Ken's Java Notes

Compiling

javac someFile.java
java someFile.class *running*

jar_manifest (sample file):

```
Manifest-Version: 1.0
Main-Class: IVInspect
Created-By: Ken Freed (kfreed@verizon.net)
```

```
jar cvfm IVInspect.jar jar_manifest *.class
ParsingPkg\*.class MapEditPkg\*.class
```

See: (later in this document):
Sample Setup and Build Instructions
Sample detailed instructions from a former project

Simple Variables-Objects

Most of the Java utility classes require the use of objects. Thus, if you needed to store an integer in a hashtable, you would have to "wrap" an Integer instance around it.

integer anInt = new Integer(55); //obj
- or -
integer anInt = 55; // simple var

// convert from obj to simple var:
int anInt2 = anInt.intValue();

// string to int
int anInt2 = Integer.parseInt("25");

Simple types-objs: Boolean, Byte, Character, Double, Float, Integer, Long, Short

Decisions

Not equal: !=

if ( ) {
} **else if** ( ) {
} else {
}

Float aFloat1 = new Float(3.0f);
Float aFloat2 = new Float(3.05f);
if (aFloat1.equals(aFloat2)) {
}

Looping

int aLength = anArray.length();
for (i = 0; i < aLength; i++) {
}

do {
} while (   );

while (   ) {
}

Arrays:

int[] aIntArray = new int[250];
aIntArray[0]++; //1st is zero
String aName[];
String reindeerNames[] = { "Dasher", "Dancer" };
Arrays.**sort**(reindeerNames);

String mostFamous = "Rudolf red-nosed";
char[] mfl = mostFamous.toCharArray();

```
/* Note: 2D array initialization is COLUMN major */
                                                /* west */
       int gWaferMap[][] =  { {0,0,0,0,0,0,0,0,0,0},
                              {0,0,0,0,1,1,0,0,0,0},
                              {0,0,0,1,1,1,1,0,0,0},
                              {0,1,1,1,1,1,1,1,0,0},
                              {0,1,1,1,1,1,1,1,1,0},
                   /* north */  {1,1,1,1,1,1,1,1,1,1},
                              {0,1,1,1,1,1,1,1,1,1},
                              {0,1,1,1,1,1,1,1,1,1},
                              {0,0,1,1,1,1,1,1,1,0},
                              {0,0,0,1,1,1,1,0,0,0},
                              {0,0,0,1,1,1,1,0,0,0} };
                                             /* east */
```

toString

```
String []array = new String[3];
array[0] = "Australia";
array[1] = "Denmark";
array[2] = "Italy";

System.out.println(Arrays.toString(array));
```

//output:
[Australia, Denmark, Italy]

Vectors *can grow or shrink at any time*

import java.util.*;
Vector<String> aStringVec =
                new Vector<String>();

```
aStringVec.add("Vance");
aStringVec.add("Ken");
Int theSize = aStringVec.size();
String aFirstOne = aStringVec.get(0);

if (aStringVec.contains("Ken") ) {
   System.out.println("Ken found");
}
```

vector iterator

```
for (String aName : aStringVec ) {
   System.out.println(aName);
}
```

Stack: a type of vector

```
import java.util.*;           // for Stack
...
private Stack<Integer> defect_locs_stack =
      new Stack<Integer>();
...
defect_locs_stack.push(x);
x=defect_locs_stack.pop();
x=defect_locs_stack.peek();
aBool=defect_locs_stack.empty();
anInt=defect_locs_stack.search(look4);
```

Hash Map

```
HashMap aHash = new HashMap();
aHash.put(aKey0, anObject0);
aHash.put(aKey1, anObject1);
```

Hash Table

```
import java.util.Hashtable;
import java.util.Enumeration;

Hashtable <String,Integer> hashtable =
   new Hashtable<String,Integer>();

hashtable.put( "One", new Integer(1) );
hashtable.put( "Two", new Integer(2) );
hashtable.put( "Three", new Integer(3) );

hashtable.remove("One");

hashtable.size()

if( hashtable.contains( new Integer(1) ) ){
if( hashtable.isEmpty() ){
if( hashtable.containsKey("One") ){
```

```
// get method returns Object, so we need to cast it
Integer one = (Integer) hashtable.get("One");
//Retriving all keys from the Hashtable
Enumeration e = hashtable.keys();
while( e. hasMoreElements() ){
    System.out.println( e.nextElement() );
}
```

```
//    Retriving all values from the Hashtable
   import java.util.Hashtable;
   import java.util.Iterator;
   import java.util.Set;

   String aDevice;
   DEVICE_CSV aDevice_csv;

   Set<String> devices =
   gDEVICE_CSV_HASH.keySet();

    Iterator<String> itr = devices.iterator();
   while (itr.hasNext()) {
       aDevice = itr.next();

    aDevice_csv = gDEVICE_CSV_HASH.get(aDevice);

    logmsg("Device:" + aDevice +
           " dieSize:<" +
Double.toString(aDevice_csv.dieSizeX) + "," +
Double.toString(aDevice_csv.dieSizeY) + ">"
        );
}    // while
```

StringBuffer

String are "immutable". once a String object has been created there is no way to modify the string of text it represents. Thus, each method that operates ona String typically returns a new String object that holds the modified String.

