


# An Introduction to Kawa 3.13

## Prerequisites

Before beginning to use Kawa you will need to be familiar with Windows95 or NT to the point where you understand folders, files, drives and their differences. Further, you will need to know how to:

- navigate the folder structure;
- open and close folders;
- do basic text editing;
- save files in specific folders;
- select commands from menus;
- maximise, minimise (iconify), restore (deiconify), and close an application window and a document window;

You will need to have installed Kawa from the CD-ROM so you can perform the operations and exercises suggested. Installation instructions for Kawa appear in the document *install.doc*

also available on the CD-ROM. You should have the Kawa shortcut token,  Kawa, on your desktop.

## Overview

We investigate *Kawa*, an integrated development environment (IDE) that you may elect to use throughout *Java Genesis* to do the exercises involving Java programs. Here you will learn:

- about Kawa
- how to set up projects in Kawa:
  - create new projects
  - register/deregister projects
- about Help's Search facility
- a guide to safe practice when using Kawa with *Java Genesis*

The purpose of this document is to get you started using Kawa. It is not a complete guide to all of Kawa's capabilities. As you become adept at using Kawa you are encouraged to explore Kawa's menu system and investigate the commands.

## About Kawa

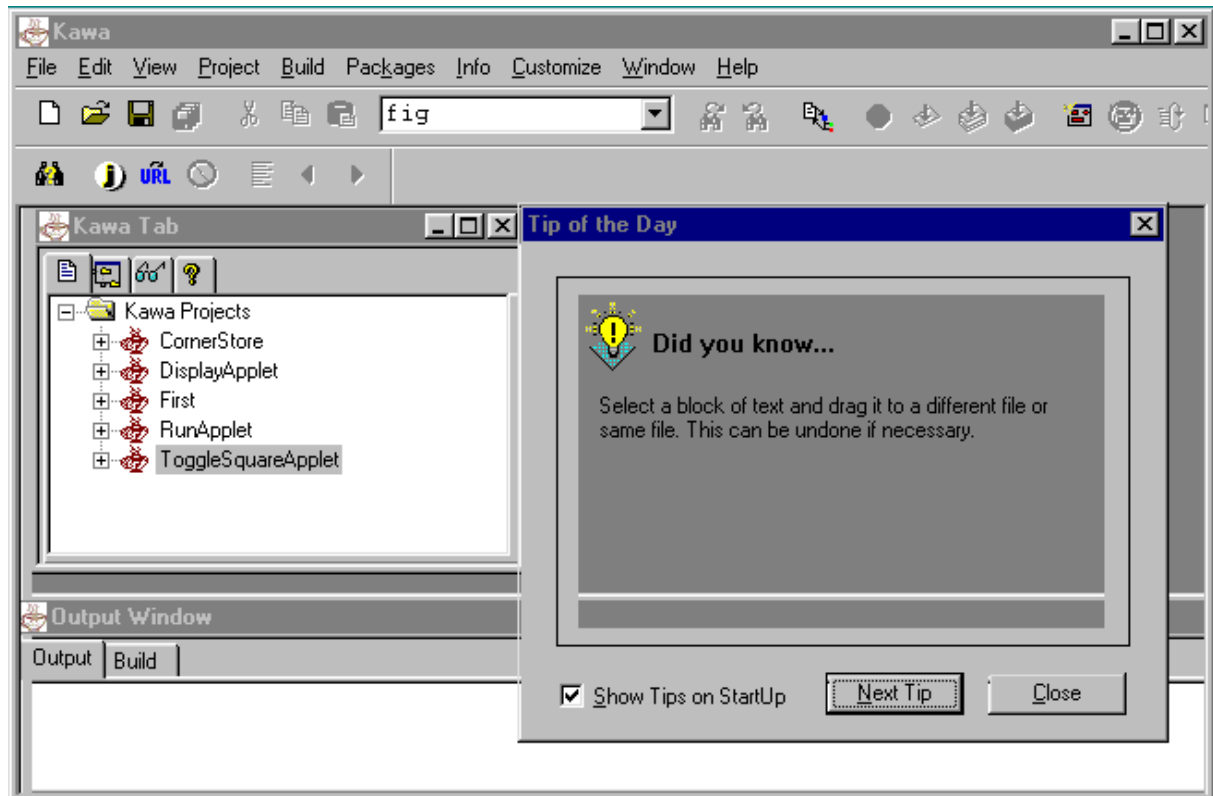
Kawa is an integrated development environment (IDE) developed by *Tek-Tools* for constructing Java applications in Windows95, 98 or NT. IDE's provide us with a graphical user interface and a set of tools that make creating and managing programs easier. Some of the tools that Kawa integrates are: an editor, a compiler, an output window, a mechanism to associate program files with a specific application or project and a library browser to allow us

to study the documentation on the packages released with Java. In this document we will learn how to use these tools.

