

## Interactive Development Environments

Philip Johnson  
Collaborative Software Development Laboratory  
Information and Computer Sciences  
University of Hawaii  
Honolulu HI 96822

(1)

## What is an IDE?

An interactive development environment is a software development tool with the following properties:

- Language-aware editing
- Project definition facilities
- Integrated compilation
- Integrated stepwise execution (debugger)

Other common features:

- Wizards
- Diagramming
- Integration with configuration management

(2)

## IDEs and Non-IDEs

Is an IDE:

- Emacs
- Jbuilder
- Code Warrior
- Visual Studio
- NetBeans
- etc

Is not an IDE:

- Notepad
- Textpad
- Edit
- Vi (debatable)
- etc

(3)

## Why use an IDE?

IDEs are:

- complicated to learn to use
- take time to master
- slower to bring up than non-IDEs

So why use them?

You use an IDE because developers who are skilled with an IDE can successfully:

- understand large software systems faster
- correct large software systems faster
- improve large software systems faster

(4)

## One other reason to use an IDE

As told to me by a senior software developer in charge of hiring at a local/national high tech firm:

- My first question to a potential new hire at an interview is, "What editor do you like best for writing software?" If their answer is "Notepad" or "Wordpad", then the interview is effectively over right then.

(5)

## Some analogies

Tricycle	vs.	Mountain bike
Fingers	vs.	Calculator
Inflatable raft	vs.	Boat
Notepad	vs.	Emacs

The left hand side tools are:

- easier to learn
- easier to use
- less complicated

They just don't "scale" well.

(6)

## Specific example: compilation

### Notepad:

- Edit text in notepad, switch to command window, type "javac <file>", get listing of compilation errors, switch back to notepad window, pull down file menu, select open, find file, start scrolling through buffer looking for line where error occurred, find error, edit line, save file, switch back to command window, type "javac <file>", get new listing of errors.

### Jbuilder:

- Edit text in jbuilder, hit "make" button, get listing of compilation errors, click on error, edit line, hit "make" button, get new listing of errors.

(7)

## If things can be accomplished fast, you're likely to do them:

Traverse a package hierarchy, searching for all instances of a specific string.

Single-step through a function.

Generate a UML diagram.

Get the completions for a method name prefix.

Reformat code.

Check matching braces.

(8)

## JOSSE Standard

You must use JBuilder Personal Edition, Version 5 or 6 in this course.

### Reasons:

- It's a reasonably good IDE.
- It's free for non-commercial use.
- It runs on Mac/Unix/Windows.
- Even if it's not your favorite IDE (it's not mine), it is better for everyone to use the same one so that collaboration and knowledge spreads faster.

(9)

## Basic IDE tasks

Configuration to support coding standards

- two space indent; no tabs; align brace

Defining a new project

Editing source code

Compiling a project

Running a project

Debugging a project

(10)

## JBuilder Demo

(11)

## JOSSE Directory Standards (Part 1)

accountinfo1/

- README.html

- doc/

- src/

- lib/

- jbuilder/

- accountinfo1.jpj

- build/

- classes

- bak

(12)

## JOSSE JBuilder project structure standards

Jbuilder project should have an internal folder structure that "mirrors" the package folder structure.

Exceptions:

- Do not have to mirror subdirectories that do not contain editable components (i.e. lib/Ant)
- Do not have to mirror "top-level" package directories in src folder:
  - src/edu/hawaii/johnson/accountinfo1/ can become just:
    - accountinfo1/

(13)

## More advanced issues

JBuilder has a significant number of free add-on tools contributed by the development community:

•<http://www.borland.com/jbuilder/resources/jbopentools.html>

Two that I use are:

- MouseTracker (for wheelie mice)
- BraceCheck (brace highlighting)

(14)

## IDEs vs. Build tools

IDEs provide support for:

- code writing
- incremental compilation and testing
- IDEs are developer-specific

Build tools provide support for IDE-independent, cross-platform, standardized:

- compilation
- testing
- packaging

Build tools are the bridge that allows developers and users to work together, even though they are:

- on different platforms
- prefer different IDEs.

(15)

## Installation Tips

System can be downloaded from

•<http://www.borland.com/jbuilder>

Install package is 55MB compressed (includes a Java JRE). Docs are an additional 50MB.

They email you a license key to unlock the product after you fill out a short form at their web site.

If windows do not repaint correctly, you must perform workaround for MS bug in Direct Draw:

- <http://community.borland.com/article/0,1410,20480,00.html>
- add vmparam -Dsun.java2d.noddraw
- useful fix for other Java GUIs as well!

(16)

## The JOSSE Software CD

I've burned two CDs containing most of the software you'll need for this class:

- JBuilder6 (Windows), JDK1.3 (Windows), WinCVS (Windows), Tomcat, Junit, Ant, Checkstyle, HttpUnit, etc.
- ~175MB compressed

One CD is for in-lab use (POST 307, CSDL).

- You must bring your computer to the lab.
- You can download the software right away.

One CD is for up to 24 hour lending.

- You must wait in line to get it, first come, first served.

(17)

Any questions?

(18)