Java ME on Symbian OS

Roy Ben Hayun

With
Sam Mason, Daniel Rocha, and Ivan Litovski, assisted by Sam Cartwright

Reviewed by
Gavin Arrowsmith, Brendan Donegan, Erik Jacobson, Martin de Jode, Mark Shackman, Jo Stichbury

Head of Symbian Press
Freddie Gjertsen

Managing Editor
Satu McNabb

A John Wiley and Sons, Ltd., Publication

Java ME on Symbian OS

Inside the Smartphone Model

Java ME on Symbian OS

Roy Ben Hayun

With
Sam Mason, Daniel Rocha, and Ivan Litovski, assisted by Sam Cartwright

Reviewed by
Gavin Arrowsmith, Brendan Donegan, Erik Jacobson, Martin de Jode, Mark Shackman, Jo Stichbury

Head of Symbian Press
Freddie Gjertsen

Managing Editor
Satu McNabb

A John Wiley and Sons, Ltd., Publication

John Wiley & Sons Ltd, The Atrium, Southern Gate, Chichester,
West Sussex PO19 8SQ, England

Telephone (+44) 1243 779777

Email (for orders and customer service enquiries): cs-books@wiley.com
Visit our Home Page on www.wiley.com

Other Wiley Editorial Offices

John Wiley & Sons Inc., 111 River Street, Hoboken, NJ 07030, USA

Jossey-Bass, 989 Market Street, San Francisco, CA 94103-1741, USA

Wiley-VCH Verlag GmbH, Boschstr. 12, D-69469 Weinheim, Germany

John Wiley & Sons Australia Ltd, 42 McDougall Street, Milton, Queensland 4064, Australia

John Wiley & Sons (Asia) Pte Ltd, 2 Clementi Loop #02-01, Jin Xing Distripark, Singapore 129809

John Wiley & Sons Canada Ltd, 6045 Freemont Blvd, Mississauga, Ontario, L5R 4J3, Canada

Wiley also publishes its books in a variety of electronic formats. Some content that appears in print may not be available in electronic books.

Library of Congress Cataloging-in-Publication Data

Hayun, Roy Ben.
Java ME on Symbian OS : inside the smartphone model / Roy Ben Hayun, with Sam Mason ... [et al.].
p. cm.
Includes bibliographical references and index.
ISBN 978-0-470-74318-8 (pbk. : alk. paper) 1. Smartphones–Programming. 2. Java (Computer program language) 3. Symbian OS (Computer file) I. Title.
TK6570.M6H33 2009
005.26'8–dc22

2008053889

British Library Cataloguing in Publication Data

A catalogue record for this book is available from the British Library

ISBN 978-0-47074318-8

Typeset in 10/12 Optima by Laserwords Private Limited, Chennai, India
Printed and bound in Great Britain by Bell & Bain, Glasgow
This book is printed on acid-free paper responsibly manufactured from sustainable forestry in which at least two trees are planted for each one used for paper production.

Contents

Foreword

James Coplien

It was great re-discovering through this book just how much of a nerd I am. Roy's descriptions and exercises painted amazing pictures in my head of the internal workings of my own Nokia S60 phone, to the extent that I unearthed new and useful menus previously invisible to me. While I can't promise that every casual reader will find this book suitable as a user's guide, and while I can't promise that every programmer will become an expert user of their own handset by reading the chapters that follow, I can promise the intent programmer a treasure chest of knowledge not only for understanding what goes on inside your phone but also for changing what goes on inside your phone.

Java ME on Symbian OS is a title that sounds like nerd's heaven – and it is (though it's more than that, as I'll come back to below). Though the title bears two trade names, the meat of the book proudly displays its inclusiveness across the market: Sun, Symbian, Nokia, Samsung, LG, Sony-Ericsson, Motorola . . . It's the ultimate mash-up. This is a book less about niche technologies than about a broad phenomenon. It isn't only a technological phenomenon but a sociological one; if you don't view it that way today, I believe that vision is rapidly approaching tomorrow.

To the everyday programmer, I think this book will open a new world of wonder, as history has similarly favored our industry several times in the past. Twenty years ago, someone would have thought you to be drunk if you asked him to take a picture of you with his phone. Today we have lost touch with photographic film and, to a large degree, with those ancillary devices called cameras. We used to have a shelf-full of gadgets for voice dictation on the run, digital photography, calendar management, and, yes, even communication – all of which are

converging on single small, convenient computers that are the brain-grand-grand-grand-grand-children of Alexander Graham Bell. And those boxes need software – software such as you will find in this book, along with guidance for bringing it to life.

It was inevitable that the evolution of consumer programming should turn to the telephone. I remember the visions of unification of computers and telephones in the minds of people such as Jerry Johnson back in Bell Lab in the 1970s. We can now look back at the past 20 years and conclude that computers' primary contribution to society has been not as calculating engines in their own right, but as tiny points on an almost fully-connected network that ties together most computers in the world. The phone network is more than just a metaphor for the worldwide web: it is often its backbone. As the pundits predicted, the intelligence has evolved out of the network into its periphery, out of its bowels and into its eyes, ears, and fingers.

The intelligence hasn't stopped at phones. Next year, Nokia and RWE are expected to market a box that extends the phone network to control heating, window shades, and home security cameras. The possibilities are endless. Programmable phones will replace not only cameras and small dictation machines, but will reach ever further into both our immediate and physically remote realities. They have already made a significant foray into the video game space, taking advantage of advancing display technology, complementing our newfound connections with reality with newfound escapes into fantasy. In the not so distant future, these two worlds may blend and merge. Whereas yesterday's play was a personal Tamagotchi and today's play includes multiple Tamagotchis talking via an infrared Tamacom protocol, tomorrow will see our phones as a Tamagotchi interface whereby we feed or clean up after the flesh-and-blood Fido back home – all with a sequence of a few DTMF tones.