## Starting Kawa

Start Kawa by double-clicking on the Kawa shortcut on your Windows desktop.

When Kawa starts, a window like the following (Figure 1) appears.



**Figure 1: the initial Kawa window**

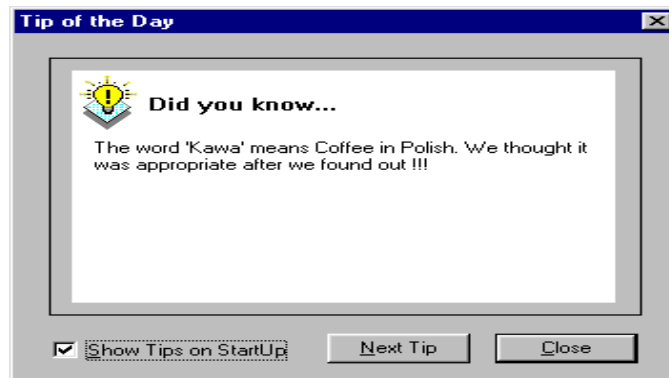
There are actually four windows here. They are:

- the whole GUI named *Kawa* and the identification of the current project we are working on (if any) ;
- a *Project Window* named *Kawa Tab*;
- an *Output Window*; and
- a *Tip of the Day* Window.

The Kawa window contains the Project Window and the Output Window; the *Tip of the Day* window is really a separate window positioned over the main Kawa window. We will see the role of each shortly.

## ***Tip of the Day Window***

The *Tip of the Day* window opens each time you start Kawa unless you tell Kawa you don't want to see it anymore. While you are getting familiar with using Kawa we recommend that you read the tips (hints) as they will help you become more proficient at using Kawa. Figure 2 shows a *Tip of the Day* window - a new tip appears each time you start Kawa. Don't worry if you don't always understand what the tips mean. We don't. Some tips are very helpful and well written, while others are cryptic and best ignored. Click on the *Close* button to dismiss this window.



**Figure 2 - Tip of the Day Window.**

## **The Kawa Window**

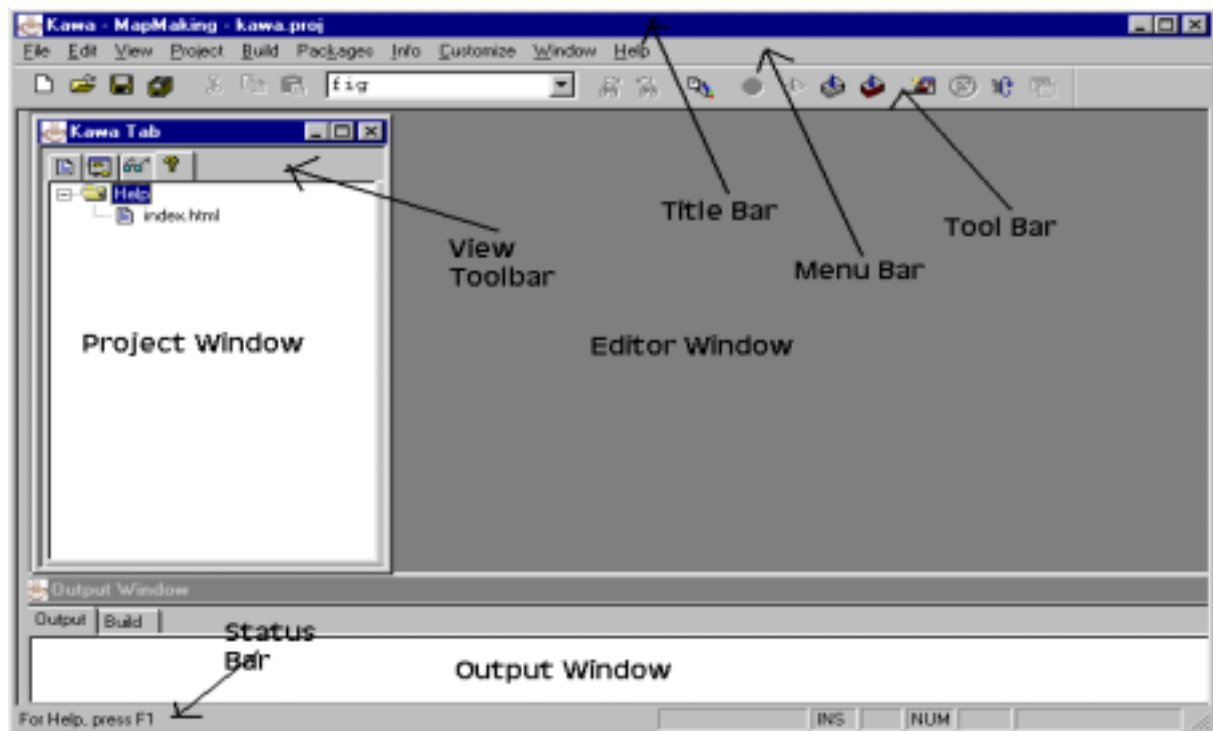
With the Tip of the Day window removed you can see Kawa's main window (Figure 3). We have labeled some key components for reference. Look at the Kawa window on your screen and make sure you can locate the various parts. There may be some differences between your Kawa window and Figure 3.

