DECLARATION

I, Jeremy Burton, have personal knowledge of the matters stated or referenced herein and if called to testify about them, I could and would be able to do so:

1. I am currently the Chief Technology Officer of Wonolo, a startup company which provides an online platform for staffing and recruiting.
2. From 1993 until 1999, I worked for Psion Plc. and Psion Software Plc. (collectively, "Psion"), and Symbian Ltd. ("Symbian"). In particular, from May 1997 until May 1998, I was a Business Development Manager at Psion; and from May 1998 until June1999, I was a Technology Manager at Symbian.
3. Attached as Exhibit A is a true and correct copy of the Java User Manual for Psion Series 5mx, titled "Psion Series 5mx Java^{TM} User Guide" ("Psion Java User Guide"). Based on my experience in the development of Symbian, my involvement with the Psion, and my personal knowledge, the Psion Java User Guide attached as Exhibit A was publicly available as of May 1999.
4. Attached as Exhibit B is a true and correct copy of the EPOC technical paper, titled "EPOC's JAVA^{TM} Implementation" ("EPOC Technical Paper"). Based on my experience in the development of Symbian, my involvement with the Psion, and my personal knowledge, the EPOC Technical Paper attached as Exhibit B was publicly available as of March 1999.
5. I have personal knowledge that the Psion Series 5mx was sold in the United States at least as of 1999 and that customers who purchased a factory-sealed Psion Series 5mx would have received a copy of the Psion Java User Guide. The

EPOC Technical Paper was also available to developers through the Symbian website.

6. The Psion Java User Guide and the EPOC Technical Paper were made at or near the time of development of the Psion Series 5mx, by a person with knowledge of the matters stated; they were kept in the course of the regularly conducted activities of Psion.

I declare under penalty of perjury that the foregoing is true and correct.

Executed on November 13, 2014 in San Francisco, CA.

By: [illegible]

Jeremy Burton

EXHIBIT A

Java™

User Guide

Version 1.00
May 1999
English

Introduction

Java™ technology is a programming language developed by Sun Microsystems Inc. Programs written using Java technology may be run on a variety of computing platforms which have a Java Virtual Machine (JVM). This software includes the following components:

- A Java Virtual Machine (JVM) for EPOC.
- Class Libraries and other files to allow you to run Java 'applets' and 'applications'.

Memory requirements

Approximate memory requirements:

- This software requires a minimum of 5700K of free memory when installed on the 'C' drive.
- If you install this software on the 'D' drive (i.e. a memory disk), you will need 2900K of free memory on the 'D' drive and 2800K on the 'C' drive.
- Running Java applets and applications programs will require additional memory, according to the size and memory requirements of the program.

Running Java programs

You can run 'applets' and 'applications' written for the Java platform.

- A Java **applet** is a program which is normally embedded in a Web page, and it not designed for a particular platform. Applets can be viewed using the Web application.
- A Java **application** is a program that will normally be packaged specifically for your Series 5mx.

Important: You cannot run JavaScript programs using the Java Virtual Machine for EPOC. For example, you will not be able to view Web pages which implement JavaScript.

If you wish to run Java applications that have not been specially designed for your Series 5mx, see the 'Advanced Information' section later for help.

Running Java applets

When you open a Web page which includes a Java applet, the Web program will run the applet. Some Web sites allow you to download Java applets onto your Series 5mx, so that you can run them when you aren't connected to the Internet.

To run a downloaded applet:

1. Move to the System screen and select the applet file.
2. Tap on the applet to open it in the built-in applet viewer, or the Web program if you have installed it on your Series 5mx.
   Note: You can install Web from the PsiWin CD ROM.

See our Web site at **www.series5mx.com** for demonstration Java applets.

Running Java applications

We recommend that you only run Java applications that have been designed to run on your Series 5mx. Many Java applications have been designed to run on desktop PCs and may require resources that are not available on your Series 5mx. You can install Java applications that have been written for your Series 5mx, and run them in the same way as other third party programs, e.g. by tapping on the application's icon on the Extras bar.

If you have problems running Java programs...