Roy's book takes us on a journey into a land that is more than just nerd heaven. The history of computing has drawn more and more everyday folk into programming. This trend sits squarely in the center of the vision of people such as Alan Kay and Adele Goldberg: that computers should be the sidekicks of human beings from early life, enabling children of all ages to extend their memory and processing power by manufactured, rather than biological, means. This vision of a Dynabook – a truly personal computer, an extension of self – includes information tethered to the outside world as one of its central building blocks. Because my Dynabook is part of me, I get to program it. Today, that programming means setting a date on my calendar or setting an alarm for a meeting. Those humble acts may not entail using Java, but

it's still programming: as Dan Barstow used to say, "No matter how high-level, it's still programming." To me, as a C++ programmer, Java looks almost like a scripting language. And that's not even an insult. Java has perhaps finally realized its vision of ubiquitous expression of general-purpose computation that is portable and intelligible enough that Everyperson can at least tweak the code. I know that such æsthetics are difficult to judge from the seat of a professional programmer, but we can at least say that Java code provides a glimpse of such a world. Perhaps even my neighbor, the real-estate salesman, could read some of the code in this book and understand it well enough to play with it. Or my dentist. Or my banker. None of them are professional programmers. But each has a computer on his or her desk: a machine which 40 years ago would have terrified most secretaries, and which still terrifies some executives today. Machine creatures inhabit more and more of the human ecosystem.

It's interesting to note that such programmability has played out not in desktop phones (yes, they still exist) but in hand-held portable phones. Why? Perhaps it is because only personal cellular phones are close enough to our heart – or some other part of our anatomy – to truly be part of Self. It was a revolution when people who bought the early Ataris and Amigas discovered that they could actually program them themselves. We are perhaps on the threshold of a revolution that unleashes more powerful programming into the hands of what will be over one billion owners of Symbian OS phones in two or three years. Even if great programmers arise from only one in a million of these users of small-handheld computers that take pictures, play games, and even place phone calls, it will be enough to move the world.

This book, as a nerd's text, is nonetheless a bold foray into Dynabook territory. If nothing else, it will make you realize that if you own a regular personal computer and a mobile phone, that it's within your reach – today – to write software that can be an at-hand extension of yourself any time, anywhere. It is perhaps one small step on the road to a future vision of even further integration between man and machine. Such integration must go forward with intense ethical scrutiny and social care; if we choose such a path, machines can make us even more human, humane, and social than we are today.

I leave you with a poem which, when published, was wrongly attributed to me alone. It was written by the attendees at VikingPLoP 2004 as we took the human–computer symbiosis to its limit. Think of *Java ME on Symbian OS* as an important step on this journey. Nerds, children of all ages, and everybody out there: Happy Programming!

Comfortable as blue jeans

She fits.
Part of me, yet not me;
Unplugged: oxygen is her food
And the breeze her power adaptor.

Her identity mine, yet not me:
  I put her down, I take her up
  like a well-worn wallet plump
  with life's means and memories.
A thin mask worn in life's play
  And both of us are players.
I call her Self; she does not call me User
A curse unique to programmers and other pushers.
I and my computer are one
But we own each other not.

Envision this: my new machine, a soulmate,
With whom I talk in whispers unencumbered by flesh and bone
Software begotten, not made
Of one being with its person.
She hears my voice, and perhaps my thoughts
rather than the drumming of my fingers
that burdened her forebears:
35 calcium levers linking brain to digits,
slow and tired machines of the information age.
Now, mind to mind,
there for tea and sympathy
all day, and our night
minds together process the day,
sharing dreams – yet
She slumbers not, nor sleeps.

An invisible mask the birthright of all humanity;
Her software penned by
  My soul
And like my soul, her hardware
Im-mortal, invisible
  Us together wise.
Transparent personæ mine, yet not me,
That with me loves, and lives, and walks life's journey –
A path both real and virtual at once
  as only a True and rich life is.

With me she dies
– our immortality inscribed in
the fabric of the network.
No duality of community here
and Internet there –
The Network is the Com-
munity:

One Web of life.

Jim Coplien
Mørdrup, Denmark
2 July 2008
en.wikipedia.org/wiki/James_O._Coplien

Kicking Butt with Java Technology on Symbian OS

Mobile phones have changed the lives of billions of people and it's clear that the revolution is just beginning. The quickly evolving world of mobile technologies is an exciting place for developers to create new applications to connect people and information.

Java technology has enjoyed dizzying success in just under a decade in the mobile phone industry. As I write, more than 2.1 billion Java technology devices are deployed worldwide. At the same time, Symbian OS dominates the fast-growing smartphone market (with approximately 70%). This book is about the confluence of these two technologies and how to create your own powerful mobile applications.

Now is a great time to be a mobile developer. You can help shape the coming landscape of advanced mobile applications by harnessing the power of both Java technology and Symbian OS. Learn the technologies, let your imagination run free, and have fun!

Jonathan Knudsen
jonathanknudsen.com
Author of *Kicking Butt with MIDP and MSA*

About This Book

In 2001, Symbian published its first book devoted to Java on Symbian OS. Jonathan Allin's *Wireless Java for Symbian Devices* provided an in-depth exposition targeted at programming PersonalJava on Symbian OS. In 2004, Martin de Jode's *Programming Java 2 Micro Edition on Symbian OS* focused on programming MIDP, particularly MIDP 2.0 on Symbian OS. This book is the third in the series and there is no longer a need to explain what Java is. Therefore, the primary goal of this book is not to teach Java programming but to introduce you specifically to Java Platform, Micro Edition (Java ME) on Symbian OS.

