Eclipse FOR DUMMIES®

by Barry Burd

Wiley Publishing, Inc.

Eclipse FOR DUMMIES®

by Barry Burd

Wiley Publishing, Inc.

Eclipse For Dummies®
Published by
Wiley Publishing, Inc.
111 River Street
Hoboken, NJ 07030-5774

Published by Wiley Publishing, Inc., Indianapolis, Indiana

Published simultaneously in Canada

Library of Congress Control Number: 2004116454

ISBN: 0-7645-7470-1

Manufactured in the United States of America

10 9 8 7 6 5 4 3 2 1

1B/RW/RS/QU/IN

About the Author

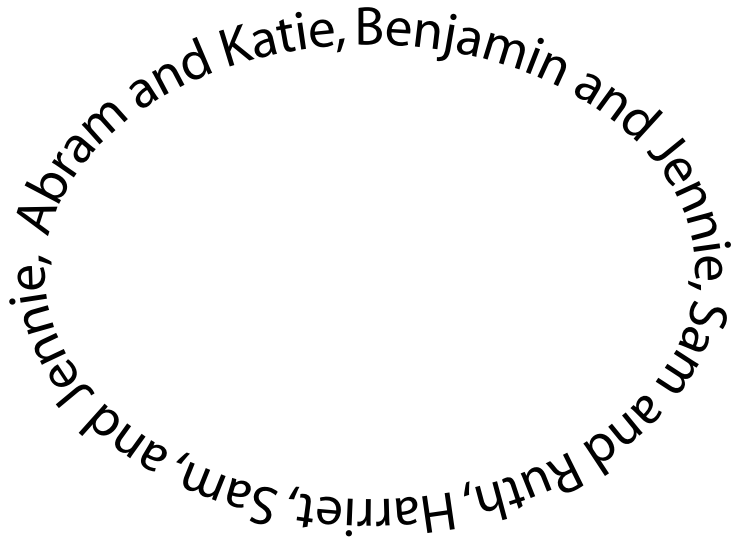
Dr. Barry Burd received an M.S. degree in Computer Science at Rutgers University and a Ph.D. in Mathematics at the University of Illinois. As a teaching assistant in Champaign-Urbana, Illinois, he was elected five times to the university-wide List of Teachers Ranked as Excellent by their Students.

Since 1980, Dr. Burd has been a professor in the Department of Mathematics and Computer Science at Drew University in Madison, New Jersey. When he's not lecturing at Drew University, Dr. Burd leads training courses for professional programmers in business and industry. He has lectured at conferences in the United States, Europe, Australia, and Asia. He is the author of several articles and books, including *JSP: JavaServer Pages,* published by Wiley Publishing, Inc.

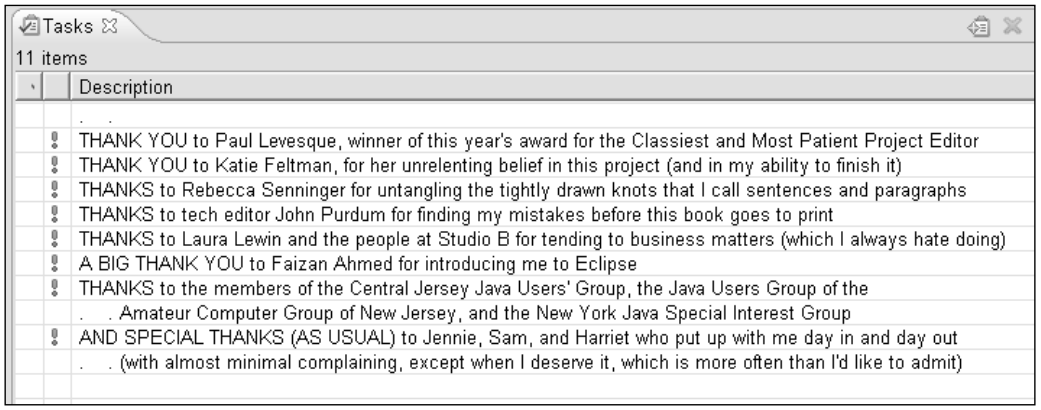
Dr. Burd lives in Madison, New Jersey, with his wife and two children. In his spare time, he enjoys being a workaholic.

Dedication

For



Author's Acknowledgments



Publisher's Acknowledgments

We're proud of this book; please send us your comments through our online registration form located at www.dummies.com/register/.

