# CS 535 Object-Oriented Programming & Design
# Fall Semester, 2001
# Doc 1 Introduction
# Contents



## References

VisualWorks Application Developer's Guide, doc/vwadg.pdf in the VisualWorks installation. Chapter 1 The VisualWorks Environment.

Software Productivity Research, Inc. (www.spr.com)

# Introduction
# Why Smalltalk

- Best language to learn object-oriented thinking
- Productivity Gains

**Software Productivity Research Study**

| Language | Lines of code/function point |
|---|---|
| Smalltalk | 21 |
| Ada 95 | 49 |
| Java | 53 |
| C++ | 53 |
| COBOL | 107 |
| C | 128 |

- Contains features not covered elsewhere in curriculum

# Learning Smalltalk

- Smalltalk language syntax

  While Smalltalk syntax is simple it is not like C/C++/Java

- Smalltalk Programming Environment

  Requires more effort to learn at first, but worth the effort

- Smalltalk Class Library

  Smalltalk has a large library of useful code

  Don't code without it

- Object-oriented thinking

  This is the hardest part

- Smalltalk culture

  Smalltalker's have standard ways to code & solve problems

  See Smalltalk Best Practice Patterns by Kent Beck

# Some History

1967 Simula-67

Language developed in Norway for simulations

Use classes and objects

Late 1960's Alan Kay – Father of Personal Computer

Kay Ph. D. thesis addresses the question:

How will we interact with notebook size computers?

Dynabook

1970-80 Xerox Parc

Alan Kay, Dan Ingalls, Ted Kaehler and others work on Smalltalk

Small – Originally for children

talk - Code is to communicate

# Smalltalk Influences

Object-Oriented Programming

GUI

- Macintosh
- Windows

Refactoring

Extreme Programming

# Versions of Smalltalk

VisualWorks

VisualAge for Smalltalk

Squeak

Dolphin
Smalltalk MT

Smalltalk X

Smallscript

PocketSmalltalk

# Smalltalk & Bytecode

Smalltalk is compiled to a bytecode for a virtual machine

Bytecode is same on all machines

VisualWorks has VM's for:

- Windows
- Macintosh
- Unix

VisualWork’s virtual machine (VM) uses a JIT to compile bytecodes

Just-in-time compilers (JIT)

- Compile bytecode to native machine code
- Cache the native machine code
- Run the native machine code
- Usually runs faster than interpreting bytecode

Smalltalk started using just-in-time compilers in early 1980s

# VisualWorks
# Parts of VisualWorks

Executable Virtual Machine (visual, visual.exe)

This is the VM that interprets Smalltalk bytecode

visual.sou

Source code for most of class library

visual.cha

Source code for changes & new classes
Does not exist until after you first use VisualWorks

visual.im

Bytecode of sources that are executed

At first the image will appear to be an IDE for Smalltalk

As you use Smalltalk the image file will change. When you make changes to code they are saved in the change file. At some point you will want to start over with a clean copy of the image. Before you use VisualWorks make a backup copy of the image file. When you want a fresh copy of the image, use copies of the backups.

parcels

Code bundles

## Starting VisualWorks

See the class wiki for instructions on downloading VW 5i4

## Before Starting VisualWorks

Before you start VisualWorks make a copy of visual.im

You will need it later

## Starting VisualWorks on Windows

Method 1

Drag and drop the image file on the Visual application or visual.exe

Method 2

Double click on the image file

The first time you do this you may get a dialog asking for the application to run the image. Select visual. You will have to find it first. It is in the bin directory.

## Starting VisualWorks on Macintosh

Method 1

Drag and drop the image file on the visual application

Method 2

Double click on the image file

## Starting VisualWorks on UNIX

Type:
visual imageFilename &

where you need to replace imageFilename with the actual name of the image file you wish to run

You path has to be set to include the program visual

# Some VisualWorks Environment

It is hard to explain on paper how to use GUIs. The best thing is to use the GUI and have an expert user nearby. The following may help you get started. For more information see Chapter 1 The VisualWorks Environment, VisualWorks Application Developer's Guide, doc/vwadg.pdf in the VisualWorks installation.