This book covers various topics from different perspectives. Our approach has been to start with the basics and prerequisites that are assumed in the rest of the book, then zoom into Java ME on Symbian OS specifically and then out again to discuss a few key areas of Java ME development today. We finish the book by delving deep down to expose what's happening 'under the hood'.

The book logically divides into four main parts:

1. Introduction to Java ME and programming fundamentals

   In Chapter 1, we describe the playground of Java ME on Symbian OS. It is recommended reading if you are not familiar with Java ME, Symbian OS or the smartphone industry. Chapter 2 is targeted at developers with no experience in Java ME. It sets the baseline information that is a prerequisite for reading the rest of the book, which assumes a certain level of knowledge of Java programming and, specifically, Java ME. Chapter 2 deals with the basic concepts of a MIDlet, LCDUI, MMAPI, and so on.

2. Java ME on Symbian OS

   Part 2 assumes you are familiar with Java ME but are interested in learning about the Java ME platform hosted on Symbian OS. Chapter 3 explains what makes Java ME on Symbian OS different from other Java ME platforms; what makes it more powerful; and what else is unique to Symbian smartphones. Chapter 4 provides guidelines on how to handle differences between different Symbian smartphones. Chapter 5 provides an overview of development SDKs for Java ME on Symbian OS.

3. Drill down into MSA, DoJa and MIDP game development

   Part 3 expands the discussion into various key areas of Java ME development. We show how to leverage the power of MSA, give an introduction to how Java evolved very differently in the Japanese market, provide a glimpse of games development and equip you with best practices for your next application.

4. Under the hood of Java ME

   In Part 4, we reveal information that has rarely been exposed before. Chapter 10 explains the internal architecture of the Java ME subsystem as a Symbian OS component and Chapter 11 explains how Java ME on Symbian OS is tightly integrated with the native services.

Appendix A introduces WidSets which is a run-time environment that makes use of the Java language and the Java ME platform. Appendix B introduces SNAP Mobile for game developers.

This book contains a number of web addresses. If any of them are broken, please consult the wiki page for this book at ***developer.symbian.com/javameonsymbianos*** to find the updated location.

We welcome beginners and professionals, third-party application developers and OEM developers working inside Symbian OS. Whether you're just starting out in the exciting world of mobile development, want to learn Java ME or just like reading interesting stuff, this book is a mix of different topics that combine in the same way as a dish full of spices.

Author Biographies

Roy Ben Hayun

Roy Ben Hayun has more than 10 years' experience in various Java technologies in various software engineering roles: engineer, consultant, team lead, tech lead, and architect. He started his career in Lotus IBM, working on JDK 1.1.8, and then became a founding member of eMikolo networks, which pioneered P2P in the early days of Project JXTA. He later worked in various roles in other startup companies in Java SE, Java ME and Java EE. From 2002 until the end of 2007, he worked for Symbian, mostly in the Java group working on the design and implementation of JSRs and CLDC-HI VM tools support. In his last year at Symbian, he worked in the team supporting developers, during which time he authored a number of technical articles and white papers about run-time environments.

After JavaOne 2007, Roy joined Sun Microsystems. He is currently working as a system architect in the Engineering Services group, which leads the development, marketing and productizing of Java ME CLDC and CDC on various platforms.

Ivan Litovski

Ivan joined Symbian in 2002, working first in the Java Team and later with the System Libraries, Persistent Data Services and Product Creation Tools teams. Ivan has 11 years of professional experience with Java, embedded software, information security, networking, scalable servers and Internet protocols. He enjoys solving difficult problems and research and development. He has authored several patents and papers in peer-reviewed,

international journals. He holds a BSc in Electronics and Telecommunications from the University of Nis, Serbia and is an Accredited Symbian Developer.

Sam Mason

Sam spent about two years working on a Java-based, multi-lingual, video-on-demand system for SingTel and other Asia-Pacific Economic Cooperation (APEC) telecommunications agencies. He has spent most of the last four years working with and learning about mobile phone technologies, while picking up a couple of Java certificates and completing a Master of Information Technology (Autonomous Systems) at the University of New South Wales along the way. He was a contributing author to *Games on Symbian OS*, writing the chapters on Java ME development and DoJa. He is an Accredited Symbian Developer.

Daniel Rocha

Daniel is a software engineer with 10 years' experience in application development, having worked with web (Perl, ASP, PHP, JavaScript, JSP, Flash), enterprise (Java EE) and mobile software (Symbian C++, Java ME, Flash Lite, Python). He currently works as a Forum Nokia Technology Expert.

Author's Acknowledgements

I feel vividly that this project was an amazing experience, primarily because of the people involved. I am blessed that there are so many people to thank – editors, contributing authors, reviewers, test readers and colleagues.

Two of the people whom I had the honor to be guided by, Jo Stichbury and Jim Coplien, I met for the first time on the same day. The place was the ACCU conference in 2006 and the idea of writing the next book about Java ME for Symbian OS was too embryonic to be called 'an idea'. Jo Stichbury (who had just returned from Nokia to Symbian) knew from her own authoring experience how to deliver a book from start to finish, and I am thankful for her personal guidance, her professionalism and wisdom as a technical editor, which has always managed to find the right solution at any time and on any matter.

