Advanced Java

Anil Joseph

Created By Anil Joseph(anil.jos@gmail.com)

Agenda

Concurrency	Threading	Concurrency Utilities	Generics
Annotations	Java 6 & 7 New Features	Java 8 New Features	I/O
	Reflection	JVM & GC	

Created By Anil Joseph(anil.jos@gmail.com)

Software

JDK 1.8

Eclipse IDE

Java Documentation

Created By Anil Joseph(anil.jos@gmail.com)

Concurrency

- Concurrency is the ability of a program to execute several computations simultaneously(in parallel).
- A modern computer has several CPU's or several cores within one CPU.
  - The ability to leverage these multi-cores can be the key for a successful high-volume application.
- Process
  - Processes are an execution environment provided by the operating system that has its own set of private resources
  - In Java, processes correspond to a running Java Virtual Machine (JVM).
- Thread
  - Threads are lightweight processes that live within a process and share their resources with the other threads of the process.
  - Threads live within the same JVM and can be created and stopped by the Java application dynamically during runtime

Created By Anil Joseph(anil.jos@gmail.com)

Thread

- Threads are light-weight processes within a process.
- They share the memory(Heap) with other threads.
- Each thread has its own call stack
- In Java the java.lang.Thread class is used to create a thread.
- Every thread has an
  - Identifier
  - Name
  - Priority
  - State

Created By Anil Joseph(anil.jos@gmail.com)

Creating Threads

Extend from the Thread class and override the run method

Implement the interface Runnable and override the run method

Created By Anil Joseph(anil.jos@gmail.com)

Thread States

New

- A thread that has not yet started is in this state.

Runnable

- A thread executing in the Java virtual machine is in this state.

Blocked

- A thread that is blocked waiting for a monitor lock is in this state.

Waiting

- A thread that is waiting indefinitely for another thread to perform a particular action is in this state.

Timed Waiting

- A thread that is waiting for another thread to perform an action for up to a specified waiting time is in this state.

Terminated

- A thread that has exited is in this state.

Created By Anil Joseph(anil.jos@gmail.com)

Sleep, Join and Interrupt

sleep() puts the current Thread to sleep without consuming any processing time.

The current thread removes itself from the list of active threads and the scheduler doesn't schedule it for the next execution

Interrupts are a mechanism to send a message to a thread from another

join() lets the thread to wait for the termination of another thread.

Created By Anil Joseph(anil.jos@gmail.com)

Thread Local

- The *TheadLocal* construct allows us to store data that will be **accessible only** by **a specific thread**.
- This used to store context information like session or transaction identifies with the Thread.

Created By Anil Joseph(anil.jos@gmail.com)

Concurrency Issues

- Threads have their own call stack but can also access shared data.
- Therefore you have two basic problems
  - Visibility
  - Access problems.
- Visibility: Occurs if thread A reads shared data which is later changed by thread B and thread A is unaware of this change.
- Access: Occur if several thread access and change the same shared data at the same time.

Created By Anil Joseph(anil.jos@gmail.com)

Locks and Synchronization

- Java provides *locks* to protect certain parts of the code to be executed by several threads at the same time.
- To create a lock on an object use the ***synchronized*** keyword.
- Synchronization is necessary for mutually exclusive access to blocks of and for reliable communication between threads.

Created By Anil Joseph(anil.jos@gmail.com)

Object monitors: wait & notify

The wait() method pauses the execution of the current thread.

The notify() methods wakes up a thread in the paused/wait mode.

The thread that calls the wait() method must hold a lock to a resource

Calling wait() the lock is released and the threads waits until notify is called.

Another thread that now owns the lock calls notify() on the same object instance

Created By Anil Joseph(anil.jos@gmail.com)

Java.util.concurrent package

- The java.util.concurrent packages provide a set of ready to use classes for concurrency.
- Thread Pool
- Executor
- ExecutorService
- ExecutorCompletionService
- Callable
- Future
- CompleteableFuture
- Semaphore
- CountDownLatch
- CyclicBarrier

Created By Anil Joseph(anil.jos@gmail.com)

Fork And Join

Fork (split) a large task into smaller subtasks which can be executed concurrently.

Each subtasks can be executed in parallel by different CPUs, or different threads on the same CPU.

Once the subtasks are completed the task may *join* (merge) all the results into one result.

