

CS 535 Object-Oriented Programming & Design

Spring Semester, 2003

Doc 3 Classes Part 1

Contents

VisualWorks Tools & Windows	8
Launcher	8
Viewing a Class	9
Viewing Point Methods	13
Viewing Inherited Methods	15
Finding Methods	19
Viewing Class Methods	20
Class Comments	21
Defining a New Class	22
Creating a Category	22
Adding Methods	24
Versions of Methods	26

References

Ralph Johnson's University of Illinois, Urbana-Champaign CS 497 lecture notes, <http://st-www.cs.uiuc.edu/users/cs497/>

Reading

VisualWorks Application Developer's Guide, Chapter 1 pages 44-56. The VisualWorks Application Developer's Guide is the file AppDevGuide.pdf in the docs directory of the VisualWorks directory. On Rohan this file is /opt/smalltalk/vw7nc/doc/AppDevGuide.pdf

Copyright ©, All rights reserved. 2003 SDSU & Roger Whitney, 5500 Campanile Drive, San Diego, CA 92182-7700 USA. OpenContent (<http://www.opencontent.org/opl.shtml>) license defines the copyright on this document.

Procedural Paradigm

Programs consists of data and procedures

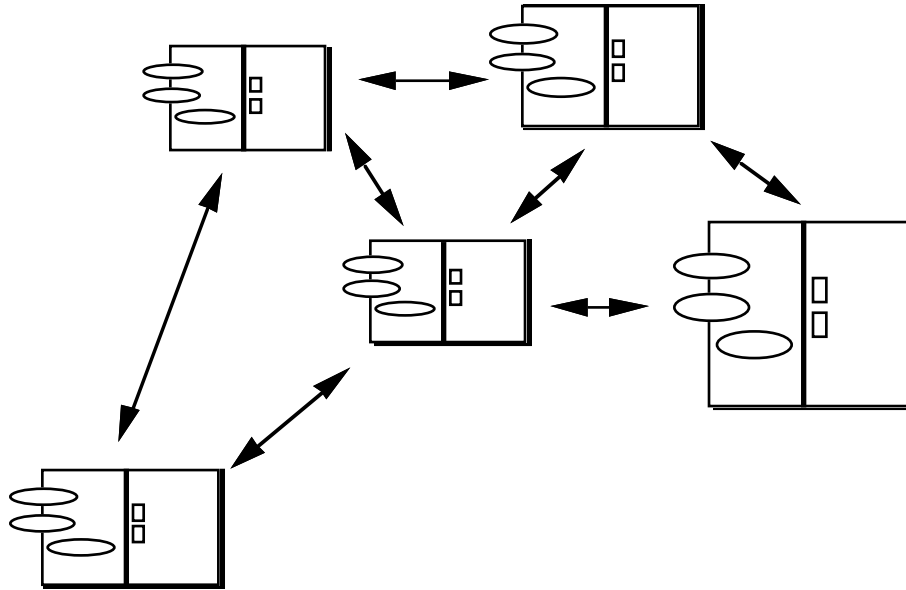
Procedures manipulate data passed to them

Programs organized by

- Functional decomposition
- Dataflow
- Modules

Object-Oriented Paradigm

Program consists of objects interacting



Objects:

- Know things
- Do things
- Collaborate with other objects to perform task

Program are organized by:

- Classes
- Inheritance hierarchies
- Subsystems

Software Development & Modeling

Each stage of software development is modeling

- Analysis

Modeling the problem

- Design

Modeling the solution

- Implementation

Making the model run on a computer

- Maintenance

Fixing and extending your model

Modeling & Objects

Base system design on structure of problem

Objects are the bases of OO design

Objects considered more natural

Match way people think

Objects come from the problem domain

Example Problem

A refrigerator has a motor, a temperature sensor, a light and a door. The motor turns on and off primarily as prescribed by the temperature sensor. However, the motor stops when the door is opened. The motor restarts when the door is closed if the temperature is too high. The light is turned on when the door opens and is turned off when the door is closed.

What are some objects?

What are some relationships between objects?

What are some attributes of the objects?

What are some behaviors of the objects?

Two Views of Objects & Classes

Model View verses Language Syntax View

Object - Modeling Perspective

Abstraction in the domain with both

Attributes (things it knows)

Behavior (things it can do)