(This layout reflects the preference of the authors. Feel free to experiment. The key is to position components that suit your style. You'll learn how to do this as we proceed.)

## **Kawa Window's Main Components**

We will give an overview of each of the key essentials of the Kawa window before we demonstrate how to use the features. We'll introduce the following:

- Title Bar
- Menu Bar
- Tool Bar
- Editor Window
- Project Window
- Output Window
- Status Bar



**Figure 3: the Kawa window and its components**

### **Title Bar**

The *Title Bar* at the top of Kawa's main window displays the name of the current project and, if a file is open, the name and (possibly) the path of this file. At the right hand end of the Title Bar we have the familiar *minimise*, *maximise/restore* and *close* buttons for the Kawa main window.

### **Menu Bar**

The *Menu Bar* appears directly below the Title Bar. It provides us with a selection of menus, each full of commands. Some menu names may be familiar, such as *File* and *Edit*; others will not be quite so familiar being specific to IDE's. Let's take a few moments to investigate the commands in some of the menus.

The menus follow the standard Windows format in that a command followed by ellipses (...) means that a dialog box will appear requesting further information if we select that command. Many commands in the menus have a keystroke combination displayed to the right of the command which can also be used to invoke that command if we get tired of operating the pull-down menus. Commands greyed-out are not available for use at the current time. For example, opening the *Edit* menu with no file opened will give the complete set of Edit commands none of which can be started.

Let's investigate the *File* menu.

- Open the *File* menu.  
Notice that the *Open* command is followed by ellipses (...).
- Click on *Open*.  
A dialog box appears asking you to provide more information (the file to open in this case).
- Click on the *Cancel* button to close the dialog box without opening a file.
- Open the *File* menu again. Notice that to the right of *Open*, the menu tells us we can also invoke this command by using the keystroke combination *Ctrl* and *O*.
- Close the *File Menu* by clicking anywhere in Kawa's *Editor Window*.
- Press the *Ctrl* and *O* keys on the keyboard. The same dialog box we saw before will appear.
- Click on the *Cancel* button.
- Open the *File* menu yet again. Notice the *Close All* command is greyed out indicating that it is currently unavailable.
- Close the *File* menu without making a selection.

## Tool Bar

The *Tool Bar* appears below the *Menu Bar*. It contains buttons for the most frequently used commands from the menus. That is, each command an icon represents can be found in one of the menus in the *Menu Bar*. Some icons may look familiar.

Below, each button icon of the *Menu Bar* is displayed and its command explained.



*New* - creates a new file by opening a window for editing text in the *Editor Window*.



*Open* - opens an existing file and displays it in a window positioned in the *Editor Window*.



*Save* - saves the active file. The active file has its window bar highlighted.



*Save All* - saves all of the active project's files.



*Cut* - cuts (removes) the text selection from the active file and puts it on the clipboard.



*Copy* - copies the text selection from the active file and puts it on the clipboard.



*Paste* - inserts the clipboard's contents at the cursor's position.



