Introduction

NetBeans IDE 3.6, a free open-source development environment for Java developers, is a highly productive tool for developing applications based on Java technology, from stand-alone executables to JSP web pages. Not only does it offer standard IDE tools including a code editor with syntax highlighting, debugging facilities, file management and versioning system support, NetBeans IDE is well support by free third-party plug-ins allowing for extensibility and flexibility.[1]

When I was choosing an IDE for Java, the key factor was similarity in functionality to the familiar Windows development tool, Microsoft Visual Studio .NET; in particular, pop-up documentation and code completion were important productivity considerations. In fact, over the last few months of use, NetBeans IDE has surpassed all initial expectations, and this article, which targets amateur to intermediate level programmers, describes the experience during this period of use, and comments on its usefulness as a Java IDE.

[1] NetBeans.org, "NetBeans IDE Features", http://www.netbeans.org/products/ide/features.html, (Retrieved 3 May 2004).

The User Interface

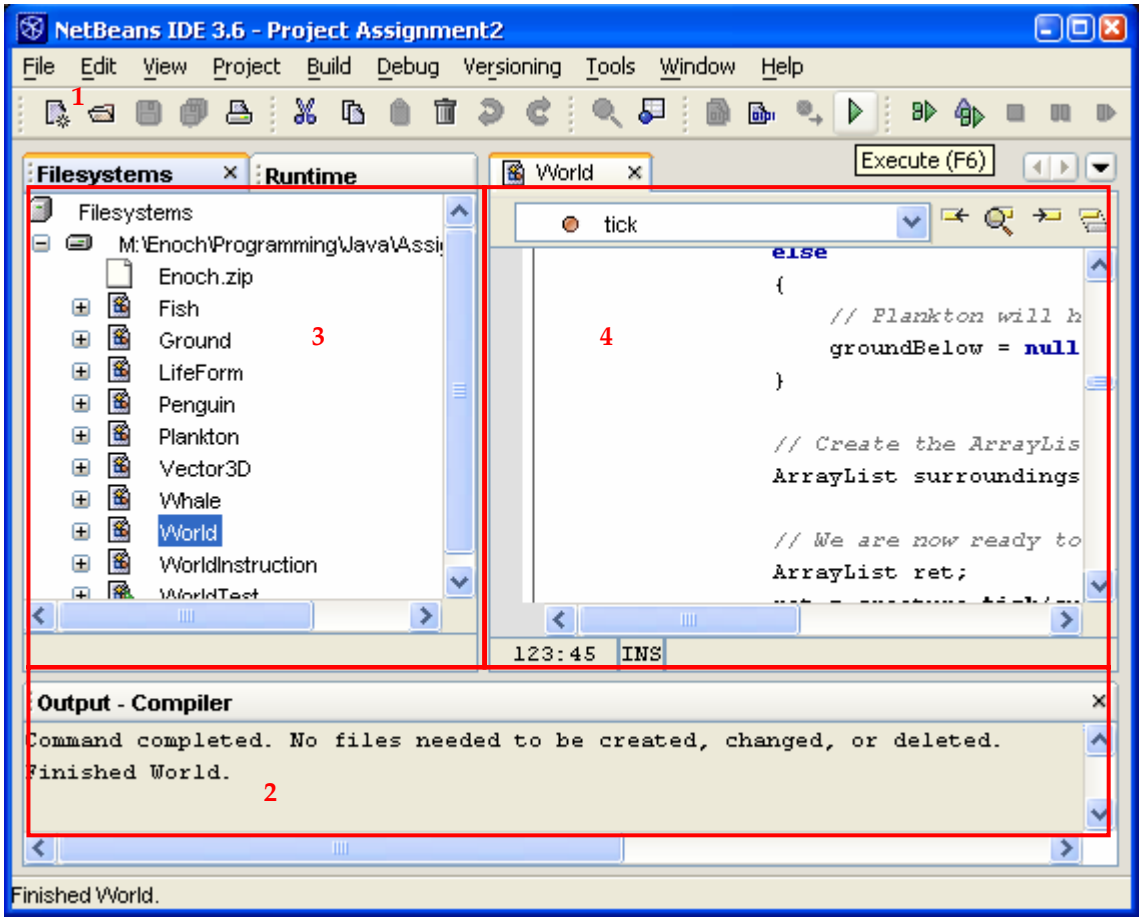

Figure 1 The NetBeans IDE 3.6 user interface

The recently introduced NetBeans IDE 3.6 user interface design is a significant improvement over that of version 3.5; it is substantially easier and more comfortable to use. It has been well designed and conformant to the familiar paradigms of menus, context-menus, toolbars and tabbed multi-document windows. Key tasks, such as compilation and debugging are quickly accessible from the toolbar (1 in **Figure 1**).

It integrates well with the Java SDK, and is able to use its services without user configuration, which is advantageous for smoothing the learning curve for programmers new to Java. The Output pane (2) is a centralised output location, providing consistency in usage, displaying output from the compiler as well as output from console programs and debugging sessions. Despite its convenience, its default size is rather small, and it is most effective when maximised by double clicking on its title bar. This applies to all the toolboxes, which unfortunately cannot be undocked from the sides and float freely.

The project's files are displayed on the left (3), and files are opened in the right-hand pane (4), which also holds the form editor when creating GUI applications. The projects are displayed in a hierarchical manner, meaning that locating a file within a project or a method within a class is quick and does not require excessive scrolling. Indeed, the ability to inspect the public interface of a class in the file list without opening the class file is a key strength to this IDE. Navigation within a document itself is best achieved by using the bookmark feature in the margin, or the method drop-down list on the top of the code window.

