Version 1.0 Pre–Beta 1

Changes Since the Last Release

The current (1.0 Pre–Beta1) release of the Java Development Kit is significantly different from the 1.0 Alpha3 release. From an applet programmer's point of view, the differences fall into three categories:

- Java tool changes
- API changes
- Java language changes

Java Tool Changes

This is the first release of the Applet Viewer.
For information, see Converting Applets and the appletviewer reference documentation.

This is the first release of the Java Debugger.
For information, see Converting Applets and the jdb reference documentation.

javaprof has been removed from the release
Also, the related –prof option for java has been removed.

The Java interpreter (java) has changed a bit.
See the java reference documentation for more information about the following changes:

- A new option (–mx) sets the maximum amount of memory that the heap is allowed to grow to.
- A new option (–D) lets you set properties and their values.
- The default size for the –ss option has grown from 64KB to 128KB.
- The default size for the –ms option in the interpreter has shrunk from 3MB to 1MB.

The Java compiler (javac) no longer supports the –ng option.

The Java disassembler (javap) has a new option (–l) to print line and local variable tables.

API Changes

The API has changed significantly in 1.0 Pre–Beta1. The most noticeable changes are described below. For details on the API, refer to The Java Class API. For information on how to convert 1.0 Alpha3 applets to this API, see Converting Applets.

The Applet class no longer has any public instance variables.
The following table shows where they went (changes that the **upgrade** script

handles automatically are prefixed with `[u]`):

```
    IVAR        REPLACEMENT
    ----        -----------
[u] appletURL   getCodeBase()
[u] documentURL getDocumentBase()
[u] height      size().height [Component method]
[u] width       size().width [Component method]
    bgColor     getBackground() [Component method]
    fgColor     getForeground() [Component method]
    font        getFont() [Component method]
    item        new Applet AWT superclass makes this unneeded
```

The Applet class now inherits from java.awt.Panel.

This makes Applets inherently graphical objects that can contain other graphical objects. This means that applets can (and should) use the the standard graphical user interface classes, such as Button, TextArea, List, and so on. It also means that Applets use the same event–handling and drawing interfaces as other graphical objects.

Applet subclasses must be declared public.

The **upgrade** script automatically makes this change for you.

The AWT package has changed significantly.

Not only does it have a new name (java.awt), but its API has been cleaned up. Converting applets that used the old AWT is tedious, but writing new programs that use the AWT is much easier than before. Here are a few of the changes:

- The Frame constructors are much simpler than before. For example, the parentFrame argument is gone; now you use show() or add() to set the parent. Instead of specifying w and h arguments, you use reshape(). Instead of specifying bg, you use setBackground().
- The Frame resize() method now requires an argument specifying the new size.
- The Window paint() method now requires a Graphics object as an argument.

The exception hierarchy has changed.

The browser.audio.AudioData class is no longer in the API.

It has been replaced in the applet API by the java.applet.AudioClip interface.

Images are now processed asynchronously.

This can greatly improve perceived performance since Image objects can now be created without having to load the entire image into memory first. There are two ways in which this affects the way that Applets use images.

First, when an applet calls getImage(), an Image object is returned immediately without making a net connection to load it. This allows the init() methods of applets to run much faster if they call the getImage() method. This also means, however, that an Image object is returned even if the indicated URL does not exist. If there is an error loading an image, then the Applet is notified using the ImageObserver interface.

Secondly, when an applet calls any of the methods that return information or operate on an image, such as

- anImage.getWidth()

- anImage.getHeight()
- anImage.getProperty()
- aGraphics.drawImage()

the method returns immediately, whether or not the information is available. Each method has a way to indicate that the information is not yet available. getWidth() and getHeight() return −1, getProperty() return null, and drawImage() return false. Each of these methods also takes an ImageObserver parameter. This ImageObserver is notified when the requested information becomes available. The Applet class already implements this interface and by default calls the repaint() method whenever new data is available, so most applets can just pass "this" for the ImageObserver argument so they will automatically repaint themselves as the images are loaded.

DIBitmaps are no longer supported.

Due to the need for supporting asynchronous loading of images and the desire to support images with more than 256 colors, a new image architecture was designed around ImageProducer and ImageConsumer interfaces. There is now a supported method to create an image from a Java array of pixels which hides the details of this interface for most applet writers. There is also a new ImageFilter class that makes it easier to perform manipulations of existing images within the new asynchronous architecture without having to read the image's data into a DIBitmap and create a new Image from that DIBitmap. See the RGBImageFilter, FilteredImageSource, and MemoryImageSource class documentation for examples.

Java Language Changes

For information about the Java language, refer to the **Java Language Specification.** (The specification does not yet include the following changes. We will update it soon.)

- Methods must declare (using the new **throws** keyword) the exceptions they can throw.
  For example:

```
void doSomething() throws AnException, AnotherException {
    . . .
}
```

- Only Throwable objects can be thrown and caught.
- Default type of real constants is now **double** (was **float**).
- Comma (,) operator has been deleted.
- Method matching algorithm changed.
- You can now use \uXXXX anywhere.
- Interface variables must be initialized with a constant expressions.
- Expression statements are restricted.
- No more warnings, only errors.
- No more `new("...")` operator; use `Class.forName("...").newInstance()`.
- Threadsafe is volatile again.
- Meaning of the **instanceof** operator changed (null is no longer an instance of

everything).

- Restrictions on use of ==, !=, instanceof, checkcast to cases of superclass or subclass.
- The meaning of floating point % got clarified.
- Shift operators with large counts got clarified.
- The definition of what a constant is got more precise.
- It is a compile time error to overflow a constant.
- \<NL> is no longer legal.
- Abstract classes must be declared **abstract.**
- Static methods are by default final.
- The **final** keyword is no longer legal for local variables.
- Cast to **void** is no longer legal.
- Several reserved words have been added to the Java language but are not currently used: **cast, future, generic, inner, operator, outer, rest, var**
