Java Dependency Analysis Tool

jdeps is a new command-line tool added since JDK 8 for developers to use to understand the static dependencies of their applications and libraries. jdeps is a static analysis tool on the given class files and dynamic class dependencies (Class.forName or loading of service providers etc) are not reported.

It also provides an ***-jdkinternals*** option to find dependencies to any JDK internal APIs that are unsupported and private to JDK implementation (see Why Developers Should Not Write Programs That Call 'sun' Packages).

Prepare for JDK 9

Most of the JDK's internal APIs are encapsulated in JDK 9. JEP 261 specifies the critical internal APIs that remain accessible until a replacement API is available in a future release. Other internal APIs are inaccessible by default.

To prepare for JDK 9, download JDK 9 early-access build and run jdeps to find out if your application and libraries depends on any JDK's internal API.

```
$ jdeps -dotoutput <dot-file-dir> -jdkinternals <one-or-more-jar-files....>
```

This jdeps command will output the dependencies in DOT file format and one output .dot file per JAR file.

Replace uses of the JDK's internal APIs

Below lists some of the JDK's internal APIs and the recommended way to replace their usage. See JEP 261 for the `--add-exports` command-line option to break in the encapsulation as a short-term migration purpose

Unsupported API (not for use)	Supported APIs (please use instead)	Note
core-libs		
protected java.lang.ClassLoader::defineClass method	java.lang.invoke.MethodHandles.Lookup::defineClass @since 9	Frameworks may use java.lang.invoke.MethodHandles::privateLookupIn to obtain a Lookup object with the permission to access the private members a target class in a different module if the framework is granted with deep reflection access to the target class.
sun.io	java.nio.charsets @since 1.4	
sun.misc.BASE64Decoder, sun.misc.BASE64Encoder, com.sun.org.apache.xml.internal.security.utils.Base64	java.util.Base64 @since 8	See http://openjdk.java.net/jeps/135
sun.misc.ClassLoaderUtil	java.net.URLClassLoader.close() @since 7	
sun.misc.Cleaner	java.lang.ref.PhantomReference @since 1.2	JDK-6417205 may help with the resource issues that can arise when mapped byte buffers are not unmapped in a timely manner. Libraries accessing sun.misc.Cleaner have to be fixed as direct byte buffer no longer uses sun.misc.Cleaner class; instead jdk.internal.misc.Cleaner. See JDK-6685587 and JDK-4724038
sun.misc.Service	java.util.ServiceLoader @since 1.6	
sun.misc.Timer	java.util.Timer @since 1.3	

sun.misc.Unsafe	java.lang.invoke.VarHandle since 9 java.lang.invoke.MethodHandles.Lookup::defineClass @since 9 java.lang.invoke.MethodHandles.Lookup::defineHiddenClass @since 15 java.lang.invoke.MethodHandles.Lookup::ensureInitialized @since 15	sun.misc.Unsafe consists of a number of use cases. The following features are identified to provide support in the future releases: • JEP 193: Enhanced Volatile • JEP 187: Serialization 2.0 • JEP 189: Shenandoah:Low-Pause GC • Arrays 2.0 • Project Panama • JEP 191: FFI • JEP 370: Foreign-Memory Access API (Incubator) & JEP 383: Foreign-Memory Access API (Second Incubator) • JEP 371: Hidden Classes See also • JDK-8044082 Efficient array comparison intrinsics • JDK-8033148 Lexicographic comparators for arrays
sun.reflect.Reflection.getCallerClass	java.lang.StackWalker::getCallerClass @since 9	See JDK-8043814 (Stack Walking API)
sun.util.calendar.ZoneInfo	java.util.TimeZone or java.time API @since 8	
security-libs		
sun.security.action.*	java.security.PrivilegedAction to call System.getProperty or other action @since 1.1	`AccessController.doPrivileged(` `  (PrivilegedAction<String>) () -> System.getProperty(key));`
sun.security.krb5.*	Some provided in com.sun.security.jgss javax.security.auth.kerberos.EncryptionKey @since 1.9 javax.security.auth.kerberos.KerberosCredMessage @since 1.9 javax.security.auth.kerberos.KerberosTicket.getSessionKey() @since 1.9	If internal classes are used to get the session key of Krb5Context, we now have ExtendedGSSContext for this purpose. JDK-8043071 resolved in JDK 9 b25
sun.security.util.SecurityConstants	java.lang.RuntimePermission, java.net.NetPermission, or specific Permission class @since 1.1	
sun.security.util.HostnameChecker	javax.net.ssl.SSLParameters.setEndpointIdentificationAlgorithm("HTTPS" or "LDAPS") can be used to enabled hostname checking during handshaking javax.net.ssl.HttpsURLConnection.setHostnameVerifier() can be customized hostname verifier rules for URL operations.	See also JDK-7192189 RFE to support the new endpoint identification.
sun.security.x509.*	javax.security.auth.x500.X500Principal @since 1.4	JDK-8056174 defines jdk.security.jarsigner.JarSigner API in JDK 9. This API can also be used to generate self-signed certificates.
com.sun.org.apache.xml.internal.security	javax.xml.crypto @since 1.6	
com.sun.net.ssl.**	javax.net.ssl @since 1.4	