All elements of the user interface can be accessed via convenient keyboard shortcuts, which appear in the tool tips, as illustrated in **Figure 1**. Some of these shortcuts, grouped by function, are listed in **Table 1**. However, NetBeans IDE has inconsistently assigned different shortcuts to many of the compilation and debugging functions compared with Visual Studio, making transition more difficult.

Compilation and Execution	
Compile current file	F9
Build current file (includes cleaning of cache)	F11
Execute current class	F6
Execute current class in debugger	Alt + F5
Execute project	Ctrl + Shift + M
Next error	F12
Previous error	Shift + F12
Debugging	
Terminate a process	Shift + F5
Step over	F8
Step into	F7
Run to Cursor	F4
New breakpoint	Ctrl + Shift + F8
New watch	Ctrl + Shift + F7

Be aware of the inconsistencies in the assignment of shortcuts

Table 1 Important shortcut keys required for a productive programming experience

Furthermore, it is interesting to note that the default action is not to attach the debugger to the program, which is strange when you consider that the IDE is for software development. In addition, there is no key combination to break out of the program once it begins execution, and lockups can only be resolved by terminating the process without locating the source of the error, or attempting to use the pause menu item, which is rarely effective, as NetBeans IDE has already become unresponsive. This is possibly because it too is a Java program, and the virtual machine has become tied up.

The Coding Experience

Features of the Code Editor

How an IDE responds to, and manipulates, code has a strong impact on developer productivity, because a large proportion of time is spent in the code editor. NetBeans IDE offers syntax highlighting, error detection, automatic indentation and reformatting and code editor shortcuts.

The extensive **colour highlighting** facilitates quick and enjoyable reading of large amounts of code. In particular, the choice of grey as the comment colour is commendable, as the comments can be easily found while unobtrusive when coding. In addition, a welcome feature that improves efficiency and reduces errors is the highlighting of the matching bracket, especially when dealing with multiple nested loops or 'if' blocks in Java; placing the cursor next to one of the braces causes the matching opening/closing brace to turn pink (**Figure 2**).