*Find Text List* – recalls the last five text strings that have been used in a *Find Next* or *Find Previous* – commands that appear under the Edit menu and have short-cuts in the Tool Bar. The list can be looked at to make a selection for searching. A selected item may be edited as well before invoking the search.



*Find Next* - finds the next occurrence of the text sought in the active file.



*Find Previous* - finds the previous occurrence of the text sought in the active file.



*Toggle Project View* - Toggles the view in the *Project Window* between a class and file view. The class view presents the classes making the project; the file view shows the files.



*Stop* - stops the current compile.



*Compile* - compiles the active file.



*Rebuild All* - compiles all of the files in the active project.



*Rebuild Dirty* - compile all the files in the active project that have had changes made.



*Run Java* – runs (executes) the active project's application.



*Stop Java* - stops the currently running application.



*Start/Continue* - starts Kawa's debugger, or if paused continues with debugging. We don't consider this button further.



*Display Windows List* - displays a list of the open file windows if 3 or more files are open. If 2 file windows are open the inactive window is made active. The pull-down menu *Window* in the Menu Bar provides similar information.

Generally buttons provide a quick way of executing a command because we only have to click on them once but sometimes it can be hard to recall what commands the buttons execute when we are not familiar with them. To assist, each button displays a *Tool Tip* when we place the mouse pointer on the button. Additionally, the role of the button is displayed in the *Status Bar*. Inactive buttons appear grey and do not invoke a command when the mouse is positioned over them and clicked.

Let's explore.

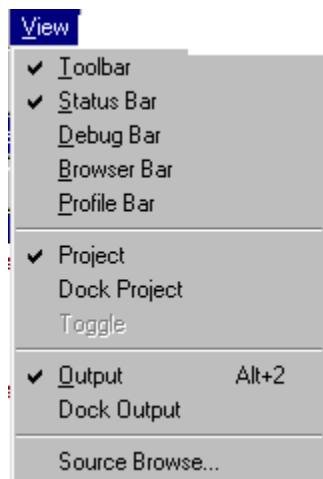
- Place your mouse pointer over the first button in Kawa's toolbar. A small box appears below your mouse pointer labeled *New*, indicating a new file will be created by clicking this button. Additionally, the phrase *create a new document* appears in the Status Bar.
- Investigate what the other buttons in the *Toolbar* do by placing your mouse pointer over each one. Don't worry if you still don't understand what some of the commands mean. The more you use Kawa the more familiar you will become with them.

## Editor Window

The entire region below the Tool Bar is the *Editor Window*. This window can contain many other windows, in particular the *Project Window*, the *Output Window* and text windows displaying the contents of opened files. This means we can have many files open at one time. Text windows displaying Java program files or pure text can be modified and the files *Saved* or *Saved As* (for a new file or copying an existing file).

## Project Window

Open *View* menu by clicking on View in the Menu Bar. The following menu (Figure 4) appears.



**Figure 4: View Menu**

If you choose to use Kawa in your studies with *Java Genesis*, you will make extensive use of the Tool Bar so we suggest you leave it visible as indicated by the tick beside *Toolbar* choice.

Using the View Menu, the status of the *Project Window* can be set to *docked* in which case the project window appears attached to the left-hand border of the Editor Window without its *Kawa Tab* (as it can't be moved). Freeing it by clicking *Project* in View Menu allows the window to float and be positioned anywhere in the Editor Window by dragging the window or moving it in the usual ways. With the View Toolbar's Tab visible it can be minimised, maximised, etc.

The *View Toolbar* of the project window allows us to change what is displayed in the Project Window. Everything is displayed in the Project Window in a tree-like structure so dependencies are easily identifiable. We look at the meanings of the icons within this tool bar.



*Project View* - displays projects currently registered with Kawa and their associated files. A project can be viewed by requesting the files comprising the project be shown. The alternative view is the class view. Here the classes making up the project are presented. This window allows us to organise and navigate through our projects and the many files and classes that will be contained in them.



*Package View* - displays the Java packages available for use. This is most useful as the library of packages that come with Java are automatically included and we can also incorporate our own. What this button permits is an analysis of what classes are available in the various packages and what methods are in the classes. We can quickly expand to the level of detail we need.



*Debugging View* - displays the current debugging information. We do not use this feature in *Java Genesis*.



*Help View* - displays information about the version of the Java Development Kit we are using and leads to documentation on advanced Java features not used in *Java Genesis*. We will say no more about this button or about these features.