security provider implementation class such as • com.sun.net.ssl.internal.ssl.Provider • sun.security.provider.Sun • com.sun.crypto.provider.SunJCE	java.security.Security.getProvider (NAME) @since 1.3 where NAME is the security provider name such as "SUN", "SunJCE".	In general, you should avoid depending on a specific provider as it may not be available on other Java implementations. See Oracle security providers documentation for more rationale.
sun.security.provider.PolicyFile() or sun.security.provider.PolicyFile(URL)	java.security.Policy.getInstance ("JavaPolicy", new java.security.URIParameter(uri)); @since 1.6	
client-libs		

java.awt.peer and java.awt.dnd.peer	Instead of doing:
	
	if (c.getPeer() != null) { .. }
	
	could be replaced with:
	
	if (c.isDisplayable()) { ... }
	
	To test if a component has a LightweightPeer, use:
	
	public boolean isLightweight() ; @since 1.2
	
	To obtain the color model of the component comes from the peer, instead of doing:
	
	getPanel().getPeer().getColorModel()
	
	could be replaced with:
	
	public ColorModel getColorModel();
		java.awt.peer.* and java.awt.dnd.peer.* types are encapsulated.
		
		API reference to java.awt.peer.* and java.awt.dnd.peer.* types are removed in JDK 9. See JDK-8037739 and awt-dev discussion

com.sun.image.codec.jpeg.** sun.awt.image.codec	javax.imageio @since 1.4	See JDK-6527962
com.apple.eawt	java.awt.Desktop @since 9	Seehttp://openjdk.java.net/jeps/272
JDBC		
com.sun.rowset.**	javax.sql.rowset.RowSetProvider @since 7	
JAXP		
org.w3c.dom.{html, css, stylesheets}	org.w3c.dom.{html, css, stylesheets} APIs are JDK supported APIs @since 9.	JDK-8042244 resolved in JDK 9 b62
org.w3c.dom.xpath	org.w3c.dom.xpath API is now JDK supported API @since 9	JDK-8042244 resolved in JDK 9 b62 JDK-8054196 for XPath support any API resolved in JDK 9 b49
com.sun.org.apache.xml.internal.resolver.**	javax.xml.catalog @since 9	See JDK-8023732 (XML Catalog API)

org.relaxng.datatype	org.relaxng.** will be repackaged in JDK 9. Users should include the org.relaxng.** types in the classpath.	See JDK-8061466
Others		
com.sun.tools.javac.**	javax.tools, javax.lang.model @since 1.6 com.sun.source.* @since 1.6	com.sun.tools.javac.Main is a supported API.
jdk.nashorn.internal.ir.**	JEP 236 Parser API for Nashorn	JDK-8048176 (Nashorn Parser API) resolved in JDK 9 b55