VisualWorks uses three logical buttons

- **Select** button
  Selects objects and text

- **Operate** button
  Opens a menu with context-sensitive commands

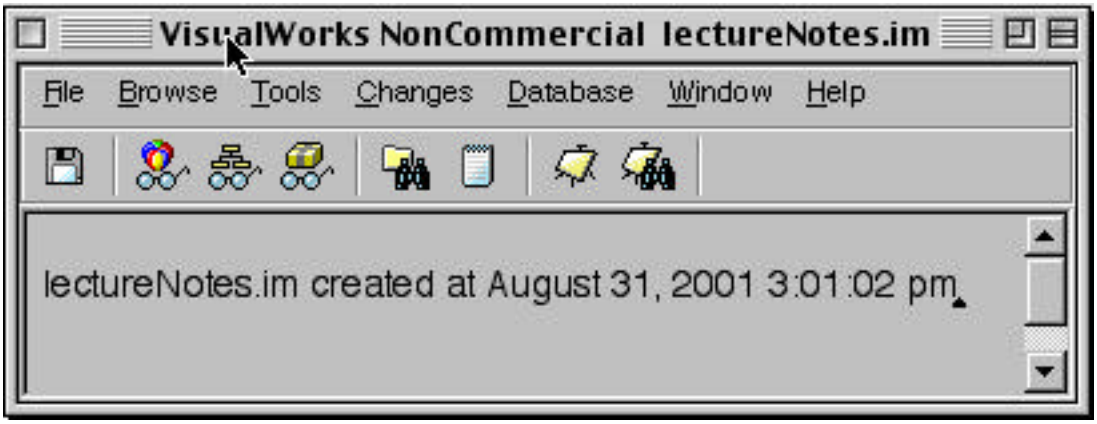
- **Window** button
  Opens a menu with window commands

## Mapping Logical to Physical Mouse Buttons

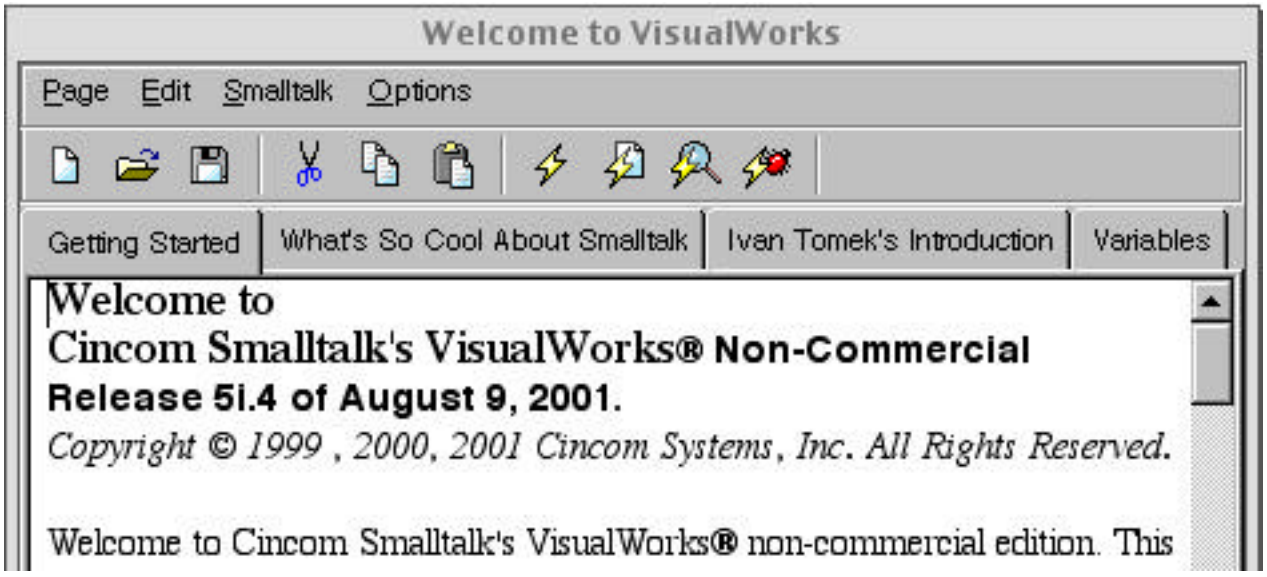
| | 3-Button | 2-Button | 1-Button |
|---|---|---|---|
| Select | Left button | Left Button | Button |
| Operate | Right button | Right button | Ctrl+Button |
| Window | Middle button | Ctrl+Left button | command+Button |

You should perform the action described in the next few pages. One learns what one does.