Class - Modeling Perspective

A class is a specification for an arbitrary number of objects

Objects that share the same behavior belong to the same class

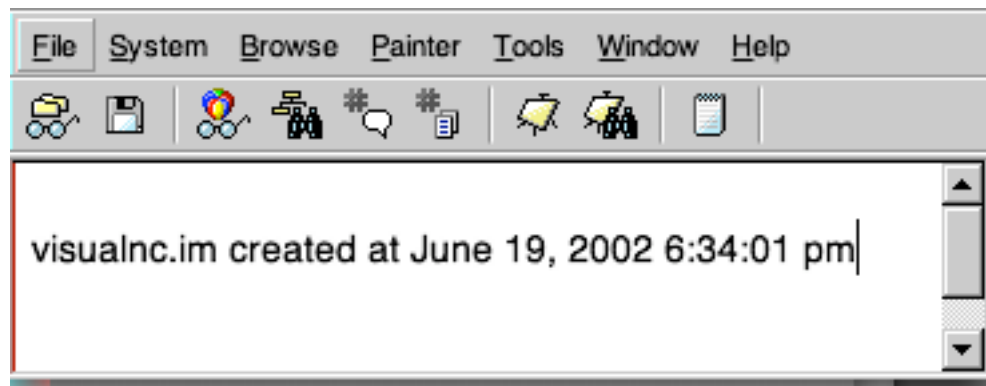
An object is an instance of a class

VisualWorks Tools & Windows

Smalltalk classes are created and viewed with a class browser. As we cover Smalltalk class syntax it will be useful for you to work with a browser. So we will first cover some aspects of the browser, then we will look at the language details. There is a chicken-and-egg problem here. To understand what the browser is showing, you need to know the language details. It helps to use the browser to understand the language details.

Launcher

The Launcher is the window (see below) in VisualWorks used to open (launch) tools. Don't close the Launcher window.



Viewing a Class

In most languages, source code is stored in a flat file and viewed with a text editor. Smalltalk provides a browser to view and edit classes. We will look at some of the features of the system browser using the Point class as an example.

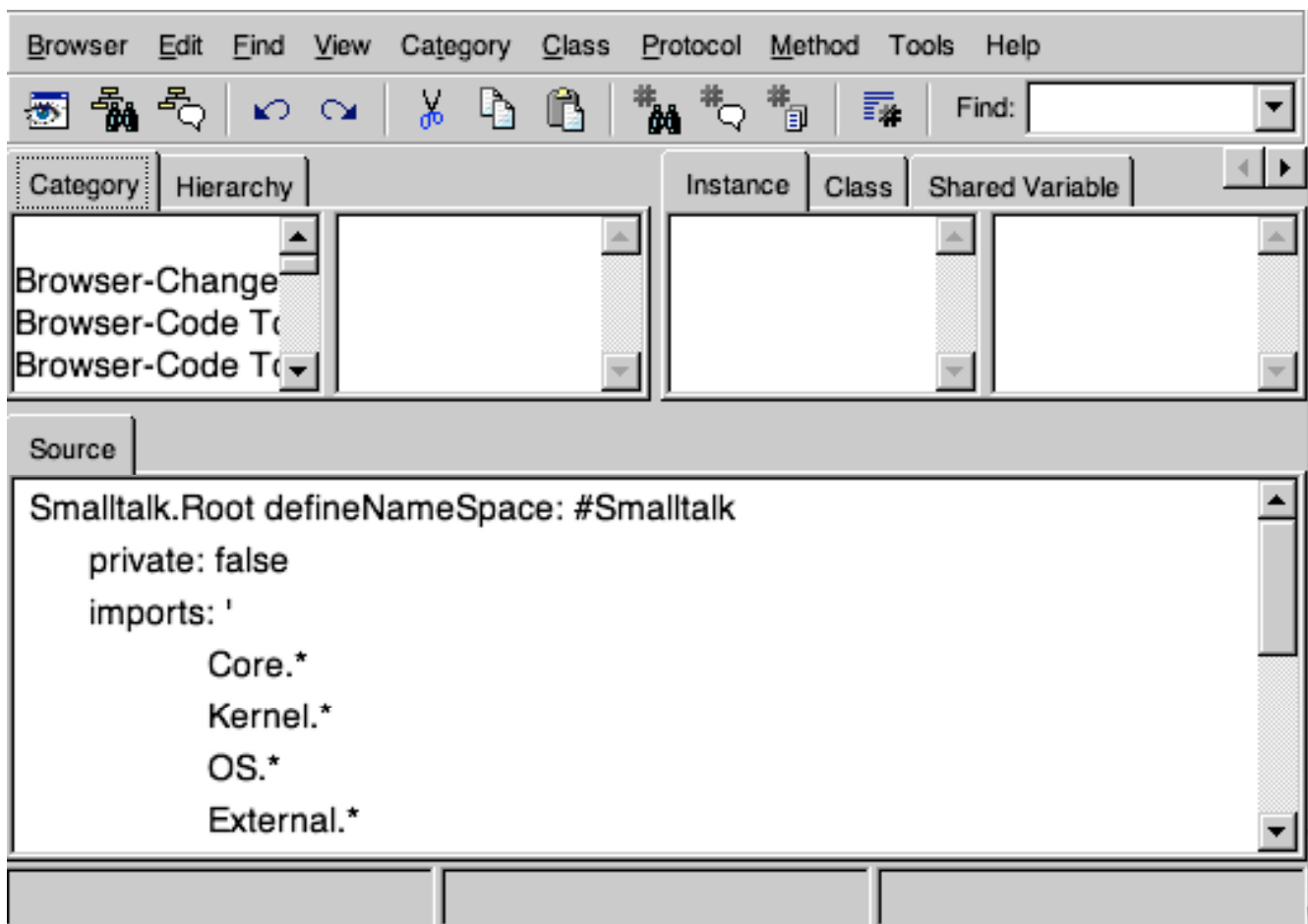
System Browser

The System Browser is used to view, manage and create code in VisualWorks. Open a System Browser by:

