


Business Culinary Architecture  
Computer General Interest  
Children Life Sciences Biography  
Accounting Finance Mathematics  
History Self-Improvement Health  
Engineering Graphic Design  
Applied Sciences Psychology  
Interior Design Biology Chemistry

**WILEY**  **BOOK**

WILEY

JOSSEY-BASS

PFEIFFER

J.K.LASSER

CAPSTONE

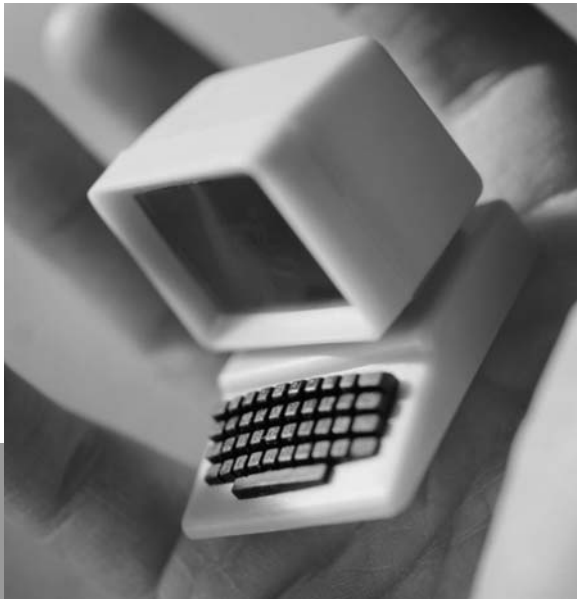
WILEY-LISS

WILEY-VCH

WILEY-INTERSCIENCE

*Professional Developer's Guide*

# Java™ 2 Micro Edition



Eric Giguère

Wiley Computer Publishing



John Wiley & Sons, Inc.

NEW YORK • CHICHESTER • WEINHEIM • BRISBANE • SINGAPORE • TORONTO

Publisher: Robert Ipsen  
Editor: Carol Long  
Managing Editor: John Atkins  
Associate New Media Editor: Brian Snapp  
Text Design & Composition: D&G Limited, LLC

Designations used by companies to distinguish their products are often claimed as trademarks. In all instances where John Wiley & Sons, Inc., is aware of a claim, the product names appear in initial capital or ALL CAPITAL LETTERS. Readers, however, should contact the appropriate companies for more complete information regarding trademarks and registration.

This book is printed on acid-free paper. ∞

Copyright © 2000 by Eric Giguere. All rights reserved.

Published by John Wiley & Sons, Inc.

Published simultaneously in Canada.

No part of this publication may be reproduced, stored in a retrieval system or transmitted in any form or by any means, electronic, mechanical, photocopying, recording, scanning or otherwise, except as permitted under Sections 107 or 108 of the 1976 United States Copyright Act, without either the prior written permission of the Publisher, or authorization through payment of the appropriate per-copy fee to the Copyright Clearance Center, 222 Rosewood Drive, Danvers, MA 01923, (978) 750-8400, fax (978) 750-4744. Requests to the Publisher for permission should be addressed to the Permissions Department, John Wiley & Sons, Inc., 605 Third Avenue, New York, NY 10158-0012, (212) 850-6011, fax (212) 850-6008, E-Mail: PERMREQ @ WILEY.COM.

This publication is designed to provide accurate and authoritative information in regard to the subject matter covered. It is sold with the understanding that the publisher is not engaged in professional services. If professional advice or other expert assistance is required, the services of a competent professional person should be sought.

***Library of Congress Cataloging-in-Publication Data:***

ISBN: 0-471-39065-8

Printed in the United States of America.