I am also thankful for meeting Jim Coplien, a truly visionary guy. Full of passion, equipped with his one-of-a-kind way of thinking, he has that rare long memory into the past of software engineering and a vision into the future of what will come out of all that we are currently doing.

I am also in debt to the wonderful people who have contributed from their knowledge and experience: Ivan Litovski, Sam Mason and Daniel Rocha. Ivan has been a great friend from day one in the Java group at Symbian, and is a reviewer and contributing author to this book. Sam is a passionate mobile expert and the founder of Mobile Intelligence in Australia. I have been lucky enough to work with Sam on this book and to meet and become friends at JavaOne 2008. Daniel has constantly given this project wonderful support and his contributions, from his viewpoint inside Forum Nokia, have been invaluable. He came to the rescue on many occasions, and I'm truly appreciative of his hard work and energy.

I'd also like to thank Mark Shackman for the pleasure of working with him during my four and a half years in Symbian. I am grateful for his constant insistence on the build up of logical arguments and perfect phrasing. Mark always delivers, and he manages to get the best out of hundreds (by now probably thousands!) of embryonic articles and book chapters.

Thanks also to Martin de Jode for his valuable comments from his own experience in writing the second book in this series and much gratitude to Satu McNabb for her support and for always being there to help.

I'd also like to acknowledge:

- the people at John Wiley & Sons for working with us on this book with their high standards of professionalism and quality
- my current group, Engineering Services in CSG, Sun Microsystems, which had the ability to see the future of Java from the remote distance of the 1990s and leads the way towards the next generation of Java technologies
- the Java group in Symbian, of which I had the pleasure to be a member during most of my time in London
- the Developers' Consulting team and the Developer Product Marketing team in Symbian
- the many people who took time and patience to read all the suggestions and ideas that were yet to become chapters: Arnon Klein, Assaf Yavnai, Yevgeny (Eugene) Jorov, Jonathan Knudsen, Tomas Brandalik, Erik Hellman, Per Revesjö, Magnus Helgstrand, Lars Lundqvist, David Mery, and Emil Andreas Siemes
- my parents and sister and all of my friends.

Finally, this book is for my friend, partner and wife – Gabi – and our wonderful, beloved children, Noa and Ariel.

For being blessed with everything that has happened and the things that are yet to happen.

I hope you will enjoy reading this book.

Symbian Press Acknowledgements

Symbian Press would like to thank Roy, Sam and Daniel for the hard work and energy they've put into this book. Roy piloted this project with particular passion, and the results you hold before you are testament to his efforts and those he inspired in his co-authors. We were privileged indeed to put together such a team of talented writers and developers, and they in turn were fortunate to be able to draw on an extensive and experienced support crew, including Sam Cartwright, Martin de Jode, Ivan Litovski and Mark Shackman.

We'd like to thank everyone involved in writing and reviewing the text of this book and in helping prepare the final manuscript. As usual, we're grateful to the Wiley team (Birgit, Colleen and Claire), and to Sally Tickner and Shena Deuchars.

In particular, Jo would like to thank Antony and Bruce for supporting the completion of this title.

Part One

Introduction to Java ME and Programming Fundamentals

1

Introduction to Java ME, Symbian OS and Smartphones

Symbian OS is the operating system that powers more than 70% of smartphones worldwide. In addition to providing one of the industry's most powerful native platforms for after-market applications, Symbian OS pushes the boundary further by allowing third-party software developers to work with a wide variety of mobile technologies including Symbian C++, Flash Lite, Python, POSIX-compliant C and, of course, Java ME – the focus of this book.

The Symbian OS ecosystem has flourished over the last decade and, like any success story, is made up of many parts. Handset manufacturers, such as Nokia, use Symbian OS as the foundation stone of their user-interface platforms. In Japan alone, Symbian OS powers over 30 million phones that use NTT DoCoMo's MOAP platform. The latest version, Symbian OS v9, is used in almost all of Nokia's famous Nseries (feature-focused) and Eseries (business-focused) devices.

In this chapter, we explore the Java Platform, Micro Edition (Java ME) technologies and see how they are uniquely positioned to address the on-going mobile phone software revolution. We start with a look at recent history, follow with a brief discussion of the Java language itself, the composition of Java ME and the benefits it inherits from Symbian OS. Finally, we wrap up with a look at the mobile phone market and get a glimpse of what the future holds.

1.1 2003: Rise of the Mobile

One of the problems with turning points in history is that they're often only apparent in retrospect. Don't expect to see Monday 16 June 2003 in any documentary about days that changed the world because it's not there – I checked. However, by the end of that day, a startling new

concept in mobile device hardware had been released, ushering in an era of exponential progress in mobile computing.

On that rather special Monday, Nokia released a completely new type of mobile phone. The Nokia 6600 was the most advanced mobile phone anyone had seen, and it is still remembered today as a clear indication of where the future was headed. It had a large, 16-bit color screen, a four-way joystick for navigation and games, and a VGA, 640 × 480 camera for video recording and photography. The phone was a 2.5 G, Internet-enabled GPRS phone running Symbian OS v7.0 s with 6 MB of memory for storage, 3 MB RAM and had Bluetooth wireless technology connectivity. It was also easy to use and looked great at the time (it looks like a bit of brick today, as you can see from Figure 1.1).

Figure 1.1 Nokia 6600

This new device was clearly going to cause quite a stir, so the Nokia marketing team came up with a special phrase just for the launch – something that was prophetic and sounded really cool: Image is Power.

