

JDK 7 Reference Card

NEWLY INCLUDED JSRs

JSR 203: More New I/O APIs for the Java Platform ("NIO.2")

File System Access

Starting with JDK 7, all files should be referenced through the new **java.nio.file** package, instead of the now deprecated `java.io.File`. JDK 7 will now use the **java.nio.file.Path** class to manipulate a file in any file system (`FileSystem`). This `FileSystem` can use any `FileStorage`.

Before Java 7, you did this:

```
File file = new File("some_file");
```

With Java 7, you can do this:

```
Path path = Paths.get("some_file");
```

Now, the `File` class has a new method `toPath()` that allows you to transform `File` to `Path`:

```
Path path = new File("some_file").toPath();
```

Socket-Channel Binding and Configuration

The network-oriented channel is updated! Now you can bind channel's socket and manipulate socket options directly. JDK 7 provides platform-/implementation-specific socket options beyond those required by the specification.

- JDK 7 includes a new `ByteChannel`, the **SeekableByteChannel**, which maintains a current position.
- **NetworkChannel** is the new super interface for the network-oriented channels. You easily bind the channel socket, and set and query for socket options.
- **The MulticastChannel** interface enables you to create Internet Protocol Multicasting. The multicast implementations are directly bound to the native multicast facility.

Scalable Asynchronous I/O

Run tens of thousands of streams in separate threads with true asynchronous I/O! JDK 7 provides asynchronous operations for both sockets and files. Some of the coolest JDK 7 channels are:

- **AsynchronousFileChannel**: An asynchronous channel for reading and writing from and to a file.
- **AsynchronousSocketChannel**: A simple asynchronous channel to a socket. The methods are asynchronous and support timeouts.
- **AsynchronousServerSocketChannel**: An asynchronous channel to a `ServerSocket`.
- **AsynchronousDatagramChannel**: A channel to datagram-oriented socket.

JSR 292: Dynamically Typed Languages on the Java Platform

Da Vinci Machine Project (JSR 292) is intended to extend the JVM to support languages other than Java – especially **dynamic languages**. The supported languages should co-exist with Java as if they were there from the beginning – no discriminations are permitted.

JSR 334: Small Enhancements to the Java Programming Language

OpenJDK's **Project Coin** (JSR 334) is intended to add a set of **minor language changes** for improving day-to-day Java development. The list of changes includes:

- Strings in switch statements
- Binary integral literals and underscores in numeric literals
- A single catch block to handle multiple exception types
- Improved type inference for generic instance creation
- **Try-with-resources** statement to automatically close "resources"

CHANGED JSRs (in Maintenance Reviews)

JSR 199: Java Compiler API

JSR 199 was added in JDK 6 and provides the ability to invoke a Java compiler via an API. Beyond the `javac` command-line tool, JSR 199 provides Java compiler-to-programmatic interaction as well. The Java compiler API serves three main goals:

- Invocation of a compiler or **other tools**
- Programmatic access to structured diagnostic messages
- Ability to customize file input and output

JSR 206: Java API for XML Processing (JAXP)

JSR 206 is known as **Java API for XML Processing** (JAXP), an implementation-independent portable API for **processing XML with Java**.

The main JAXP 1.3 features include:

- **DOM Level 3** core
- A built-in validation processor for **XML Schema**
- A thorough implementation of XML Schema data types in **javax.xml.datatype**
- XSLTC, the fast compiling transformer, which is now the default engine for XSLT processing
- An implementation of **XInclude**, which makes it possible to build a document using text or XML stored in reusable segments elsewhere

JDK 7 will contain JAXP 1.3, which is the latest release of JAXP.

JSR 222: Java Architecture for XML Binding (JAXB)

JSR 222 is known as **Java Architecture for XML Binding** (JAXB). The goal of JAXB is to allow Java developers to **map Java classes to XML representations**.

