

## PROGRAMMING IN JAVA (using the NetBeans IDE)

There are free systems available from several organizations. First, there is an SDK (software development kit) , now called a JDK (Java development kit), and it has the stuff of which JAVA is made. Its pieces can be used at the command level. Second, there are several IDEs (integrated development environments), one is NetBeans, however, others exist. Some are graphical, some are not. NetBeans is graphical, and is used here. JPadPro is a non graphical IDE, not used here.

NetBeans and the appropriate JDK can be downloaded together from the following download sites. (urls may change over time).

<http://www.netbeans.info/downloads/index.php>  
<http://java.sun.com/javase/downloads/index.jsp> [what I used]

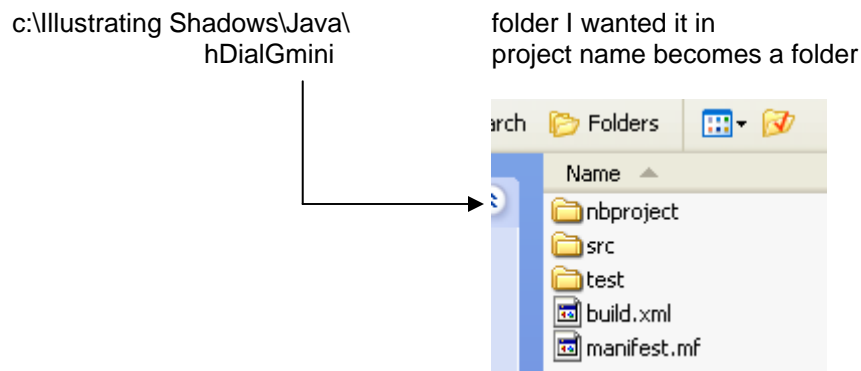
There are several books that are helpful, one is "JAVA In Easy Steps" by Mike McGrath, and is based on a text based non graphical development environment, called the "command line", such as JPadPro. Another includes an SDK on a CD and is "Programming With JAVA in 24 Hours" by Rogers Cadenhead that gets into graphical IDEs such as NetBeans. A third is "Core JAVA 2 – Volume 1 – Fundamentals" by Cay Horstmann and Gary Cornell which has useful internal theory.

This section will continue the "PBE" philosophy, namely "Programming By Example".

## DEVELOPING A PC BASED APPLICATION (versus a web applet)

Begin by starting the JAVA IDE (NetBeans is the IDE depicted in these pages).

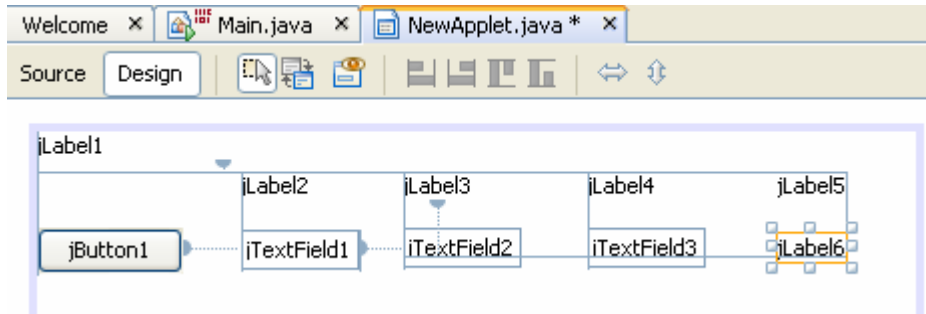
Then tell the IDE what you intend to construct. Select **FILE, NEW PROJECT, GENERAL, JAVA APPLICATION**, and enter a name. Select a folder first so that you can find the thing later. The project name you enter will become a new folder inside the folder that you selected. Plan your folder and project names in a clear manner, In this the project was called:



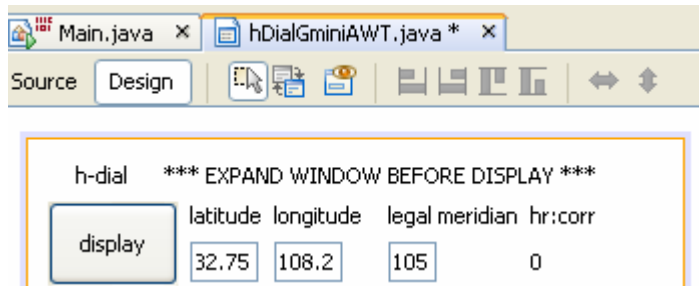
And then for human interaction, select **FILE, NEW FILE, JAVA GUI FORMS, AWT FORMS, APPLETT FORM**, and following the "NEXT" links, ensure it goes to your preferred folder.