As you work through *Java Genesis*, you will work mainly in the Project View. You will also find that the Package View can be of great assistance when constructing your programs. We will investigate this later in this document.

## Output Window

The *Output Window* displays various types of information. When we compile our Java programs, the Output Window will display compiler information and list the errors that were found and the line numbers where they occurred. We can also write our programs so that when they are executed they write information to the Output Window.

Opening the View Menu, you will see that the Output Window can be docked or remain floating. When docked, it appears at the bottom of the Editor Window and its width can be adjusted (even to nothing). When floated, its width and height can be altered; it can even be removed. To recover it, simply go to the view Menu and click on Output; it will return at its original size and position.

## The Status Bar

The *Status Bar* displays useful information to help us use Kawa. To illustrate, the Status Bar displays the purpose of a button in the Tool Bar when we position the mouse pointer on the button. When we are compiling a program, Kawa will indicate what it is doing by displaying messages in the Status Bar. This is particularly helpful if our program takes a long time to

compile by letting us know that Kawa is still performing. It will also indicate the result of the last compile: “No errors” or “Errors occurred during compiling.”

This completes our introductory discussion of the layout of the Kawa window. Now we demonstrate how these apparently distinct features cooperate.

## Setting up Projects

Kawa allows us to organise our Java programs by creating *projects*. Kawa creates special project files to keep track of which program files belong to which projects. (Project files have a *.kawa* extension.) We can create new projects or open existing ones previously created with Kawa but since deregistered from the Kawa system.

In this section we’ll show how to create a new project containing a new file and how to open an existing project provided with *Java Genesis*. Let’s begin.

When the Kawa system is first installed, there are no projects registered with it. All projects registered will be listed in the Project Window and visible under Project View. The Project Window should look like Figure 5.

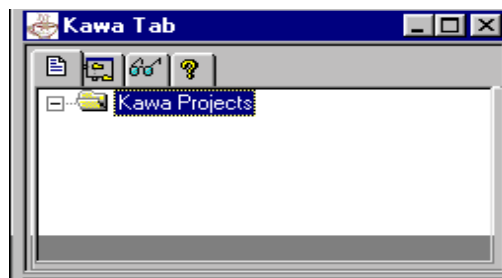


Figure 5: the project window with no registered projects

## Creating a New Project

To create a new project in Kawa, we normally create a separate folder for the project in order that each project is in its own folder.

- Open the *Project* menu and select *New*. The *New Project* dialog box appears asking you to enter the name of your new project in the *File name* box.
- Navigate to the folder *C:\genesis\projects\working\First*. This will mean creating the two new folders *working* and *First*.
- Type *First.kawa* into the *File name* box.
- Click on the *Save* button.

The Project Window now takes the appearance of Figure 6.



**Figure 6: Registering First**

We are now able to enter files into this project – either existing files or new files we can now create. We will create a brand new file by typing a small program into a text window.

### Creating a New File and Registering it in a Project

- To get a window to create a file, go to the File Menu of the Menu Bar and select *New*. A text window with the tag *Untitled* appears.
- Make sure the cursor is in the window (the window tag will be highlighted) and type in the following text ensuring what you type is *exactly* as the text appears here.

```
public class First {
    public static void main (String [] args) {
        System.out.println("This is a piece of cake.");
    }
}
```

- Now we need to **save** this file and **include** it in this project. Go to (click on) the File Menu on the Menu Bar. Select *Save As*. A dialog box appears.
- Navigate to the folder *C:\genesis\projects\working\First* if you are not already there
- Type the name *First.java* in the *Name* box . Note it is spelt with a capital 'F'.
- Click the *Save* box.
- When the file is saved, a dialog box appears asking if you would like the file added to the project. Respond by clicking the **yes** button. The Project Window appears as in Figure 7.



**Figure 7: Project First containing file First.java**

Even with this partially built simple project, we can demonstrate several features of Kawa.

### **Moving About the Project**

The minus tag before the project name **First** indicates the project is displaying the files which make up the project. Click on the minus and see what happens. Click on the plus that replaced it and the above should be restored.

Only one project can be *open* at once. By *open* we mean have its execution controlled by Kawa. The open project appears at the top of the project list, its name in boldface. The list of registered closed projects is presented alphabetically. As we have only one project registered, we won't see this feature just now.