The main features of JAXB 2 include:

- Support for all W3C XML Schema features (JAXB 1.0 did not specify bindings for some of the W3C XML Schema features.)
- Support for **binding Java to XML**, with the addition of the `javax.xml.bind.annotation` package to control this binding.
- A significant reduction in the number of generated schema-derived classes
- Additional validation capabilities through the JAXP 1.3 validation APIs

JDK 7 will contain JAXB 2.2, which is the latest release of JAXB.

JSR 224: Java API for XML-Based Web Services (JAX-WS)

JSR 224 is known as the Java API for XML-based Web Services (JAX-WS), a programming model based on a standard annotation model for developing Web services applications and clients.

The main JAX-WS 2 features include:

- Support for JAXB 2.1 APIs (JSR 222)
- Support for Web Services Addressing 1.0
- APIs for EndpointReference (EPR): Creation (`BindingProvider.getEndpointReference()`, `Endpoint.getEndpointReference()`, `MessageContext.getEndpointReference()`), Propagation (Using JAXB 2.1 to bind W3C EPR to `W3CEndpointReference` class, Marshall/Unmarshall `W3CEndpointReference` class using JAXB)
- User friendly APIs to enable/disable features, such as MTOM and Addressing

JDK 7 will contain JAX-WS 2.2, which is the latest release of JAX-WS.

JSR 269: Pluggable Annotation-Processing API

JSR 269 is known as **Pluggable Annotation-Processing API**

- Starting with JDK 5, annotations became a powerful mechanism for annotating classes, fields and methods. Commonly, annotations are processed at build-time or run-time to get new semantic effects. This JSR defines APIs to allow annotation processors to be created using a standard pluggable API.
- The specification will include at least a section of the API modeling the Java programming language and a section for declaring annotation processors and controlling how they are run.
- Since annotations are placed on program elements, an annotation-processing framework needs to reflect program structure.
- Annotation processors will be able to specify what annotations they process and multiple processors will be able to run cooperatively.
- The processors and program structure API can be accessed at build-time.