**IMPORTANT NOTE:** this form by default is very limited. So to make it unlimited, right click in the new form that popped up, select SET LAYOUT, and then select FREE DESIGN. At this point a practical user interface can be designed. The "FREE DESIGN" is critical to being able to build a meaningful panel for data entry. It is critical for data display if you intend to use LABEL or TEXTFIELD elements. Text can also be displayed using the graphical operations where you name an x,y coordinate.

The user interaction area, or "form", was started, as shown below.



then the text areas for the objects on the form were entered so they would make sense. This program would display hour line angles using graphics rather than the method shown in the Visual Basic example which used labels for the data. And consequently, there would be no table of hour line angles since the dial would be graphically portrayed.



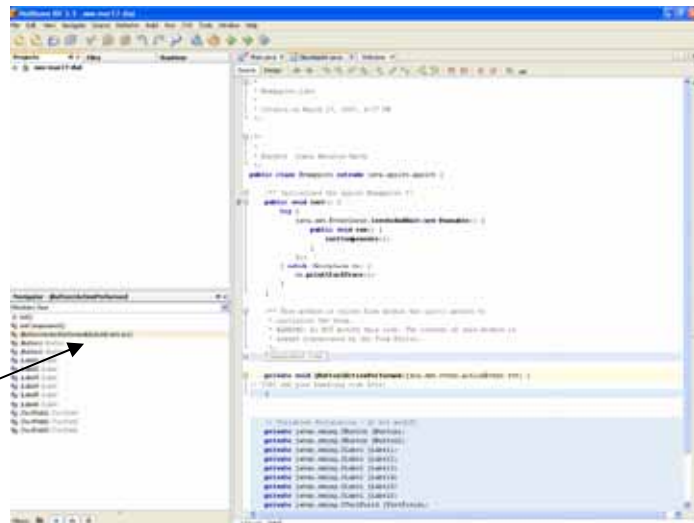
Before proceeding, a FILE, SAVE ALL was done and the system exited. Then NetBeans was reloaded and work resumed, this was to ensure the project was saved properly, and still available.

The screen to the right shows what comes back up.

To add the code for the "display" button, called

jButton1

the list on the lower left half of the IDE shows the objects, and clicking on the jButton1ActionPerformed brings up skeleton code.



Alternatively, in the "form" display the button can be right clicked and EVENTS, ACTION, ACTION PERFORMED can be used to achieve the same result. The end result is the a stub for code is generated in the SOURCE tab for the form.

The skeleton code looks something like the following:-

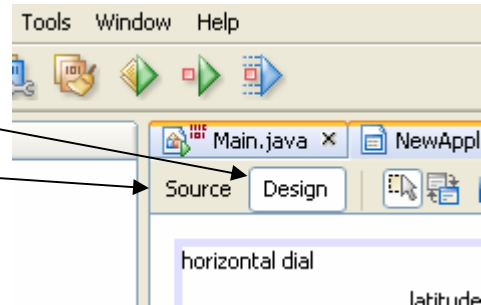
```

-      ~/
]   Generated Code

]   private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
// TODO add your handling code here:
-   }

```

and into here goes the code. By the way, to switch between code and the form, the buttons "Design" and "Source" are used.



The code we wish we could add would be something like the following:

```

jButton1 = calculate hour line angles and draw them
jLabel6.Text = 4 * (jTextField2.Text - jTextField4.Text) / 60

```

which is simple enough, but it is not JAVA syntax, nor does it meet the rules of modifying objects in JAVA. First, some "import" statements are needed so the JAVA system knows the format for some of the functions, and their parameter types.

```

import java.awt.image.renderable.RenderableImage;
import java.io.*;
import java.awt.*;
import java.awt.Graphics;
import java.awt.Graphics2D;
import java.awt.font.*;
import java.awt.geom.*;
import java.awt.image.*;
import java.text.AttributedString;
import java.util.Map;
import javax.swing.*;
import javax.imageio.*;

```

Code would be added to extract the TEXT STRINGS of longitude and legal meridian, convert them to FLOAT, perform the math, and convert them back to STRING. This sounds simple however the JAVA IDE "help" system is not overburdened with practical examples. A search on the web did locate the "float to string" function using HELP, SUPPORT AND DOCS ONLINE, then selecting "The JAVA Tutorial" and following links until the helpful url was found:-

<http://java.sun.com/docs/books/tutorial/java/data/strings.html>

This showed the "String.format" function. Finding useful functions complicates learning JAVA since what are normally language functions may now be an object's methods.

```

// derive the dial location and hour correction
s1 = jTextField2.getText();
s2 = jTextField3.getText();
t1 = Float.parseFloat(s1);
t2 = Float.parseFloat(s2);
t3 = 4*(t1-t2)/60;
// show hour correction
s3 = String.format("%F", t3);
jLabel6.setText ( s3 );

```

## ADDING HOUR LINE GRAPHICAL AND ANGULAR DATA TO THE DISPLAY

At this point the math functions are needed, and the online help was used using HELP, SUPPORT AND DOCS ONLINE, then selecting "The JAVA Tutorial" and following links until the helpful url was found:-