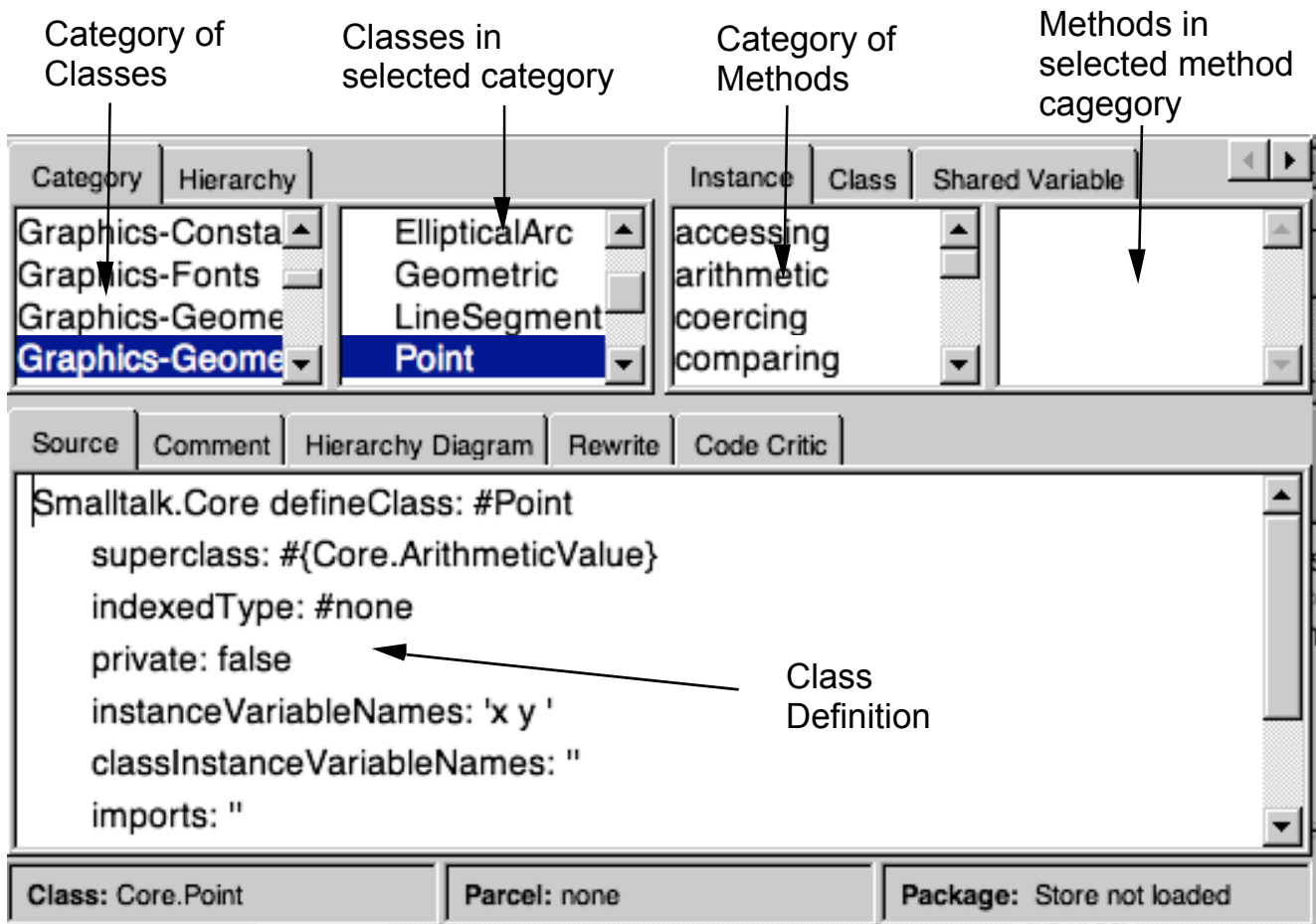
Selecting the “System” item of the “Browse” menu in the Launcher or



Clicking on the “System Browser” icon in the icon bar of the Launcher



We will look at the Point class. In the Find window enter Point and hit the return key. You should get something like:



Point Definition Partially Explained

```
Smalltalk.Core defineClass: #Point
  superclass: #{Core.ArithmeticValue}
  indexedType: #none
  private: false
  instanceVariableNames: 'x y '
  classInstanceVariableNames: ''
  imports: ''
  category: 'Graphics-Geometry'
```

Smalltalk.Core

Namespace in which Point class is defined

Core.ArithmeticValue

Parent class of Point

x and y

Points instance variables

All methods in Point can access x & y

No direct access to x & y outside of Point

Each Point object has separate copy of x & y

Using Point



Open a workspace by clicking on the workspace in the launcher.

In a workspace type:

| a b |

a := Point "x:y: is a class method that creates a point object"

 x: 1

 y: 1.

b := Point

 x: 3

 y: -1.

c := a + b.

Transcript

show: a x printString;

cr;

show: b x printString.

cr;

show: c printString.

