APPENDIX H

Java Version History

H.1 INTRODUCTION

Java is the most important advance in the programming technology invented during the last decade of the 20th century. Java technology is still evolving and is likely to be the primary programming language of the next millennium.

Java technology has many facets that work together to get the job done. Java environment basically consists of the following three entities:

- The language itself
- A class library (Application Program Interface (API) packages)
- A set of development tools.

Sun Microsystems, the inventors of Java, calls these three entities put together as *Java Development Kit* (JDK). This means that the JDK contains everything we need for developing Java programs. The new versions of Java are released as JDK versions by Sun Microsystems. That is, version JDK x.y means version Java x.y and vice versa.

Since the release of the original version of Java (known as Java 1.0) in May 1995, Sun Microsystems has been regularly releasing updates (changes and enhancements) of Java systems. Java 1.1 was released in March 1997 and Java 1.2 in early 1998. Fortunately, most of the changes and enhancements are related to the API packages and tools and very little changes have been introduced in the language itself. In early 1999, the Java 1.2 was renamed as Java 2 by Sun Microsystems and therefore, JDK 1.2, Java 1.2, and Java 2 all refer to the same thing.

The core API packages that contain numerous classes and interfaces have grown from around 200 in version 1.0 to more than 1500 in version 2. Table H.1 summarizes the core API packages available in Java 2 and their functions. The table shows not only the packages added during the stages 1.1 and 2 but also the number of classes and interfaces added to the various existing packages at these stages. These packages together contain more than 1500 classes and interface and define more than 13,000 methods. It is beyond the scope of this book to provide a complete description of all these classed and methods. (For more details, readers may refer to the JDK 1.2 documentation available in one of the Sun web sites or refer to the book

Java Developers Almanac 1998, Patrick Chan, Addison Wesley). More statistics of Java packages are given in Appendix I.

Table H.1 *API Packages Available in Java 2*

Sl No.	*Package*	*Stage First Added*	*Total Classes and Interfaces*			*Task/Function*
			1.0	*1.1*	*2*	
1.	`java.applet`	1.0	4	4	4	Provides basic functionality needed to implement applets.
2.	`java.awt`	1.0	46	61	78	Provides the standard GUI controls as well as drawing, printing and other capabilities
3.	`java.awt. accessability`	2	–	–	13	Supports the use of assistive technologies for disabled users.
4.	`java.awt.color`	2	–	–	7	Supports the ability to work with different color models.
5.	`java.awt.`	1.1	–	6	6	Supports clipboard operations, for transferring data between applications.
6.	`jawa.awt.dnd`	2	–	–	17	Supports drag–and–drop operations.
7.	`java.awt.dnd.peer`	2	–	–	3	Provides capability to access platform–dependent drag–and–drop facilities.
8.	`java.awt.event`	1.1	–	30	33	Provides foundation for processing events fired by AWT components.
9.	`java.awt.font`	2	–	–	15	Provides support for advanced font capabilities.
10.	`java.awt.geom`	2	–	–	33	Supports standard geometrical objects and transformations, in 2D.
11.	`java.awt.im`	2	–	–	3	Supports the Input Method API for internationalization.
12.	`java.awt.image`	1.0	12	14	52	A Java 2D API package that supports image processing.
13.	`java.awt.image.codec`	2	–	–	6	Supports image compression.
14.	`java.awt.image. renderable`	2	–	–	7	Supports functions for producing rendering-independent images.
15.	`java.awt.peer`	1.0	22	27	26	Provides support for interfacing with the underlying window system.
16.	`java.awt.print`	2	–	–	12	A Java 2D API package that supports printing of text and graphics.

(Contd)