StringBuffe*r - can grow, has an append; e.g.:*

```
StringBuffer aString = "Now";
aString.append(" is the time");

StringBuffer theLot = new StringBuffer();
theLot.insert(0,aString);
```

String buffers are used by the compiler to implement the binary string concatenation operator +. For example, the code:

```
x = "a" + 4 + "c"
```

is compiled to the equivalent of:

```
x = new
StringBuffer().append("a").append(4).append("c")
        .toString()
```

// erase StringBuffer, ready for reuse:
gBIN_TITLE[bin].setLength(0);

String

```
String nameBuf[] = {"Joe","Ken"};
int length = nameBuf.length;
```

```
java.util.Arrays.sort(nameBuf);

String aName = "Hello";
int nameLength = aName.length();
String uaName = aName.toUpperCase();
String laName = aName.toLowerCase();

String aSubString = aName.substring(0,5);

String aTrimmedString = aName.trim();

boolean bIsSame = uaName.equals(laName);
boolean bIsSame =
            uaName.equalsIgnoreCase(laName);
```

if (uaName == laName) *compares object refs*

```
String aString = "Now is the time";
int index = aString.indexOf("the");
```

String to Integer:

String **string_count** = "25";
int int_count = **Integer**.parseInt(**string_count**);

Integer to String:

```
Integer.toString(x + 1);
```

Split and join

```
String aString = "ken,bob,richard,harry";
String pieces[] = aString.split(",");
- or -
String lines[] = null;
lines = aString.split("\n");

for (int i = 0; i < pieces.length; i++){...}

StringTokenizer aStrTok =
 new StringTokenizer(aStrTok, ",", false);
```

An instance of StringTokenizer behaves in one of two ways, depending on whether it was created with true or false:

* If the flag is false, delimiter characters serve to separate tokens. A token is a maximal sequence of consecutive characters that are not delimiters.
* If the flag is true, delimiter characters are themselves considered to be tokens. A token is thus either one delimiter character, or a maximal sequence of consecutive characters that are not delimiters.

Hash

```
JTextField mInitials_entry = new ...
String anEntry = mInitials_entry.getText();
if (anEntry.length()  < 3) {
```

Iteration

for (Iterator it = **collection**.iterator();
    it.hasNext();   ) {
  Object element = it.next();
}

A **Map** is an associative array, or dictionary (, or Perl hash)

java.util
Interface Map<K,V>

Type Parameters:
K - the type of keys maintained by this map
V - the type of mapped values

```
Set keys = map.keySet(); //set of keys in map
Iterator keyIter = keys.iterator();
while (keyIter.hasNext()) {
```

for (Iterator it = **map.keySet().**iterator();
    it.hasNext();   ) {
  Object element = it.next();
}

for (Iterator it = **map.values**().iterator();
    it.hasNext();   ) {
  Object element = it.next();
}

```
map.entrySet() //returns a Set that
contains all the associations from
the map.
```

```
Set entries = map.entrySet();
Iterator entryIter = entries.iterator();
while (entryIter.hasNext()) {
 Map.Entry entry = (Map.Entry)entryIter.next();
 Object key = entry.getKey(); // Get the key from the entry.
 Object value = entry.getValue(); // Get the value.
}
```

for (Iterator it = **map.entrySet**().iterator();
    it.hasNext();   ) {
  Map.Entry entry = (Map.Entry)it.next();
  Object key = entry.getKey();
  Object value = entry.getValue();
}

File I/O

```
//open for reading
BufferedReader hFile = new
   BufferedReader(
      new FileReader("somefile.txt")
                    );

File aFile = new File("somefile.txt");
InputStream hFile = new FileInputStream(aFile);
long fileSize = hFile.length();
byte fileBytes[] = new byte[(int)fileSize];
int offset = 0;
int numead = 0;
while ((offset < bytes.length) &&
        (numRead hFile.read(bytes,offset,
                 bytes.length-offset))
      >= 0)) {
    offset += numRead;
}
hFile.close();


//open for writing
BufferedReader hFile = new
   BufferedReader(
      new FileWriter("somefile.txt")
                    );
```

OO Misc

If more than one class is defined in the same source file, only one of the classes can be public.

