hiJaC

Safety Critical Java

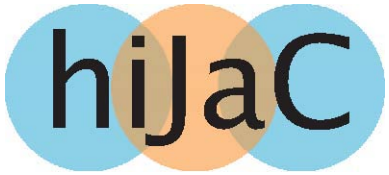
University of York

15 November, 2011

Safety Critical Java

- Background
- Introduction to SCJ
- The Expert Group
- Key components of the Specification
- Current status

Background



- Success of Java led to a wish to use it for real-time systems
- In 1999, NIST identified requirements for real-time extensions to Java
- The **Java Community Process** or **JCP**, established in 1998, is a formalized process that allows interested parties to get involved in the definition of future versions and features of the Java Platform
- The JCP involves the use of **Java Specification Requests** (JSRs) that proposed specifications and technologies for adding to the Java platform.
- JSR 1: Create a Real-Time Specification for Java

Background: The RTSJ (Version 1.02)

Enhances Java in the following areas:

- memory management
- time values and clocks
- schedulable objects and scheduling
- real-time threads
- asynchronous event handling and timers
- asynchronous transfer of control
- synchronization and resource sharing
- physical and raw memory access

Background: JSR 282

- Affinity
- User-defined Clock
- Happenings
- Rich AsyncEvents
- Scope Pinning
- Periodic Phasing
- Raw Memory
- Physical Memory
- Timed restart
- Timer info
- SCJ Constructors
- waitForNextRelease
- Blocking time
- CPU time reporting

Introduction

- SCJ Goal – a specification for Safety Critical Java capable of being certified under DO-178B Level A and other safety critical certification standards
  - Certification implies a small, reduced complexity infrastructure (i.e., JVM)
  - Emphasis is on defining a minimal set of capabilities required of safety critical applications using Java implementations
- Based on the Real-Time Specification for Java (JSR-1 and JSR-282)

Introduction

- This effort being done as a Java Specification Request (JSR) under the Java Community Process
  - JSR-302
  - As a JSR, this means that three things are being created:
    - A Safety Critical Java Specification
    - A Reference Implementation
    - A Technology Compatibility Kit

Expert Group

Under the JCP, specifications are created by a Specification Lead (in this case, The Open Group) and an Expert Group.

- The currently active EG members are:
  - James Hunt (aicas)
  - Doug Locke (TOG & LC Systems Services)
  - Johan Nielsen (DDC-I)
  - Kelvin Nilsen (Atego)
  - Martin Schoeberl (T.U. Denmark)
  - Jan Vitek (Purdue U.)
  - Andy Wellings (U. of York)
- Previously active authors of the Specification are:
  - Scott Anderson (Verocel)
  - Ben Brosgol (AdaCore)
  - Mike Fulton (IBM)
  - Thomas Henties (Siemens)

SCJ Specification

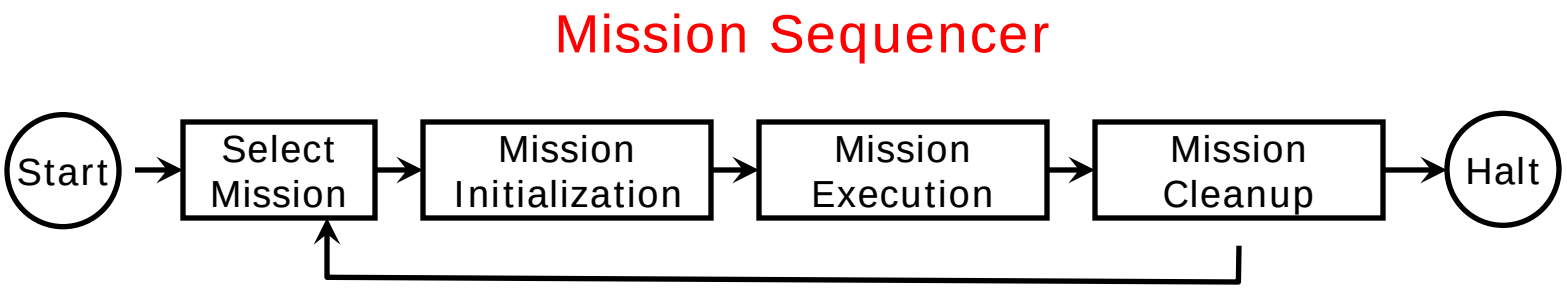
- Essentially a subset of Java and the RTSJ but
  - augmented with some SCJ-specific classes that impose a programming paradigm
  - with annotations to facilitate static analysis tools
- Problem: wide range of definitions of safety critical systems
  - Single thread, single function, simple timing constraint
  - Highly complex application, multiple modes of operations, complex timing constraints