Some of the people who helped bring this book to market include the following:

Acquisitions, Editorial, and Media Development

Project Editor: Paul Levesque

Acquisitions Editor: Katie Feltman

Copy Editor: Rebecca Senninger

Technical Editor: John Purdum

Editorial Manager: Kevin Kirschner

Media Development Manager: Laura VanWinkle

Media Development Supervisor: Richard Graves

Editorial Assistant: Amanda Foxworth

Cartoons: Rich Tennant (www.the5thwave.com)

Composition Services

Project Coordinator: Maridee Ennis

Layout and Graphics: Andrea Dahl, Lauren Goddard, Joyce Haughey, Barry Offringa, Lynsey Osborn, Jacque Roth, Heather Ryan, Julie Trippetti, Mary Gillot Virgin

Proofreaders: Leeann Harney, Joe Niesen, TECHBOOKS Production Services

Indexer: TECHBOOKS Production Services

Publishing and Editorial for Technology Dummies

Richard Swadley, Vice President and Executive Group Publisher

Andy Cummings, Vice President and Publisher

Mary Bednarek, Executive Acquisitions Director

Mary C. Corder, Editorial Director

Publishing for Consumer Dummies

Diane Graves Steele, Vice President and Publisher

Joyce Pepple, Acquisitions Director

Composition Services

Gerry Fahey, Vice President of Production Services

Debbie Stailey, Director of Composition Services

Contents at a Glance

Introduction 1

Part I: The Eclipse Landscape 7

Chapter 1: Reader, Meet Eclipse; Eclipse, Meet the Reader 9
Chapter 2: Installing Eclipse 19
Chapter 3: Using the Eclipse Workbench 41
Chapter 4: Changing Your Perspective 65
Chapter 5: Some Useful Perspectives and Views 83

Part II: Using the Eclipse Environment 103

Chapter 6: Using the Java Editor 105
Chapter 7: Getting Eclipse to Write Your Code 119
Chapter 8: Straight from the Source's Mouse 137
Chapter 9: More Eclipse "Sourcery" 155
Chapter 10: Refactoring: A Burd's Eye View 173
Chapter 11: Refactor This! 189
Chapter 12: Looking for Things in All the Right Places 225

Part III: Doing More with Eclipse 249

Chapter 13: Working with Projects 251
Chapter 14: Running Code 281
Chapter 15: Getting Help 299
Chapter 16: Squashing Bugs 315

Part IV: The Part of Tens 323

Chapter 17: Ten Frequently Asked Questions (And Their Answers) 325
Chapter 18: Ten Great Plug-Ins for Eclipse 331

Index 335

Table of Contents

Introduction 1

Conventions Used in This Book 2
What You Don't Have to Read 2
Foolish Assumptions 3
How This Book Is Organized 4
Part I: The Eclipse Landscape 4
Part II: Using the Eclipse Environment 4
Part III: Doing More with Eclipse 5
Part IV: The Part of Tens 5
Additional Web Sources! 5
Icons Used in This Book 5
Where to Go from Here 6

Part I: The Eclipse Landscape 7

Chapter 1: Reader, Meet Eclipse; Eclipse, Meet the Reader 9

An Integrated Development Environment 10
A Little Bit of History (Not Too Much) 10
The Grand Scheme of Things in Eclipse 11
The Eclipse project 11
The Eclipse Tools project 13
The Eclipse Technology project 13
What's the Best Way to Create a Window? 14
Here comes Swing 15
The Standard Widget Toolkit 15
Relax and Enjoy the Ride 17

Chapter 2: Installing Eclipse 19

Setting Up Eclipse on Your Computer 19
Having enough hardware 20
Getting and installing the Java Runtime Environment 20
Downloading Eclipse 24
Installing Eclipse 25
Running Eclipse 26
Turning the ignition key 26
Revving up before you leave the driveway 29

Hello World, and Goodbye Moon 31
Getting started 31
Creating a new Java project 32
Creating a package 34
Creating and running a Java class 36
Oops! 39

Chapter 3: Using the Eclipse Workbench 41

What's All That Stuff on the Eclipse Workbench? 41
Views and editors 44
What's inside a view or an editor? 47
Understanding the big picture 49
Action sets 50
Juggling among perspectives 50
Working with Views 53
Using a working set 53
Using filters 59
Linking views with the editors 61

Chapter 4: Changing Your Perspective 65

Changing the Way a Perspective Looks 65
Adding views 65
Repositioning views and editors 68
Detaching a view 71
Fast views 72
Changing the Way a Perspective Behaves 76
The Shortcuts page 76
The Commands page 79
Saving a Modified Perspective 80

Chapter 5: Some Useful Perspectives and Views 83

Some Useful Perspectives 84
Resource perspective 84
Java perspective 84
Java Browsing perspective 85
Java Type Hierarchy perspective 86
Debug perspective 86
Some Useful Views 86
Navigator view 86
Package Explorer view 86
Outline view 87
Console view 89
Hierarchy view 89
Call Hierarchy view 93