10 9 8 7 6 5 4 3 2 1



## ***Professional Developer's Guide Series***

Other titles in the series:

*Advanced Palm Programming* by Steve Mann and Ray Rischpater,  
ISBN 0-471-39087-9

*WAP Servlets* by John L. Cook, III, ISBN: 0-471-39307-X



***To Lisa, Victoria, Dino, and Taffy, with all my love.***





# Contents

---

Preface	xv
Introduction	xvii
<b>Part One: Java and Small Devices</b>	<b>1</b>
<b>Chapter 1: It Really Is a Small World After All</b>	<b>3</b>
Small Computing Devices	4
Memory and Storage Capacity	5
Processor Power	7
Input/Output Methods	8
Input Methods	8
Output Methods	12
Form Factor	13
Networking	13
Putting It All Together	15
Chapter Summary	15
<b>Chapter 2: Java: Fat and Slow?</b>	<b>17</b>
The Architecture of Java	18
Overview	18



---

The Execution Engine	19
The Virtual Machine	20
The Garbage Collector	29
The Class Loader	31
The Class Verifier	33
The Native Code Interface	33
Runtime Libraries	34
The Evolution of Java	35
Java 1.02: Client Programming	36
Applets	36
Applications	38
Java 1.1: Server Programming	39
Just-in-Time Compiling	39
Object Serialization	40
RMI	41
Database Connectivity	41
JavaBeans	42
Servlets	42
Browsers and the Java Plug-In	43
JRE	44
Java 2: Enterprise Programming	44
New Names and New Beginnings	45
One Version, Three Editions	45
Java 2 Standard Edition (J2SE)	46
Java 2 Enterprise Edition (J2EE)	47
Java 2 Micro Edition (J2ME)	47
What Is Next for Java 2?	47
The Devolution of Java	48
Chapter Summary	49
<b>Chapter 3: Programming Strategies for Small Devices</b>	<b>51</b>
If in Doubt, Do Not Use Java	51
Move Computation to the Server	52
Simplify the Application	53
Build Smaller Applications	54
Use Less Memory at Run Time	56
Use Scalar Types	56
Do Not Depend on the Garbage Collector	57
Help the Garbage Collector	57
Use Lazy Instantiation	58
Release Resources Early	59

---

Reuse Objects	59
Avoid Exceptions	61
Code with Performance in Mind	62
Use Local Variables	62
Avoid String Concatenation	63
Use Threads, but Avoid Synchronization	64
Separate the Model	65
An Introduction to MVC	65
Why Separate the Model?	66
How to Build a Model	66
The Tic-Tac-Toe Example	68
Chapter Summary	73
<b>Part Two: Java 2 Micro Edition (J2ME) Specifications</b>	<b>75</b>
<hr/>	
<b>Chapter 4: Java 2 Micro Edition (J2ME)</b>	<b>77</b>
Introducing the Micro Edition	78
A New Virtual Machine	78
New and Changed Classes	79
Configurations and Profiles	79
The KVM	81
The Spotless System	82
Early KVM Controversy	83
The KVM Today	84
Related Technologies	86
Chapter Summary	87
<b>Chapter 5: Configurations</b>	<b>89</b>
Overview	89
The Purpose of a Configuration	90
Connected Device Families	90
Dynamic Application Delivery	91
Identifying a Configuration	91
The CLDC	91
CLDC Quick Summary	92
Detailed Requirements	94
Language Support	94
Virtual Machine Support	95
Sandbox Security	96
Inherited Classes	97
CLDC-Specific Classes: Generic Connections	98

The CDC	101
CDC Quick Summary	101
Comparison to the CLDC	103
Chapter Summary	104
<b>Chapter 6: Profiles</b>	<b>105</b>
Overview	106
The Purpose of a Profile	106
How Profiles Are Used	106
Identifying a Profile	107
MIDP	107
Mobile Information Devices	108
MIDP Quick Summary	109
MIDlets and MIDlet Suites	109
What Defines a MIDlet?	110
What Defines a MIDlet Suite?	111
The Manifest	111
Application Descriptors	113
Application Lifecycle	114
User Interface Classes	115
The MIDP UI APIs and Abstract	
Windowing Toolkit (AWT)	115
Screens and Events	116
Drawing and Repainting	119
Threading Issues	120
Other MIDP Classes	120
The Record Management System	120
HTTP Connections	123
Timer Notifications	124
Other Profiles	125
The PDA Profile	125
The Foundation Profile	125
The Personal Profile	126
The RMI Profile	126
Chapter Summary	126
<b>Part Three: Java 2 Micro Edition (J2ME) Implementations</b>	<b>127</b>
<b>Chapter 7: The Connected Limited Device Configuration (CLDC) Reference Implementation</b>	<b>129</b>
Overview	129
Installation	130
Running the Samples	131