Java applets and applications that have not been specifically developed for your Series 5mx may not run correctly. This may be due to a number of reasons, such as:

- **Size:** the application is designed for a PC desktop and is too large to fit on your Series 5mx's screen.
- **Security:** the application may require a level of security not available on your Series 5mx.
- **Resources:** the application may require resources, e.g. memory, that are not available on your Series 5mx.
- **Speed:** applets and applications that have been designed for desktop PCs may run considerably slower on your Series 5mx due to the slower processing speed.

If an applet does not run it will either be stopped automatically, or will simply not work. Moving to a different Web page will automatically stop the affected applet.

Important: You cannot run JavaScript programs using the Java Virtual Machine for EPOC.

If you are experiencing problems using Java programs, contact the program's supplier for assistance.

Viewing a Java program

Some Java programs are designed to be used on platforms with a larger screen size than the Series 5mx. If an area of the program is not displayed on the Series 5mx's screen, you can move the program so that it becomes visible.

To do this:

1. Press Ctrl+Spacebar. A flashing 'Move Mode' box will appear to the bottom right of the screen.
2. Use the arrow keys to move the window around the screen. Hold down the Shift key whilst using the arrow keys to move the window more quickly.
3. Press Esc to exit Move Mode.

Advanced information

This information is intended for readers who are familiar with Java programs.

- The Series 5mx Java implementation is a Java 1.1 run time environment based on the Sun 1.1.4 Java Development Kit, and is compatible with applications that support Java 1.1.
- Java applications that are not packaged for the Psion may be run provided they are compliant with Java 1.1, and that they operate within the memory and processor resources available on the computer.
- You may run a class file by tapping its icon twice from the system screen. The class file must contain the "main" class for the application.
- To run applications that are packaged in 'jar' or 'zip' files, you need to install the application into the correct place on the Series 5mx and then create an EPOC program to run the Java application. The tools to do this are contained within the Symbian *EPOC Java Software Development Kit*, which may be found in the EPOCWorld area of the **www.symbian.com** Web site.

Where to find out more

To learn more about Java technology, the Java Virtual Machine and running Java applications on your Series 5mx, more information can be found at the following Web sites:

www.java.sun.com	For general information on Java technology.
www.series5mx.com	For links to a Java specific area on our Web site, including demonstration Java applets for your Series 5mx.
www.symbian.com	For information on the Java platform implementation on the Series 5mx, toolkits, Software Development Kits (SDKs) and support for Series 5mx developers.

EXHIBIT B

EPOC's Java™ Implementation

Revision 1.0(009), 10th March 1999
Unclassified

Summary

In EPOC Release 5 Symbian adds Java capabilities to the EPOC platform by providing an EPOC Runtime for Java and an EPOC SDK for Java.
This paper is intended for Java developers interested in writing Java applets or applications that will be deployed on devices running EPOC: it describes in detail what Symbian has implemented and what features this offers developers.

Contents

1. About Java 1
1.1 Introduction 1
1.2 Basic concepts 1
1.3 Runtime environment architecture 2
1.4 Java platforms 3
2. Java on EPOC 3
2.1 Introduction 3
2.2 Architecture 4
2.3 Java and ROM-based devices 4
2.4 AWT 4
3. Using Java on EPOC 4
3.1 Why use Java on EPOC? 4
3.2 Java applications on EPOC 5
3.3 Using the EPOC SDK for Java 6
3.4 Application deployment 6

1. About Java

1.1 Introduction

This section provides enough information to understand the description of the Java implementation on EPOC. For more information about Java, follow the references in **http://java.sun.com/** indicated at the end of this paper.

1.2 Basic concepts

The term Java applies to two related but distinct concepts: it describes a *programming language* and a *runtime environment* in which programs written in the Java language can be executed. These are illustrated in Figure 1.
The Java language, from Sun Microsystems, Inc, is an object oriented language with a syntax similar to C++. It adds features such as garbage collection, and removes others such as pointers. These make Java an easier language in which to program in the sense that they remove the causes of significant numbers of bugs that are encountered when using C++. They also make it possible to check that programs written in Java are safe to run.
Java is an interpreted language. Unlike C++, it is not normally compiled to object or machine code that contains instructions that can only be executed by a specific CPU, or run on a specific operating system. Instead, Java source files are compiled to byte code — a compact representation of the source that can be interpreted on any computer that has a Java Virtual Machine.
Java programs can be stand-alone applications, or applets which run in the context of a web browser, or other program types such as servlets which run in the context of a web server.
Byte code is usually distributed in individual **.class** files, or collectively in **.zip** or **.jar** files.

