concatenation
- Converting to String
- split method, basic usage
- StringBuffer
- StringBuilder

Regex
- patterns
- Compiling
- Use with split method

Exceptions
- try/catch
- Creating an Exception Class
- throwing exceptions
- finally
- Exception Types
- throws
- Exception handling
- Best Practices

Inheritance
- Inheritance vs Aggregation /Composition
- Protected Access
- Overriding Methods
- Constructor Chaining
- Inheritance and Finalization
- Abstract Classes
- Casting Between Class Types
- Use Eclipse to generate code
- Polymorphism

I/O Streams
- Introduction
- Class InputStream
- Class OutputStream
- Using InputStream and OutputStream
- File I/O
- Memory Stream
- Filtered Streams
- Buffered I/O
- Data Input
- Data Output
- Printed Output (to console)

Java for COBOL Programmers part 1 continued

Validating input
- Avoiding redundant code
- Best practices for location of code

Separation of concerns

Collections – Most common ones
- Arrays
- ArrayList
- Vector
- TreeMap
- Hashtable
- Code to the interface
- Wrapper Classes
- Customized wrapper classes
- thread-safety versus speed
- sorting/searching/expanding

Autoboxing/unboxing
- When to use or avoid
- Performance considerations
- Under the covers

Enums
- Simple and complex enums
- Adding methods
- switch with enums
- Under the covers
- Best practices with enum

Constants
- Consider possible locations, and make good choices
  (separate class, interface, enum, in class, in method, XML, etc.)

JUnit Testing and TDD (2 days)

JUnit Testing
- Testing Approach
- Typical Testing Stages
- Summary of Testing

Testing Steps
- JUnit Setup in Eclipse/RAD
- Create an Interface with Three Methods
- Generate Java code with empty methods
- Create Test Program
- @Test
- fail Method
- Run JUnit
- Add Tests
- "assert" test methods
- @Before
- @Before /@BeforeClass
- @After / @AfterClass
- Test Floating Point with Deltas
- JUnit Annotations / Order
- JUnit Rules for Test Class
- JUnit Terminology

TDD concepts

Test Runners
- Test Case
- Test Suite
- JUnitCore façade - Runner
- AllTests.java – JUnit Test Suite

More JUnit Testing
- assertSame versus assertEquals
- @Ignore – Good Practice
- Testing for Expected Exceptions
  - Old and new approaches
- @Test(expected=
- @Test (other options

Parameterized Tests
- @RunWith
- @Parameters
- Limitations in JUnit 4
- No Test Suites for Parameterized Tests

Theories
- Theories Testing
- Imports for Theories
- Parameterized Tests
- Theories Tests
- Theories Output
- @Theory
- @DataPoint

JUnit Best Practices

JUnit 5 vs JUnit 4
- Additional annotations
- Changed meanings from JUnit 4
- Jupiter
- Platform
- Vintage

Debugging Java in IDz (1 day)

Debugging Techniques

- Breakpoints
- Step mode
- Visuals for debugging
- Monitoring/changing variables/memory/expressions
- Jump – altering program flow
- Interactive statement reorder/playback
- Frequency counters
- Customizing displays
- Integrating with the editor
- White box (clear/glass box) and black box testing

Smart Testing

- Saving test customizations

Code Coverage

- Debugging with code coverage analysis

White box versus black box testing

- Why both are important

Eclipse-GUI-based vs 3270-based debugging

Java for COBOL Programmers part 2 (5 days)

Review (as necessary)
- Naming standards
- Interfaces
- Inheritance, abstract classes
- Arrays
- Strings
- StringBuffer
- Converting Strings to Numbers
- Line Input
- Tokenizer Classes
- Deprecated Methods
- Data types
- Passing Data types to a method
- Method names
- Constructors and initialization
- Object Cloning
- Garbage collection
- Static blocks
- CLASSPATH
- Packages and JAR files

Design Patterns
- What are Design Patterns
- Why they are important
- Creational Patterns
  - Singleton
  - Factory
- Builder Structural Patterns
  - Façade
  - Adapter
  - Composite
- Behavioral Patterns
  - Template
  - State
  - Observer
- Null Object