---

Compiling, Preparing, and Running Classes	134
The Hello World Application	136
The Hello World Spotlet	137
Memory Usage	139
The JAM	139
Debugging	141
Internet Resources	142
Chapter Summary	142
<b>Chapter 8: Java for Palm Connected Organizers</b>	<b>143</b>
Using the CLDC with Palm Devices	143
A Palm Primer	144
A User's Perspective	144
A Developer's Perspective	145
The Palm OS Emulator	149
Installing the Palm CLDC	150
Running the Samples	151
KVMutil	152
Revisiting the Hello World Application	153
Additional MakePalmApp Options	155
Additional APIs	157
The User Interface	158
The Spotlet Model	158
Input Events	158
Drawing	160
Controls	161
Databases	163
Understanding Palm Databases	163
Using Databases in Java	164
Network Connectivity	165
The Connector Class	165
HTTP Connections	167
The Tic-Tac-Toe Example	167
Usage	168
How It Works	168
The Automatic Player	171
Third-Party Tools and Extensions	172
kAWT	172
Color KVM	172
JBuilder Handheld Express	173
Jbed MicroEdition (CLDC)	173
Chapter Summary	175

---

<b>Chapter 9: The Mobile Information Device Profile (MIDP) Early Access Release</b>	<b>177</b>
Overview	177
Installation	178
Using the Cellular Phone Simulator	179
Running the Simulator	180
Web Server Setup	180
Example Invocations	181
Running the Samples	182
Compiling and Preparing Classes	183
The HelloMIDlet Application	184
The Tic-Tac-Toe Example	187
Chapter Summary	191
<b>Chapter 10: Java for Motorola Devices</b>	<b>193</b>
The Motorola J2ME SDK	194
Installing the SDK	194
Using the SDK	194
Using the Emulator	195
Running the Samples	196
Compiling and Preparing Classes	198
The Tic-Tac-Toe Example	198
Chapter Summary	200
<b>Chapter 11: Java for BlackBerry Wireless Handhelds</b>	<b>201</b>
A BlackBerry Primer	201
A User's Perspective	202
A Developer's Perspective	203
The BlackBerry JDE	203
Installing the JDE	203
Starting the IDE	204
Using the IDE	204
Running and Debugging Code	206
Using the Simulator	208
Running the Samples	211
RIM's J2ME Implementation	213
Overview	213
A First Application	213
A Better Application	216
Building Applications by Hand	218
The Application Model	219
Application Entry	219
Event Dispatching	220

---

Screens and Graphics	220
User-Interface Components	221
Network Communication	221
Miscellaneous Classes	222
The Tic-Tac-Toe Example	222
The TicTacToe Class	223
The TicTacToeUI Class	223
Chapter Summary	225
<b>Chapter 12: Waba: An Alternative to Java</b>	<b>227</b>
What Is Waba?	227
The Waba VM	228
The Waba Run Time	228
The Bridge Classes	229
Using Waba	230
Installation	230
Running the Samples	231
Compiling and Running Classes	233
The HelloWorld Application	235
Debugging	235
The Waba Foundation Classes	235
Overview	235
The Application Model	235
The User Interface	236
Input/Output (I/O)	236
The Tic-Tac-Toe Example	237
Chapter Summary	239
<b>Chapter 13: Final Thoughts</b>	<b>241</b>
Alternatives to J2ME	241
Conclusion	242
<b>Appendix A: Tic-Tac-Toe Source Code</b>	<b>243</b>
<b>Index</b>	<b>289</b>





# Preface

---

**W**riting a book about software is a frustrating task, because you know that by the time the book makes it into print, something about the software will have changed. This is especially true when writing about Java technology, because it evolves at Internet speeds. Rather than just listing things, I have tried to explain them. How do I use them? Where do I get them? What can I ignore? Where can I get more information about them? How do they relate to each other and to other things?