If the file window **First.java** is still visible, click on its *Close* button. Ensure the file name is displayed under its project, *First*. Now double click on the icon representing the file and the text window will reappear. Magic, but wonderfully convenient!

To remove text windows to avoid clutter, they can be iconified (to a rectangle) by clicking the minimise button.

### **Closing the Open Project**

Close the project by right clicking on its project name *First* and selecting *Close* from the menu that appears. (See Figure 8). Reopen by right clicking on the project and selecting *Open* on the menu that appears.

All of this and we haven't even compiled or run a program yet!


### **Compiling and Running Programs**

To run a program, we must first compile it. Compiling is the process of taking a Java text file (a file with a *.java* extension) and translating it to a form the Java 'virtual machine' can understand and execute. While we can compile any Java program in any registered project, the program which runs on the *run project* command is within the project currently open (also called the *active* project). Note that if we make changes to a program by editing it, it must be recompiled before running so the change has an effect.



**Figure 8 : right clicking on a project name**

Let's compile and run the program **First.java** in the project *First*.

- Make sure the project *First* is open
- Open the files in the project by clicking on the + tag before the project's name
- Double click on **First.java** so its (text) window appears
- Click on the *Compile* button, , in the Kawa toolbar.

In the Output Window, Kawa will display information related to the compiling. Don't worry if you do not understand what is displayed - as you progress so will your understanding. For the moment just use the Output Window to observe whether Kawa found any errors in our program. The Output Window will report *No errors* if there were none found, or list the errors it did find. For **First.java** there should be no errors. (If there were, check your typing and spelling. Edit your program and retry until there are no errors. You'll possibly pick up clues to the problem if you look carefully in the Output Window.)

Of additional use to us at the moment is the information displayed in the Status Bar. While Kawa is compiling **First.java**, the Status Bar displays the message *Wait, compiling*. When the compilation is complete the message displayed is *No errors*. These messages are very helpful in determining what Kawa is doing especially if things seem to be taking a long time. Also, the *Stop* icon in the Tool Bar shows red. When the compilation is complete, it reverts to its normal colour. The Stop icon can be clicked to stop compilation for some reason. **Be warned:** you must wait until you receive the *File compiled* message before you run or recompile a program otherwise the resulting behaviour exhibited by Kawa may be strange and misleading.

Now that *First.java* is compiled you will notice that its icon in the *Project Window* is green. Files that Kawa regards as compiled and up to date have green icons. Files that

have been changed and have not been recompiled have red icons. This gives us an easy visual way of determining what files we still need to compile.

- Click on the *Run Java* button, , in the Kawa toolbar.

Now the file icon has a red box drawn around it. This tells us that this file contains the main program and it is this program that will be run whenever we request the open project be run.

Amongst other information, we see displayed in the output Window:

```
This is a piece of cake.  
Process Exit...
```

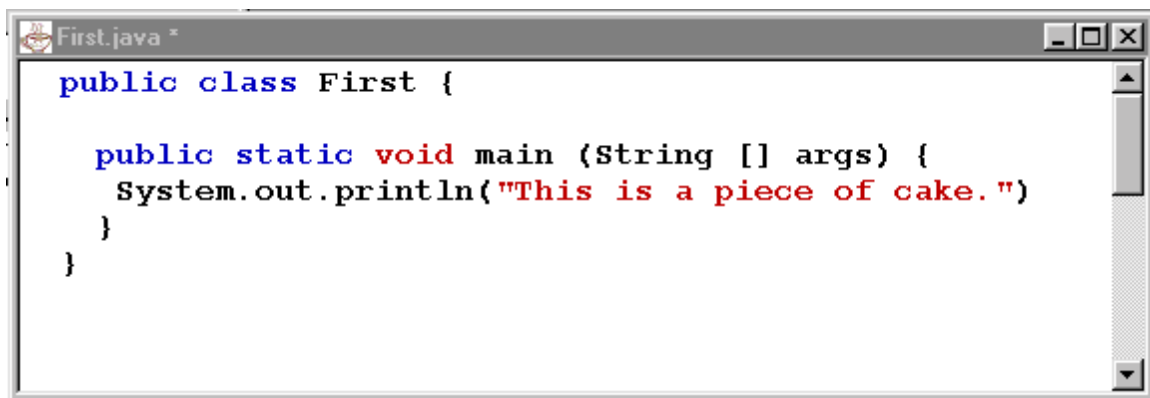
The first line is the output from our program; the second line indicates that the program has finished executing. Good news indeed!