The name of the .java source file should match the name of the public class.

If no superclass constructor is called, Java automatically calls one with no arguments.

You do not need to create an instance of a public static/class method

Different Java Constructs

1: Special initialization for a particular instance

```
JPanel aJPanel = new JPanel() {
        <some code for this aJPanel instance >
};
```

2: Scoping the instance

```
class MapPanel extends JPanel {
        <some code for any MapPanel instance>
}

MapPanel aMapPanel = new MapPanel();
```

3: Nominal case

```
public class IVproto3 extends JFrame {

        <some code>

        public IVproto3( ) {

                <some constructor code>
        }

        public static void main (String[] arguments) {

                IVproto3 anIVproto3 = new IVproto3();
      }
}
```

4: Interfaces

```
public class IVproto3 extends JFrame implements MouseMotionListener {
//  Register in constructor of the JComponent that we want to pickup these events
// (1) this = the JComponent instance that we want to pickup these events
   addMouseMotionListener(this); // (2) this=an instance which implements MouseMotionListener

// You MUST stub out ALL the interfaces for MouseMotionListener
// in the class that implements MouseMotionListener
     /*--------------------------*/
     /* mouse moved event        */
     /*--------------------------*/
     public void mouseMoved(MouseEvent e) {
     }

     /*--------------------------*/
     /* mouse dragged event      */
     /*--------------------------*/
     public void mouseDragged(MouseEvent e) {

     }
}
```

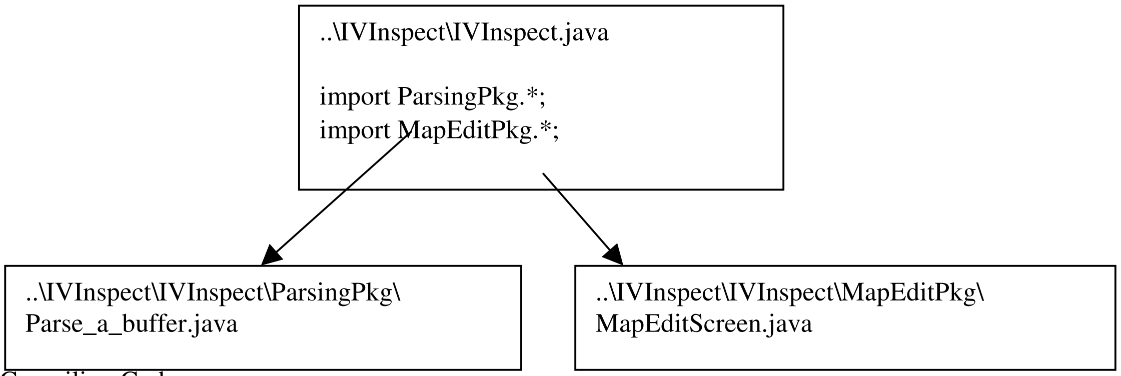
Sample Setup and Build Instructions

Sample detailed instructions from a former project

Compiling the Java Code

Overview

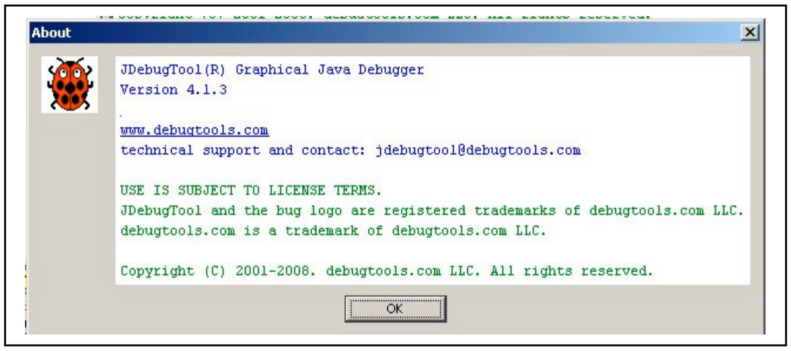
The IVInspect Java source code is made up of a Main Screen and two Java packages as follows:


..\IVInspect\IVInspect.java
import ParsingPkg.*;
import MapEditPkg.*;
..\IVInspect\IVInspect\ParsingPkg\ Parse_a_buffer.java
..\IVInspect\IVInspect\MapEditPkg\ MapEditScreen.java

Compiling Code

All compilations were done in a Dos shell using the command line compiler.