Table H.1 *(Contd)*

Sl No.	*Package*	*Stage First Added*	*Total Classes and Interfaces*			*Task/Function*
			1.0	*1.1*	*2*	
17.	`java.awt.swing`	2	–	–	169	Provides foundation for the swing API, for creating completely portable GUI.
18.	`java.awt.swing.border`	2	–	–	10	Implements borders and border styles, around a swing component.
19.	`java.awt.swing.event`	2	–	–	38	Implements Swing events and event listeners.
20.	`java.awt.swing.plaf`	2	–	–	40	Supports the use of the pluggable look-and-feel, capabilities.
21.	`java.awt.swing.plaf.basic`	2	–	–	107	Provides default look-and-feel of Swing components.
22.	`java.awt.swing.plaf.metal`	2	–	–	42	Provides "metal" look-and-feel of Swing components.
23.	`java.awt.swing.preview`	2	–	–	21	Contains classes whose API have not been finalized yet.
24.	`java.awt.swing.table`	2	–	–	13	Implements the Swing table component for managing tables.
25.	`java.awt.swing.text`	2	–	–	87	Implements text processing functions such as selection, editing, etc.
26.	`java.awt.swing.text.html`	2	–	–	39	Provides basic HTML editing capabilities.
27.	`java.awt.swing.text.rtf`	2	–	–	1	Provides the capability to edit Rich Text Format documents
28.	`java.awt.swing.tree`	2	–	–	11	Provides capability to construct and manage tree-type data structure.
29.	`java.awt.swing.undo`	2	–	–	9	Provides undo and redo capabilities in applications.
30.	`java.beans`	1.1	–	23	25	Provides the basic Java beans functionality.
31.	`java.beans.beancontext`	2	–	–	19	Supports the implementation of execution environment for beans.
32	`java.io`	1.0	31	69	77	Performs a wide variety of input and output functions.

(Contd)

Table H.1 *(Contd)*

Sl No.	*Package*	*Stage First Added*	*Total Classes and Interfaces*			*Task/Function*
			1.0	*1.1*	*2*	
33.	`java.lang`	1.0	64	69	76	Provides support for implementing fundamental Java objects.
34.	`java.lang.ref`	2	–	–	6	Supports Reference Objects that are used to refer to other objects.
35.	`java.lang.reflect`	1.1	–	7	9	Provides capability to obtain reflective information about classes, and objects.
36.	`java.math`	1.1	–	2	2	Provides capability to perform arbitrary-precision arithmetic.
37.	`java.net`	1.0	19	26	32	Supports features for network programming.
38.	`java.rmi`	1.1	–	19	20	Provides capability to access objects on remote computers.
39.	`java.rmi.activation`	2	–	–	16	Supports persistent object reference and remote object activation.
40.	`java.rmi.dgc`	1.1	–	3	3	Supports functions for distributed garbage collection.
41.	`java.rmi.registry`	1.1	–	3	3	Supports distributed registry operations.
42.	`java.rmi.server`	1.1	–	23	24	Provides capabilities for supporting the server side of RMI.
43.	`java.security`	1.1	–	26	57	Provides basic foundation for the Security API.
44.	`java.security.acl`	1.1	–	8	8	Provides capability for implementing security access controls.
45.	`java.security.cert`	2	–	–	12	Provides support for parsing and managing digital certifications.
46.	`java.security.interfaces`	1.1	–	5	5	Supports implementation of the NIST digital signature of algorithm.
47.	`java.security.spec`	2	–	–	10	Provides specification for cryptographic keys
48.	`java.sql`	1.1	–	17	24	Provides support for the Java database connectivity.
49.	`java.text`	1.1	–	19	28	Provides support for internationalization of text and messages.

(Contd)

Table H.1 *(Contd)*

Sl No.	*Package*	*Stage First Added*	*Total Classes and Interfaces*			*Task/Function*
			1.0	*1.1*	*2*	
50.	`java.util`	1.0	14	26	49	Supports a variety of common programming needs.
51.	`java.util.jar`	2	–	–	8	Provides support for working with JAR (Java Archive) files.
52.	`java.util.mime`	2	–	–	3	Provides capability to work with MIME type objects.
53.	`java.util.zip`	1.1	–	17	17	Provides support for working with compressed files.
54.	`org.omg.CORBA`	2	–	–	104	Implements the foundation for supporting Java-CORBA integration.
55.	`org.omg.CORBA. ContainedPackage`	2	–	–	1	Describes a CORBA object in a CORBA container.
56.	`org.omg.CORBA. ContainerPackage`	2	–	–	1	Describes a CORBA container object.
57.	`org.omg.CORBA. InterfaceDefPackage`	2	–	–	1	Describes a CORBA interface definition.
58.	`org.omg.CORBA. ORBPackage`	2	–	–	1	Raises an exception when an invalid name is passed to an object request broker.
59.	`org.omg.CORBA.`	2	–	–	2	Signals exceptions related to type usage and constraints.
60.	`org.omg.CORBA. portable`	2	–	–	5	Supports vender-specific CORBA implementation.
61.	`org.omg.CosNaming`	2	–	–	22	Implements a tree-structured data type naming.
62.	`org.omg.CosNaming. NamingContextPackage`	2	–	–	18	Implements nodes within the tree-structured naming scheme.

