Version 1.0 Beta 2

Changes Since the Last Release

There have been several bug fixes, security improvements, and API changes since the JDK Beta release −− some of the major changes are listed below. For information on the changes between Alpha 3 and Beta, look at the Alpha 3 to Pre−Beta 1 changes document, and the Pre−Beta 1 to Beta changes document.

API Changes

These API changes (for the 1.0 Beta 2 version of the JDK) are compatible with Netscape's 2.0 Beta 4 (and later) version of Navigator.

- Class changes:
  - Added java.awt.image.PixelGrabber
  - Added java.io.FileDescriptor()
  - Added java.lang.Compiler
  - Added java.lang.CloneNotSupportedException
  - Added java.lang.IllegalMonitorStateException()
  - Added java.net.DatagramPacket
  - Added java.net.DatagramSocket
  - Removed java.lang.UNIXProcess
  - Removed java.lang.Win32Process
- Method changes:
  - Added java.applet.Applet.resize(java.awt.Dimension)
  - Added java.applet.AppletContext.showDocument(java.net.URL, java.lang.String)
  - Added java.awt.Container.add(java.awt.Component, int)
  - Added java.awt.FontMetrics.getMaxDescent()
  - Added java.awt.Frame.getCursorType()
  - Added java.awt.Graphics.drawImage(java.awt.Image, int, int, java.awt.Color, java.awt.image.ImageObserver)
  - Added java.awt.Graphics.drawImage(java.awt.Image, int, int, int, int, java.awt.Color, java.awt.image.ImageObserver)
  - Added java.awt.GridBagLayout.getLayoutDimensions()
  - Added java.awt.GridBagLayout.getLayoutOrigin()
  - Added java.awt.GridBagLayout.getLayoutWeights()
  - Added java.awt.GridBagLayout.location(int, int)
  - Added java.awt.Image.flush()
  - Added java.awt.Insets.clone()
  - Added java.awt.List.addItem(java.lang.String, int)
  - Added java.awt.List.replaceItem(java.lang.String, int)
  - Added java.awt.MediaTracker.getErrorsAny()

- Added java.awt.MediaTracker.getErrorsID(int)
- Added java.awt.MediaTracker.statusAll(boolean)
- Added java.awt.MediaTracker.statusID(int, boolean)
- Added java.awt.MediaTracker.waitForAll(long)
- Added java.awt.MediaTracker.waitForID(int, long)
- Added java.awt.Scrollbar.getLineIncrement()
- Added java.awt.Scrollbar.getPageIncrement()
- Added java.awt.Scrollbar.setLineIncrement(int)
- Added java.awt.Scrollbar.setPageIncrement(int)
- Added java.awt.TextArea.appendText(java.lang.String)
- Added java.awt.Toolkit.checkImage(java.awt.Image, int, int, java.awt.image.ImageObserver)
- Added java.awt.Toolkit.getColorModel()
- Added java.awt.Toolkit.getFontMetrics(java.awt.Font)
- Added java.awt.Toolkit.getImage(java.lang.String)
- Added java.awt.Toolkit.getImage(java.net.URL)
- Added java.awt.Toolkit.prepareImage(java.awt.Image, int, int, java.awt.image.ImageObserver)
- Added java.awt.peer.FileDialogPeer.setFile(java.lang.String)
- Added java.awt.peer.ListPeer.addItem(java.lang.String, int)
- Added java.awt.image.DirectColorModel.getRGB(int)
- Added java.awt.image.ImageFilter.clone()
- Added java.awt.image.ImageFilter.resendTopDownLeftRight(java.awt.image.ImageProducer)
- Added java.io.File.delete()
- Added java.io.PrintStream.checkError()
- Added java.lang.Boolean(java.lang.String)
- Added java.lang.Boolean.hashCode()
- Added java.lang.Boolean.valueOf()
- Added java.lang.Character.equals(java.lang.Object)
- Added java.lang.Character.hashCode()
- Added java.lang.Double(java.lang.String)
- Added java.lang.Double.toString(double)
- Added java.lang.Float(java.lang.String)
- Added java.lang.Object.finalize()
- Added java.lang.Runtime.exec(java.lang.String, java.lang.String[])
- Added java.lang.Runtime.exec(java.lang.String[], java.lang.String[])
- Added java.lang.SecurityManager.checkConnect(java.lang.String, int, java.lang.Object)
- Added java.lang.SecurityManager.checkDelete(java.lang.String)
- Added java.lang.SecurityManager.checkPropertyAccess(java.lang.String)
- Added java.lang.SecurityManager.checkPropertyAccess(java.lang.String, java.lang.String)
- Added java.lang.SecurityManager.checkRead(java.lang.String, java.lang.Object)
- Added java.lang.SecurityManager.checkTopLevelWindow(java.lang.Object)
- Added java.lang.SecurityManager.getInCheck()
- Added java.lang.SecurityManager.getSecurityContext()
- Added java.lang.String(java.lang.StringBuffer)
- Added java.lang.Thread.stop(java.lang.Throwable)
- Added java.lang.ThreadGroup.uncaughtException(java.lang.Thread,