The -g option is to allow debugging. The jde debug tool was used; i.e.:



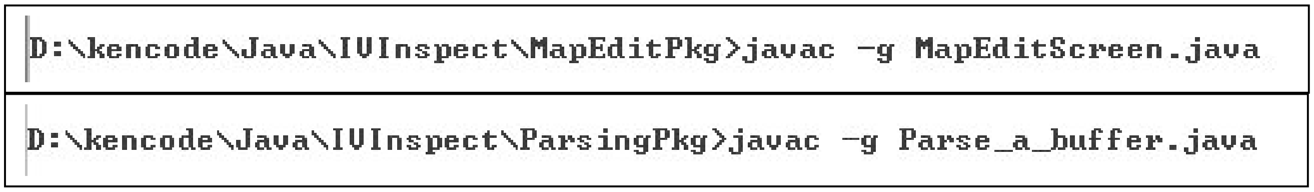
JDebugTool(R) Graphical Java Debugger
Version 4.1.3

www.debugtools.com
technical support and contact: jdebugtool@debugtools.com

USE IS SUBJECT TO LICENSE TERMS.
JDebugTool and the bug logo are registered trademarks of debugtools.com LLC.
debugtools.com is a trademark of debugtools.com LLC.

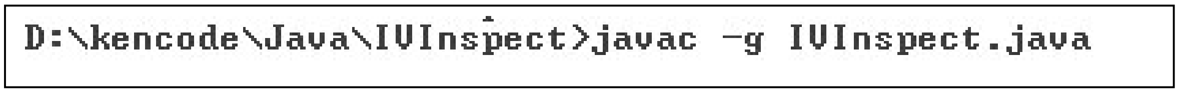
Copyright (C) 2001-2008. debugtools.com LLC. All rights reserved.

1. First compile the packages, e.g.:



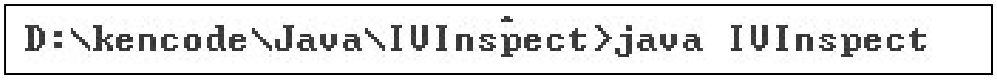
```
D:\kencode\Java\IVInspect\MapEditPkg>javac -g MapEditScreen.java
D:\kencode\Java\IVInspect\ParsingPkg>javac -g Parse_a_buffer.java
```

2. Then compile the file holding the "main" (entry point) routine:



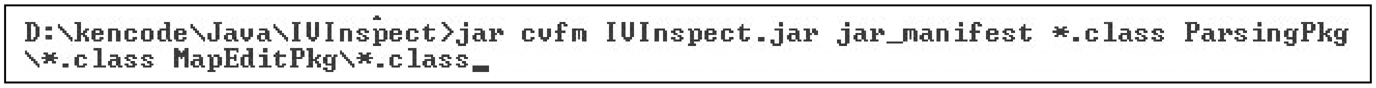
```
D:\kencode\Java\IVInspect>javac -g IVInspect.java
```

At this point, upon successful compiles, the java app can be run by issuing the command:



```
D:\kencode\Java\IVInspect>java IVInspect
```

3. To package the compiled code into a "jar" file (so that you don't have to maintain the development directory structure in order to run it), issue the command:



```
D:\kencode\Java\IVInspect>jar cvfm IVInspect.jar jar_manifest *.class ParsingPkg
\*.class MapEditPkg\*.class_
```

Where the manifest file (jar_manifest) is a text file that defines which file has the "main" (entry point), and is:

```
Manifest-Version: 1.0
Main-Class: IVInspect
Created-By: Ken Freed (kfreed@verizon.net, kxf@cypress.com)
```

4. To run the jar file (as exemplified in the IVInspect.bat file) issue the command:

```
java -jar IVInspect.jar
```

Note that the IVInspect.bat file uses:

```
start javaw -jar IVInspect.jar
```

Where:

javaw : Means don't leave a Dos (console) window open.

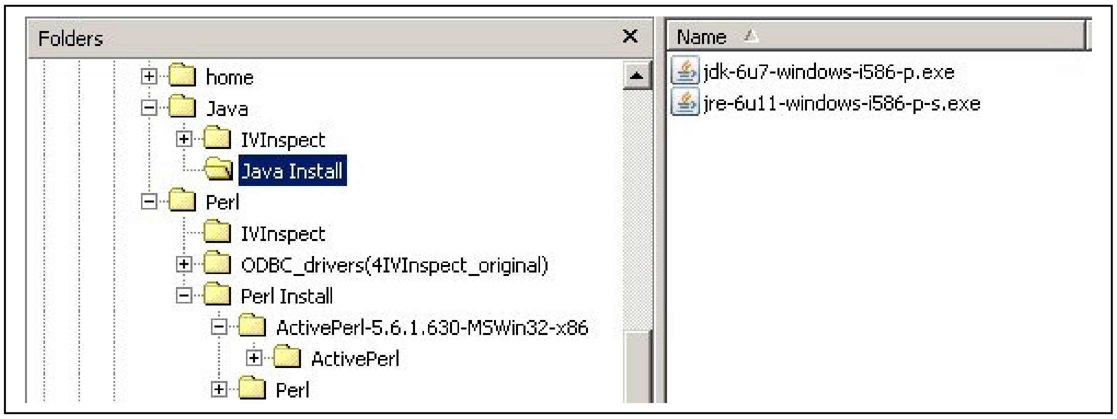
- You might want to change this to just use "java", so you can see error messages if you have problems