Declaration view 93
Javadoc view 96
Problems view 97
Tasks view 97
Projects, Packages, Types, and Members views 100
Search view 101

Part II: Using the Eclipse Environment 103

Chapter 6: Using the Java Editor 105

Navigating the Preferences Dialog 106
Using Keyboard Shortcuts 106
Using Structured Selections 107
Folding Your Source Code 111
Letting Eclipse Do the Typing 112
Configuring the smart typing options 112
Using smart typing 113
Getting Eclipse to Mark Occurrences 115
Marking and unmarking 116
Some marking magic 116

Chapter 7: Getting Eclipse to Write Your Code 119

Code Assist 120
Using code assist 120
Filtering code assist suggestions 124
Auto activation 125
Templates 126
Using templates 127
Creating your own template 130

Chapter 8: Straight from the Source's Mouse 137

Coping with Comments 137
Slash that line 138
Block those lines 138
Formatting Code 139
Eclipse's Format menu actions 140
Configuring Eclipse's formatting options 143
Fixing indentation 147
Shifting lines of code 148
Sorting Members 150
Dealing with Imports 151
The Organize Imports action 151
The Add Import action 153

Chapter 9: More Eclipse "Sourcery" 155

Creating Constructors and Methods 155
Override and implement methods 155
Better getters and setters 156
Don't wait. Delegate! 158
Creating constructors 160
Creating try/catch Blocks 162
"I18n" 164
Preparing your code for internationalization 165
Adding other languages to your code 169

Chapter 10: Refactoring: A Burd's Eye View 173

Eclipse's Refactoring Tools 174
The Three Ps 175
Parameter pages 175
The preview page 179
The problem page 182
More Gossip about Refactoring 184
Selecting something 184
Why is that menu item gray? 186
Calling Eclipse's bluff 187

Chapter 11: Refactor This! 189

What Am I Doing Here in Chapter 11? 190
Renaming Things 190
Moving Things 192
Hiring a mover 193
Dissecting a parameter page 196
An immovable object meets irresistible source 197
Using views to move things 198
Changing a Method's Signature 199
Kicking Inner Classes Out 202
Pulling Up; Pushing Down 206
Extracting an Interface 206
Eclipse dodges bullets 209
Promoting types 210
Moving Code In and Out of Methods 212
Eclipse practices conflict resolution 215
Eclipse becomes stubborn (for good reasons) 216
Creating New Variables 218
But I thought I selected an expression! 220
Giving higher status to your variables 220
The Facts about Factories 223

Chapter 12: Looking for Things in All the Right Places 225

Finding versus Searching 225
Finding Text 227
Using the Find/Replace dialog 227
Using the Selected Lines option 230
Searching 231
File Search 232
Java Search 235
Using the Exception Occurrences action 247

Part III: Doing More with Eclipse.....249

Chapter 13: Working with Projects 251

The Typical Java Program Directory Structure 251
Working with Source Folders 252
Creating a separate source folder 253
Oops! I forgot to create a separate source folder. 256
Working with even bigger projects 258
Renaming your new output folder 261
Working with colossal applications 263
Adding extra stuff to a project's build path 266
Importing Code 269
Using drag and drop 269
Dragging and dropping selected directories 271
Using the Import Wizard 273
Adding Javadoc Pages to Your Project 276

Chapter 14: Running Code 281

Creating a Run Configuration 281
Using Program Arguments 284
Running with program arguments 285
Is there such a thing as a rerun configuration? 287
Piling on those run configurations 288
Using Properties 288
Using Other Virtual Machine Arguments 290
Using Environment Variables 294

Chapter 15: Getting Help 299

Searching for Help 299
Things you can use in a search expression 301
Using a help working set 302
Some useful Search view tricks 304

Using the Help View 305
A ten-cent tour of Eclipse's Help view 306
Some useful Help view tricks 309
Need More Help? 312

Chapter 16: Squashing Bugs 315

A Simple Debugging Session 316
The Debug View's Buttons 319
Experimenting with Your Code 320

Part IV: The Part of Tens 323

Chapter 17: Ten Frequently Asked Questions (And Their Answers) 325

I Can't My New Project 325
A New File Doesn't Appear 326
Failure to Format 326
Renaming Is Broken 327
Searching Is So Complicated 327
Large Isn't the Same as Maximized 327
Illegal Imports 328
Finding a Bookmark 328
The Case of the Missing Javadocs 329
Get Right to the Source 329

Chapter 18: Ten Great Plug-Ins for Eclipse 331

Checkstyle 332
Cheetah 332
Eclipse Instant Messenger (Eimp) 333
Gild (Groupware enabled Integrated Learning and Development) 333
Jigloo 333
Lomboz 333
Open Shell 334
PMD 334
VE (Visual Editor) 334
XMLBuddy 334

Index 335

Introduction

"There's no such thing as a free lunch."