Now select the all the text and execute it. You execute text by:

Right clicking in the workspace and selecting "Do it" in the popup menu or



Clicking on the Do it icon or

Control-d

Viewing Point Methods

If you click on a category of methods the browser shows all methods in that category. If you click on a method, the bottom pane will display the source code of the method. Instance methods are methods that you send to an instance of the class.

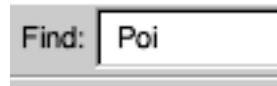
The screenshot shows a software interface with a toolbar at the top containing icons for viewing, navigating, and editing. Below the toolbar is a search bar with the text "Find: Point". The main area is divided into three panes: "Category", "Hierarchy", and "Instance". The "Category" pane shows a list of categories: "Graphics-Consta", "Graphics-Fonts", "Graphics-Geome", and "Graphics-Geome" (selected). The "Hierarchy" pane shows a list of classes: "EllipticalArc", "Geometric", "LineSegment", and "Point" (selected). The "Instance" pane shows a list of methods: "accessing", "arithmetic", "coercing", and "comparing". The "accessing" method is selected, and its source code is displayed in the bottom pane. The source code is as follows:

```
x
"Answer the x coordinate."

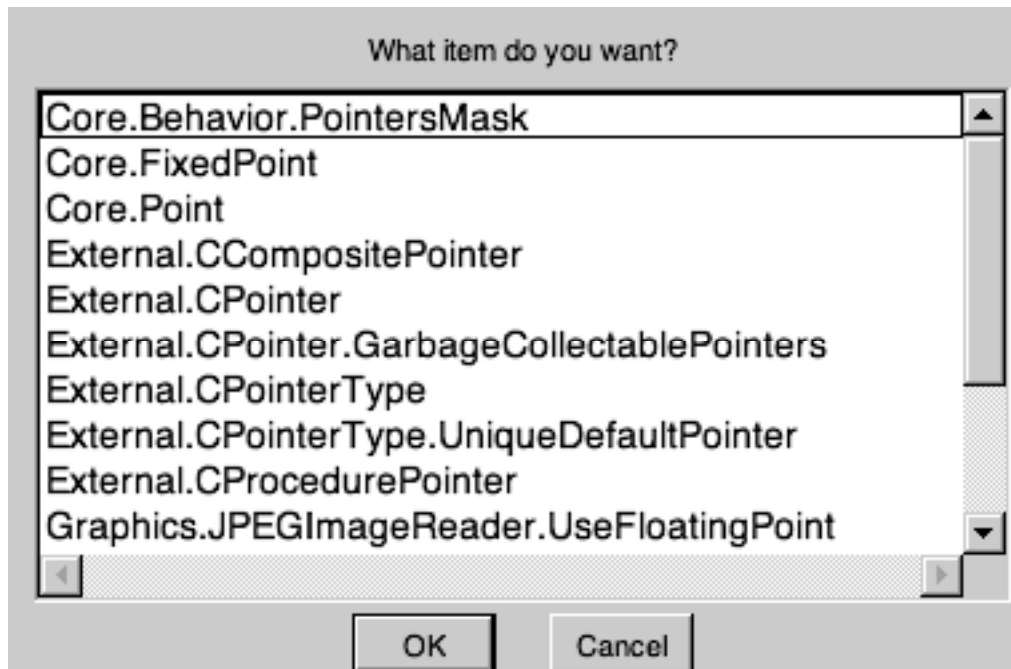
^x
```

Using Part of a Class Name

In the find window you can type in only part of the class name. The browser will find all matches for the partial name. If there is only one match the browser will show that class. If there are multiple matches the browser will give a list to select from. When I give the name: Poi