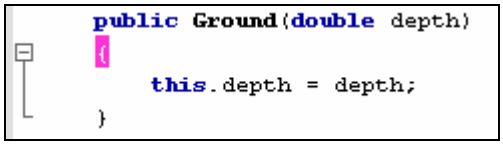

Figure 2 NetBeans IDE colour highlighting facilities

Error checking in the IDE is implemented at a standard, but sufficiently comprehensive, level. It attempts to compile each line incrementally, and compile errors are conveniently flagged with a wavy underline and a cross in the margin (**Figure 3**); the error that triggered it can be viewed by placing the mouse pointer over the line. This speeds up the development process by minimising the number of errors before the compiler is even invoked by the user, and gives instant feedback, unlike the traditional, and traditionally frustrating, edit-compile loop. This feedback forces the programmer to check documentation immediately when errors occur, instead of delaying it until later; on the other hand, it may give a false sense of security, as logic errors cannot be determined by checking performed by the IDE.

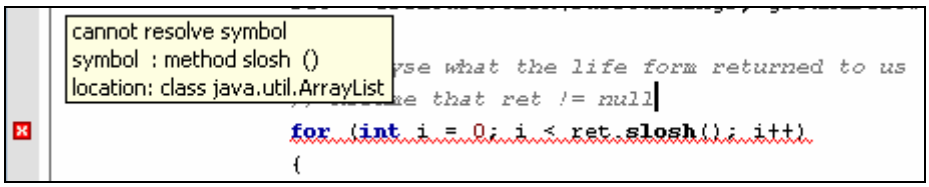

Figure 3 NetBeans IDE's inline error checking

Productivity Hint Unlike Visual Studio, which highlights the exact part of the offending line, the entire line is marked as an error. Use the compiler to locate the character where the error occurs (**Figure 4**). Then click on the blue compiler error to take you to the exact spot.

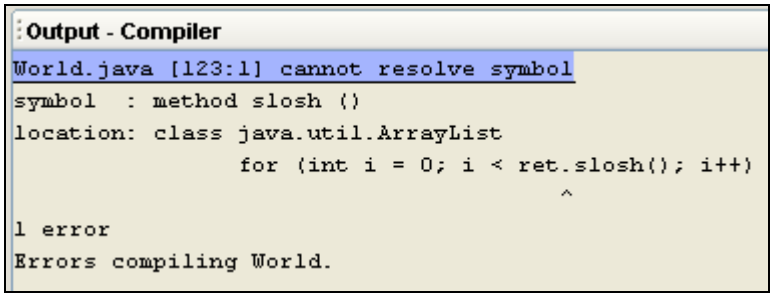

Figure 4 Using the compiler to pinpoint the location of the error – see the caret (^) symbol. An excellent feature of NetBeans IDE's integration with the Java compiler is that the error messages become links that, when clicked, take you to the line where the error occurred.

Code formatting, which is important in producing self-documenting and readable code, is made substantially easier by NetBeans IDE. When typing a structure with braces, indentation is automatically applied to the lines inside, avoiding unnecessary keystrokes. If the formatting has been completely ruined, for example, by code rearranging operations, simply pressing Ctrl + Shift + F will cause the entire file to be re-indented correctly according to the user's preferences.

Shortcuts are not only employed by NetBeans IDE to facilitate access to the menu system, but they are also used in the code editor. For example, typing `sout` and then [space] automatically expands it out into `System.out.println("")`.

Abbreviation	Expansion
Ex	Exception
Psf	public static final
Psfi	public static final int
Psfs	public static final String
ca	catch (
df	default:
impj	import java.
iof	Instanceof
psf	private static final
psfi	private static final int
psfs	private static final String
serr	System.err.println("")
sout	System.out.println("")
twne	throw new Error()

Table 2 List of commonly used editor abbreviations[2] – type the abbreviation followed by space to expand out the string

[2] NetBeans.org, "Appendix B – Editor Abbreviations", http://editor.netbeans.org/doc/UserView/apdx_b_abbrevs.html, (Retrieved 3 May 2004).