start : is a batch file command that says

- "start this script and then don't wait around for it to finish, just go away"

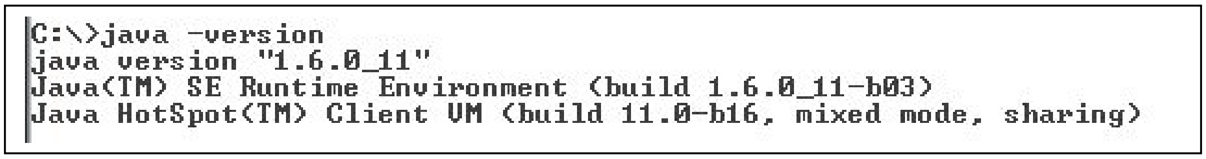
Java Installation

The Java directories are included on the CD ROM as follows:


Folders
home
Java
IVInspect
Java Install
Perl
IVInspect
ODBC_drivers(4IVInspect_original)
Perl Install
ActivePerl-5.6.1.630-MSWin32-x86
ActivePerl
Perl
Name
jdk-6u7-windows-i586-p.exe
jre-6u11-windows-i586-p-s.exe

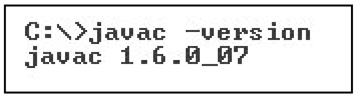
The "jre..." is the "Java Runtime Environment". You need this to run Java programs. You might already have this on your PC.

- To check, open up a Dos command prompt shell and type "java -version", e.g.:



```
C:\>java -version
java version "1.6.0_11"
Java(TM) SE Runtime Environment (build 1.6.0_11-b03)
Java HotSpot(TM) Client VM (build 11.0-b16, mixed mode, sharing)
```

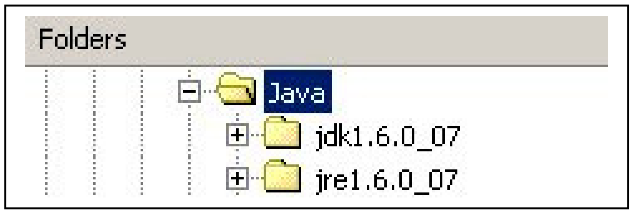
The "jdk..." is for "Java Development Kit". You need this to change the Java source code (more on how to rebuild the Java IV Inspect application later). To see if you already have this on your PC, type "java**c** -version", e.g.:



```
C:\>javac -version
javac 1.6.0_07
```

Setting the Java Classpath Environment Variable

Most of the time, the Java Runtime Environment and the Java Development Kit are installed under the "c:\Program Files\..." directory, e.g.:

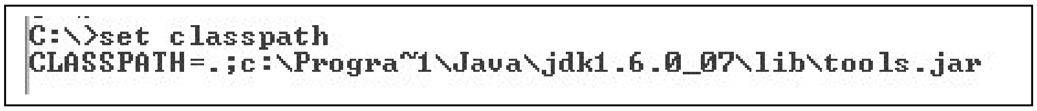


The JDK includes another copy of the JRE.

The separate JRE directory is needed if you do not have the JDK installed (hence you are not rebuilding Java code).