<http://java.sun.com/javase/6/docs/api/java/lang/Math.html>

The code for the DISPLAY BUTTON CLICK is as follows.

```
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
    /* ***** this is the displays graphics button handler ***** */
    Graphics swsxy = getGraphics(); // THIS LINE IS CRITICAL
    // x = 050 is a good start for the graphical depiction, y = 400 also
    int leftx, rightx, ctrx, topy, boty;
    leftx = 050;
    rightx = 650;
    ctrx = (leftx+rightx)/2;
    topy = 150;
    boty = 450;
    swsxy.drawString("(*)", ctrx-5, boty+5);
    swsxy.drawRect(leftx,topy,rightx-leftx,boty-topy);
    swsxy.drawRect(leftx-30,topy-20,60+rightx-leftx,40+boty-topy);

    // establish some variables for the major loop
    float t1 , t2, t3, t4, t5, t6;
    String s1, s2, s3, s4, s5;
    double slat, thra, t4d, t7;

    // derive the dial location and hour correction
    s1 = jTextField2.getText();
    s2 = jTextField3.getText();
    t1 = Float.parseFloat(s1);
    t2 = Float.parseFloat(s2);
    t3 = 4*(t1-t2)/60 ;

    // show hour correction
    s3 = String.format("%f", t3);
    jLabel6.setText ( s3 );

    // get the dial latitude into t4
    s4 = jTextField1.getText();
    t4 = Float.parseFloat(s4);
    t4d = t4;
    slat = Math.sin( t4d*(2.0*3.1416)/360.0 ); // sin of the latitude

    for ( int hr= -6; hr<7; hr++) // loop the hours
    {
        thra = Math.tan( ((2*3.1416)/360)*15*(hr-t3) ); // tan of the suns hour angle
        t6 = (float)slat * (float)thra; // t6 is sin(lat)*tan(hra)
        t7 = Math.atan(t6); // t7 is atan radians
        // calculate the hour line angle
        t5 = (float)(360/(2*3.1416)) * (float)t7; // t5 is hour line angle
        s5 = String.format("%f", t5); // s5 is hr angle in text

        // *** t5 is float and is the hour line's angle in degrees ***
        // *** t7 is the hour line angle in radians also ***
        float xxx, yyy;
        float t7f = (float) t7;

        // handle cases where the hour line angle is greater than 45 degrees
        if (Math.abs(t5)>45)
        { // x is the full amount, derive y
            // this is hours close to the 6's
            xxx = (rightx - leftx)/2;
            yyy = xxx / ((float) Math.tan(t7f));
            if ( hr < 0 )
            {
                xxx = -xxx; // if morning then xxx is negative
                yyy = -yyy;
            }
        }
        else
    }
}
```

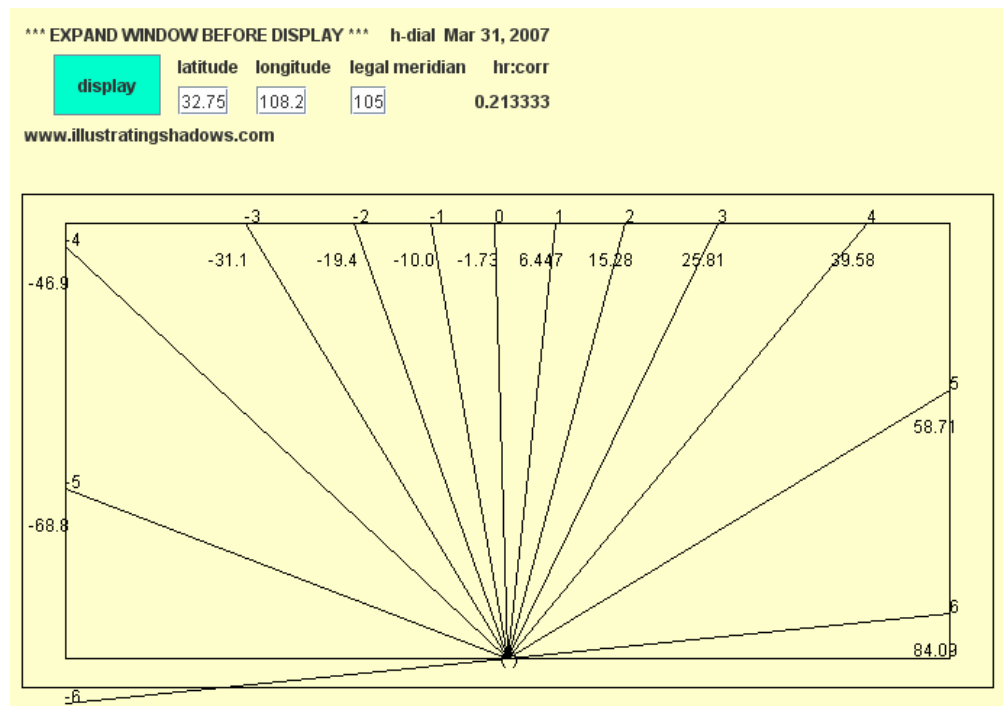
```
{ // y is the full amount, derive x
  // this is hours close to noon
  yyy = (float)(boty - topy) ;
  xxx = yyy * ((float) Math.tan(t7f));
}

int xxxxx, yyyyy; // end of the line coordinates
xxxxx = ((int) ctrx + (int)xxx);
yyyyy = ((int) boty - (int)yyy);

swsxy.drawLine(ctrx,boty, xxxxx, yyyyy);
String h = ""+hr;
swsxy.drawString(h, xxxxx, yyyyy);
swsxy.drawString(s5.substring(0,5), xxxxx-25, yyyyy+30);
}
}
```