Less important at the time but far more relevant to us today, the Nokia 6600 was the first mobile phone to come with the very latest in Java technology – the Mobile Information Device Profile (MIDP) 2.0, which had recently been finalized. Java ME was already dominating the mobile software market; it was about to take a huge leap forward and the Nokia 6600 was the springboard. Even today, I keep my Nokia 6600 in a special place and consider it with reverence.

In 2003, while the rest of the software industry was dragging itself out of the Dot-com Bust, the constrained-device development sector was at the start of a vertical rise. Hundreds of business applications had already been written and shipped in expectation of an amazing shift in the way people did business. The momentum for mobile hardware and software

built over the years following the release of the Psion Organiser 1 in 1984 and, by 2003, millions of portable information devices had sold around the world. These included pagers, iPods and generic MP3 players, personal organizers (both the original handheld computers and the latest PDAs), *Game & Watch*[1] devices and, of course, mobile phones.

People were getting busier and more 'connected'. Business was becoming more demanding, requiring decisions faster, for higher stakes, and 24 hours per day. It was an exciting time and mobile technology, it seemed, had all the answers. Except that it didn't work out that way.

What the oracles didn't predict was that while there would be a worldwide explosion in the use of mobile phones, they would largely be used to send text messages, buy ringtones and wallpapers, and take photos or video. People didn't buy them to do business after all. They bought them as personal statements and as fashion accessories. They bought them as a convenient communications device. They also bought games. Quite a lot of games, actually. Thousands of titles were developed and sold directly to the public or shipped on devices. The market for mobile phone games quadrupled between 2002 and 2003 and it is estimated that over 50 million phones that allow after-market game installation were shipped in 2002 alone.

Fast forward five years to 2007 and there are nearly three billion mobile phone subscribers worldwide.[2] The mobile phone has now become the personal information accessory of choice – everyone is 'texting' using appalling new pseudo-grammars (wch is gr8 if u cn do it), crowded buses sound like call centers, the mobile games market has exploded into the public eye with huge titles, such as ***The Sims 2***,[3] and in many parts of the world a new kind of digital life has emerged and become the norm. The newest devices are a far cry from the Nokia 6600. January 2007 saw the iPhone launched into the marketplace, followed closely by the Nokia N95 in March of the same year with the slogan 'It's what computers have become'. In October 2008, the T-Mobile G1 became the first smartphone to use the Android platform. Clearly innovation and demand caused many changes in just those five years!

Let's look at some of the figures to put this into perspective – in 2007, mobile handset annual sales exceeded 1 billion units for the first time. That means that more than 2.7 million phones were sold *each day*. Moreover, sales of Nokia handsets alone in 2007 were larger than the entire industry in 2001 and 2002! Figure 1.2 charts annual sales and shows how the market almost tripled over a six-year period, with an annual growth rate of nearly 20%.

---

[1] If you missed the 1980s, find out about the *Game & Watch* craze at ***en.wikipedia.org/wiki/Game_%26_Watch***

[2] Global cell phone penetration reached 50% in November 2007, according to ***www.reuters.com/article/technologyNews/idUSL2917209520071129***

[3] ***www.thesims2.com***

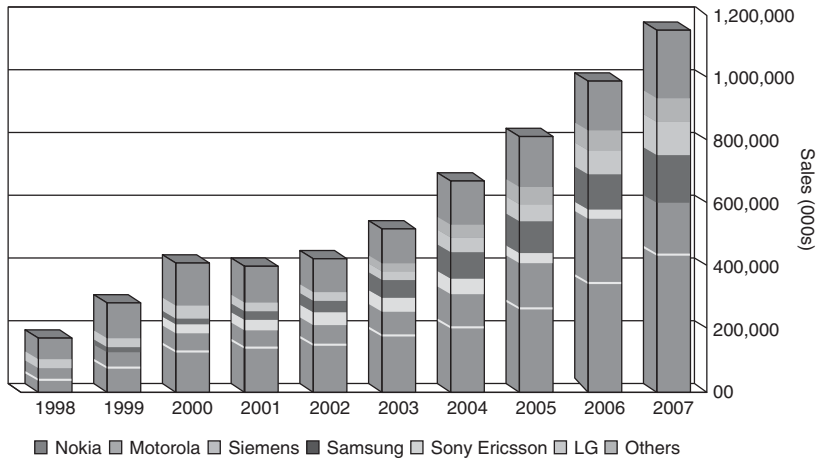

Figure 1.2 Annual number of devices sold[4]

Add five more years and it is 2012. At current growth rates, it is estimated that smartphones will comprise nearly 30% of the market (compared to 10% in 2007) and that subscriber levels worldwide will reach 3.5 billion. Just think about that number – we can't really conceive figures like this in our minds, but this is a *really* big market. Eight out of ten devices support some form of Java-based technology – quite an opportunity for the budding technically-minded entrepreneur. If you sell every thousandth person a $1 Java ME MIDlet, pretty soon you'll have made over $3 million – job done, time to retire.

1.2 2008: Mobile Generation

Our world has changed. Like all great upheavals, the 'rise of the mobile' has had both positive and negative effects on us. As a society, we're irrevocably addicted to our information devices now. Everywhere we look we see iPod MP3 players, Tom-Tom personal navigation devices, mass-market feature phones, smartphones, managers who can't get off their Blackberries, people talking while walking, people setting up meetings, messaging, listening to music, reading, using devices for entertainment, organization, romance, learning. It goes on and on.