That's what New York City Mayor Fiorello LaGuardia said back in 1934. Not many people understood the meaning or the impact of Mayor LaGuardia's statement, because he said it in Latin. ("E finita la cuccagna," said the mayor.) But today, most people agree with the spirit of LaGuardia's proclamation.

Well, they're all wrong. I have two stunning examples to prove that there is such a thing as a free lunch.

- I'm the faculty representative to the Dining Service Committee at Drew University. During the regular academic year, the committee meets once every two weeks. We meet in the university commons to evaluate and discuss the dining facilities. As a courtesy to all committee members, lunch is free.
- Open source software doesn't cost a dime. You can download it, use it, modify it, and reuse it. If you have questions about the software, you can post your questions for free in online forums. Usually someone answers your question quickly (and for free).

Many people shy away from open source software. They think open source software is unreliable. They believe that software created by a community of volunteers is less robust than software created by organized business. Again, they're wrong. The open source Linux project shows that a community of volunteers can rival the effectiveness of a commercial software vendor. And some of my favorite Windows utilities are free for download on the Web.*

This harangue about open source software brings me to one of my favorite subjects: namely, Eclipse. When you download Eclipse, you pay nothing, nada, zip, bupkis, goose egg, diddly-squat. And what you get is a robust, powerful, extensible Java development environment.

**The free CatFish program from Equi4 software does a better job cataloging CD-ROMs than any commercial software that I've tried. Mike Lin's Startup Control Panel and MCL utilities beat the competition without costing any money. You can find CatFish at* `www.equi4.com`*, and Mike Lin's programs live at* `www.mlin.net`.

In a recent survey conducted by QA Systems, Eclipse has a 45 percent share in the Java IDE market.* That's nearly three times the market share of the highest-ranking competitor — Borland JBuilder. In June 2003, the editors of the *Java Developer's Journal* gave two Editors' Choice awards to Eclipse. As one editor wrote, "After being anti-IDE for so long I've finally caved in. It (Eclipse) has nice CVS utils, project frameworks, code refactoring and 'sensible' code generation (especially for beans). Add industry backing and a very fired up user base and you have one winning product."**

Conventions Used in This Book

Almost every technical book starts with a little typeface legend, and *Eclipse For Dummies* is no exception. What follows is a brief explanation of the typefaces used in this book.

- New terms are set in *italics*.
- If you need to type something that's mixed in with the regular text, the characters you type appear in bold. For example: "Type **MyNewProject** in the text field."
- You also see this `computerese` font. I use computerese for Java code, filenames, Web page addresses (URLs), on-screen messages, and other such things. Also, if something you need to type is really long, it appears in computerese font on its own line (or lines).
- You need to change certain things when you type them on your own computer keyboard. For instance, I may ask you to type

  ```
  public class Anyname
  ```

  which means that you type **public class** and then some name that you make up on your own. Words that you need to replace with your own words are set in *`italicized computerese`*.

What You Don't Have to Read

Eclipse For Dummies is a reference manual, not a tutorial guide. You can read this book from the middle forward, from the middle backward, from the inside out, upside down, or any way you want to read it.

* *For more information, visit* `www.qa-systems.com/products/qstudioforjava/ide_marketshare.html`.

** *For details, visit* `www.eclipse.org/org/press-release/jun92003jdjadv.html`.

Naturally, some parts of the book use terms that I describe in other parts of the book. But I don't throw around terminology unless I absolutely must. And at many points in the book I include Cross Reference icons. A Cross Reference icon reminds you that the confusion you may feel is normal. Refer to *such-and-such* chapter to rid yourself of that confused feeling.

The sidebars and Technical Stuff icons are extra material — stuff that you can skip without getting into any trouble at all. So if you want to ignore a sidebar or a Technical Stuff icon, please do. In fact, if you want to skip anything at all, feel free.

Foolish Assumptions

In this book, I make a few assumptions about you, the reader. If one of these assumptions is incorrect, you're probably okay. If all these assumptions are incorrect . . . well, buy the book anyway.

- **I assume that you have access to a computer.** You need a 330 MHz computer with 256MB RAM and 300MB of free hard drive space. If you have a faster computer with more RAM or more free hard drive space, then you're in very good shape. The computer doesn't have to run Windows. It can run Windows, UNIX, Linux, or Mac OS X 10.2 or higher.
- **I assume that you can navigate through your computer's common menus and dialogs.** You don't have to be a Windows, UNIX, or Macintosh power user, but you should be able to start a program, find a file, put a file into a certain directory . . . that sort of thing. Most of the time, when you practice the stuff in this book, you're typing code on your keyboard, not pointing and clicking your mouse.