Productivity Hint Use NetBeans IDE's inbuilt abbreviations functionality to your advantage by using it to fix common typographical errors automatically, much like how AutoCorrrect works in Microsoft Word. In the Editor Settings panel (**Figure 5**), you can add and remove abbreviations. For example, you may want to map `pruvate` to `private`. Overall, one of NetBeans IDE's key strengths is its customisability, probably a reflection on its opens source roots; you can change everything from syntax colours to how the automatic indentation engine indents.

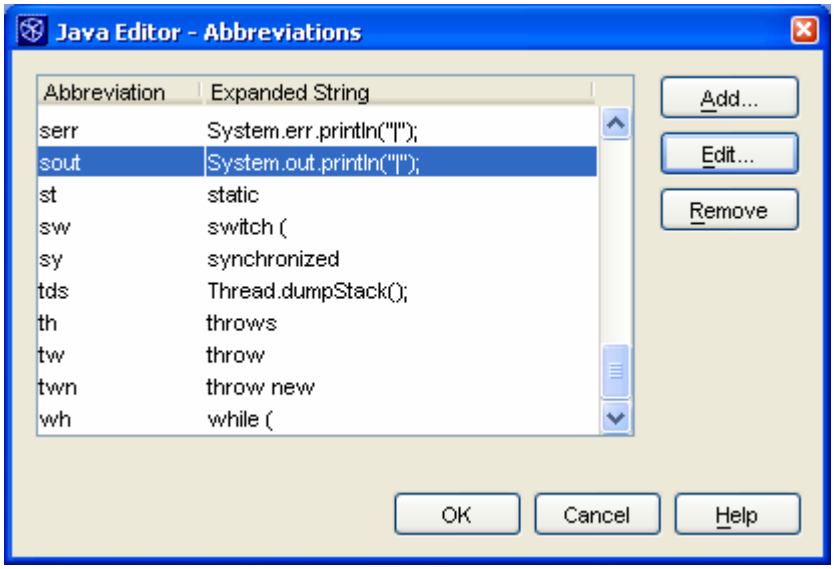

Figure 5 Customising editor abbreviations in Options

In fact, not only do these varieties of features save time, but they also make coding a far more enjoyable and pleasant activity.

JavaDoc Integration and Popup Documentation

Integration with JavaDoc is a very important consideration in choosing an IDE because of its wide adoption as the format of choice for Java documentation. Not only does NetBeans IDE link via the menus a command to invoke the JavaDoc generator, but it also utilises JavaDoc at every step of the coding process, including popup menus and a built-in utility that can write various aspects of the JavaDoc documentation automatically.

When the period (.) character is typed after an object's name, or open brackets are typed after the name of a class, all the JavaDoc information relevant to the issue at hand is presented onto the screen into two distinct panels (**Figure 6**). This is the reason why, as a programmer using NetBeans IDE, you are compelled to write documentation for all methods you write. The fact that programmers who will write classes that consume the earlier class, whether you or someone else, will be able to refer instantly to documentation about parameters, pre-conditions and post-conditions means that the time writing the documentation will prove fruitful. On a more practical level, this popup documentation negates the need to refer to the Java API documentation continually, because NetBeans IDE will use the JavaDoc statements in the Java library source code to generate the popup help panels.