The primary purpose of this book, then, is to explain what Java 2 Micro Edition (J2ME) is and how you can use it—or *not* use it because it does not meet your needs or expectations. When you get to the end, you will be ready to start programming small computing devices in Java. You will understand how configurations and profiles work and what the KVM is and is not. You will know where to go in order to get the tools and reference implementations that you need, and you will know how to use them (which is not always obvious). And you will know where to get updates on everything discussed here: the book's Web site at [www.ericiguere.com/microjava](http://www.ericiguere.com/microjava).



Before I close this preface, I would like to make a few acknowledgements. As usual, I would like to thank my editor at John Wiley & Sons, Carol Long, for putting up with a busy part-time author who tends to push the envelope when it comes to deadlines. And I would like to thank all of the other good people at Wiley as well as my agent, Carole McClendon. Jose Lacal at Motorola and David Yach at Research In Motion also receive my thanks for allowing me to include their software on the CD-ROM. My employer, iAnywhere Solutions, a new subsidiary of Sybase, also deserves recognition for providing a wonderful work environment and a chance to explore and use new technologies while providing real solutions for our customers and for allowing me to write this kind of book. And finally, to my wife Lisa, whom I love dearly, and my baby daughter Victoria, whose birth immediately followed the “birth” of my first book, for providing a warm and fun home environment and reminding me that there is more to life than programming—no matter how much you enjoy it.

*Eric Giguère  
Waterloo, Ontario  
October 2000*



# Introduction

---

**T**his book introduces you to Java programming for hand-held and embedded systems, devices such as personal digital assistants (PDAs), cellular telephones, and so-called intelligent appliances. The focus is on the Java 2 Micro Edition (J2ME), a new Java platform from Sun Microsystems that is geared specifically toward devices that have limited memory and/or processor power (which we refer to simply as small computing devices). J2ME is still a developing platform, however, so we also will look at other options when appropriate. By reading this book, you will learn to apply your Java programming skills to an ever-increasing number of small computing devices.

## Why Java?

---

As a software developer, you choose the programming language that helps you get the job done as effectively and efficiently as possible. For desktop- and server-based applications, many developers now choose to program in Java because of the features that it offers:

**Object-orientation.** Apart from a few primitive types, everything in Java is an object. This situation forces you to decompose your application into classes and interfaces from the start, imposing a certain structure on your code. We are not saying that you cannot write bad code in an object-oriented system, but this system does require you to think a bit about what you are doing (which is the first step toward writing good code).

**Automatic garbage collection.** Java frees you from having to track who is using a specific piece of memory and when it is available for reuse by another part of the application. If nobody is referring to an object, the system automatically reclaims that object. Without garbage collection, you have to resort to schemes such as reference-counting (like the `auto_ptr` type in C++ or the `AddRef` and `Release` methods in COM) in order to ensure that an object is truly unreferenced. These schemes require careful cooperation of all users of the object, or else memory leaks or heap corruptions inevitably result. (That said, garbage collection is not a solution to all memory woes—as we will see in Chapter 3, one of your jobs as a programmer is to help the garbage collector do its job.)

**Exception handling.** Exceptions are notifications that something has gone wrong—something that might or might not be fixable—and that execution must stop until the notification is handled. Java includes support for throwing and catching exceptions.

**Portability.** Java promises that you can run the same program on any platform (write once, run anywhere) with a conforming Java interpreter and run-time library. Developers can reuse their skills and knowledge in order to target a wider audience than would otherwise be possible.

**Multithreading.** Java is thread-aware and includes built-in support for thread creation, synchronization, and notification. Care is still required to make code thread-safe, but the framework that the language provides makes it easier and more portable than using the operating system's threading capabilities (if any).

**Similarity to C++.** Java is as close to C++ as any other programming language not based on C, which makes it easier to transfer programming skills.