Mobile information devices are the instruments of an unprecedented wave of fundamental social change. The very basis upon which we interact with each other has suddenly changed without notice and the results are not necessarily all good. SMS bullying is on the rise; onboard cameras have changed our expectations of personal privacy; train carriages are

[4] ***java.sun.com/developer/technicalArticles/javame/mobilemarket***

like mobile offices; and I've yet to go to a funeral where someone's mobile doesn't ring during the eulogy. It's fashionable to talk about people being more 'connected', as I did above, but a world full of people with iPods and earphones can feel like an increasingly isolated one. Even our basic standards of interaction are affected – people think nothing of sending text messages while talking to you and the legalities involved with using evidence from mobile phones are still being ironed out.

On the positive side though, the plethora of lifestyle applications and services available on mobile phones now definitely makes life a lot more interesting. GPS and navigation applications make it hard to get lost in most places in the world and you can send goodnight kisses to your kids via MMS from wherever you end up! Messaging is by nature asynchronous so you don't *have* to deal with inter-personal communication immediately any more, which allows you to manage your time better. And if you have a two-hour train trip to work why not listen to a podcast on some topic of interest instead of reading the newspaper?

I was thinking the other day about the phrase 'Information is power' and about mobile phones, games, Symbian OS, business applications, Nokia, the future, mobile phones (again), robots, Linux and just general technology. As sometimes happens, I had an acronym attack and realized that information is indeed POWER because it affords us Pleasure, Opportunity, Wealth, Experience and Reputation. To me, this acronym spells out five very good reasons to start learning Java ME today. If you're not yet convinced, stick with me, since the rest of this chapter, and the book, considers the use of Java technology, specifically the Java ME platform, as the candidate to address the growing hardware diversity and the challenges of the future.

1.3 Meet the Host – Symbian OS

Symbian OS is a market-leading, open operating system for mobile phones. Cumulative sales of Symbian smartphones reached 226 million units worldwide in June 2008. By the end of that month, more than 250 different models of smartphone had been created by handset manufacturers licensing Symbian OS; eight handset vendors, including the world's five leading vendors, were shipping 159 mobile phone models.[5]

On 24 June 2008, the Symbian Foundation was announced[6] and, at the time of writing, it is on track to create an open and complete mobile software platform, which will be available for free, enabling the whole mobile ecosystem to accelerate innovation. The Symbian Foundation provides, manages and unifies Symbian OS, S60, MOAP(S) and UIQ for its members. It is committed to moving the platform

[5] ***www.symbian.com/about/fast.asp***

[6] ***www.symbian.com/news/pr/2008/pr200810018.asp***

to open source within two years of the Foundation's announcement. See ***www.symbianfoundation.org*** for more information.

There have been many operating systems available over the last decade – so what is it about Symbian OS that has enabled it to stay so far ahead of the competition? The design of Symbian OS is highly modular, which means that it is constructed from well-defined, discrete parts. Most Symbian OS components expose a Symbian C++ API[7] and a large number of these APIs are available to third-party application developers.

At a high level, Symbian OS can be thought of as a layered model, with a flexible architecture that allows different UI layers to run as platforms on top of the core operating system. That is, Symbian OS provides a framework that supplies a common core to enable custom components to be developed on top of it. The generic UI framework of Symbian OS supplies the common behavior of the UI and supports extension by licensees who define their own look and feel. For example, the S60 platform, supplied by Nokia, sits on top of Symbian OS. S60 is the world's most popular smartphone platform, and is currently used by Nokia, LG and Samsung. Almost 180 million S60 devices had been shipped by S60 licensees by the end of June 2008.[8]

The power and configurable design of Symbian OS are the key contributing factors to its success. Today, Symbian OS devices are being sold in every market segment as you can see from the variety of hardware shown in Figure 1.3. If you want to find out more about Symbian and Symbian OS, there are a range of resources including books from Symbian Press, available from ***developer.symbian.com/books***, and ***www.forum.nokia.com***. If you are new to Symbian OS and interested in a slightly more detailed overview about Symbian OS, a recommended resource is the first chapter of [Babin 2007].[9]

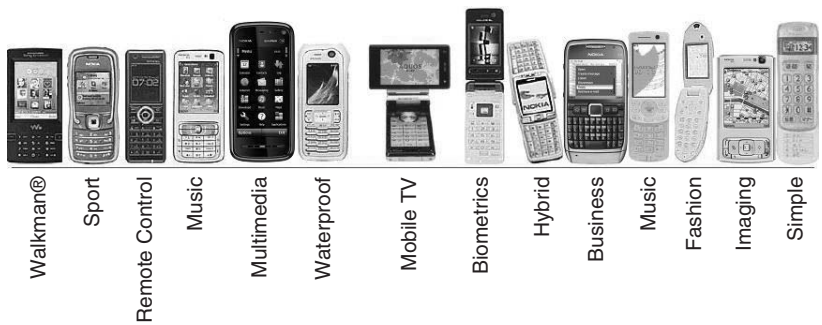

Figure 1.3 Symbian OS is everywhere

[7] Symbian C++ is a dialect of C++ with idiomatic extensions.

[8] ***www.forum.nokia.com/main/platforms/s60***

[9] The chapter can be downloaded from ***developer.symbian.com/main/documentation/books/books_files/dasos***.

1.4 What Is Java?

Let's jump back to August 1998. The word 'Java' is synonymous with the Internet in the public mind. The most common question at the time is 'What is Java?' If you can explain that then you have a job.

The company behind Java technology is Sun Microsystems, which was founded in 1982 and today provides a diversity of software, systems, services, and microelectronics that power everything from consumer electronics, to developer tools and the world's most powerful data centers. Other than Java technology, the core brands include the Solaris operating system, the MySQL database management system, StorageTek[10] and the UltraSPARC processor.

