

## Apprentice Programming 2 Module

Matt DesVoigne

July 3, 2008

VI Tutorial: <http://www.eng.hawaii.edu/Tutor/vi.pdf>

SubVersion tutorial:

<http://www.shodor.org/programmers/guides/svnuser.pdf>

Old SubVersion challenge/assignment (if you are interested):

<http://www.shodor.org/programmers/guides/svnChallenge.pdf>

### I. DAY 1 - Account Setup and Daily procedures using svn

- Learning Java is BIG. A lot to Java. Not a 15 minute exercise.

LAB NOTEBOOK USAGE

login to [login.shodor.org](http://login.shodor.org)

- Terminal editor and compiler used to start as opposed to an IDE which may be introduced as experience develops.

cd public\_html (or create it if it does not exist)

mkdir programming2

cd programming2

svn checkout <https://svn.shodor.org/repos/training/interactivate>

cd interactivate

chmod 777 build.sh (Explain – changing permission to execute a script.)

- Briefly display contents of build.sh.

Cross platform compatibility of Java. Write once compile anywhere. Java VM runs as a separate machine in your computer which is independent of the OS.

### II. Daily Routine

./build.sh graphit

<https://www.shodor.org/~mdesvoig/programming2/interactivate/graphit>

(“programming2” will be omitted when you use the production repository.)

pico (or vi or whatever UNIX editor) graphit/GraphIt.java

Make modifications

./build.sh graphit

Check in browser. Repeat until satisfied with work.

svn commit -m “Changed XYZ in graphit” graphit (PLEASE INCLUDE DESCRIPTIVE COMMENTS.)

Assign RT Ticket back to Matt DesVoigne (mdesvoig) for review.

### III. Sharing code

Some families of applets with similar functionality share code so that one change may propagate to all applets which share the same code. Simple Plot shares code with GraphIt.

./build.sh graphit simpleplot

<https://www.shodor.org/~mdesvoig/programming2/interactivate/simpleplot>

("programming2" will be omitted when you use the production repository.)

pico (or vi or whatever UNIX editor) graphit/GraphIt.java

Make modifications

./build.sh graphit simpleplot

Check in browser. Repeat until satisfied with work.

svn commit -m "Changed XYZ in graphit" graphit (PLEASE INCLUDE DESCRIPTIVE COMMENTS.)

Assign RT Ticket back to Matt DesVoigne (mdesvoig) for review.

#### IV. Adding an applet

- cd ~/public\_html/programming2/interactivate
- mkdir newappletmdesvoig
- cp graphit/index.html newappletmdesvoig
- cp graphit/parameters.txt newappletmdesvoig (Link to code sharing section. Explain why parameters are moved to a text file from html – no html parameters in an application. **Change parameters in parameters.txt, not index.html. Size still read by index pages (and database)**)
- mkdir newappletmdesvoig/META-INF (BRIEFLY explain what this means.)
- cp graphit/META-INF/MANIFEST.MF newappletmdesvoig/META-INF
- Modify newappletmdesvoig/index.html, newappletmdesvoig/parameters.txt, and newappletmdesvoig/META-INF/MANIFEST.MF to port to newappletmdesvoig's specifications.
- Add .java files to newappletmdesvoig directory. (For DEMO and FIRST EXERCISE, copy newAppletFiles/TextCanvasTest.java to newappletmdesvoig and change the package description. Good time to discuss packaging.)
- cd ~/public\_html/programming2/interactivate
- ./build.sh newappletmdesvoig
- <https://www.shodor.org/~mdesvoig/programming2/interactivate/newappletmdesvoig> (Run as applet)
- <https://www.shodor.org/~mdesvoig/programming2/interactivate/newappletmdesvoig/newappletmdesvoig.jar> (Run as application)
- Test and debug.
- cd ~/public\_html/programming2/interactivate
- svn add newappletmdesvoig
- svn commit -m "Adding newappletmdesvoig to Interactivate" newappletmdesvoig

#### V. EXERCISES

Correct spelling in Graph Title – graphit or simpleplot. (NO COMMIT)

Correct aspect of graph paper so graph paper is square - graphit. (NO COMMIT)

Create TextCanvasTest applet in the repository. (USES SVN) (REPEAT STEP IMPLEMENT SECTION 4 AS IT APPLIES TO YOU.)

## VI. DAY 2 – Org and Event handling

Org library documentation is located at

<http://www.shodor.org/programmers/libraries/doc/>

All interactivate applets import at least one org class.

Only import the classes you need (import org.shodor.gui11.ShodorGraphPaper). Do not import the whole package (import org.shodor.gui11.\*). This makes it easier to decipher which classes are actually used.

Description of packages:

- data – Encapsulates various data types into a single object. Not a standalone package. Data package has no functionality itself but must be used by other packages which implement the functionality (building blocks per se).
- events – Shodor-defined events (eg. TimerEvent). More on events later.
- gui11 – Graphical User Interface components, both controls (sliders, numerical inputs, etc) and views (ShodorGraphPaper, TextCanvas, etc). (gui14 is Swing components for a couple Swing applets. Shodor does not use Swing in daily development.)
- io – Input/Output classes – Helper classes for Save State functionality for activities run as applications (if Save State functionality is implemented).
- javaws – Java Web Start- for applets requiring Java 1.5+. Not typically used in Interactivate.
- layout – Shodor defined layout managers. Layout covered later in class.
- listeners – Shodor defined event listeners. Event listeners covered later in day.
- math11 – Shodor math utilities including a “Formula” object for displaying functions graphically or as a table of (x, y) values.
- util11 – Things that don’t belong above. (DataParser and StrLib are most widely used.)