  On those rare occasions when you need to drag and drop, cut and paste, or plug and play, I guide you carefully through the steps. But your computer may be configured in any of several billion ways, and my instructions may not quite fit your special situation. So, when you reach one of these platform-specific tasks, try following the steps in this book. If the steps don't quite fit, consult a book with instructions tailored to your system.
- **I assume that you can write Java programs, or that you're learning Java from some other source while you read *Eclipse For Dummies*.** In Chapter 1, I make a big fuss about Eclipse's use with many different programming languages. "Eclipse is . . . a Java development environment, a C++ development environment, or even a COBOL development environment."

  But from Chapter 2 on, I say "Java *this*," and "Java *that*." Heck, the beginning of Chapter 2 tells you to download the Java Runtime Environment. Well, what do you expect? I wrote *Java 2 For Dummies*. Of course I'm partial to Java.

In fact, Eclipse as it's currently implemented is very biased toward Java. So most of this book's examples refer to Java programs of one kind or another. Besides, if you don't know much about Java, then many of Eclipse's menu items (items such as Add Javadoc Comment and Convert Local Variable to Field) won't make much sense to you.

As you read this book, you may not know Java from the get-go. You may be using *Eclipse For Dummies* as a supplement while you learn Java programming. That's just fine. Pick and choose what you read and what you don't read.

If a section in this book uses unfamiliar Java terminology, then skip that section. And if you can't skip a section, then postpone reading the section until you've slurped a little more Java. And . . . if you can't postpone your reading, then try reading the Eclipse section without dwelling on the section's example. You have plenty of alternatives. One way or another, you can get what you need from this book.

How This Book Is Organized

This book is divided into subsections, which are grouped into sections, which come together to make chapters, which are lumped finally into four parts. (When you write a book, you get to know your book's structure pretty well. After months of writing, you find yourself dreaming in sections and chapters when you go to bed at night.)

Part I: The Eclipse Landscape

To a novice, the look and feel of Eclipse can be intimidating. The big Eclipse window contains many smaller windows, and some of the smaller windows have dozens of menus and buttons. When you first see all this, you may experience "Eclipse shock."

Part I helps you overcome Eclipse shock. This part guides you through each piece of Eclipse's user interface, and explains how each piece works.

Part II: Using the Eclipse Environment

Part II shows you what to do with Eclipse's vast system of menus. Edit Java source files, use refactoring to improve your code, search for elements within

your Java projects. Everything you can think of doing with a Java program lies somewhere within these menus. (In fact, everything that everyone ever thought of doing with anything lies somewhere within these menus.)

Part III: Doing More with Eclipse

What more is there to do? Lots more. Part III describes ways to customize a Java project and the run of a Java program. This part also tells you how to find help on Eclipse's murkier features.

Part IV: The Part of Tens

The Part of Tens is a little Eclipse candy store. In The Part of Tens, you can find lots of useful tips.

Additional Web Sources!

One of my favorite things is writing code. But if your idea of a good time isn't writing code, I include every code listing in this book on a companion Web site at `www.dummies.com/go/eclipse_fd`. Feel free to download any code listings into Eclipse to follow along with my examples in this book or for any of your own projects.

And be sure to visit my Web site, `www.BurdBrain.com`, for any updates to *Eclipse For Dummies* and my additional ramblings about Eclipse.

Icons Used in This Book

If you could watch me write this book, you'd see me sitting at my computer, talking to myself. I say each sentence in my head. Most of the sentences I mutter several times. When I have an extra thought, a side comment, something that doesn't belong in the regular stream, I twist my head a little bit. That way, whoever's listening to me (usually nobody) knows that I'm off on a momentary tangent.

Of course, in print, you can't see me twisting my head. I need some other way of setting a side thought in a corner by itself. I do it with icons. When you see a Tip icon or a Remember icon, you know that I'm taking a quick detour.

Here's a list of icons that I use in this book.

A tip is an extra piece of information — something helpful that the other books may forget to tell you.

Everyone makes mistakes. Heaven knows that I've made a few in my time. Anyway, when I think people are especially prone to make a mistake, I mark it with a Warning icon.

Sometimes I want to hire a skywriting airplane crew. "Barry," says the white smoky cloud, "if you want to rename a Java element, start by selecting that element in the Package Explorer. Please don't forget to do this." Because I can't afford skywriting, I have to settle for something more modest. I create a Remember icon.

"If you don't remember what *such-and-such* means, see *blah-blah-blah,*" or "For more information, read *blahbity-blah-blah.*"

This icon calls attention to useful material that you can find online. (You don't have to wait long to see one of these icons. I use one at the end of this introduction!)