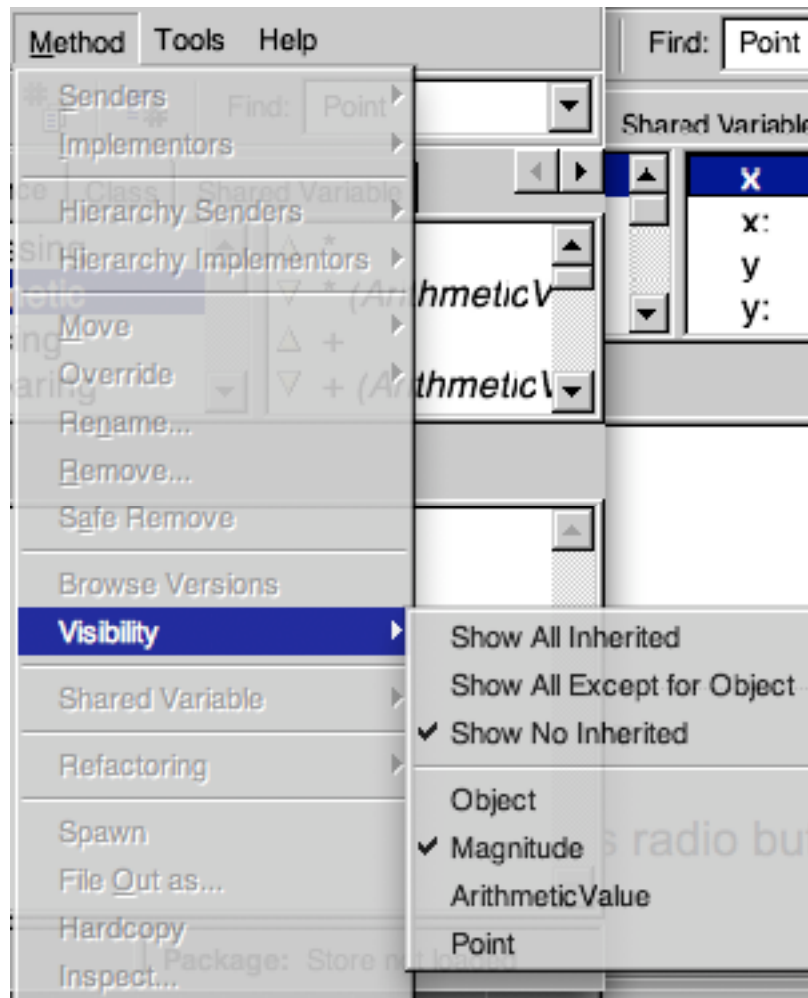


I get the following list of classes:

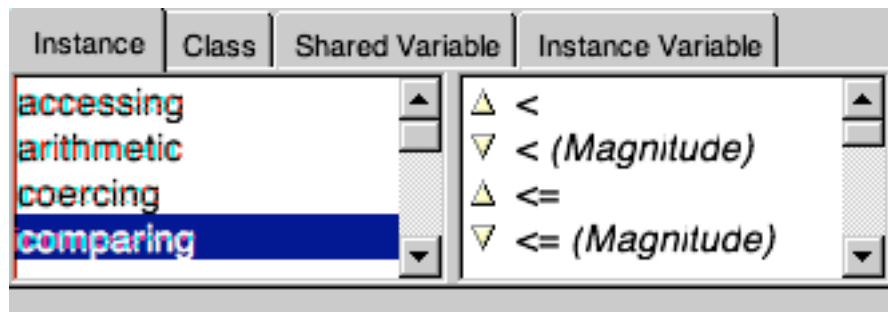


Viewing Inherited Methods

The view of the Point class only shows the methods defined in the Point class. However Point class inherits methods from its parent class. We can view these methods by viewing the Parent classes or by setting the visibility for the browser. To set the visibility of the browser select the “Visibility” item in the “Method” menu of the browser. The bottom of the popup menu shows the inheritance hierarchy of the class. You can select how high in the hierarchy you are interested in. Below I select to show all methods in the classes Magnitude, ArithmeticValue and Point.

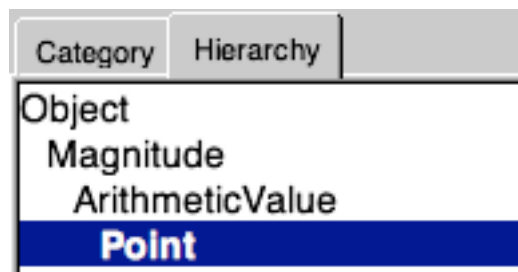


Now the name of Point methods will indicate if they are defined in a parent class.



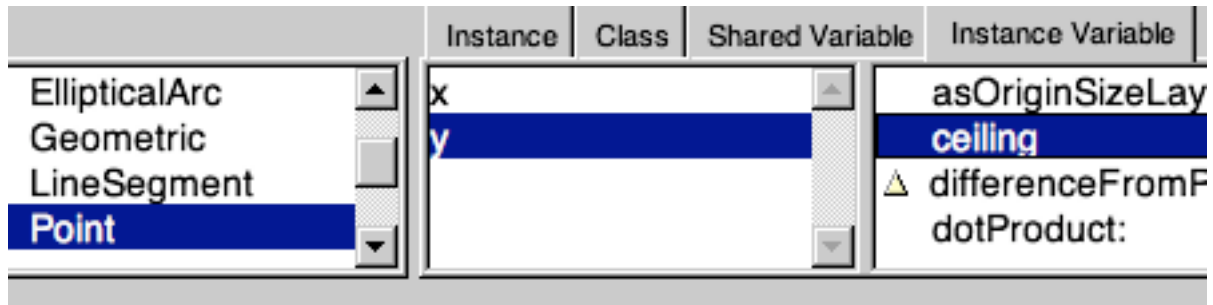
Viewing the Class Hierarchy

If you select the "Hierarchy" tab in the browser you will see the inheritance hierarchy. Now by selecting a class you can look at the methods and instance variables defined in the class.