These features also appeal to developers who work on smaller devices. Garbage collection, for example, avoids memory leaks, which can be

disastrous in an embedded system. Some would argue that these features are even more important on these devices because they add a robustness to programs that is only achieved with careful and disciplined programming in languages such as C or C++. There is a reluctance, however, to adopt Java as a programming platform because of its perceived shortcomings:

**Java is fat.** The original Java platform, what is now called the Java 2 Standard Edition (J2SE), has grown steadily in size since it was first released. To reduce the installation footprint, the part required to run Java programs (as opposed to the part required to compile them) was split into a separate Java Runtime Environment (JRE) containing the Java interpreter, run-time classes, and associated support files. Even by itself, however, the JRE portion of J2SE is large—about 26MB on the Windows platform at this writing—and it grows as more and more classes are added to the core Java run time. The JRE is simply too large for many devices, considering that a typical Palm-connected organizer has between 2MB and 8MB of total memory. Add to that the memory that is required to load and run a Java program, and you will see why Java has been slow to make inroads as a viable programming platform on these devices. You can also make the argument that certain features of the language itself—its object orientation, exception handling, and symbolic run-time method resolution—lead to larger programs.

**Java is slow.** Java is an interpreted language, executing more slowly by one or more orders of magnitude than comparable native code. To make matters worse, a slower processor is often used in small computing devices in order to extend battery life, because slower processors use less power. Java performance is acceptable on desktop and server systems not just because they use faster processors, but also because of the run-time conversion of Java bytecodes to native code by using technologies such as HotSpot and just-in-time compilation (technologies that are not feasible on smaller systems).

**Portability is unnecessary.** The promise of write once, run anywhere appeals to developers who are writing applications that need to run on different systems, but if you are writing code for a specific embedded system or hand-held device, portability can be more of a hindrance than a help. If you cannot access the unique features of your device with the Java platform, then Java is useless. As well,

portability is often never completely achieved in any case because of bugs in the Java interpreter and in the run-time classes (write once, debug everywhere as the joke goes).

As small computing devices evolve, they will undoubtedly include more memory and faster processors—making standard Java a viable programming language on these platforms. If you have 16MB of memory available on your PDA, however, you would probably rather have most of that memory available to store addresses, images, sound recordings, and other data (as opposed to using half of it just to run your programs). The end user does not care if a program is written in Java or C; instead, his or her only concern is that the application works well and makes efficient use of the device's limited resources. From this viewpoint, Java would seem to fail.

The irony is that Oak, the early form of the Java language, was specifically designed as an embedded programming language for consumer devices. (For some interesting reading about Oak and the Green Project at Sun Microsystems, refer to the Web site <http://java.sun.com/people/jag/green/index.html>). As such, there is nothing inherent in the design of the language that limits it to usage on high-powered devices; rather, it is all a matter of implementation. The recent surge in popularity of hand-held devices is pushing Java to return to its roots as a programming language for small devices. This situation is happening in a number of ways, ranging from the simple (like writing tools in order to compact Java code) to the complex (such as reimplementing the Java interpreter).

The question remains, “Is Java usable on small computing devices?” The answer is yes, at least with J2ME. When Sun Microsystems announced J2ME at the JavaOne Conference in 1999, the major significance of the announcement was that it demonstrated a will to make Java work well on smaller platforms. It might take a while to get there, but Java is on its way to acceptance as an alternative to C or C++ for all platforms.

---

## What to Expect from This Book

---

You will obtain two things from this book: a general understanding of what makes Java programming on small devices different, and specific instructions to get you started. You will still have to download and read

the various specifications for yourself, but the book will provide you with valuable guidance to the material.

---

## Who Should Read This Book?

---

This book is written for software developers. We assume that you have a moderate level of Java programming experience. You do not need any background in hand-held or embedded systems, however, because the concepts that you need to understand are introduced as we go along. It helps, however, to have access to some device—real or simulated—that you can use for experimentation and to better understand the user experience. If you do not own a device, do not worry—we will show you how to get one or two simulators.