Occasionally I run across a technical tidbit. The tidbit may help you understand what the people behind the scenes (the people who developed Java) were thinking. You don't have to read it, but you may find it useful. You may also find the tidbit helpful if you plan to read other (more geeky) books about Eclipse.

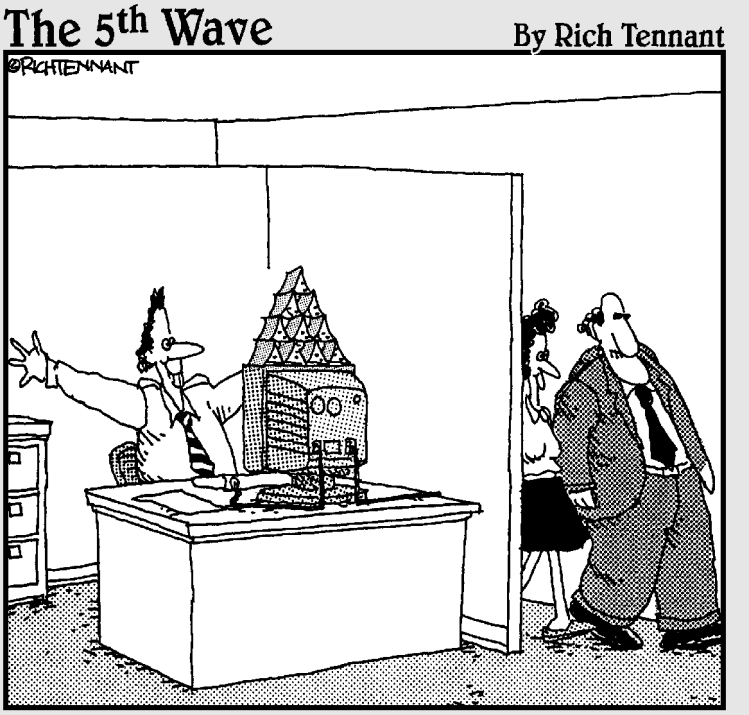
Where to Go from Here

If you've gotten this far, you're ready to start reading about Eclipse. Think of me (the author) as your guide, your host, your personal assistant. I do everything I can to keep things interesting and, most importantly, help you understand.

If you like what you read, send me a note. My e-mail address, which I created just for comments and questions about this book, is `Eclipse@BurdBrain.com`. And don't forget you can get the latest *Eclipse For Dummies* information at `www.BurdBrain.com`.

Part I

The Eclipse Landscape



"Why, of course. I'd be very interested in seeing this new milestone in the project."

In this part . . .

I'll be the first to admit it. When I started working with Eclipse, I was confused. I saw an editor here, tabs and panes everywhere, and dozens upon dozens of menu options. Eclipse is more complicated than your run-of-the-mill programming environment. So your first taste of Eclipse can be intimidating.

But if you calm down and take things step by step, then Eclipse's options make sense. Eventually you become comfortable to the point of using Eclipse on autopilot.

So this part of *Eclipse For Dummies* contains the "calm down" chapters. This part describes Eclipse's user interface and tells you how to get the most out of Eclipse's grand user interface.

Chapter 1

Reader, Meet Eclipse; Eclipse, Meet the Reader

In This Chapter

- How I learned to love Eclipse
- How the Eclipse project is organized
- How Eclipse puts widgets on your screen

The little hamlet of Somerset, New Jersey, is home to an official Sun Microsystems sales office. Once a month, that office hosts a meeting of the world-renowned Central Jersey Java Users' Group.

At one month's meeting, group members were discussing their favorite Java development environments. "I prefer JBlipper," said one of the members. "My favorite is Javoorta Pro," said another. Then one fellow (Faizan was his name) turned to the group and said, "What about Eclipse? It's pretty sweet."

Of course, Faizan's remark touched off an argument. Everyone in the group is attached to his or her favorite Java development tools. "Does Javoorta do refactoring?" "Does JBlipper support Enterprise JavaBeans?" "Does Eclipse run on a Mac?" "How can you say that your development environment is better?" "And what about good old UNIX `vi`?"

Then someone asked Faizan to demonstrate Eclipse at the next users' group meeting. Faizan agreed, so I ended the discussion by suggesting that we go out for a beer. "I don't drink," said one of the group members. "I don't either," I said. So we went out for pizza.

At the next meeting, Faizan demonstrated the Eclipse development environment. After Faizan's presentation, peoples' objections to Eclipse were more muted. "Are you sure that Eclipse runs well under Linux?" "Can you really extend Eclipse so easily?" "How does the open source community create such good software for free?"