### Compiling in the Presence of Errors

Here we look at the features Kawa provides when we compile a program with error. This is a common situation when we are developing programs.

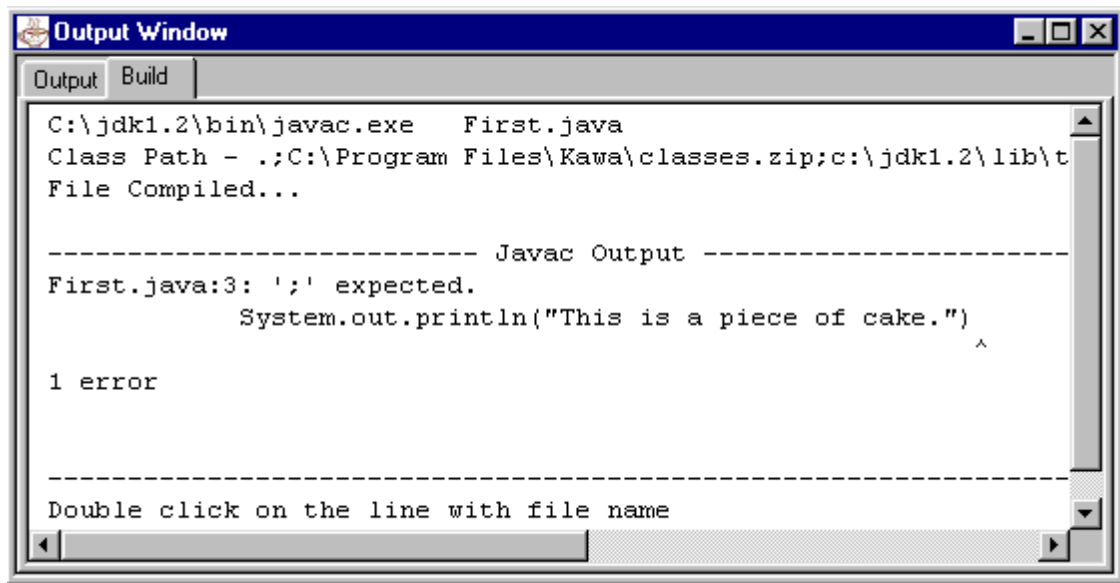
In Figure 9 is our program **First.java**. We show its window but this time note we have an error – we have left a semi-colon off the line:

```
System.out.println("This is a piece of cake.");
```



**Figure 9: an erroneous program**

This time when we compile, we get the Output Window reproduced in Figure 10.



**Figure 10: output from program with error**

We now show a neat feature of Kawa. Position the cursor over the 3 in **First.java:3** in the Output window and double click. A red arrow appears in the text window containing the program file with this error allowing us to locate the error quickly as the red arrow points directly to the line in error.

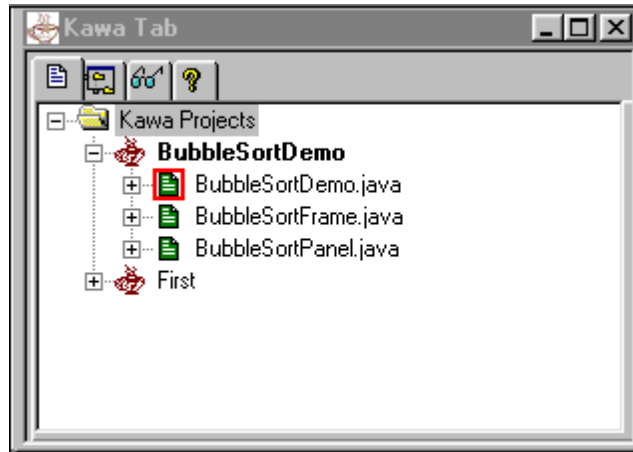
## Opening an Existing Project

Frequently throughout *Java Genesis*, you will be instructed to open a folder with a specific name. For each problem in each chapter in the text, there is a distinct folder containing a prepared Kawa project. When instructed to *open a folder called something*, it is time to register that application with Kawa. We store within each project folder all the files required for that problem as well as the Kawa project file for Kawa's use.

(*Aside: To keep the number of projects registered to a manageable number, we would advise as each chapter is completed that the projects of the completed chapter be deregistered before starting the next chapter. End Aside.*)

Let's open the project *BubbleSortDemo* in *Chapter 5*.