---

## Chapter Summaries

---

This book has 13 chapters, grouped into three parts. The chapters are best read in sequence.

### Part One: Java and Small Devices

Chapter 1, “It Really Is a Small World After All,” defines what small computing devices are all about.

Chapter 2, “Java: Fat and Slow?” looks at the architecture of Java and its evolution so that we can better understand the need for something like J2ME.

Chapter 3, “Programming Strategies for Small Devices,” discusses design and coding strategies that you can use in order to write Java programs that work on small devices.

### Part Two: J2ME Specifications

Chapter 4, “Java 2 Micro Edition (J2ME),” introduces the Micro Edition and explains how it fits into the bigger Java picture. This chapter traces the development of one of its key components, a new virtual machine called the KVM.

Chapter 5, “Configurations,” describes the two initial J2ME configurations: the Connected Device Configuration (CDC) and the Connected, Limited Device Configuration (CLDC). Configurations define basic Java language and run-time library support.

Chapter 6, “Profiles,” describes the first J2ME profile: the Mobile Information Device Profile (MIDP). This chapter also briefly describes other profiles that are currently in development. Profiles build on top of configurations by adding classes to support specific types of applications or uses of devices.

## **Part Three: J2ME Implementations**

Chapter 7, “The Connected Limited Device Configuration (CLDC) Reference Implementation,” shows you how to use the reference implementation of the CLDC on your desktop computer.

Chapter 8, “Java for Palm Connected Organizers,” shows you how to use the Palm operating system (OS) port of the CLDC reference implementation. This chapter includes a short discussion of the unique architecture of Palm devices and how to use and obtain the Palm OS Emulator so that you can try out the port even if you do not own a Palm device yourself.

Chapter 9, “The Mobile Information Device Profile (MIDP) Early Access Release,” shows you how to use the Early Access release of the MIDP reference implementation. This release includes a cellular phone simulator that lets you try your MIDP applications in a different environment than Palm.

Chapter 10, “Java for Motorola Devices,” describes Motorola’s implementation of the MIDP, the first complete MIDP implementation from a non-Sun party. A beta version of the Motorola J2ME SDK, complete with emulators for various Motorola cellular telephones, is available on the CD-ROM accompanying this book.

Chapter 11, “Java for BlackBerry Wireless Handhelds,” describes the BlackBerry Java Development Environment—another non-Sun implementation of J2ME. An early-access version of the BlackBerry JDE is available on the CD-ROM accompanying this book.

---

Chapter 12, “Waba: An Alternative to Java,” explores a similar but different take on making Java work on small computing devices.

Chapter 13, “Final Thoughts,” concludes our exploration with a few thoughts about the future of J2ME.

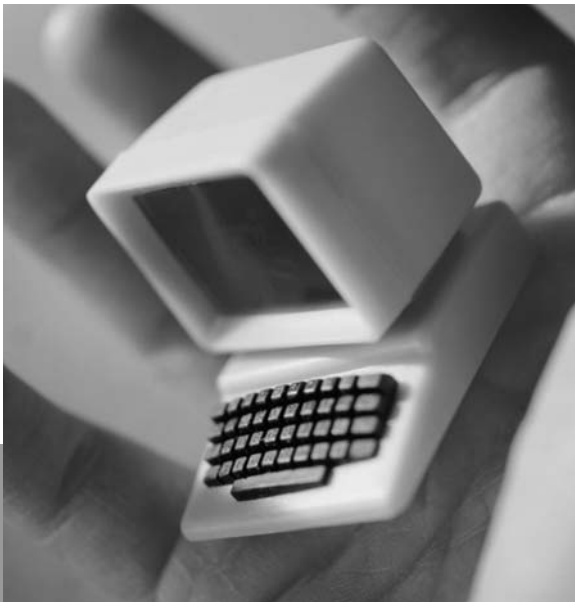
## Appendices

Appendix A, “Tic-Tac-Toe Source Code,” lists the source code for the tic-tac-toe game developed in this book.