SMALLER ENHANCEMENTS

java.util.Objects	A set of nine brand new static methods to deal with objects . Two methods that check if an object is null (nonNull), two methods to compute a toString() value for objects supporting null objects, two similar methods for hashing, two similar methods for equality checks, and finally, a method named compare that returns 0 if a == b or if both are null (Otherwise, it returns c.compare(a, b).
Swing JLayer Component	The JXLayer is a component decorator that provides a simple solution for decorating compound components and catching all mouse, keyboard and FocusEvent events for all its subcomponents. It employs only the public Swing API and it has no effect over global settings, such as Event-Queue or RepaintManager. (Speaking of Swing additions, JDK 7 will probably bring JXDatePicker and possibly CSS-based styling.)
Concurrency and Collections APIs	Under JSR 166 , concurrency and collections APIs have been enhanced with flexible and reusable synchronization barriers, a concurrent-reference HashMap, transfer queues, the lightweight fork/join framework and thread-local pseudo-random number generators.
Class Loader Architecture	The class loader architecture was upgraded to avoid deadlocks in non-hierarchical class loader topologies. JDK 7 contains an important enhancement for multithreaded custom class loaders: a concept called parallel-capable class loader. Loading a class by a parallel-capable class loader now synchronizes on the pair consisting of the class loader and the class name.
Locale Enhancement	Java Locale should avoid loss of data from small changes. In addition, Locale should support more features, such as IETF BCP 47 and UTR 35 (CLDR/LDML).
Separate User Locale and User-interface Locale	JDK 7 separates formatting locales from UI language locales, as is done in Vista and later versions of Windows.
Strict Class-file Checking	Per the Java SE 6 specification, class files of version 51 (SE 7) or later must be verified with the type-checking verifier introduced by JSR 202 in Java SE 6. The VM must not fail over to the old inferencing verifier.
Elliptic-Curve Cryptography (ECC)	Starting with JDK 7, Java provides a portable implementation of the standard ECC algorithms approach to public-key cryptography based on the algebraic structure of elliptic curves over finite fields.
Nimbus Look-and-feel for Swing	The Nimbus is the new look-and-feel for the JDS (Java Desktop System). This is available as a GTK theme in Solaris 11 pre-release.
XRender Pipeline for Java 2D	JDK 7 brings a new Java 2D graphics pipeline based upon the X11 XRender extension. This provides access to much of the functionality of modern GPUs.
TLS 1.2	TLS (Transport Layer Security) is a secure protocol for transferring data over the Internet by preventing eavesdropping, tampering, or message forgery. The main goal of TLS is to provide privacy and data integrity between two communicating applications. TLS was standardized as RFC 5246, and JDK 7 comes with support for version 1.2.
JDBC 4.0/4.1	JDBC 4.1 features are present only in a JDK 7 or higher environment. JDBC 4.1 includes minor modifications to JDBC 4.0. A few JDBC 4.0/4.1 features are worth noting: • DataSources – Derby introduces new implementations of javax.sql.DataSource. • Autoloading of JDBC drivers – Applications no longer need to resolve JDBC drivers through Class.forName() on the driver name. Instead, the DriverManager will find an appropriate JDBC driver when the application requests a connection. • Wrappers – This is a new concept in JDBC 4.0 and represents a mechanism used by application servers to find vendor-specific extensions inside standard JDBC objects like Connections, Statements, and ResultSets. • Statement events – Connection pools can listen for Statement closing and Statement error events. New methods were added to javax.sql.PooledConnection: addStatementEventListener and removeStatementEventListener. JDK 7 provides complete support for JDBC 4.1.
Translucent and Shaped Windows	Create new platform APIs for 6u10 graphics features with translucent (simple translucency and per-pixel translucency) and shaped windows (set any shape to an undecorated window), heavyweight/lightweight mixing, and the improved AWT security warning. The translucent and shaped windows feature is available through the new com.sun.awt.AWTUtilities class.
Unicode 6.0	Unicode 6.0 brings new features such as 2.088 characters, properties corrections for existing characters, format improvements, and new properties and data files. JDK 7 was upgraded to support Unicode 6.0.
Method to Close a URLClassLoader	With URLClassLoader, applications can load classes and resources from a search path of URLs. JDK 7 comes with a method (close()) that cleans up resources held by URLClassLoader (invalidates the loader).
Enhanced JMX Agent and MBeans	This enhancement is ported from JRockit, which makes it easier to connect to the platform MBean server through firewalls and provides a richer set of MBeans, which expose additional information about the internal operation of the VM.
New Garbage Collector	JDK 7 has a new garbage collector that achieves lower pause times and better predictability than the current CMS collector.

ENHANCED JSRs

JSR 901: Java Language Specification (JLS)

- JSR 901 includes all changes, clarifications and amendments made to the Java programming language since the first language spec. The language is specified by the JLS.
- Changes to the JLS are managed under JSR 901.
- JDK 7 will come with the latest JSR 901.

JSR 924: JVM SpecificationPlatform

- JSR 924's scope is to maintain review of changes to the Java Virtual Machine Specification, Second Edition for J2SE 1.5.

Java SE APIs

- Java SE APIs remain under Maintenance Review to incorporate changes made by routine maintenance and small-scale enhancement.

DEFERRED TO JDK 8 OR LATER

- **JSR 294: Language and VM support for modular programming** – The main goal of this JSR is to provide modular programming at compile time and run time.
- **JSR 308: Annotations on Java types** – This will be an extension to the current annotation syntax for allowing annotations on any occurrence of a type.
- **JSR 296: Swing application framework** – The main goal is to eliminate boilerplate code and provide a simpler structure for Swing applications.
- **Modularization** – An implementation-specific, simple, low-level module system focused upon the goal of modularizing the JDK, and the application of that system to the JDK itself
- **JSR TBD: Project Lambda** – Lambda expressions (informally, “closures”) and defender methods for the Java programming language
- **JSR TBD: Language support for collections** – Literal expressions for immutable lists, sets, and maps, and indexing-access syntax for lists and maps
- **Swing JDatePicker component** – Adds the SwingLabs JXDatePicker component to the platform