Specification Content Compliance Levels

- Three Compliance Points (Levels 0, 1, 2)
  - Level 0 essentially supports a timeline/frame-based/cyclic executive model
  - Level 1 essentially supports a highly constrained multithread application under the control of a fixed-priority preemptive scheduler: equivalent to Ravenscar-Ada
  - Level 2 essentially supports much more complex multithread application under the control of a fixed-priority preemptive scheduler
- All three Levels targeted for DO-178B Level A certifiability but with increasing complexity & cost

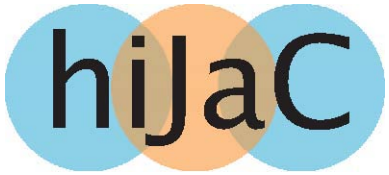
Specification Content
Missions and Mission Sequencers

- Application Structure



- Data structures created at initialization in a shared scoped memory area called mission memory
- Application consists of schedulable objects (SOs)
- Each SO has a private scoped memory area used for dynamic allocation
- Application will not require heap memory, even at startup
- A mission is managed by the infrastructure

Specification Content Compliance Levels Revisited



- Level 0 provides a cyclic executive (single thread shared among SOs), no wait/notify
  - Synchronization ignored
  - No application threads – only Periodic Event Handlers
- Level 1 provides a single mission sequence with multiple concurrent SOs, no wait/notify
  - Multiple concurrent SOs use synchronized methods
  - Periodic and Aperiodic Event Handlers
- Level 2 provides nested missions with additional capability
  - May have NoHeapRealtimeThreads, wait/notify on (this),

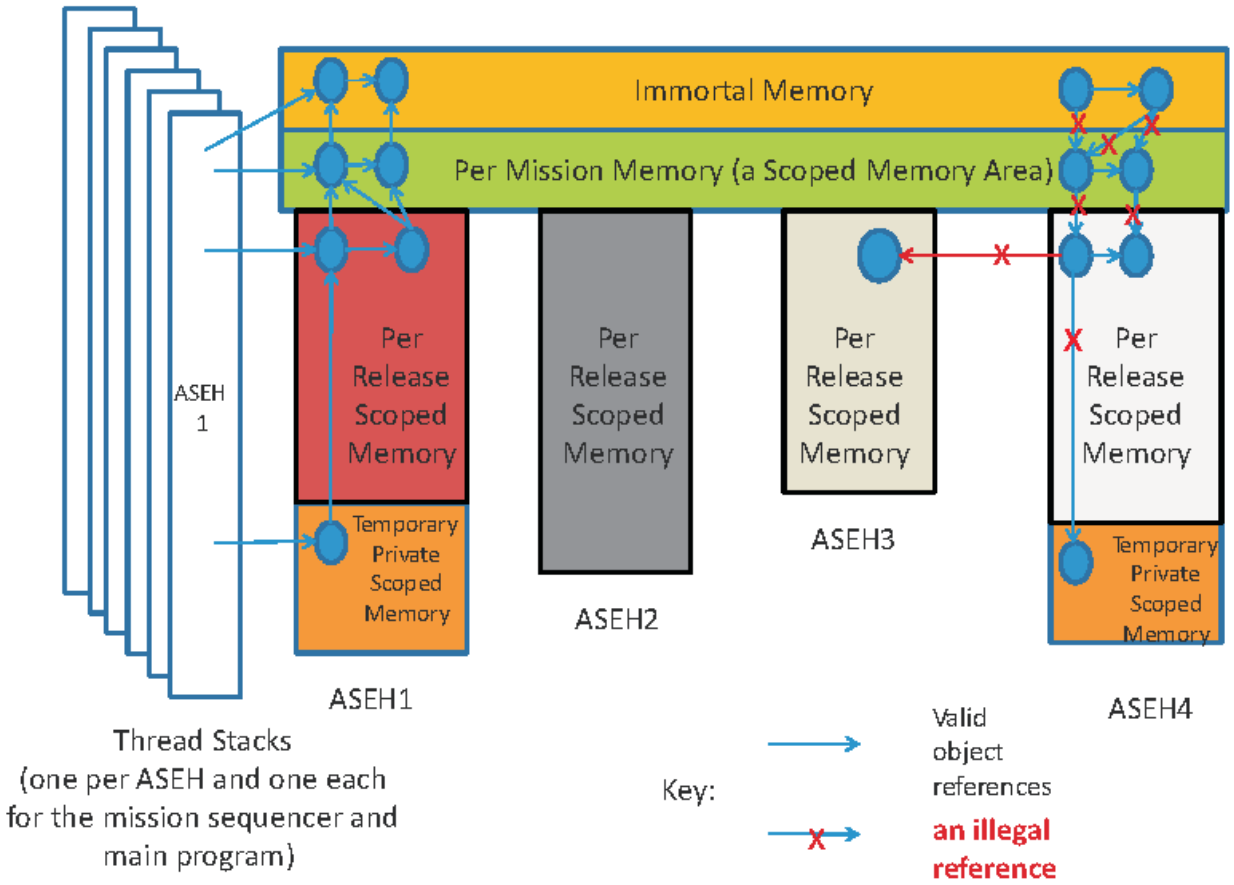
Specification Content Concurrency & Synchronization