In the early 1990s, Sun formed the 'Green Team' whose job it was to prepare for the future. One of the team members was James Gosling (known as 'the father of Java'). The aim was to create a software platform that could run independently of the host platform – and the 'Write Once Run Anywhere' mantra was created. By creating a controlled virtual environment, or machine, software written on one hardware platform could run unaltered on any other hardware platform.

The Java programming language was introduced by Sun Microsystems in 1995 and is designed for use in the distributed environment of the Internet. It was designed to have familiar notation, similar to C++, but with greater simplicity. The Java programming language can be characterized by:

- Portability: Java applications are capable of executing on a variety of hardware architectures and operating systems.
- Robustness: Unlike programs written in C++, Java instructions cannot cause a system to 'crash'.
- Security: Features were designed into the language and run-time system.
- Object orientation: With the exception of primitive types, everything in Java is an object in order to support the encapsulation and message-passing paradigms of object-based software.
- Multithreading and synchronization: Java supports multithreading and has synchronization primitives built into the language.
- Simplicity and familiarity: This is, of course, relative to C++. Don't imagine that Java can be learned in a day.

Java is a technology rather than just a programming language. It is a collection of software products and specifications that provide a system

[10] StorageTek is a tape-based storage solution from Sun. See ***www.sun.com/storagetek*** for more information.

for development and deployment of cross-platform applications. After you've worked with Java technologies for a while, you'll realize that they're also much more than that – they're a way of thinking about software systems and how they work together.

Java is an interpreted language. This means that program code (in a `.java` source file) is compiled into a platform-independent set of bytecodes (a `.class` file) instead of into machine-specific instructions (which is what happens to a C program). These bytecodes are then executed on any machine that hosts a Java platform, a software-only platform that runs on top of a native platform (the combination of an operating system and the underlying hardware). A Java platform is divided into two components:

- The Java virtual machine (JVM) that executes the Java language bytecode can be ported to various native platforms.
- The collection of Java APIs are ready-made software libraries that provide programming capabilities.

Together, the JVM and the APIs form the Java platform which insulates an application from the native platform.

This was a remarkable achievement but it was not achieved without cost. Support for low-level power and fine-grained control was sacrificed, as well as explicit memory management and pointers. It was almost impossible to write Java software that would corrupt memory, but programmers couldn't dispose of memory themselves (to the horror of many there was a `new` keyword but no `delete`); instead it was handled automatically by the run-time environment using the 'garbage collector'. This was part of the environment that monitored object references at run time and would decide when to reclaim allocated memory on behalf of the program.

Added to all of these new ideas was the 'applet'. This probably had the largest role in making Java a 1990s buzz word. Java applets are small programs that can run inside a web browser. Code resides on a remote server, is downloaded over the network and is executed locally inside a 'sandbox', restricting operations to those considered 'safe'. At the same time, class file verification and a well-defined security model introduced a new level of software security.

Java applets had limited use and were quickly outdated by the rise of Flash as a browser-based, client-side technology but applets had served a special role in the Java world. They demonstrated the power of this new language called Java, raised awareness of the Internet and helped create a strong worldwide Java developer base that today numbers over six million.

As time went on, Java moved to have more of a focus on desktop applications and then server applications: suddenly, everyone was talking

about Enterprise Java Beans. Following this was a big leap to smart cards and the emerging market with the rise of mobile phones.

Sun quickly realized that while there were Java solutions for all of these areas, no one set of technologies could adequately address all requirements. It was at this point that Java was separated into the four core technologies that we see in Figure 1.4.[11]

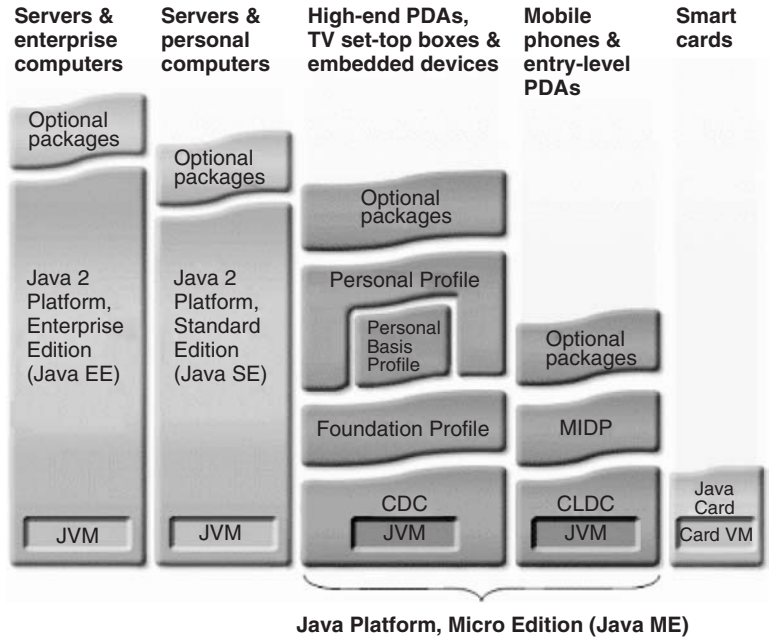

Figure 1.4 Java technologies

The Java Platform, Micro Edition (Java ME) fork was created to address the constrained-device market. Java Card is a separate technology that has nothing to do with Java ME. These days, most mobile phones include some kind of Java technology. Blackberries run a superset of Java ME, extended with RIM libraries. Google's Android platform uses a 'Java-like' platform and even Microsoft's Windows Mobile and Pocket PC support Java ME. Nokia's Series 40 phones, which form the backbone of the feature phone market, all have support for Java ME as do *all* phones running Symbian OS.[12]

If you want to know more about the other Java technologies or learn about the Java language, this is probably the wrong book for you. If that's what you want, then probably the best place to start learning about Java

[11] Java ME consists of two 'flavors', which is why there are five groups in Figure 1.4.