# Windows on Startup
# Launcher

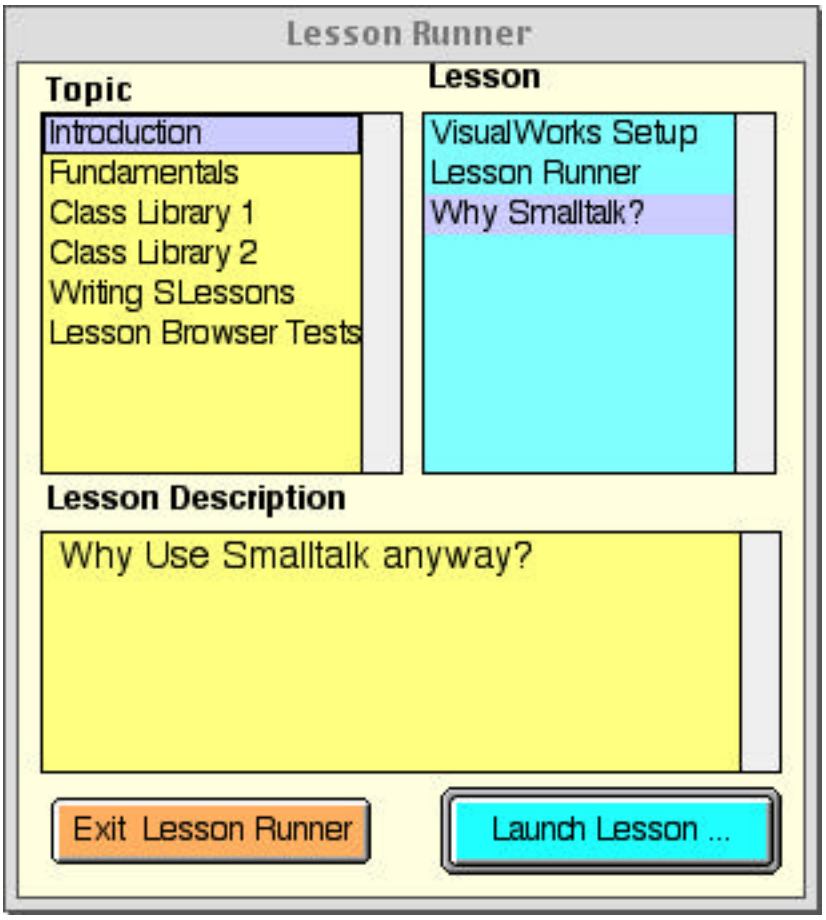



The Launcher is the main window in VisualWorks. It is your main starting point in the VisualWorks IDE. Do not close this window. We will come back to this window.




The welcome to VisualWorks window contains some useful information. Ivan Tomek's introduction will lead you through much of Smalltalk. Try going through it.

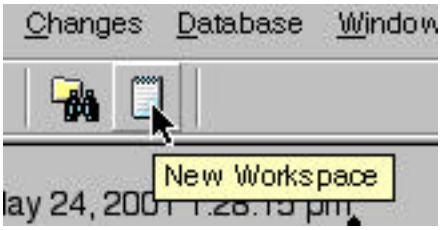
# Lesson Runner

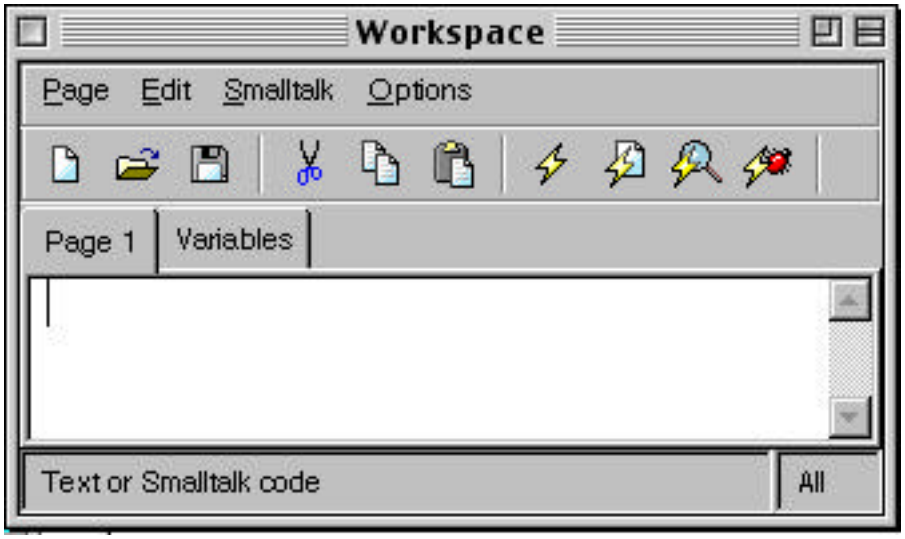



Lesson runner leads you through Smalltalk syntax, some of the Smalltalk classes and some tools.