Architecture vs Framework vs Design Patterns

Interfaces
- In-depth: purpose, design and use
- Code to the Interface
- Holding default code
- Polymorphism with interfaces

Collections
- Review of collections from Java Part 1
- Array vs Collections /groups
- Terminology - Aggregates of Data
- Collections Framework
- Raw types and @SuppressWarnings
- LinkedList
- ArrayList vs LinkedList
- Using methods Specific to LinkedList
- Iterators
- Operations on Collections
- Converting between collections
- List, Map, Queue, Set, Collection, etc.
- Interfaces Hierarchy
- Typical Implementations
- ArrayList in-depth:
  - Constructors and Methods
- Iterator vs ListIterator
- Create a static List
- Thread Safety
- Synchronizing a collection
- Performance Strategies – Choosing a collection

Formatting numbers for output
- printf and format Methods
- Formatter converters, flags, specifiers
- DecimalFormat class
- NumberFormat class

Annotations
- Aspect-Oriented Programming and Java
- The Annotations Model
- Annotation Types and Annotations
- Built-In Annotations
- Annotations vs. Descriptors (XML)
- Standard annotations
- Reflecting annotations

Assertions
- When to use Assertions
- Enabling Assertions

Static Imports
- When to use and when to avoid
- Use with enums

varargs
- Variable-length argument lists

Bit manipulation

Central exception handling

Java for COBOL Programmers part 2 - continued

Object Serialization
- What is Serialization
- Serializable Objects
- Writing an Object
- Reading an Object
- Handling Exceptions

File I/O
- Review from Part 1 course
- Files and streams
- I/O exceptions
- File object
- File class methods
- Text and binary
- Object I/O
- File locking/sharing
- Multi-thread considerations

Regular Expressions
- Methods of the Matcher Class
- Syntax of Regular Expressions
- Methods
- Cascade/Telescope Methods & "this."
- Method Signatures and Overloading
- Enhanced Setter methods and constructors

JavaBeans
- Properties
- Packaging JavaBeans
- Multithreading Considerations

BigDecimal
- Handling COBOL COMP-3/Packed Decimal
- Why BigDecimal?
- BigDecimal arithmetic
- BigDecimal is immutable
- Example: BigDecimal instead of float
- Rounding and Scaling BigDecimal
- Rounding BigDecimal Numbers

SQLJ
- Assumes knowledge of basic SQL
- Static SQL
- Contrast with JDBC
- Creating and executing SQL with SQLJ

Java for COBOL Programmers part 3 (5 days)

Logging
- log4j logging
- log formats
- logging best practices
- Reading logs
- Logging levels
- Best practices

XML and Java
- Data is often enclosed in XML
- XML Syntax
- Elements
- Attributes
- Comments
- Unicode and Character Sets
- Character References
- Entity References
- Character Data Sections (CDATA)
- Processing Instructions
- Parsing XML – DOM and SAX
- JAXP - A Plugability Layer
- Trees and Nodes
- Processing Child Nodes
- Error Handling
- Building the Node Tree

Dates in Java
- Java Dates
- Java has two important Date classes
- java.util.Date
- java.util.Date constructors
- java.util.Date non-deprecated methods
- What happened to the Date class?
- Get Today's Date
- Java Classes Typically Used for Dates
- Setting Dates Using Calendar Factory
- Java Date Validation
- Parsing Dates from Strings - Example
- Date Setting and Comparison
- Time Zones — java.util.TimeZone
- java.text.SimpleDateFormat
- SimpleDateFormat Patterns
- Some Pattern Examples
- SimpleDateFormat is Not Thread-safe
- Calendar Instance - first, get instance
- Calendar Instance - next, set time zone
- Calendar Instance - lots of information
- Calendar Instance - output
- Locales
- SimpleDateFormat and locales
- SimpleDateFormat is locale-sensitive
- DateFormat Example
- DateFormat to Set a date
- GregorianCalendar class
- GregorianCalendar Example
- Date Arithmetic
- java.sql.Date constructor
- Java util vs sql Dates
- Convert java.util.Date to java.sql.Date