Viewing Methods that Use Instance Variables

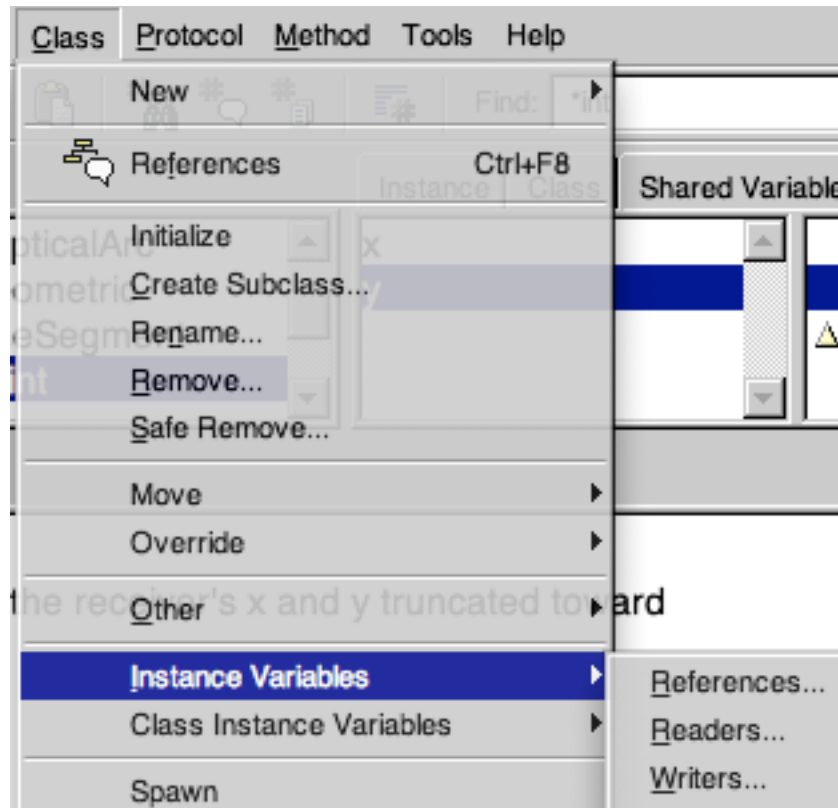
There are several ways to find methods that access an instance variable defined in a class. The first method is to select the “Instance Variable” tab.



When you do this the third list pane from the left contains a list of the instance variables of the class, in this case x & y. When you select one of the variables the fourth pane lists all the methods in the current class that access that variable.

Viewing Methods in all classes that Read or Write Instance Variables

Select the “Instance Variables” item in the Class menu of the browser. The submenu will allow you select all methods that access an instance variable, read a variable or write to a variable. When you select an option you will be asked to select one variable from a list of instance variables. You will then get a list of methods in the class inheritance hierarchy that access, read or write to the variable.



Finding Methods

In the find field of the System browser you can type the name of a method. If the method is not a keyword method you will be asked if you wish to look for a method. If there is more than one implementation of the method you will be given a list to select from. This does not work well for non-keyword methods, as the browser will first try to find a match with a class first.

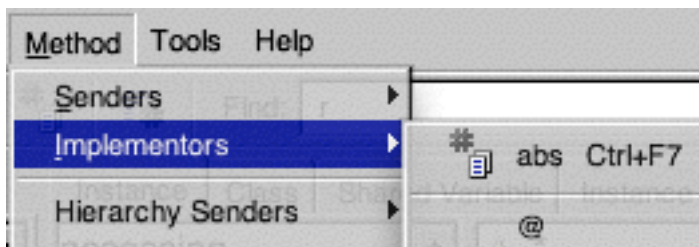
Often one is looking at a source code and comes across a method you would like to look up. For example go to the Point class and in the arithmetic category (or protocol) of methods find the abs method. The source of this method is:

abs

"Answer a new Point whose x and y are the absolute values of the receiver's
x and y."

\wedge x abs @ y abs

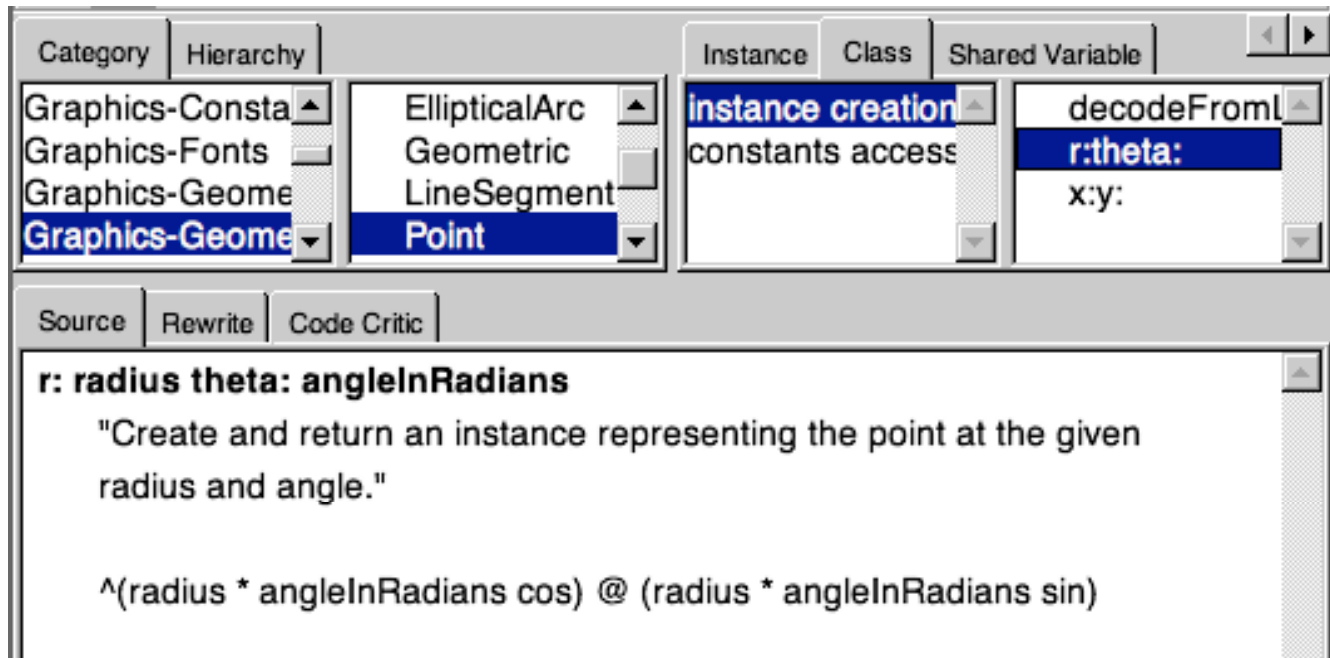
What is this @ message? To find out more about it we would like to see its implementation. To see the implementations of the @ message select the "@" item in the "Implementators" item of the "Method" menu of the system browser. A new browser will open listing all the classes that implement @.