## VII. Event Handling

Show eventtest.TimerTest

- TimerTest registers itself as a TimerListener with the Timer object and tells the Timer how long to wait between notifications (100 milliseconds).
- Every 100 milliseconds (every tenth of a second), the Timer notifies all registered TimerListeners that a TimerEvent has occurred by calling the timerEventTick() method on all registered TimerListeners. (In this case the only registered TimerListener with Timer is TimerTest.) TimerTest which implements the TimerListener interface handles the event by implementing the timerEventTick method for the TimerListener interface. (Show in code where this happens.
- Major AWT Listeners:
  - ActionListeners registered with buttons and menus

- ItemListeners registered with drop-down boxes, checkboxes, and radio buttons.
- WindowListeners are registered with windows and frames.
- Major Shodor Listeners
  - SliderListeners registered with SliderBars.
  - GraphListeners registered with ShodorGraphPaper to listen to changes in window limits.
  - RandomSeedListeners registered with RandomNYUSeedFrame to listen for user-defined seeds for Random Number Generator?
  - Save State Listeners and Adaptors listen for Save State Events (Advanced).

### VIII. EXERCISES:

- Turn eventTest into a digital clock, showing the hours, minutes, and seconds of the current time. The clock should update every second. *Hint:* Use the method getTime() in the TimerTest class. *Caution:* There are 1000 milliseconds in one second. What do you need to change the Timer constructor to?
- The applet currently adds object from org called org.shodor.gui11.ShodorButton, which works like a Java awt button only we at Shodor can control how it looks. When clicked, currently nothing happens. When the button is clicked, the clock should pause if it is running or resume if it is paused. Fix it. To stop a timer, call the stop() method on the Timer class. *Hint 1:* Register the TimerTest object as an ActionListener with the bStartStopButton. *Hint 2:* Add to the actionPerformed method. *Hint 3:* Use the isRunning boolean variable in TimerTest to determine if the timer is running or not. *Hint 4:* Use the method ShodorButton.setLabel(String) to toggle the button label between “Play” and “Pause.”
- Examine the getTime() method in the TimerTest class. Modify the method so the applet instead displays the date that Matt was born. Modify it again to display the date you were born.

### IX. DAY 3 – LAYOUT MANAGERS

- Layout managers control the positioning of components in your applet or on a frame. Layout managers specific to Shodor are in the org.shodor.layout package.
- In general, I use StackLayout to stack components vertically, HStackLayout to align components horizontally, and FlexGridLayout to align components in a grid. Occasionally I will use FlowLayout (awt layout manager) for left-anchored components butted end to end in a single row. I very rarely use any other layout manager.
- Show and Explain layouttest. Uses Java Console. Explain opening console on PC, Mac, and run as application.
- Implements ItemListener. Tie back to yesterday.

- `gui11.BorderPanels` are used to show the outlines of the panels. Usually we don't want the border and just use `java.awt.Panel`. The most common constructor is `StackLayout` anchored north, instantiated like this:
- `Panel myPanel = new Panel(new StackLayout(StackLayout.NORTH));` which is equivalent to
- `Panel myPanel = new Panel();`
- `myPanel.setLayout(new StackLayout(StackLayout.NORTH));`
- Naming – Start Panel names with a “p”, Checkboxes with “cb”, Buttons with “b” and `SliderBars` with “sb”. This helps easily recognize object types.

## X. EXERCISES

- Modify `LayoutTest` such that the dropdown in the Grid panel at the top is in the top-right cell. *Caution:* Java starts counting from 0, not 1.
- Add a new panel to `LayoutTest` that has three components in a top row and one component in the bottom row. Use at least one control (`SliderBar`, `Button`, `Choice`, `Checkbox`, etc.) When the control is activated (clicked on), print information about the control to the console.

## XI. DAY 4 – Procedures

- Lab Notebook
  - Flow Chart / Logic / Sketches
- Incremental Programming
- VVA
- Descriptive Variable Naming
- Accessor variables – Private variables for Eclipse monitoring what is used.
- In-line comments as well as a general overview.
- OOP – Encapsulate data and behavior into objects mindset.
- Avoid “catch-all” event handling (ie. An `actionPerformed` method that does not check the source of the event. What if another control is added that notifies `ActionListeners` via the `actionPerformed` method?)
- Catch the exception that was thrown (not just catch `Exception`)
- Handle the exception.
- Lean on static constants that can be changed once to affect various components instead of hard-coded numbers used multiple times and if changed, multiple changes would be required.)
  - `public static final int CANVAS_WIDTH = 300;`
- Similarly, ask yourself “How easy would it be to modify this if need be?”

## XII. Projects

For each project, add a new applet in the training repository. You may copy existing files to your directory and then modify them to add functionality.)

- Add a default function to `graphit`.
- Add a default data set to `Simple Plot`.
- Add user-defined title/labels/description to `stemleaf`. – RT 6573
- Create an applet with two rows. The top row contains an `org.shodor.gui11.SliderBar` and a button. The bottom row contains an

org.shodor.gui11.TextCanvas. When the slider is moved, it's value is displayed in the TextCanvas. *Hint:* Applet should implement SliderListener and the sliderMoved method should update the TextCanvas. When the button is pressed, the TextCanvas should display the number one greater than the slider value.

- Add ShodorButtons to fracfinders. – RT 6415 – *Caution:* DO NOT change copyright or Keep Score button to ShodorButton (or else it will pop up behind the window). Also, do not alter Scoreboard buttons.
- Histogram min/max/step – RT 7838
- Others???

### XIII. DAY 5 – Project Work