# Workspace

Workspaces are good places to try out Smalltalk code. There are two ways to open a Workspace from the Launcher. First you can click on the Notepad icon in the Launcher. The second way is to select the “Workspace” item in the Tools menu in the Launcher.




There are a number of interesting things one can do with the Workspace. The most important is to execute code. The code can be run with four different options: do it, print it, inspect it, and debug it. To run the code first select the code you wish to run. The code should now be highlighted. Now run the code using the option you wish by: clicking on the corresponding icon, selecting the corresponding item in the “Smalltalk” menu, or using the short-cut keys.

 Do it (ctrl-d)
Compile and execute the selected code

 Print it (ctrl-p)
Same as "do it" but also prints the result of running the code

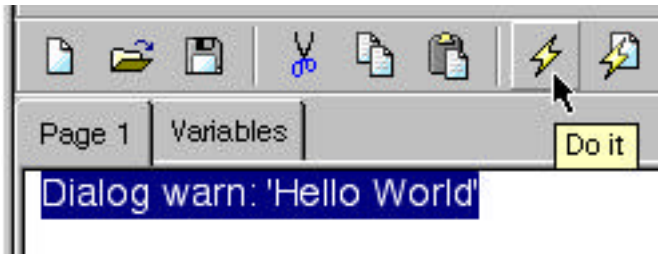
 Inspect it (ctrl-i)
Same as "do it" but also opens an inspector window on the result of running the code

 Debug it (ctrl-d)
Opens the debugger to allow you to step through the selected code. We will cover the debugger later.

# Examples

We will go through a few examples. To keep the images small, only part of the windows will be shown.

## Do it

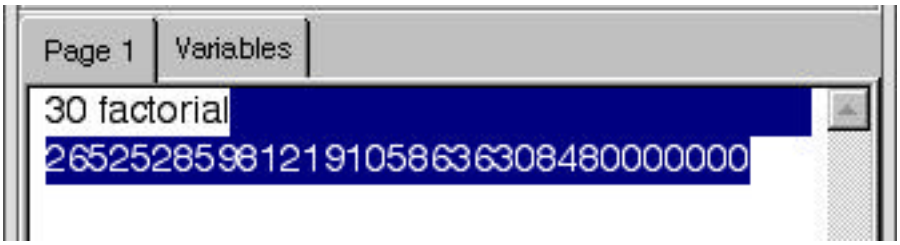



In a workspace window type: Dialog warning: ‘Hello World’. Select the text and then run the code with "do it". In the window above, “do it” was done via the icon. When this is done you will see the following dialog. When code is run via do it, the code is first compiled to Smalltalk byte code. The byte code is translated to machine code by the JIT, which is then run.

# Print it

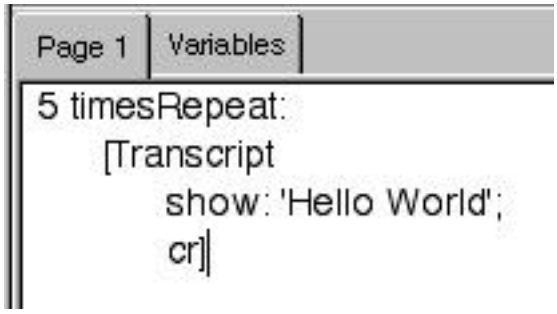
For the following example, the workspace tool bar was closed to make the graphic a bit smaller. This is done using the option menu in the workspace.


In a workspace, type the text: 30 factorial. Now try selecting the text and running the code using “do it”. Seemingly nothing happens. The code is run, but there is no indication of this. Now run the code using “print it”. This will run the code and print the result of the code in the workspace. As you can see below, the result is a large integer. Smalltalk can handle any integer that your machine as memory for.