- In Kawa, select the Project Menu and choose *Open*.
- Navigate to the *Chapter 5* folder, thence to the *Bubble Sort Demo* folder. At this point, a file *BubbleSortDemo.kawa* will appear in the list of Kawa files in this folder.
- Click on this file and then press open. Our Project Window changes to include the registration of this project. Its appearance is now as in Figure 11.



**Figure 11: new project BubbleSortDemo added**

Now we have two projects registered with Kawa. Furthermore, the project *BubbleSortDemo* is the open project. We can see that this project is made up of three files. Studying the colours of the file icons indicates the programs have been compiled earlier and **BubbleSortDemo.java** is the main program – it has a red box around its icon. (Note: if this latter information is incorrect, it can be corrected by right clicking on the **BubbleSortDemo.java** and selecting *Main Class* from the menu that pops up – see Figure 12.)

You might try doing some of the operations we looked at thus far.

Run this project.

We will demonstrate one additional feature before leaving this example – adding a file to a project. We elect to do this to the *BubbleSortDemo* project. We'll delete a file then subsequently add it.

First we look at the pop-up that appears when we right click on a file within a project within the project window (Figure 12).



**Figure 12: file right-click menu**

To delete the file **BubbleSortFrame.java** from the project, ensure the project's files are displayed, move the cursor to this file's name, right click and from the menu that appears (Figure 12) select *Delete*.

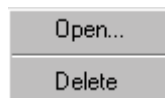
To reinstate this file:

- go to the name of the project in the project window;
- right click on the project name (**BubbleSortDemo**), a pop-up window appears (Figure 8);
- select *Add File*. A dialog box appears. Navigate to the directory `c:\genesis\projects\Chapter 5\Bubble Sort Demo`;
- select the file **BubbleSortFrame.java**;
- add it to the project either by double clicking on it, or by clicking on it and then clicking the *Add* button in the dialog box.

You can have many projects visible in the *Project Window* (those registered with Kawa) at the same time. When the Kawa system is started, it remembers its state from when it was previously closed and has the same set of registered projects.

### Deregistering a project

- First make sure the project is closed. An open project cannot be deregistered. This is done by right clicking on the project name and selecting *Close* from the menu that pops up (Figure 8).
- Right click on the closed project and select *Delete* from the following pop-up menu (Figure 13):



**Figure 13 : pop-up menu obtained by right clicking on a closed project**

Try deregistering each project. Now reregister them again. To register any project, we must recall in which folder we saved it. For example, the project *First* was in the folder `c:\genesis\projects\working\First`.

## Help's Search Facility

Java comes with a gigantic set of packages available for Java users. It is virtually impossible to remember every class let alone every method and the way each method must be used. Sun, the company that developed Java, has produced HTML pages containing full documentation. Kawa can be set up to make this documentation available on-line using *Search* under the *Help* menu. This requires downloading the Java 2 SDK documentation.

## Guides to Safe Practice

### Folder Structure

As we have seen, we can organise our Java programs in Kawa by creating projects and then adding program files to these projects. Files from different directories can be added to the same project. This can create confusion because the tree-like folder structure in the *Project Window* may create the impression it mirrors the folder structure on your hard disk drive. To prevent this confusion we recommend that you do indeed keep the two structures the same. Taking our *BubbleSortDemo* program as an example, we created a project called *BubbleSortDemo* in its own unique folder. We then created the three files we needed within this folder.

Following this recommendation makes it easy to keep track of your projects and files.

### Maintaining a Clean Copy of the Files

As you work through *Java Genesis*, you will create many programs of your own and modify programs that are given to you. Often in the modification process you will change the program so much that it will be difficult to remember how it looked originally. Sometimes you will need to return to the original version of programs. The files you need for *Java Genesis* are in the *genesis.zip* file on the CD-ROM and you can retrieve the original versions of the files at any time.

Before making significant modifications to your programs we strongly advise that you keep a backup copy of the files so that you can return to them if necessary. A major modification is one that, if you had to change the code back to its unmodified form, would take more time to accomplish than first making a backup. As most problems within the text involve modifications to one file in the project only, it should be a simple matter to save a copy of that file. This can be done easily by opening the file and using the *Save As* option under the *File* menu. Remember to give the back-up file a distinctive, recognisable name and to work on the original source.