A few months later, I ran into a group member at a local Linux conference. "Does Javoorta Pro run under Linux?" I asked. "I don't use Javoorta Pro anymore. I've switched to Eclipse," he said. "That's interesting," I said. "Hey, let's go out for a beer."

An Integrated Development Environment

An *integrated development environment* (IDE) is an all-in-one tool for writing, editing, compiling, and running computer programs. And Eclipse is an excellent integrated development environment. In a sense, that's all ye need to know.

Of course, what you absolutely need to know and what's good for you to know may be two different things. You can learn all kinds of things about Java and Eclipse, and still benefit by learning more. So with that in mind, I've put together this chapter full of facts. I call it my "useful things to know about Eclipse" (my "uttkaE") chapter.

A Little Bit of History (Not Too Much)

In November 2001, IBM released $40 million worth of software tools into the public domain. Starting with this collection of tools, several organizations created a consortium of IDE providers. They called this consortium the Eclipse Foundation, Inc. Eclipse was to be "a universal tool platform — an open extensible IDE for anything and nothing in particular."* (I know, it sounds a little like Seinfeld's "nothing." But don't be lead astray. Eclipse and Seinfeld have very little in common.)

This talk about "anything and nothing in particular" reflects Eclipse's ingenious plug-in architecture. At its heart, Eclipse isn't really a Java development environment. Eclipse is just a vessel — a holder for a bunch of add-ons that form a kick-butt Java, C++, or even a COBOL development environment. Each add-on is called a *plug-in,* and the Eclipse that you normally use is composed of more than 80 useful plug-ins.

While the Eclipse Foundation was shifting into high gear, several other things were happening in the world of integrated development environments. IBM was building WebSphere Studio Application Developer (WSAD) — a big Java development environment based on Eclipse. And Sun Microsystems was

**Quoted from the eclipse.org Web site:* `www.eclipse.org`.

promoting NetBeans. Like Eclipse, NetBeans is a set of building blocks for creating Java development environments. But unlike Eclipse, NetBeans is pure Java. So a few years ago, war broke out between Eclipse people and NetBeans people. And the war continues to this day.

In 2004, the Eclipse Foundation turned itself from an industry consortium to an independent not-for-profit organization. Among other things, this meant having an Executive Director — Mike Milinkovich, formerly of Oracle Corporation. Apparently, Milinkovich is the Eclipse Foundation's only paid employee. Everybody else donates his or her time to create Eclipse — the world's most popular Java development environment.

The Grand Scheme of Things in Eclipse

The Eclipse Foundation divides its work into projects and subprojects. The projects you may hear about the most are the Eclipse project, the Eclipse Tools project, and the Eclipse Technology project.

Sure, these project names can be confusing. The "Eclipse project" is only one part of the Eclipse Foundation's work, and the "Eclipse project" is different from the "Eclipse Tools project." But bear with me. This section gives you some background on all these different projects.

And why would you ever want to know about the Eclipse Foundation's projects? Why should I bother you with details about the Foundation's administrative organization? Well, when you read about the Foundation's projects, you get a sense of the way the Eclipse software is organized. You have a better understanding of where you are and what you're doing when you use Eclipse.

The Eclipse project

The *Eclipse project* is the Eclipse Foundation's major backbone. This big Eclipse project has three subprojects — the Platform subproject, the Java Development Tools subproject, and the Plug-in Development subproject.

The Platform subproject

The *Platform subproject* deals with things that are common to all aspects of Eclipse — things such as text editing, searching, help pages, debugging, and versioning.

At the very center of the Platform subproject is the platform *core.* The core consists of the barebones necessities — the code for starting and running Eclipse, the creation and management of plug-ins, and the management of other basic program resources.

In addition, the Platform subproject defines the general look and feel of Eclipse's user interface. This user interface is based on two technologies — one that's controversial, and another that's not so controversial. The controversial technology is called SWT — the *Standard Widget Toolkit.* The not-so-controversial technology is called *JFace.*

- The Standard Widget Toolkit is a collection of basic graphical interface classes and methods, including things such as buttons, menus, labels, and events.

  For more chitchat about the Standard Widget Toolkit (and to find out why the Toolkit is so controversial), see the second half of this chapter.

- JFace is a set of higher-level graphical interface tools, including things such as wizards, viewers, and text formatters. JFace builds on the work that the Standard Widget Toolkit starts.

The Java Development Tools (JDT) subproject

The word "Java" appears more than 700 times in this book. (Yes, I counted.) In many people's minds, Eclipse is nothing more than an integrated development environment for Java. Heck, if you start running Eclipse you see the Java perspective, Java projects, Java search tools, and a bunch of other Java-specific things.