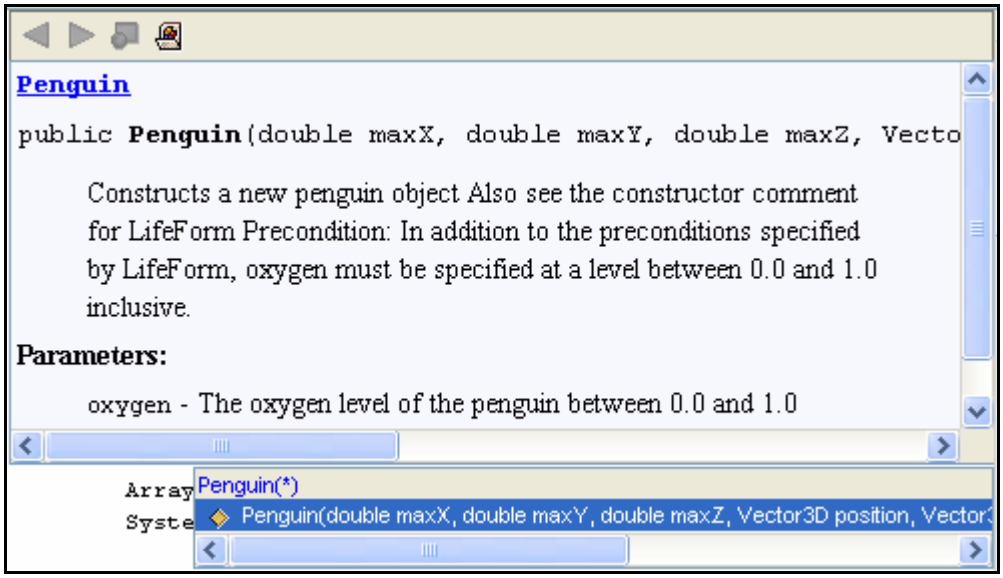

Figure 6 The top pane displays JavaDoc documentation, while the smaller pane towards the bottom displays the parameters of the current method call being constructed.

Figure 7 illustrates the Auto Comment Tool, which allows you to enter JavaDoc statements interactively instead of by code. Furthermore, by seeing a list of all the available statements, it will then be possible to continue to extend your use of JavaDoc. Naturally, writing documentation in this manner is more fun – as evident by the use of brightly coloured icons – but it is not necessarily the most efficient method, as it takes longer.

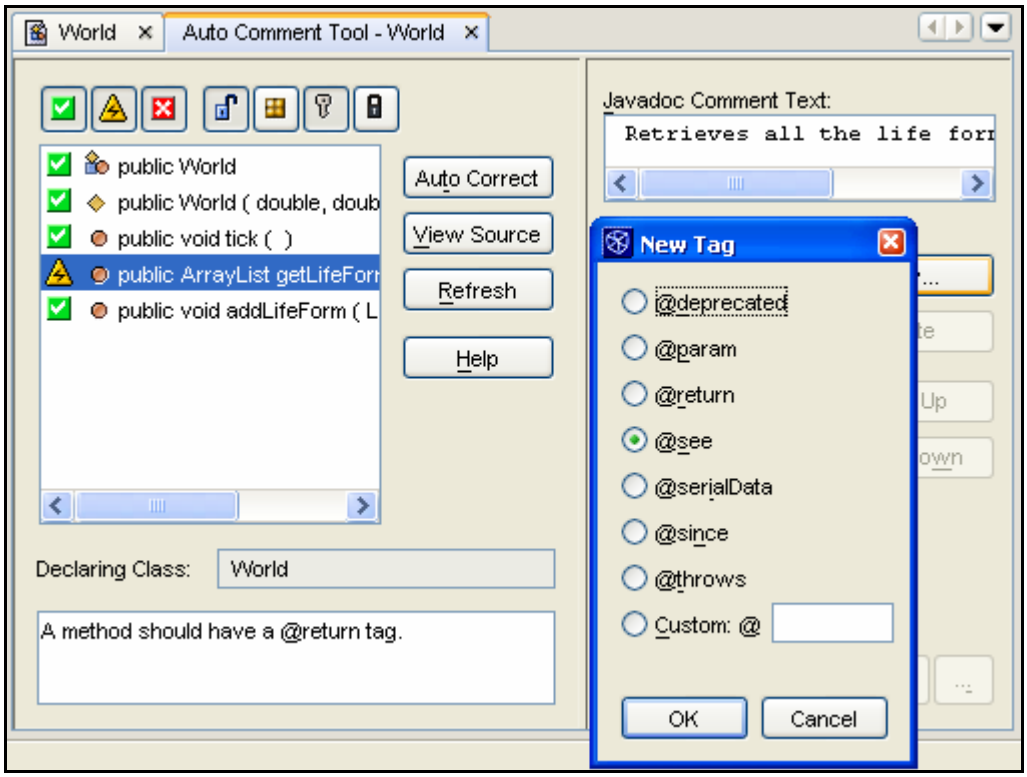

Figure 7 The Auto Comment Tool – adds and removes JavaDoc tags from above method headers. Through this attractive interface, JavaDoc becomes accessible to the majority of users, although it comes with a trade-off in coding efficiency.