# Using the Transcript

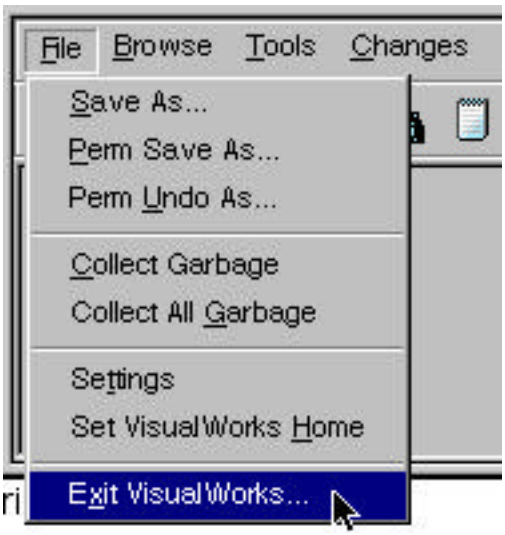
Smalltalk has a special output window called the Transcript. The text pane in the Launcher is the Transcript. You can write to the Transcript from your code. While in some languages this is done a lot in debugging, Smalltalk provides a better way to debug.




Type the code shown above in a workspace and run the code using “do it”. The following will appear in the Transcript.

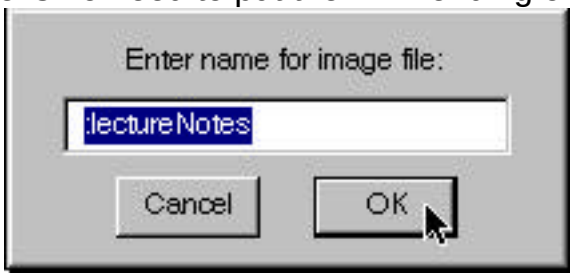
# Exiting from VisualWorks

To exit from VisualWorks, select the “Exit VisualWorks” item in the “File” menu of the Launcher.




You will get a dialog asking if you want to save the image. Saving the image put the current state of the image on disk. When you restart the image it will be in the same state as when you saved it. Normally you want to save the image. If you don’t save the image it takes some effort to recover your work.


Saving your image changes the state of the image on disk. If you save the image under the name visual, then you will lose the original state the image. So save the image under a different name. When you save an image “.im” is appended to the end of the name. Hence there is no need to put the “.im” ending on the name.



# Some Text Editing Short Cuts

Selection shortcuts (double click)

To select text, use the following double-click shortcuts.
Double-click at start or end of a window to select all text in the window.
Double-click at start of line to select the line. Does not work on the first line.
Double-click at end of line to select the line. Does on work on the last line.
Double-click just after an opening (or just before the closing) of ' or [ or ( or " selects all text surrounded by the symbols.
Double-click inside a word or selector to select the word or selector.

Ctrl keys

Press at the same time the control key and the second key to:

<Ctrl> f inserts ifFalse: into the text.
<Ctrl> t inserts ifTrue: into the text.
<Ctrl> g inserts := into the text.
<Ctrl> d inserts today's date into the text.
<Ctrl> s (search or find) finds the next instance of the string in your copy buffer (last copied or cut string)
<Ctrl> e (replace) opens a replace dialog
<Ctrl> a (find) opens a find dialog
<Ctrl> c is equivalent to Copy,
<Ctrl> z will Undo the most recent text change,
<Ctrl> v is equivalent to Paste

ESC keys

Press and release the ESC key then press the second key to:

ESC b changes the selected text to bold.
ESC i changes the selected text to italic.
ESC u underlines the selected text.
When the letter is uppercase (B, I, U), the effect is reversed: ESC U removes underline, etc.
ESC + increases the font size of the selected text
ESC - decreases font size of the selected text
ESC followed by < or ' or " or [ or ( adds surrounding < ' " [ ( to the selected text.
ESC followed by <tab> selects the text just typed in
ESC x removes style changes to current selection and returns to default font.

# Alt Keys

Alt keys are also used as menu accelerator keys. Since menu accelerator keys have priority the following keys may not work in all text windows.

<alt> a (again) repeats the last text change
<alt> A repeats the last text change for rest of text in window
<alt> c copies selected text into copy buffer
<alt> d (do it) compiles and executes selected text
<alt> f finds the next occurrence of the selected text
<alt> F finds the next occurrence of the text in the copy buffer
<alt> n finds the next occurrence of the selected text
<alt> p pastes the copy buffer into the current location
<alt> P pastes from the last 5 elements of copy buffer
<alt> z undoes the last edit