New Features in Java 7
- Coin Project - six new features
- Strings in switch
- Binary Literals
- Underscores in Numbers
- Multicatch in Exception
- Final rethrow in Exception
- final rethrow
- Try-with-resources (TWR)
- TWR can give enhanced stack traces
- Try with Resources Example
- Diamond Syntax
- Simplified varargs method invocation
- New File and Path features

Reading/Writing Flat Files & Spreadsheets in Java 6
- File I/O Concepts
- NIO classes
- CRLF on Different OS's
- Filtered Streams
- Buffer Concept
- "while" looping
- CDMI classes for Line I/O
- MyFileNotOpenException.java
- MyEofException.java
- MyIn.java / MyOut.java

Java for COBOL Programmers part 3 continued

File I/O Features in Java 7

- Java 7's New I/O
- File System Concepts
- Path vs File
- Important I/O Classes
- Improved classes in NIO.2
- Create a New Path
- Retrieve Information from a Path
- Resolving a Path
- New directory and file support
- Walk a directory
- Basic Creation and Deletion of Files
- Delete a File
- Copy a File - Files.copy(…)
- Move a File - Files.move (…)
- Open input File - Files.newBufferedReader(…)
- Open output File - Files.newBufferedWrite(…)
- Open and Read a File - Files.readAllLines(…)

Generics and More Collections

- Type Safety Features
- Generics and Raw-Types
- Example of Vector with generics
- Raw Type Vector
- Hashtable Example
- Iterate through a Hashtable
- Other Implementations of Map
- SortedMap
- Choosing an appropriate collection

Miscellaneous Topics

- Method signatures and Overloading
- Factories vs Inheritance vs Constructors, etc.
- MVC Tips
- StringTokenizer
- Tool or Utility class Tips
- Ternary if – Some Typical Uses
- Avoid Memory Leaks

Java 8 New Features

- Java 8 – java.time package
- Interface default / extension methods
- Actual code in Interfaces
- Invoking methods in Interface
- Functional programming

NOTE – Java file I/O changed significantly from Java 5 to java 6 to Java 7 to Java 8. Because there are many legacy Java programs exist, all the different I/O techniques are taught, from oldest to newest.

Java for COBOL Programmers Special Topics (1 to 10 days)

I18N - Internationalization
- Time Zones — java.util.TimeZone
- Locales
- Daylight Savings time
- Formatting dates for different countries
- Formatting Numbers for different countries
- Formatting currency for different countries
- Sending messages in different languages

Lambdas, Stream, Functional Programming
- Module Topics
- Concurrency in Java
- Streams
- Streams API (New with Java 8)
- Streams Concepts
- Pipeline Concepts
- Stream Highlights
- Passing Methods and Constructors
- Lambda Example
- Lambda Expressions
- Lambda Expression Syntax
- Methods as Arguments
- Passing methods – Syntax
- Passing methods – Example (Old)
- Passing methods – Example (New)

Writing customized annotations

Reflection
- Introduction to Reflection
- Uses for Meta-Data
- The Reflection API
- The Class<T> Class
- java.lang.reflect package
- Constructors
- Fields
- Methods
- Exception Handling and Reflection
- Reading Type Information
- Navigating Inheritance Trees
- Dynamic Instantiation
- Dynamic Invocation
- Reflecting on Generics

Customized Generics
- Generics concepts and usage
- Type Erasure
- Type Boundaries
- Wildcards
- Generic Methods
- Strengths and Weaknesses of Generics
- Legacy Code and Generics
- java.lang.Comparable
- Extending generic classes
- for/in loop

Inner Classes
- Syntax for Inner Classes
- Inner Class Categories
- Inner Classes: Pro's and Con's
- Run an inner method in static inner class
- Local Inner Classes
- Anonymous local inner Classes
- Static Nested classes

Regex in-depth
- Classes in java.util.regex package
- Pattern class
- Matcher class
- Advanced Patterns
- Matching Methods (all return boolean)
- find(...)
- matches()
- lookingAt()

Threads
- Java Thread Model
- Creating and Running Threads
- Manipulating Thread State
- Thread Synchronization
- Volatile Fields vs. Synchronized Methods
- wait and notify
- join and sleep
- The Concurrency API – newer approach
- Atomic Operations
- Thread class and Runnable Interface
- Creating / starting threads
- Timers
- Deadlock
- Performance Issues