java.lang.Throwable)
- Added java.lang.Throwable.printStackTrace(java.io.PrintStream)
- Added java.net.SocketImpl.available()
- Added java.net.SocketImpl.getFileDescriptor()
- Added java.net.SocketImpl.getInetAddress()
- Added java.net.SocketImpl.getLocalPort()
- Added java.net.SocketImpl.getPort()
- Added java.net.URLEncoder.encode(java.lang.String)
- Added java.net.URLStreamHandler.setURL(java.net.URL, java.lang.String, java.lang.String, int, java.lang.String, java.lang.String)
- Changed FileInputStream(int) to java.io.FileInputStream(java.io.FileDescriptor)
- Changed FileOutputStream(int) to java.io.FileOutputStream(java.io.FileDescriptor)
- Changed FramePeer.setCursor(java.awt.Image) to java.awt.peer.FramePeer.setCursor(int)
- Changed RandomAccessFile(int) to java.io.RandomAccessFile.getFD()
- Changed SecurityManager.checkRead(int) to java.lang.SecurityManager.checkRead(java.io.FileDescriptor)
- Changed SecurityManager.checkWrite(int) to java.lang.SecurityManager.checkWrite(java.io.FileDescriptor)
- Removed java.awt.Graphics.scale(float, float)
- Removed java.awt.GridBagConstraints.copy(java.lang.Object)
- Removed java.awt.MediaTracker.checkAll()
- Removed java.awt.peer.ListPeer.addItem(java.lang.String)
- Removed java.awt.MediaTracker.checkID(int, boolean)
- Removed java.lang.Object.copy(java.lang.Object)
- Removed java.lang.Thread.stop(java.lang.Object)
- Removed java.util.Date.setDay(int)

Other Changes

- **PROTECTED**
  The meaning of the modifier "protected" for instance variables and methods has been changed. For example, suppose that you have

```
class Parent {
    protected int x;
}
```

and

```
class Child extends Parent {   ..... }
```

The class Child can only access "x" on objects that are of type Child (or a subtype of Child). The following two uses of "x" are both legal

```
boolean equal(Child other) {
   return x == other.x
}
```

because the left−most "x" is implicitly "this.x", and both "this" and "other" are of type Child.

Now, suppose that class Child has a subclass SubChild, then in this example

```
static boolean isEqual(Parent one, SubChild two) {
    return one.x == two.x          /* "one.x" illegal */
}
```

"one.x" is illegal, and "two.x" is legal.

"static" protected variables and methods are accessible from any child class since there is no "object" through which to access them. "super.protectedMethod(...)" is always legal.

Note that "protected" still allows access of the variable/method to other classes in the same package. To allow access only to those classes that satisfy the above rules, you must use "private protected".

- **FINALIZE**
  There is now an empty finalize() method in Object

```
protected void finalize() throws Throwable { }
```

  This means that:
  - If you define a finalize() method for your class, it must be protected or public.
  - All finalizers can call super.finalize(), as appropriate, without about whether or not the parent actually has a finalize() method.

- **CLONE**
  The definition of clone() in Object is now

```
protected native Object clone() throws CloneNotSupportedException;
```

  If the class of an object has **not** been declared to implement the interface Cloneable, then a call to clone() on that object will fail and give the error CloneNotSupportedException. Remember, you can only call clone() on an object of the current class's type, or one of its subtypes.

  If a class wishes to allow others to make copies, it should do the following:

```
class FOO  ...... implements Cloneable {
     ........

     public clone() {
         try {
             Object result = super.clone()
             ... other class dependent copying . . ..
             return result;
         } catch (CloneNotSupportedException e) {
             throw appropriate error.
         }
     }
```

If FOO extends Object, then the appropriate error should indicate that you've gotten an internal error!

- **WARNINGS**
  Many Compiler warning messages are now errors, and must be dealt with.

- **OVERRIDING**
  Contravariant return types for methods has been removed, overriding methods now need to have the same return type as the overriden method.

- **EVENTS**
  In previous versions of the JDK, awt programs couldn't intercept events that were sent to native widgets. In theory, an awt program can subclass a component, say a TextField, override it's handleEvent() method and can then:
  - Pass the event along unmodified, by returning false.
  - Pass the event along with modifications by modifying the event and then returning false.
  - Consume the event without passing it on by returning true.

  If a component's handleEvent() method returns false, the event keeps travelling up the containment hierarchy. If none of the component's parents return true from their handleEvent methods, then the event is sent to the peer which will consume the event. Unfortunately, for 1.0beta2 (and fcs) the only events that will be able to be filtered by an application are keyboard events. Future versions of the JDK will allow all events to be filtered by the awt program.

  Note that if an applet's textfields don't seem to be getting any keyboard input, it's probably because the applet is (incorrectly) returning "true" from it's handleEvent() method. This causes the event to not be sent to the actual textfield.

Last Updated: 12 Dec 1995