Note that right clicking in the forth list pane from the left will also produce the Method menu.

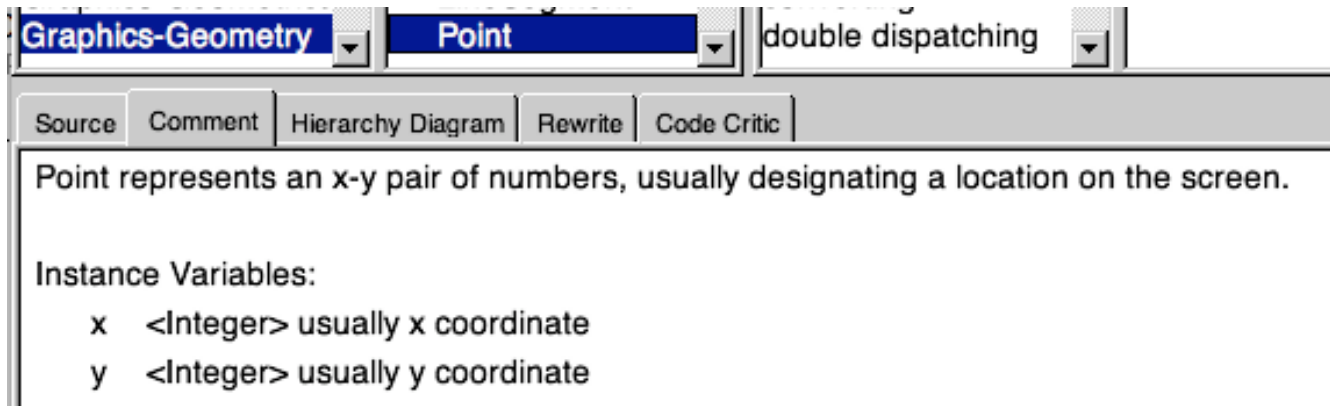
Viewing Class Methods

To view the class methods click on the class radio button in the browser.



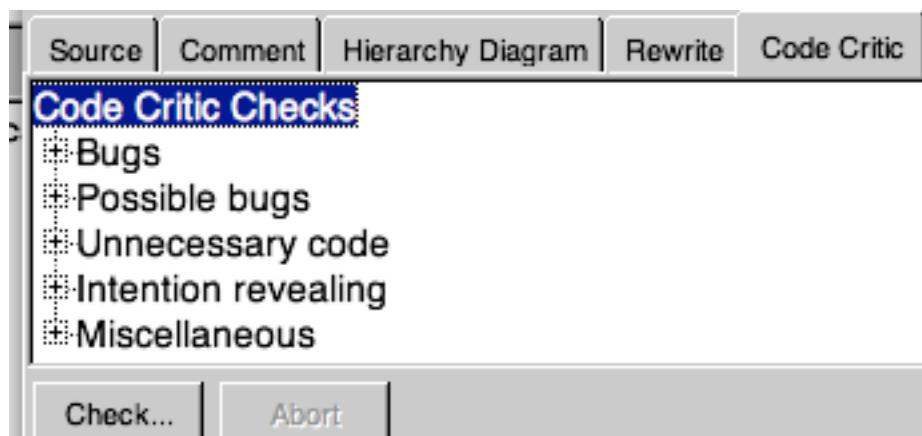
Class Comments

When you select a class in the browser (without selecting a method category). The code pane at the bottom of the browser has five tabs. Select the “Comment” tab to view the class comment.



Code Critic

The code pane at the bottom of the browser has a code critic tab. Select the types of problems want the code critic to examine, then click the “Check” button. The code critic will examine the currently selected code unit. That is you just select a category of classes, all classes in that category are checked. If you also select a class, only that class will be checked. If you also select a category of methods, only the methods in that category will be checked. By the way I do use the code critic to help me grade student assignments!