# Java and Small Devices







# CHAPTER 1

## It Really Is a Small World After All

---

These days, it seems that smaller is becoming just as important as faster—at least, when it comes to computing. Whether it is something as extreme as nanotechnology or as commonplace as internet appliances, there is an increasing desire to embed computing technology in all facets of the human experience. The popularity of personal digital assistants (PDAs), cellular telephones, and other hand-held devices is just the first wave of this surge.

Today's small computing devices are comparable in power to the computers of five years ago. Most will always lag behind what is considered adequate for a desktop computer, however, due to space and power-consumption considerations. The challenge is to adapt programs that work well on the fastest processors and that use memory indiscriminately so that they also work on these smaller devices. In this chapter, we discuss small computing devices to see what makes them different from desktop and server computers. This knowledge prepares us for the next chapter, where we will explore how Java works and why it needs to change in order to work with these devices. We will then conclude this part of the book with a discussion of programming strategies for small devices.

## Small Computing Devices

---

First, we should define a small computing device, which is simply a computing device that has limited processor speed and/or available memory when compared to a desktop or server computer. A typical small computing device is a PDA, such as a Palm V, but the category also covers devices that do not have a conventional user interface. Some devices, for example, are meant to be embedded inside other systems—with no explicit human interaction. These devices are also of interest to us, although we will be dealing mostly with interactive devices throughout this book (because they are more general and more readily available).

Admittedly, the line separating a small computing device from the capabilities of a full-fledged desktop or server computer is rather blurry as thinner notebook computers, Web pads, and other devices come onto the market. In general, however, you can use the following rule of thumb to separate the two camps. If you never worry about the speed of your application (if it is acceptable without any extra work on your part) or about the amount of memory that it requires (within normal limits), you are not working with a small computing device.

We can now examine the characteristics that separate small computing devices from their more powerful siblings.

### **Speed Is Always Important**

---

**No matter what platform you are programming for, the speed of your application is always important. What your goal is, however, is acceptable performance—not the best possible performance. Achieving the best possible performance is usually a lower-priority task than fixing bugs or otherwise improving the application. In many cases, the difference between acceptable and best possible simply is not worth the effort that it requires. The challenge with small computing devices is achieving acceptable performance, not surpassing it.**

## Memory and Storage Capacity

---

Memory capacity is an important measuring stick, and at first glance, a small computing device might not seem that limited. You will find that it is not unusual, for example, for a modern PDA to have 4MB, 8MB, or more of memory. While this amount is less than the 64MB or 128MB that you will get in a typical desktop system, it is not a debilitating amount. Programs can fit comfortably in the 0MB-to-4MB range, regardless of the platform. Focusing on memory capacity is rather deceiving, however, because what we need to focus on is total storage capacity (which is a much more important number).

Total storage capacity is the sum of a device's online and offline storage capacity. Online storage capacity is simply another term for memory capacity—the amount of memory available to store runtime application data (stack, threading information, and heap), system data, built-in applications, and the operating system. Online storage is characterized by instant availability and might or might not be persistent. Offline storage capacity, on the other hand, is the capacity of secondary, persistent storage modules such as hard disks or memory sticks. Offline storage is usually slower to access and is not normally available for storing runtime application data.

Compare the memory capacity of a desktop computer with 64MB of random access memory (RAM) and a Palm Vx with 8MB of RAM, as illustrated in Figure 1.1. The difference is not that pronounced. Now compare the total storage capacity of the two devices. Say that the desktop computer includes a 10GB hard drive for offline storage. Figure 1.2 demonstrates the difference between the two quite dramatically: the desktop computer's capacity is roughly 10GB (we can effectively ignore the 64MB of RAM for this calculation), and the Palm Vx's still stands at 8MB. The desktop computer has almost infinite storage capacity when compared to the basic Palm Vx.

Total storage capacity is not always a distinguishing feature, however. Consider the IBM Microdrive, for example—a 340MB storage device that can be plugged into any device that supports the CompactFlash Type II expansion slot. In other words, you can use it with PDAs such as the Hewlett-Packard Jornada 430se or the Psion Series 7. While