

## Java Programming with the OSGi Framework - 2 days

The OSGi framework is a specification for an execution environment supporting dynamic, loosely coupled, component-based architectures. The framework normally runs in an instance of a Java Virtual Machine and the services provided by each component are represented as Java interfaces. A component implementing a given service can be registered with a central service registry; in turn a component may query the registry to determine what other services are available and can listen for changes in the registry in order to adapt to updates of other components that it depends upon. The framework is designed to support continuous deployment of networked devices; in principle the life cycle of a component can be managed from anywhere in the network. The Gravity Service Binder is itself a component that simplifies the configuration of an OSGi-based system; using the Service Binder, component properties and dependencies on other services are described using a simple, human-readable XML file.

This course provides a detailed introduction to developing software using OSGi and the Service Binder. Topics covered include the service registry and component life cycle; creating and deploying components as OSGi bundles; accessing and using component properties; managing dependencies between components using the registry listener interface; managing dependencies between components using the Service Binder; synchronization issues; resolving package dependencies in dynamic updates; and service factories.

## Software Architectures to support integration of Ada, C, C++, and Java components - 3 days

Modernization of large, complex software systems often requires peaceful coexistence of legacy software components with more recently developed functional enhancements. Often, the legacy software is written in more traditional languages like C, C++, and Ada. To maximize architectural flexibility and reduce ongoing development and software maintenance costs, the trend is to use higher level languages like Java for development of new capabilities.

This course describes techniques to allow efficient and reliable integration of components written in different languages. Key areas of emphasis include containment and separation of concerns, efficiency, and reliability. Upon completion of this course, students will be able to oversee the architecting and integration of large software systems that are comprised of the combination of Ada, C, C++, and Java components.

## Safety Critical Development for Ada and Java - 2 days

Many industries have set, specific standards for the development, testing, and certification of safety critical software. These standards require or recommend the use of industry-best practices in all aspects of systems development. In some areas, standards mandate specific techniques for the development of safety critical systems. In all cases, a reasoned justification for the techniques actually used is required, together with evidence to show that the life cycle development processes are being followed. One area of key importance is the programming language used as the basis of the final installed system. The standards specify the use of a language which is well defined, has validated tools, enables modular programming, has strong checking properties, and is clearly readable.

This course describes the use of standards based versions of Ada and Java that are well suited to development of safety critical software. Both Ada and Java affords many software architecture characteristics, making them an ideal choice for such systems. Topics covered will include the benefits of using Ada and Java in safety critical systems, subset of Ada and Java suitable for use in the development of safety critical software, use of programming paradigms such as Ravenscar profile instead of cyclic executives. After this course, students will be able to develop safety critical software using Ada or Java.



### **NORTH AMERICA**

Phone: 800.97-AONIX  
Fax: 858.824.0212  
Email: info@aonix.com

### **UNITED KINGDOM**

Phone: +44 (0) 1491 415000  
Fax: +44 (0) 1491 571866  
Email: info@aonix.co.uk

### **FRANCE**

Phone: +33 (0) 1 4148-1000  
Fax: +33 (0) 1 4148-1020  
Email: info@aonix.fr

### **GERMANY**

Phone: +49 (0) 7243 5318-0  
Fax: +49 (0) 7243 5318-78  
Email: info@aonix.de

# Software Modernization Courseware Catalog



Software Technology Accelerated Modernization Program

## Software Modernization Courses

Concepts of object-oriented programming for C and C++ programmers

Java for C++ programmers

Java for Ada programmers

Concurrent Programming with Java

Soft Real-Time Programming with Java

Hard Real-Time Programming with Java

Embedded Development with the PERC Virtual Machine

Java Programming with the OSGi Framework

Software Architectures to support integration of Ada, C, C++, and Java components

Safety Critical Development for Ada and Java

## Concepts of object-oriented programming for C and C++ programmers - 2 days

For developers comfortable with the structured, procedural style of programming embodied by languages such as C, Pascal, or Ada-83, thinking in terms of *objects* and *classes* rather than *data* and *functions* presents a significant paradigm shift. This course provides an introduction to the terminology and key concepts of object-oriented programming: encapsulation, classes, object identity and state, inheritance, composition, polymorphism, and dynamic binding. Examples and exercises are given in the Java programming language; however, no prior knowledge of Java is assumed. This course prepares students for a more intensive course in Java such as "Java for C++ programmers," below.