Project Management

Arranging Groups of Files

NetBeans IDE arranges groups of files into projects, which contain one or more file systems. A file system is either a directory on the hard drive or a JAR archive. Each project has its own settings, including the positions of the toolboxes and the class path variable.

When NetBeans IDE is first started up, there is one project already in existence, namely the *Project Default* (see **Figure 8**). You can continue to mount completely different directories on your computer into the one project, but you run the risk of having clashes in class names. If two classes have the same name in the same project, even though they may reside nowhere near each other in the hierarchy or on the hard drive, one will be used at the expense of the other. Thus, it is best to place an individual project or assignment into a new NetBeans IDE project.

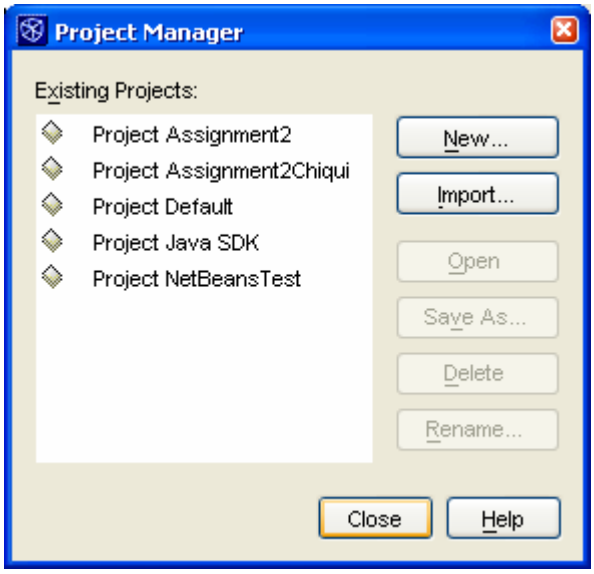

Figure 8 The Project Manager window allows you to switch between different projects

Productivity Hint When a project uses its own package system, it is very important that the correct folder is mounted as the root of those directories holding the class files, otherwise the class path will be set incorrectly. It is also a good idea to set aside a project in which to experiment, because all settings and files are particular to one project. Furthermore, by separating the project file system into one project, you can create another file system in that project to store the compiled class files, for the purpose of neatness; the compilation path can be set in External and Internal Compilation under Global Options.

Speed and Memory Usage

NetBeans IDE requires substantial resources

Although not directly related to its feature set, one cannot write an article on NetBeans IDE without mentioning its speed and memory usage, because these factors curtail users on lower-end systems to use the IDE to its full potential. Start-up times are excruciatingly slow even on a fast computer, and there are noticeable delays in the rendering of some user interface components. Memory usage averages an amazing 120MB of system memory, which, at three times the next heftiest application, may prove prohibitive for some potential users.

Conclusion

Ratings

1 = Turkey
2 = Below Expectations
3 = Satisfactory
4 = Commendable
5 = Unsurpassed

Category	1	2	3	4	5
Useability of the User Interface				●	
Features of the Code Editor					●
Documentation Support				●	
Project Management				●	

Table 3 Ratings for NetBeans IDE 3.6

In conclusion, NetBeans IDE is a solid program that permits comfortable and efficient coding, providing many facilities that enable the user to code quickly and accurately. It uses visual feedback, and a comprehensive array of commands and shortcuts that can be rapidly recalled. Project management, although initially confusing, proves to be an invaluable method to organise the files that constitute a software project.

NetBeans IDE is highly recommended to intermediate to professional programmers who need maximum control, efficiency and customisation over the pedagogical aspects of other development environments such as BlueJ.