H.2 CHANGES IN JAVA 1.1

As pointed out earlier, the updates include five kinds of changes:

1. Additions to the existing packages
2. Addition to new packages
3. Changes in the existing classes and members
4. Changes in the language itself
5. Changes in tools

Addition to the Existing Packages

Additions to the existing packages may include two things:

1. New classes in the existing packages
2. New members (fields, constructors, and methods) in the existing classes

As seen from Table H.1, new classes have been added to all the classes except **Applet** class. Similarly, a comparison of Tables J.2 and J.3 shows that new members have been added to many classes in almost all the existing packages. All these additions are aimed at enhancing the functionality of the existing API. Some important enhancements are listed as follows:

Abstract Windowing Toolkit (AWT) Enhancements

The additions have improved the functiona-lity of AWT to make the large-scale GUI development more feasible. Java 1.1 supports delegation-based event handling, data transfer such as cut-copy-paste, desktop color schemes, printing, mouseless operation, faster scrolling, popup menus and much more. These improvements have made Java 1.1 faster than Java 1.0.

I/O Enhancements

Java 1.1 has added character streams to the existing **java.io** package. These are like byte streams of Java 1.0 except that they operate on 16-bit Unicode characters rather than eight-bit bytes. Character streams make it easy to write programs that are independent of the user's culture and language and therefore easier to write "global programs". The process of writing a global program and ensuring that it can be used without change by anyone in the world is known as *internationalization*. In addition, two byte streams were added to support object serialization. Serialization lets us store objects and handle them with binary input/output streams.

Networking Enhancements

The 1.1 release made several enhancements to the networking package, **java.net**. It supports selected BSD-style options in the base classes and provides facility for finer granularity in reporting and handling network errors.

Native Methods Interface

Native methods are written in languages other than Java. The native methods interface from 1.0 has been completely rewritten and formalized. This interface is now known as the Java Native Interface (JNI). This provides capability for Java objects to access native methods.

Addition of New Packages

Java 1.1 has added the following new packages to provide new capabilities.

1. `java.awt.datatransfer`
2. `java.awt.event`
3. `java.beans`
4. `java.lang.reflect`
5. `java.math`
6. `java.rmi`
7. `java.rmi.dgc`
8. `java.rmi.registry`
9. `java.rmi.server`
10. `java.security`
11. `java.security.acl`
12. `java.security.interfaces`
13. `java.sql`
14. `java.text`
15. `java.util.zip`

Some major new capabilities are discussed as follows:

Security and Signed Applets

Java 1.1 supports the developments of digitally signed Java applications. It provides capability for key management, certificate management and security access controls.

Java Archive Files

Java Archive (JAR) files introduced in version 1.1 provide the capability for storing a number of files together by zipping them to shrink them, so the user can download many files at once. JAR files help us organize applets, applications, beans, and class libraries and support more efficient use of network resources.

JavaBeans Architecture

The new JavaBeans architecture provides specifications that describe Java objects suitable for reuse. The JavaBeans API allows third-party software vendors to create and ship reusable components (known as Beans), such as text, spreadsheets, graphic widgets, etc., that can be used by non-programmers to build applications.

Math Package

The new package **java.math** added to Java 1.1 contains two classes, **BigInteger** and **BigDecimal**. They provide support for performing arithmetic and bit manipulation on arbitrary precision decimal and integer numbers, without causing overflow or loss of precision.

Remote Method Invocation (RMI)

RMI API, introduced in 1.1, provides capability to create distributed Java-to-Java applications. Java objects in a local computer can invoke the methods of objects on a remote computer. The concept of object serialization is used to pass objects as parameters and return values in the remote method invocations.

Reflection

Reflection means identification of fields, constructors and methods of loaded classes and objects and using this information at runtime. These capabilities are used by Java Beans, object inspection tools, debuggers, and other Java applications and applets.

Java Database Connectivity (JDBC)

JDBC capability is provided by the package **java.sql**. This provides a uniform access to a wide range of relational databases from Java.

Changes in the Existing Classes and Methods

Many classes and methods have undergone changes from 1.0 to 1.1. Changes may be:

1. New members in the existing classes
2. Deprecation of classes
3. Deprecation of methods
4. Removal of classes
5. Removal of methods
6. Modification of design of classes
7. Modification of definition of methods

A complete description of all these changes is beyond the scope of this appendix. A brief description of classes and methods that have been deprecated or removed is given in Appendix I.

Changes in Language Itself

Changes in language itself were very minor. Bytes and shorts are accommodated as wrapped numbers by adding new classes **Byte** and **Short**. The abstract class **Number** gets two new concrete methods **byteValue** and **shortValue**.