Java for COBOL Programmers Special Topics continued

Writing your own Threads
- Safety versus Efficiency
- Thread Safety Choices
  - SingleThreadModel
  - Synchronization
  - Synchronize for Thread Safety
  - Synchronize Summary
  - Deadly Embrace
  - Concurrent Interfaces and Classes
  - Old technique
    - extends Thread or implements Runnable
  - New techniques for multi-threading

JSON vs XML
- JSON Syntax
- Processing JSON data

Socket Programming

More Java Performance Tuning

Oracle vs IBM Java releases

JDK 9 Change Highlights
- Removed some methods and classes

JDK 10 Change Highlights
- Removed some more methods and classes
- Added local variable type inference typing

JDK 11 Change Highlights
- Improvements for how to manage Garbage Collection

JDK 12 Change Highlights
- Added a "preview" feature for cleaner code

JDK 13 Change Highlights
- Added multi-line stripping of white space for better formatting

JDK 14 Change Highlights
- Project Valhalla
- Project Loom

New JDK releases every 6 months

Advanced IDz

JDBC SQL Programming
(knowledge of SQL is assumed)
- RDBMS
- JDBC Connectivity Model
- Database Programming
- Connecting to the Database
- Creating a SQL Query
- Getting the Results
- Updating Database Data
- Error Checking
- SQLException and SQLWarning classes
- Statement
- ResultSet
- Updatable Result Sets
- JDBC Types
- Executing SQL Queries
- ResultSetMetaData
- Executing SQL Updates
- PreparedStatement
- Parameterized Statements
- Stored Procedures
- Batch Updates
- RowSets
- Large Objects
- Savepoints
- CachedRowSets
- DataSources

Scanner
- User interactive input

Command line input (into main)

Case Study
- ***Team or individual project to bring together all the topics taught.***

Other topics as desired

Running Batch Java in USS (5 days)

IDz for connecting to a mainframe
- IDz practice is integrated into every topic
- Connection steps
  - DB2 Tools in IDz
- Data Studio Plug-in
- E/R Diagrams
- Visual Explain optional - (only for those who know it)
- IDz data editing
  - QSAM
  - VSAM
  - DB2
  - Many others available
- z/OS Integration Menu (optional)
  - Smart Analytics and Remote Search
  - Hyperlinks to results of queries
  - Export results
  - Regex
  - Editing styles
    - Eclipse style editing
    - ISPF Style Editing
    - Edit Data files

Unix System Services (USS)
- Architecture & Integration into z/OS

OMVS
- Starting OMVS and exiting OMVS
  - Essential Unix commands
    - pwd, ls, mkdir, rmdir, rm,
    - grep, mv, cp, cat, echo,
    - pr, pg, more, nl, wc,
  - Editing in OVMS
  - Unix file structure
  - Unix file types
    - sort, cut, tr, etc.
    - ordinary, special, directory, links
  - Relative and absolute pathnames
  - HOME directory
  - Important directories
  - Important environment variables
    - PATH, TERM, HOME,
    - PS1, PS2, etc.
    - LOGNAME, USER,
  - Shell concepts
  - set, env
  - Getting out of an OMVS loop
  - Executing Java from the command line

Simple Shell Programming
- Coding and running simple unix shell programs
- Positional parameters
- User-defined variables
- if, test, while, shift, exit, export
- Command substitution
- Shell quoting mechanisms

JCL for USS
- Different ways to run batch Java in z/OS
- DD statement parameters for unix files
- EXEC statements for Java, shell programs
- BPXBATCH
- Standard output (std)
- Error output (err)

COMP-3 (packed decimal)
- Handling this data in Java

ISHELL (optional)
Other ISPF editing/browsing/viewing options (optional)

Deployment (length depends on client requirements)

Because every Endevor platform is customized to client requirements, this course requires access to client's mainframe from IDz, as well as client's standards for deploying java into CA-Endevor.

Archives
- **JARs**
- **EARs**
- **WARs**

Creating JARs in appropriate USS directory/folder

Use shell to deflate JAR file and INCLUDE into CA-Endevor

BUILD PATHs

SCM Support
- **CA-Endevor Support**
- **CARMA Repositories**
  - Packages and Field reports