- Synchronization
  - Priority Ceiling Emulation – required
  - Synchronized methods provided
    - Synchronized block not permitted
    - Not allowed to self-suspend in Level 1
- Schedulable Objects (SOs) – all non-heap
  - PeriodicEventHander (PEH)
    - Required for concurrency in Level 0, provided for Level 1
    - Bound
  - PeriodicEventHandler or AperiodicEventHandler (APEH) with appropriate parameters
    - Provided for Level 1 and Level 2
    - Bound
  - NoHeapRealtimeThread
    - Permitted only in Level 2

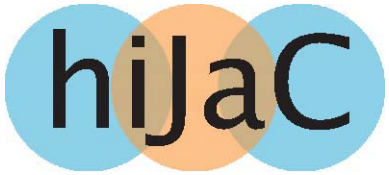
Specification Content – Memory

- Every SO has private scoped memory
  - For PEH, APEH, this memory is entered & exited (cleaned up) each time the SO is released
  - Level 2 NoHeapRealtimeThread has private scoped memory entered & exited by run() method
- SOs can also create other private scoped memory objects
  - May not share references to it or its objects it with other SOs

SCJ Memory Structure

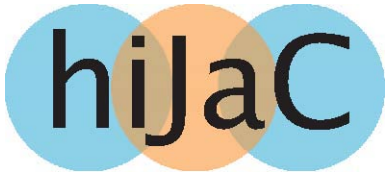


Specification Content
Key Annotation – @SCJAllowed



- @SCJAllowed(n)
  - Attached to classes, methods
  - Associated class/method may be referenced by applications at Level n or higher
  - Classes/methods without @SCJAllowed not callable by SCJ applications
    - Includes Java library objects
  - Checked as part of bytecode verification

Specification Content Exceptions



- Exception objects allocated
  - In current memory area unless newInstance used
- Every SO has thread local space for saving
  - Back trace information
  - ThrowBoundaryError exception
- When ThrowBoundaryError is thrown, (implementation-defined) back trace information is available to application

Specification Content JNI

- No reflection
- No memory allocation in JNI (unchecked)
- Primitive types, objects and arrays can be passed by SCJ program
- SCJ defines specific interfaces permitted and not permitted for each Level in JNI code

Specification Content Input / Output

- Subset of Java Micro Edition functions provided
  - Much simpler than file and socket I/O
- Interfaces and Classes
    - Open and Close methods
    - References URL named devices
  - Connector
  - Connection
- Also two simple text I/O classes
    - Uses UTF-8
    - May write only to a memory buffer if device not present
  - ConsoleConnection
  - SimplePrintStream

Specification Content Interaction With Devices

- Raw memory access
  - Permits use of, e.g., memory-mapped I/O
- Interrupt handling
  - Provides direct interrupt handling by Java event handlers
- Signals and other asynchronous external events
  - Provides interfaces for signals and happenings

Specification Content Class Libraries

- SCJ Specification lists all supported (@SCJAllowed) interfaces and classes in
  - java.io
  - java.lang
  - javax.microedition.io
  - javax.realtime
  - javax.safetycritical
  - javax.safetycritical.annotate
  - javax.safetycritical.io

Summary

- The Expert Group completed their Early Draft Review (EDR) under the Java Community Process
  - Review comments are now being considered
- The specification is being edited into a final form
- Reference Implementation now being tested as open source RTSJ-compliant Java executable on an RTSJ-compliant JVM
- Technology Compatibility Kit has been created and is being tested
- Submission to the JCP Executive Committee hoped to be ready in mid 2012