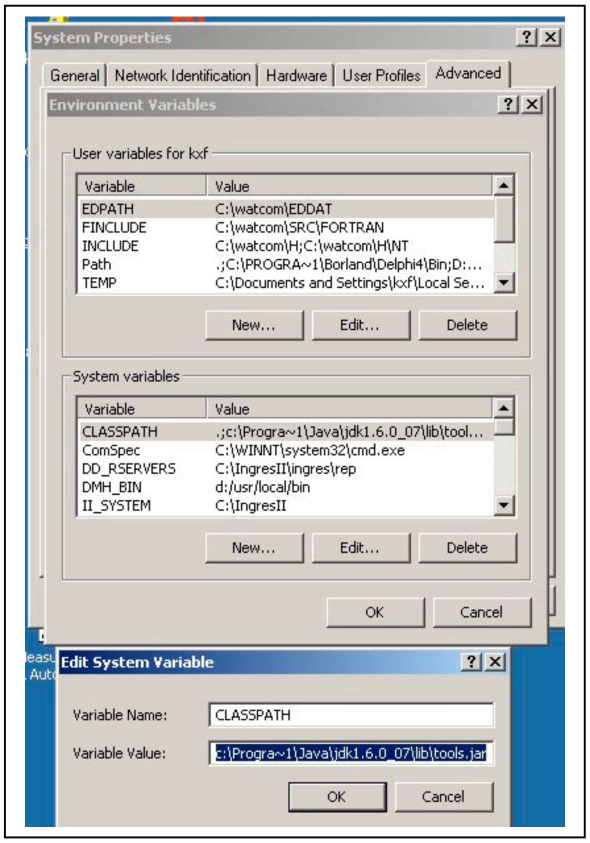
In order to run and build Java code, the CLASSPATH environment variable must be set to point to the jdk.

To see your CLASSPATH environment variable:



```
C:\>set classpath
CLASSPATH=.;c:\Progra~1\Java\jdk1.6.0_07\lib\tools.jar
```

To permanently set a system CLASSPATH environment variable, go to the windows:
control panel->system->advanced->environment variables, e.g.(for Win2K):



User variables for kxf

Variable	Value
EDPATH	C:\watcom\EDDAT
FINCLUDE	C:\watcom\SRC\FORTRAN
INCLUDE	C:\watcom\H;C:\watcom\H\NT
Path	.;C:\PROGRA~1\Borland\Delphi4\Bin;D:...
TEMP	C:\Documents and Settings\kxf\Local Se...

System variables

Variable	Value
CLASSPATH	.;c:\Progra~1\Java\jdk1.6.0_07\lib\tool...
ComSpec	C:\WINNT\system32\cmd.exe
DD_RSERVERS	C:\IngresII\ingres\rep
DMH_BIN	d:/usr/local/bin
II_SYSTEM	C:\IngresII

Variable Name: CLASSPATH

Variable Value: c:\Progra~1\Java\jdk1.6.0_07\lib\tools.jar

Packages and Disk Paths

When using the package:
  package ivproto   // sample package name

If your SET CLASSPATH=.;... has a dot, then there must be a subdirectory under where you are in the path with the same name as the package,

e.g.:

If you are at:
...\java\IVproto1\src
then to use the package "ivproto" you must have subdirectory:
...\java\IVproto1\src\ivproto
and the subdirectory must contain:
...\java\IVproto1\src\ivproto\IVproto1.class
...\java\IVproto1\src\ivproto\IVproto1$1.class
...\java\IVproto1\src\ivproto\IVproto1$2.class

With the above setup, you can run the package (which contains the main) via the command (from ...\java\IVproto1\src):
> java ivproto.IVproto1   *package_name.class_name*

<u>Example</u>

```
package ParsingPkg;
...
public class Parse_a_buffer {

    public static void parse_mospro (byte byteBuf[]) {
        System.out.println("inkless_prep called\n");
    }

    public static void main(String[] args) {
    }

}
```

```
/* if not in the CLASSPATH, assumes the directory "ParseitPkg" is just below where we are now */
import ParsingPkg.*;

...
ParsingPkg.Parse_a_buffer aParse_a_buffer = new ParsingPkg.Parse_a_buffer();
aParse_a_buffer.parse_mospro(byteBuf);
```

To run the T.main() method, use the java interpreter/run-time-compiler tool:

```
$ java T
```

where java takes the name of a Java class, T, not the name of the file it is in, T.java. The java interpreter looks for T.class and then begins execution at the main() method within.

Assume for now that java looks for T.class and any other class files in the current directory.

Lets use two Java files; one, U.java:
that refers to a method in the new file, Hello.java:
To compile these two files, use

```
$ javac U.java Hello.java
```

or, more generally:

```
$ javac *.java
$ ls | more
Hello.class
Hello.java
T.class
T.java
U.class
U.java
```

Now you can run U's main() method via:

```
$ java U
```

CLASSPATH Environment Variable
Jump into /tmp and now try to run U:

```
$ cd /tmp
$ java U
Exception in thread "main" java.lang.NoClassDefFoundError: U
```

The problem is that the Java interpreter does not know where to find U.class. You can specify where to look directly:

```
$ java -classpath ~/USF/CS601/code/tools T
Salut, Le Monde
$ java -classpath ~/USF/CS601/code/tools U
hello
```