The ForkJoinPool is a special thread pool which is designed to work well with fork-and-join task splitting

Created By Anil Joseph(anil.jos@gmail.com)

Generics

- Generics enable *types* (classes and interfaces) to be parameters when defining classes, interfaces and methods.
- Type parameters provide a way for you to re-use the same code with different inputs.
- Advantages
  - Stronger type checks at compile time.
  - Elimination of casts.
  - Enabling programmers to implement generic algorithms.

Created By Anil Joseph(anil.jos@gmail.com)

Autoboxing and Unboxing

- *Autoboxing* is the automatic conversion that the Java compiler makes between the primitive types and their corresponding object wrapper classes.
- Converting wrapper classes to primitive types is called unboxing.
- Introduced in Java 5.

Created By Anil Joseph(anil.jos@gmail.com)

Annotations

- *Annotations* provide data about a program that is not part of the program itself.
- A form of metadata
- Annotations have no direct effect on the operation of the code they annotate.
- Advantages
  - Annotations can be used by the compiler to detect errors or suppress warnings.
  - Software tools can process annotation information to generate code, XML files, and so forth.
  - Some annotations are available to be examined at runtime.
- Annotations can be applied to declarations
  - classes, fields, methods, and other program elements.

Created By Anil Joseph(anil.jos@gmail.com)

Predefined Annotation Types

- **@Deprecated**
  - annotation indicates that the marked element is *deprecated* and should no longer be used.
- **@Override**
  - annotation informs the compiler that the element is meant to override an element declared in a superclass.
- **@SuppressWarnings**
  - annotation tells the compiler to suppress specific warnings that it would otherwise generate.
- **@FunctionalInterface**
  - annotation, introduced in Java SE 8, indicates that the type declaration is intended to be a functional interface,

Created By Anil Joseph(anil.jos@gmail.com)

Annotations That Apply to Other Annotations

- @Retention
  - annotation specifies how the marked annotation is stored.
  - RetentionPolicy.SOURCE – The marked annotation is retained only in the source level and is ignored by the compiler.
  - RetentionPolicy.CLASS – The marked annotation is retained by the compiler at compile time, but is ignored by the Java Virtual Machine (JVM).
  - RetentionPolicy.RUNTIME – The marked annotation is retained by the JVM so it can be used by the runtime environment.
- @Documented
  - annotation indicates that whenever the specified annotation is used those elements should be documented using the Javadoc tool.

Created By Anil Joseph(anil.jos@gmail.com)

Annotations That Apply to Other Annotations

- @Target
  - annotation marks another annotation to restrict what kind of Java elements the annotation can be applied to.
- @Inherited
  - annotation indicates that the annotation type can be inherited from the super class.
- @Repeatable
  - annotation, introduced in Java SE 8, indicates that the marked annotation can be applied more than once to the same declaration or type use.

Created By Anil Joseph(anil.jos@gmail.com)

Varargs

- Variable argument list, introduced in Java 5
- Substitute to sending an array as a method argument.
- Example
  - public String format(String pattern, Object... arguments);

Created By Anil Joseph(anil.jos@gmail.com)

Java 7 enhancements

- Try-with-resources statement
- Underscores in numeric literals
- Strings in switch
- Improved Type Inference for Generic Instance Creation
- Multiple exception catching

Created By Anil Joseph(anil.jos@gmail.com)

try-with-resources Statement

- The try-with-resources statement is a try statement that declares one or more resources.
- A resource is an object that must be closed after the program is finished with it.
- The try-with-resources statement ensures that each resource is closed at the end of the statement.
- Any object that implements java.lang.AutoCloseable, which includes all objects which implement java.io.Closeable, can be used as a resource.

Created By Anil Joseph(anil.jos@gmail.com)

New IO API

- The NIO API introduced in JDK 1.4 provides a completely new model of low-level IO.
- New IO is block oriented unlike earlier(stream oriented).
- Two Features in NIO for better performance
  - Direct Buffers
  - Memory Mapped Byte Buffers

Created By Anil Joseph(anil.jos@gmail.com)

MappedByteBuffer

- A MappedByteBuffer allows you to map a portion of a file into a memory buffer.
- Changes made to the buffer are automatically propagated to the file by the underlying implementation of MappedByteBuffer.
- Helps is working with very large files.

Created By Anil Joseph(anil.jos@gmail.com)