A new class **Void** has been added as an uninstantiable place holder.

Inner Classes

One important change to the Java 1.1 is the ability to define classes as members of other classes. Such classes are called *nested classes*. Inner classes are one type of nested classes.

Instance Initializers

Java 1.0 supported initialization of only static variables (also known as class variables). Example:

```
class TestClass
{
static   {
     -------
     -------            // Initialization code
     -------
     }
}
```

Java 1.1 permits initialization of instance variables as well. Example:

```
class TestClass
{
     {
          -------
          -------       // Initialization code
          -------
     }
}
```

Array Initialization

Java 1.1 permits initialization of an array content in a **new** statement. For example, the following code creates an array of strings:

```
String [ ]   city =   new String [  ] {
                      "Madras"
                      "Delhi"
                      "Bombay" } ;
```

New Uses for Final

Java 1.1 allows us to declare the method parameters and local variables as final. However, a subclass can override a method and add or drop any final parameter modifiers. We can also deter initialization of a final variable, as long as we initialize it before it is used and assign a value to it exactly once.

H.3 CHANGES IN JAVA 2

Java 2 is a major upgrade of the core API and adds a standard extension architecture. Again the changes may be classified as follows:

1. Additions to the existing packages (Enhancements)
2. Adding new packages (New capabilities)
3. Changes in the existing class and methods
4. Changes in the language
5. Changes in tools

Enhancements in Java 2

Capabilities of java has been considerably enhanced by adding new classes to the existing packages as well as new members to almost all the existing classes. A comparison of Tables 1.3 and 1.4 (see Appendix I) will reveal this.

Security Enhancement

Java 2 provides users with the capability to specify security policies simply by editing the security permissions stored in their policy text files. Unless a permission is explicitly specified to code, it cannot access the resource that is guarded by that permission.

Java Beans Enhancement

Java 2 provides facilities to create more sophisticated JavaBeans components and applications. It provides capability to incorporate with other Beans and to learn information about their execution environment.

RMI Enhancement

Java 2 has significantly enhanced the RMI API. It supports remotely activated objects and object references that persist across multiple object activation.

JNI Enhancement

Java 2 extends Java native Interface (JNI) to incorporate new features to provide capabilities for controlling the manner in which native methods interact with the Java Virtual Machine.

JDBC Enhancement

Java 2 includes an improved version of JDBC–ODBC bridge driver and supports JDBC 2.0.

Audio Enhancement

Java 2 contains a new, high-quality sound engine that provides support for audio in applications as well as applets. The sound engine also provides support for the Musical Interface Digital Interface (MIDI) in addition to other traditional sounds.

JAR Enhancement

Java 2 JAR enhancements include improved tools for creating and updating JAR files and performing JAR I/O operations.

Reflection Enhancement

Reflection support was introduced in 1.1. Java 2 adds additional capability to identify a field, method, or constructor as suppressing Java language access controls. This facilitates the better use of reflection with the improved Java 2 security model.

New Capabilities in Java 2

In addition to improving the existing capabilities of the API, Java 2 has also added a number of new capabilities to it. Perhaps, the single most new feature is the addition of Java Foundation Classes (JFC) to Java 2. The JFC includes the functionalities such as Swing, Java 2D, Drag-and-Drop and Accessibility. Besides integrating JFC, Java 2 provides a number of other new capabilities. The new packages added include the following:

```
 1. java.awt.accessibility
 2. java.awt.color
 3. java.awt.dnd
 4. java.awt.dnd.peer
 5. java.awt.font
 6. java.awt.geom
 7. java.awt.im
 8. java.awt.image.codec
 9. java.awt.image.renderable
10. java.awt.print
11. java.awt.swing
12. java.awt.swing.border
13. java.awt.swing.event
14. java.awt.swing.plaf
15. java.awt.swing.plaf.basic
16. java.awt.swing.plaf.metal
17. java.awt.swing.preview
18. java.awt.swing.table
19. java.awt.swing.text
20. java.awt.swing.text.html
21. java.awt.swing.text.rtf
22. java.awt.swing.tree
23. java.awt.swing.undo
24. java.beans.beancontext
25. java.lang.ref
26. java.rmi.activation
27. java.security.cert
28. java.security.spec
29. java.util.jar
30. java.util.mime
31. org.omg.CORBA
32. org.omg.CORBA.ContainedPackage
33. org.omg.CORBA.ContainerPackage
34. org.omg.CORBA.InterfaceDefPackage
35. org.omg.CORBA.ORBPackage
36. org.omg.CORBA.TypeCodePackage
37. org.omg.CORBA.portable
38. org.omg.CosNaming
39. org.omg.CosNaming.NamingContextPackage
```

Swing

Swing is the code word used by the JavaSoft team for the improved AWT. Swing implements a new set of GUI components with a "pluggable" look and feel. Swing is implemented completely in Java. Pluggable look and feel architecture allows us to design a single set of GUI components that can automatically have the look and feel of any OS platform.