[12] Except for MOAP(S), which has DoJa; it is very similar to Java ME, but is not strictly the same.

is straight from the creator himself, James Gosling in [Arnold, Gosling and Holmes 2005]. From this point on, we're only going to talk about ME – Java ME!

1.5 Java ME

Sun launched a research and development project called Spotless in the 1990s in order to produce a JVM that could run on devices with small memory and power budgets, intermittent or non-existent connectivity, limited graphical user interfaces, divergent file system concepts (if any) and wildly varying operating systems. This could be anything from set-top boxes to Internet phones, parking meters, digital TVs, vending machines, automotive interfaces, electronic toys, personal organizers, household appliances, pagers, PDAs, high-end smartphones and mass-market feature phones.

The result of the project was the Kauai virtual machine (KVM). This was a cut-down version of the JVM that required less than 10% of the resources needed for the desktop standard edition of Java. This was achieved by targeting the size of the virtual machine and its class libraries, reducing the memory used during execution and by creating a pluggable architecture that allowed sub-systems of the KVM to be tailored to specific device architectures.

With such a diverse range of hardware targets, it was decided to adopt a more flexible model for Java ME and the final result consisted of three high-level components that together make a wide range of solutions possible. Java ME consists of:

- configurations
- profiles
- optional packages.

We'll look at each of these in detail shortly. In order to allow for future expansion and new technologies, additions to the Java ME technologies are managed through the Java Community Process[13] in the form of Java Specification Requests known as JSRs. Each JSR is reviewed by panels of experts and public drafts allow a wide range of interested parties to contribute to the process. JSRs are usually referred to by their name and number; for example, JSR-82 is the specification request for the Bluetooth API.

A lot of functionality had to be removed from the desktop Java Standard Edition (Java SE) to make Java ME. For example, some things that are missing include:

[13] ***www.jcp.org***

- reflection
- thread groups and advanced thread control
- user-defined class loaders
- the Java native interface (JNI)
- automatic object finalization.

Compatibility played a large part in the design of Java ME. In order to maintain as much compatibility with Java SE as possible, the Java ME packages were divided into two logical groupings: those that were specific to Java ME (packages whose names start with `javax`) and those that were a subset of their Java SE counterparts which followed the same package-naming convention.

One of the strongest features of the Java language is its security – both at the application layer and within the virtual machine itself. The VM security layer ensures that bytecodes are valid and safe to execute. This works really well on servers and the desktop environment but not so well on mobile phones with small processors and small power and memory budgets. The solution was a hybrid approach using the concept of *pre-verification* performed offline as part of the build process on the development machine. Only after VM acceptance may the bytecodes execute on the host hardware.

1.5.1 Configurations

Configurations define the lowest common denominator for a horizontal category of devices that share similarities in terms of memory budgets, processor power and network connectivity. A configuration is the specification of a JVM and a base set of necessary class libraries which are primarily cut-down versions of their desktop counterparts. There are currently only two configurations:

- The Connected Limited Device Configuration (CLDC) was aimed specifically at mobile phones, two-way pagers and low-level PDAs.
- The Connected Device Configuration (CDC) was aimed at devices with more memory, better network connectivity and faster processors.

We should be clear, however, that the CDC profile doesn't really have a role in the mobile phone space at this time. Although high-end devices, such as Symbian OS phones, can accommodate a CDC-based Java platform, the consumer market today is based on MIDP/CLDC applications, so CDC is not within the scope of this book.

Table 1.1 CLDC 1.1 Packages

Package	Description
`java.io`	Basic classes for I/O via data streams, Readers and Writers
`java.lang`	Reduced math library, threads and system functionality, fundamental type classes
`java.lang.ref`	Support for weak references which allows you to hold a reference to an object even though it is garbage collected
`java.util`	The Vector and Hashtable classes, basic date and calendar functionality and a pseudo random number generator which is great for games
`javax.microedition.io`	CLDC specific interfaces and factories for datagrams, connection stream etc. Implementation deferred to profiles.

So far there have been two versions of the CLDC,[14] the latest being CLDC 1.1. Most mobile phones shipped since 2003 contain version 1.1 of the CLDC. CLDC 1.1 (see Table 1.1) includes, amongst other updates, support for floating-point operations with the introduction of the `Float` and `Double` classes, thread naming and interrupts. The total minimum memory requirement was also lifted from 160 KB to 192 KB.

While the discussion so far has focused on mobile phones, the CLDC is not just for phones. As we'll see in the following section, it forms the foundation for a number of profiles that target completely different device families.

1.5.2 Profiles

Profiles sit on top of a configuration and allow it to be adapted and specialized for a particular class of devices in a vertical market such as mobile phones. A profile effectively defines a contract between an application and a set of devices of the same type, usually by including class libraries that are far more domain-specific than those available in any particular configuration.

Figure 1.5 shows the three profiles that currently extend the CLDC. The Mobile Information Device Profile (MIDP) is most relevant to our work here; the Information Module Profile[15] (IMP) targets small devices that may have no user interface at all, such as parking meters, call boxes, and

[14] See the white paper at ***java.sun.com/j2me/docs/pdf/CLDC-HI_whitepaper-February_2005.pdf***.

[15] ***java.sun.com/products/imp/index.jsp***