In the java applet window, either SOURCE or DESIGN, and not in the MAIN JAVA window, shift+F6 causes the program to be compiled and executed. The results are shown below.

An alternative to SHIFT + F6 is to RUN, PROJECT (does the build but does not run the program, then RUN, RUN FILE, RUN FILE (which is where shift + F6 comes from).

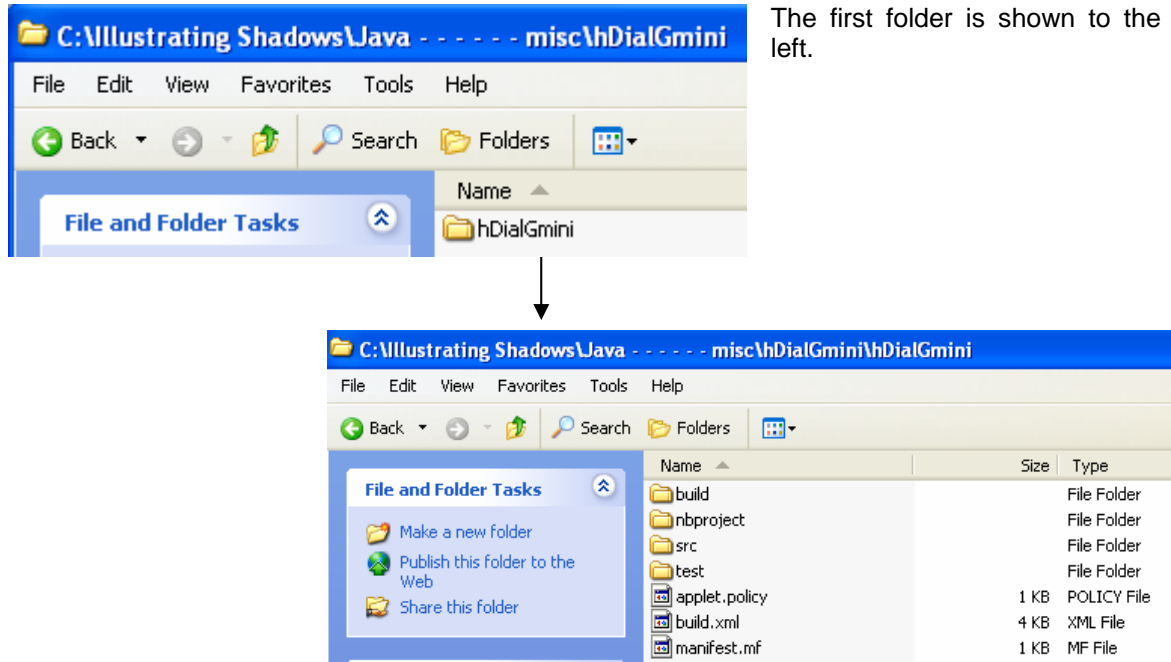


Compiled is probably not the correct word. The code is converted into an intermediate form, and that intermediate form is what the various JAVA systems on all the various platforms (operating systems and chip types) actually run.

When several projects are being developed, and NetBeans is exited, and later restarted, the next time the IDE is brought up, not everything appears. This gives the impression that all was lost.

The top left panel in NetBeans allows sub folders to be clicked and their files to be opened. In this manner the (a) main form, (b) the main code, and (c) the event driven code for the buttons will once again appear.

#### WHAT THE FOLDERS AND FILES ARE NAMED



Not shown in the program that follows is the repainting of a java window when it is covered up by some other window. A true benefit of a fully implemented object oriented system is that methods belonging to classes can be interlinked or inter-related, and invoked "when things happen". One such method may be invoked to redraw a java window when that window has been affected by some other window.

This book's web site: [www.illustratingshadows.com](http://www.illustratingshadows.com) has additional sample code, as does the CD that accompanies this book.