In general, however, you will not want to repeat the path each time. For Unix:

```
$ echo $CLASSPATH   # what is it set to currently?
.:/home/parrt/lib/antlr-2.7.2.jar:/home/parrt/lib/jguru.util.jar:...
$ export CLASSPATH="$CLASSPATH:/home/parrt/tmp"  # append
$ echo $CLASSPATH   # what is it now?
.:/home/parrt/lib/antlr-2.7.2.jar:/home/parrt/lib/jguru.util.jar:...:~/USF/CS601/code/tools
```

Executing Java Within a Jar

```
$ cd some-random-directory
$ java -classpath /tmp/tools.jar T

$ java -jar <filename>.jar
```

GUIs

GridBagLayout (tk like)

```
JPanel   top_panel = new JPanel(new GridBagLayout());
```
- or -
```
JPanel   top_wafer_panel = new JPanel();
top_panel.setLayout(new GridBagLayout());


GridBagConstraints someConstraints = new GridBagConstraints();

// resize component HORIZONTALlY but not vertically
someConstraints.fill       = GridBagConstraints.HORIZONTAL;
someConstraints.anchor    = GridBagConstraints.NORTH;
someConstraints.insets = new Insets(10,0,0,0);  //Insets(top,left,bottom,rt)

someConstraints.weighty    = 0d;   // no extra space in the y direction
someConstraints.ipadx      = 0;
someConstraints.ipady      = 0;

someConstraints.gridx      = 0;   // column 1
someConstraints.gridy      = 0;   // row    1
someConstraints.gridwidth  = 1;   // 1 columns wide = columnspan 1 (in tk)
someConstraints.gridheight = 1;   // 1 row high     = rowspan 2 (in tk)

top_panel.add (top_panel, someConstraints);    // add to toplevel
```

```
-------------------------------------------------
|FIRST_LINE_START   PAGE_START     FIRST_LINE_END|
|                                                |
|                                                |
|LINE_START           CENTER             LINE_END|
|                                                |
|                                                |
|LAST_LINE_START     PAGE_END       LAST_LINE_END|
-------------------------------------------------
```

```
    Version note:  The PAGE_* and *LINE_* constants were introduced in 1.4. Previous
releases require values named after points of the compass. For example, NORTHEAST
indicates the top-right part of the display area. We recommend that you use the new
constants, instead, since they enable easier localization.
```

<u>Anchor doesnt work:</u>

You specified the anchor, but it will always draw it to the center since you stated no weight, so the weight is always in all directions so its constant therefore it would be defaulted to center.
To fix your situation, apply an additional weight constraint for your button:
----
c.weighty = 1;
c.weightx = 1;

These variables were by default value 0 meaning there is no weight, so think of it as a floating box, they are used in calculating where space is allocated in columns and rows when frame is resized.

So if you state 1 (which is the maximum cause it is between 0 to 1 as a double) you will tell the frame that it should be allocated in that constraint northwest.

Border Layout

```
JPanel    buttons_panel = new JPanel();

BorderLayout theBorderLayout = new BorderLayout();
// needs: import java.swing.border.*
buttons_panel.setBorder(BorderFactory.createEmptyBorder(5,5,5,5));
buttons_panel.setBorder(BorderFactory.createEtchedBorder(EtchedBorder.RAISED));
```

Radio Buttons

```
/* Wafer Size radiobuttons */
JPanel   wsize_panel = new JPanel();
wsize_panel.setBorder(BorderFactory.createEmptyBorder(2,2,2,2));

wsize_panel.setBorder(BorderFactory.createEtchedBorder(EtchedBorder.RAISED));
// needs: import java.swing.border.*
wsize_panel.setLayout(flowLayout_LEFT_10_10);   // set layout type for
scaling_panel

JRadioButton winfo_6inch_rb = new JRadioButton("6 inch", true);
JRadioButton winfo_8inch_rb = new JRadioButton("8 inch", false);

ButtonGroup winfo_wsize_button_group = new ButtonGroup();
winfo_wsize_button_group.add(winfo_6inch_rb);
winfo_wsize_button_group.add(winfo_8inch_rb);

wsize_panel.add(winfo_6inch_rb);
wsize_panel.add(winfo_8inch_rb);
```

Listbox

```
String dev_string[] =
{"123","456","789","ABC","DEF","HIJ","123","456","789","ABC","DEF","HIJ"};
```
or
```
Vector<String> dev_string = new Vector<String>();
dev_string.add("123");
...

JPanel   dev_panel = new JPanel();
JList dev_list = new JList(dev_string);
      dev_list.setVisibleRowCount(6);        // listbox height
      dev_list.setFixedCellWidth(200);       // width in pixels

/* By default, a list does not automatically display a scrollbar */
/* when there are more items than can be displayed. The list     */
/* must be wrapped in a scroll pane:                              */

JScrollPane dev_list_scrollpane = new JScrollPane(dev_list);

dev_panel.add(dev_list_scrollpane);
```