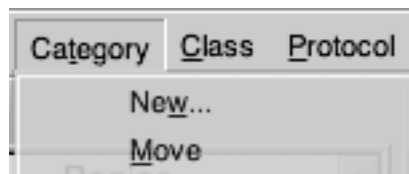
Defining a New Class

We will go through the steps of creating a new class called SimpleCircle.

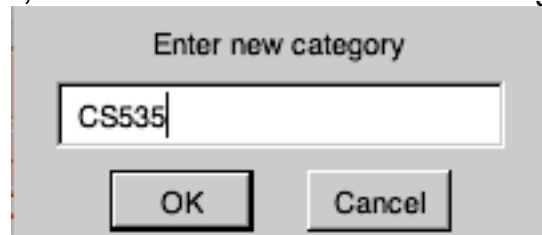
Creating a Category

First we will create a category to keep our classes separate from the system classes.

Open a system browser by clicking on the "System Browser" icon in the Launcher. Select the "New..." menu item from the Category menu in the System Browser. (You could instead right-click in the category pane.)

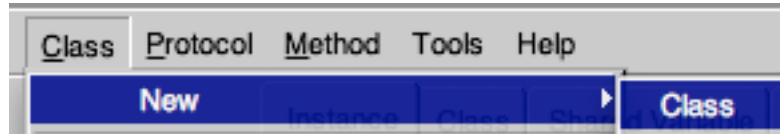


You get a dialog window, in which to enter the new category.



Creating a Class in the new Category

Select the CS535 category in the left most pane. Now select the "New" item in "Class" menu of the browser.



In the lower pane you will the template:

```
Core defineClass: #NameOfClass
  superclass: #{NameOfSuperclass}
  indexedType: #none
  private: false
  instanceVariableNames: 'instVarName1 instVarName2'
  classInstanceVariableNames: "
  imports: "
  category: 'CS535'
```

Edit the template to look like:

```
Core defineClass: #SimpleCircle
  superclass: #{Object}
  indexedType: #none
  private: false
  instanceVariableNames: 'origin radius '
  classInstanceVariableNames: "
  imports: "
  category: 'Course-Examples'
```

Accept the changes by either:

Right-clicking in the lower pane and selecting the "Accept" item in the popup menu.

Or use control-s keyboard short cut

When you do this, you will see the class listed in one of the upper level panes.

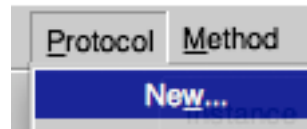
Using the Class

The class SimpleCircle does not do much, but we can still create an instance. In a workspace evaluate the following code with "Print it". What happens? Evaluate the code with "Inspect it". What happens?

SimpleCircle new

Adding Methods

Now to add a method to the class. Make sure the class is selected in the browser. Select the "New..." item in the Protocol menu.



You will be prompted to enter a category name. Enter "accessing". Once this is done the lower pane will contain the template:

```
message selector and argument names
  "comment stating purpose of message"

  | temporary variable names |
  statements
```

Edit this code to be:

```
radius: aNumber
  radius := aNumber
```

Accept these changes. (Right-click and select "Accept" or use control-s). After accepting these changes select all the text in the lower pane and replace it with:


```
area
  ^radius * radius * Float pi
```

Accept these changes (right-click or control-s). Your class now has two methods: area and radius.

In a workspace evaluate the following code with "Print it". Is your result reasonable?

```
| circle |  
circle := SimpleCircle new.  
circle radius: 2.0.  
circle area
```

You may wish to save your image so you do not lose these changes. To save

your image click on the save icon in the launcher . If you use a different name for the image, remember to start that image next time you run VisualWorks.

Versions of Methods

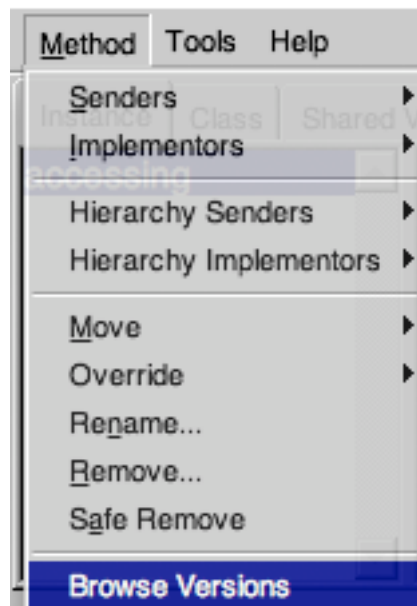
In the browser select the area method in the SimpleCircle class. Change it to be:

area

^radius + radius + Float pi

Save the changes and try running the above program to compute the area of a circle. You should get the wrong answer.

Now to browse the different versions of the method. With the area method selected in the method list pane (the right most list pane in the browser) select the “Browse Versions” item in the Method menu of the System Browser.



You will get a new window that lists the versions of the method. In this window you can view the source of each version. Select a version and use the “Replay” menu to replay a version, making it the current version. Select the “Show Conflicts” item in the “Show” menu then select a version of the method. In two panes at the bottom of the window you will see how the selected method differs from the current version of the method.