But Java is only part of the Eclipse picture. In reality, Eclipse is a language-neutral platform that happens to house a mature Java development environment. That Java development environment is a separate subproject. It's called the *Java Development Tools* (JDT) subproject. The subproject includes things like the Java compiler, Java editor enhancements, an integrated debugger, and more.

When Eclipse documentation refers to the "core," it can be referring to a number of different things. The Platform subproject has a core, and the JDT subproject has a core of its own. Before you jump to one core or another in search of information, check to see what the word "core" means in context.

The Plug-in Development Environment (PDE) subproject

Eclipse is very modular. Eclipse is nothing but a bony frame on which dozens of plug-ins have been added. Each plug-in creates a bit of functionality, and together the plug-ins make a very rich integrated development environment.

But wait! A plug-in is a piece of code, and the people who create plug-ins use development environments, too. For these plug-in creators, Eclipse is both a tool and a target. These people use Eclipse in order to write plug-ins for Eclipse.

So wouldn't it be nice to have some specialized tools for creating Eclipse plug-ins? That way, a programmer can seamlessly use Eclipse while writing code for Eclipse.

Well, whadaya' know? Someone's already thought up this specialized tools idea. They call it PDE — the *Plug-in Development Environment* — and they have an entire subproject devoted to this Plug-in Development Environment.

The Eclipse Tools project

Compared with the main Eclipse project, the *Eclipse Tools* project houses subprojects that are a bit farther from Eclipse's center. Here are some examples.

The Visual Editor subproject

If you're familiar with products like Visual Basic, then you've seen some handy drag-and-drop tools. With these tools you drag buttons, text fields, and other goodies from a palette onto a user form. To create an application's user interface, you don't describe the interface with cryptic code. Instead you draw the interface with your mouse.

In Eclipse 3.0, these drag-and-drop capabilities still aren't integrated into the main Eclipse bundle. Instead, they're a separate download. They're housed in the *Visual Editor* (VE) — a subproject of the Eclipse Tools Project.

The CDT and COBOL IDE subprojects

The *C/C++ Development Tools* (CDT) subproject develops an IDE for the C/C++ family of languages. So after downloading a plug-in, you can use Eclipse to write C++ programs.

As if the CDT isn't far enough from Java, the *COBOL IDE* subproject has its own Eclipse-based integrated development environment. (COBOL programs don't look anything like Java programs. After using Eclipse for a few years to develop Java programs, I feel really strange staring at a COBOL program in Eclipse's editor.)

The UML2 subproject

The *Unified Modeling Language* (UML) is a very popular methodology for modeling software processes. With UML diagrams, you can plan a large programming endeavor, and work your way thoughtfully from the plan to the actual code. The tricks for any integrated development environment are to help you create models, and to provide automated pathways between the models and the code. That's what *UML2* (another subproject of the Eclipse Tools project) is all about.

The Eclipse Technology project

The *Eclipse Technology project* is all about outreach — helping the rest of the world become involved in Eclipse and its endeavors. The Technology project

fosters research, educates the masses, and acts as a home for ideas that are on their way to becoming major subprojects.

As of 2004, this project's emerging technologies include *Voice Tools* — tools to work effectively with speech recognition, pronunciation, and the control of voice-driven user interfaces.

Another cool item in the Eclipse Technology project is *AspectJ.* The name AspectJ comes from two terms — aspect-oriented programming and Java. In AspectJ, you can connect similar parts of a programming project even though the parts live in separate regions of your code. AspectJ is an up-and-coming extension to the Java programming language.

What's the Best Way to Create a Window?

According to Sun Microsystems, Java is a "Write Once, Run Anywhere" programming language. This means that a Java program written on a Macintosh runs effortlessly on a Microsoft Windows or UNIX computer. That's fine for programs that deal exclusively with text, but what about windows, buttons, text fields, and all that good stuff? When it comes to using graphical interfaces, the "Write Once, Run Anywhere" philosophy comes up against some serious obstacles.

Each operating system (Windows, UNIX, or whatever) has its own idiosyncratic way of creating graphical components. A call to select text in one operating system's text field may not work at all on another operating system's text field. And when you try to translate one operating system's calls to another operating system's calls, you run into trouble. There's no good English translation for the Yiddish word *schlemiel,* and there's no good Linux translation for Microsoft's object linking and embedding calls.

When Java was first created, it came with only one set of graphical interface classes. This set of classes is called the *Abstract Windowing Toolkit* (AWT). With the AWT, you can create windows, buttons, text fields, and other nice looking things. Like any of Java's "Write Once, Run Anywhere" libraries, the AWT runs on any operating system. But the AWT has an awkward relationship with each operating system's code.

The AWT uses something called *peers.* You don't have to know exactly how peers work. All you have to know is that a peer is an extra layer of code. It's an extra layer between the AWT and a particular operating system's graphical