String aDev = (String)dev_list.getSelectedValue();   // returns selection
or
```
int selected[] = mDevList.getSelectedIndices();
if ( selected.length > 0 ) {
   String theDevice = (String)mDevList.getModel().getElementAt(selected[0]);
   status_details_update ("Device selected: " + theDevice);
...
```

Setting JList selection mode:

```
Vector<String> mDevices = new Vector<String>();
DefaultListModel dev_model = new DefaultListModel();
dev_model.ensureCapacity(mDevices.size());
for (String aDev : mDevices ) {
    dev_model.addElement(aDev);
}

JList mDevList = new JList(dev_model);
      mDevList.setSelectionMode(ListSelectionModel.SINGLE_SELECTION);
// other modes: SINGLE_INTERVAL_SELECTION, MULTIPLE_INTERVAL_SELECTION
```

JOptionPane Dialogs

While the JOptionPane class may appear complex because of the large number of methods, almost all uses of this class are one-line calls to one of the static showXxxDialog methods shown below:

Method Name	Description
JOptionPane.showConfirmDialog	Asks a confirming question, like yes/no/cancel.
JOptionPane.showInputDialog	Prompt for some input.
JOptionPane.showMessageDialog	Tell the user about something that has happened.
JOptionPane.showOptionDialog	The Grand Unification of the above three.

Each of these methods also comes in a showInternalXXX flavor, which uses an internal frame to hold the dialog box (see JInternalFrame). Multiple convenience methods have also been defined -- overloaded versions of the basic methods that use different parameter lists.

All dialogs are modal. Each showXxxDialog method blocks the current thread until the user's interaction is complete.

File Chooser Dialog

```
// Show open dialog; this method does not return until the dialog is closed
    fc.showOpenDialog(frame);
    File selFile = fc.getSelectedFile();
    // Show save dialog; this method does not return until the dialog is closed
    fc.showSaveDialog(frame);
    selFile = fc.getSelectedFile();
- or -
JFileChooser fc = new JFileChooser(new File(mMOSPRO_DIR));
fc.showOpenDialog(this);
File theFile = fc.getSelectedFile();   // returns full pathname in theFile
```

Layout Managers:

Scenario: You need to display a component in as much space as it can get.
If it is the only component in its container, use GridLayout or BorderLayout. Otherwise, BorderLayout or GridBagLayout might be a good match.

If you use BorderLayout, you will need to put the space-hungry component in the center. With GridBagLayout, you will need to set the constraints for the component so that fill=GridBagConstraints.BOTH. Another possibility is to use BoxLayout, making the space-hungry component specify very large preferred and maximum sizes.

Scenario: You need to display a few components in a compact row at their natural size.
Consider using a JPanel to group the components and using either the JPanel's default FlowLayout manager or the BoxLayout manager. SpringLayout is also good for this.

Scenario: You need to display a few components of the same size in rows and columns.
GridLayout is perfect for this.

Scenario: You need to display a few components in a row or column, possibly with varying amounts of space between them, custom alignment, or custom component sizes.
BoxLayout is perfect for this.

Scenario: You need to display aligned columns, as in a form-like interface where a column of labels is used to describe text fields in an adjacent column.
SpringLayout is a natural choice for this. The SpringUtilities class used by several Tutorial examples defines a makeCompactGrid method that lets you easily align multiple rows and columns of components.

Scenario: You have a complex layout with many components.
Consider either using a very flexible layout manager such as GridBagLayout or SpringLayout, or grouping the components into one or more JPanels to simplify layout. If you take the latter approach, each JPanel might use a different layout manager.

Thread Safety?:

```
/**
     * Create the GUI and show it.  For thread safety,
     * this method should be invoked from the
     * event-dispatching thread.
     */
    private static void createAndShowGUI() {
        //Create and set up the window.
        JFrame frame = new JFrame("AbsoluteLayoutDemo");
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
```

To write an Action Listener, follow the steps given below:

1. Declare an event handler class and specify that the class either implements an ActionListener interface or extends a class that implements an ActionListener interface. For example:

```
public class MyClass implements ActionListener {
```

2. Register an instance of the event handler class as a listener on one or more components. For example:

```
someComponent.addActionListener(instanceOfMyClass);
```

3. Include code that implements the methods in listener interface. For example:

```
public void actionPerformed(ActionEvent e) {
    ...//code that reacts to the action...
}
```