## Java for C++ programmers - 5 days

In many respects, the Java programming language derives from the successes of C and C++ while learning from and their mistakes improving upon their shortcomings. This course provides a solid overview of the Java programming language for developers already familiar with object-oriented programming in C++. (See "*Concepts of object-oriented programming for C and C++ programmers*", above.) Topics of study include Java language syntax and semantics; use of the open-source Eclipse development environment; overview of the standard edition libraries for I/O, networking, graphics, data structures, and algorithms; and introduction to important open-source third-party libraries and frameworks such as OSGi and Eclipse. Comparisons are made between the C, C++, and Java programming platforms highlighting the significance of the virtual machine, class files, garbage collection, and thread support. This course prepares students to begin developing with the Java programming language.

## Java for Ada programmers - 5 days

Though the syntax of Java more closely resembles C++ than Ada, the discipline and structure encouraged by the Java language more closely resembles Ada. This course provides a solid overview of programming with the Java programming language. Topics of study include Java language syntax and semantics; use of the open-source Eclipse development environment; overview of the standard edition libraries for I/O, networking, graphics, data structures, and algorithms; and introduction to important open-source third-party libraries and frameworks such as OSGi and Eclipse. To facilitate the learning process, comparisons are made between the Ada and Java programming platforms. This course prepares students to begin developing with the Java programming language.

## Concurrent Programming with Java - 5 days

One of the powerful strengths of the Java programming language is its built-in support for concurrency. Java programmers use special syntax to implement synchronized access to shared variables and the Java compiler, library implementations, and operating system interface cooperate to present consistent and portable behavior of correctly written concurrent Java programs across all compliant Java virtual machine implementations.

This course details the practices that must be followed in order to write portable and reliable concurrent Java programs. Programmers who misunderstand or oversimplify these rules are likely to write concurrent Java software that will misbehave under rare race conditions and/or when deployed on a different virtual machine than where the code was originally developed and tested.

Course topics include concurrency concepts; thread models and implementations; Java primitives for mutual exclusion; memory consistency issues and the Java memory model; Java primitives for waiting and signaling; introduction to the `util.concurrent` package of JDK 1.5; structuring classes for safety and liveness in concurrent systems; effective use of immutability, synchronization, and confinement techniques; the role of real-time garbage collection; and use of Real-Time Specification for Java services.

## Soft Real-Time Programming with Java - 2 days

Soft real-time systems generally have timing constraints ranging from a single ms to hundreds of ms. Compliance with soft real-time timing constraints is generally achieved through the use of empirical (statistical) analysis of resource requirements and heuristic enforcement of resource budgets. The use of real-time garbage collection techniques within the implementation of a Java virtual machine makes it possible to deploy Java components in soft real-time systems.

This course describes key considerations in selection and use of a soft real-time Java virtual machine. Topics include configuration of real-time garbage collection, avoidance of priority inversion, containment and separation of concerns through resource needs analysis and resource budgeting.

Students who complete this course will understand how to architect, design, and implement soft real-time systems using the Java programming language.

## Hard Real-Time Programming with Java - 3 days

Hard real-time components occupy the lowest layers of typical complex software hierarchies. Compliance with hard real-time constraints must be demonstrated through static analysis and theoretical proofs. To support this, hard real-time software components are much simpler and less dynamic than higher level software. The technical approaches that enable deployment of hard real-time Java software also make it possible to deploy Java software components that run at roughly the same speed, memory footprint, and worst-case real-time latencies as comparable C or C++ code.

This course describes the use of a standards-based constrained version of Java that is well suited to the lowest layers of a software system. Topics of discussion include use of annotations to describe static resource requirements and safe stack allocation of temporary objects, libraries to support the implementation of device drivers and interrupt handlers, development tools to enforce consistency between method invocations and implementations, concepts of high precision time, and efficient and reliable integration of hard real-time Java components with soft real-time components. Students who complete this course will be able to begin development using the hard real-time Java technologies.

## Embedded Development with the PERC Virtual Machine - 2 days

By using the Java programming language for development of embedded real-time systems, programmers are able leverage large-market economies of scale. Development tools are more mature and less expensive than typical embedded development environments. And there exists a huge variety of off-the-shelf portable software components, many of which are free and open-source, that easily integrate within their embedded device software systems.

This course covers the special topics that must be considered when deploying Java software in embedded and mission-critical systems. Among the topics covered, we include tools to pre-link and pre-compile Java components, cross compilation, remote debugging and profiling, ROM-targeting tools, configuration options for dynamic class loading and JIT compilation, configuration of real-time garbage collection, and use of the VM management and PERC shell tools. Students who complete this course will be able to begin targeting Java software components for execution in embedded mission-critical and real-time systems using the PERC virtual machine platform.

"One of the best courses I have taken in a long time. I recommend it to all of our employees" - Gregory K.