Java 2D

The new Java 2D API includes a set of tools for dealing with two-dimensional drawings and images. These include provision for colorspaces, text, line art and printing.

Accessibility

Accessibility API provides support for the use of *assistive technologies*, such as screen magnifiers, speech recognition systems, and Braile terminals intended for use by disabled users.

Drag and Drop

Drag and Drop capability of Java 2 facilitates data transfer across Java and native applications, between Java applications, and within a single application.

Java IDL

Java IDL in Java 2 provides a set of tools for interfacing Java objects with CORBA (Common Object Request Broker Architecture) objects and for developing CORBA objects in Java. It also includes a Java ORB (Object Request Broker) and an ORB name server.

Collections

The Collections API provides an implementation-independent framework for working with collection of objects such as sets, maps, lists, and linked lists.

Package Version Identification

A new capability of Java 2 allows applets and applications to obtain version information about a particular Java package at runtime.

Reference Objects

Reference objects (introduced in version 2) stores references to other objects. This feature can be used to implement object-catching mechanisms.

Input Method API

The new Input Method API provides support for Java's internationalisation. This enables all text-editing components to receive foreign language text input through input methods. It currently supports Japanese, Chinese and Korean languages.

Language Changes

There are not major changes in language. Only three methods of **Thread** class, **stop( ), suspend( ),** and **resume( )** have been depreciated because of errors and inconsistencies caused by them. Instead of using the **stop( )** method, it is proposed that a thread may monitor the state of a shared variable and stop execution by returning from its **run( )** method. Similarly, a thread may suspend and resume its execution based on the value of shares variables (by monitoring interface events).

Tools Changes

Java 2 has improved the tools available in the earlier versions and also added new tools. The **javakey** tool of 1.1 has been replaced by the new **keytool** and **javasinger** tools. The Java 2 now includes the following tools:

- **keytool** for maintaining a database of key pairs and digital certificates.
- **javasinger** for signing JAR files and verifying the signatures of signed files.
- **policytool** for creating and modifying the files that define security policy.
- **tnameserv** for implementing CORBA Common Object Services (COS) Naming Service.
- **rmid** for remote activation system daemon.

H.4 PERFORMANCE ENHANCEMENTS

The enhancements and new features added to Java 2 has considerably improved the performance of Java which has been a subject of criticism. Given below are some of the performance improvements achieved in Java 2.

- Improvements in multithreading performance
- Reduction in memory usage for string constants
- Faster memory allocation and garbage collection
- Improvements in the performance of the thread monitor methods

- Support of native libraries for some critical API classes
- Inclusion of just-in-time (JIT) compilers with Java 2

H.5 WHAT'S NEW IN JAVA 6

Java SE 6 is the latest version of Java that was released on December 11, 2006. It was codenamed as Mustang. Java 6 is more efficient and robust in comparison to the previous versions of Java. Higher efficiency is induced into this version through several key additions and updates. These additions have provided greater flexibility and newer options for the Java developers. Some of the key features of Java SE 6 are:

- It supports blending of scripting languages with java code.
- It provides extensive support for all JDBC databases, thus eliminating the need for configuring databases. As a result, the development of applications, which require fetching data from the database, has become simpler.
- Java SE 6 comes bundled with several new options for developing GUI-based applications. The major add-on is **SwingWorker**, which facilitates threading in the GUI applications.
- It offers specialized tools for monitoring and managing memory-heaps and other system related information.
- It offers several new features on the security front such as cryptographic services, Public Key Infrastructure (PKI) and XML-Digital signature.

Table H.2 lists some of the key additions in Java SE 6:

Table H.2 *Additions in Java SE 6*

Interfaces	Deque BlockingDeque NavigableSet NavigableMap ConcurrentNavigableMap isObjectMonitorUsageSupported isSynchronizerUsageSupported findDeadlockedThreads
Classes	ArrayDeque ConcurrentSkipListSet ConcurrentSkipListMap LinkedBlockingDeque AbstractMap.SimpleEntry AbstractMap.SimpleImmutableEntry Console
Methods	getTotalSpace getFreeSpace getUsableSpace setWritable setReadable setExecutable