The Java Virtual Machine (JVM) forms part of a Java runtime environment, which provides a self-contained execution environment for Java byte code that is independent of the operating system of the computer that it runs on. The runtime environment comprises:

- class loaders that load the byte codes from **.class**, **.jar** or **.zip** files
- the JVM interpreter that processes the byte codes to execute them
- class libraries that contain implementations of APIs that the Java source code can use. For example the AWT class library provides the Abstract Windowing Toolkit application framework, including event handling, windowing and graphics support.

The specification of the JVM is published by Sun Microsystems, Inc., as are the specifications of the standard Java APIs.

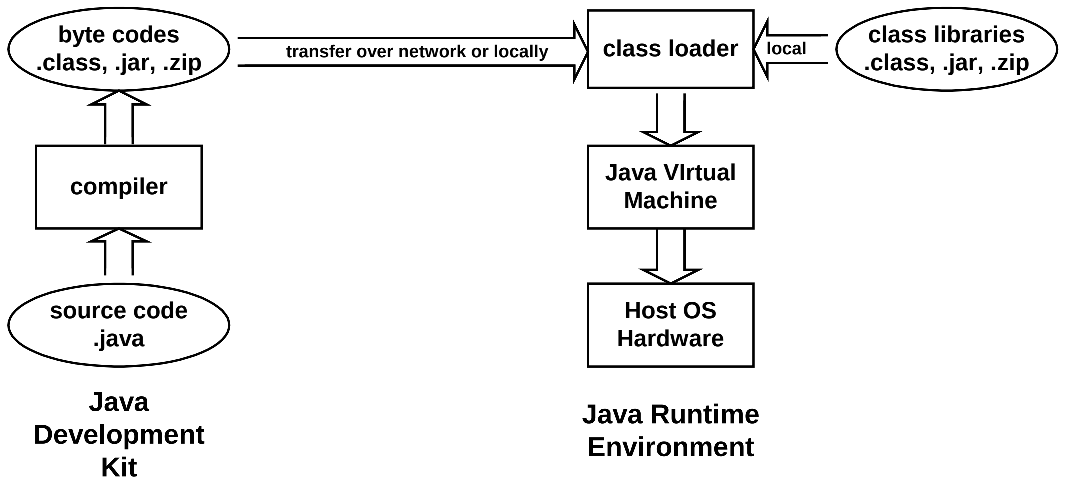

Figure 1: Java Development and Runtime Environments

1.3 Runtime environment architecture

This section describes how a Java runtime environment is implemented. Broadly speaking, the runtime environment will be developed using a combination of platform-independent Java code, and platform-dependent code written in C, C++ or assembler.

The Java language and JVM specifications define a way of declaring a *native method*. These methods belong to Java classes, but are implemented in C or C++ and hence are platform-dependent. It is necessary to use native methods to access the services of the host operating system. The interface to the native method is defined by the Java Native Interface (JNI).

Application code that uses native methods directly becomes platform-specific and therefore loses one of the main advantages available to Java developers. Java code that uses class libraries that are implemented using native methods remains platform-independent.

A Java runtime environment will typically comprise several layers of code:

- Application or applet code, compiled from Java source files. This will normally be platform-independent code, as it has been written in terms of the functionality provided by the standard Java class libraries. As explained above, however, the application code can use native methods to access operating system services.
- Class libraries. These may be implemented in Java purely in terms of the services provided by the JVM, but many libraries need to use the services of the host operating system (such as file i/o). This must be done using native methods, which means the runtime environment must include native libraries that are platform-dependent.
- Virtual machine layers. The virtual machine is defined such that part of it is written in Java and is platform-independent. This layer runs on top of a platform-dependent layer that implements an API called the Host Porting Interface (HPI).

The virtual machine includes an interpreter that can be implemented in C or assembler. Using C means that the interpreter needs to be recompiled for each new platform, but not rewritten, as an assembler interpreter would be. However, it is often possible to improve the performance of the interpreter significantly by writing some of it in assembler.

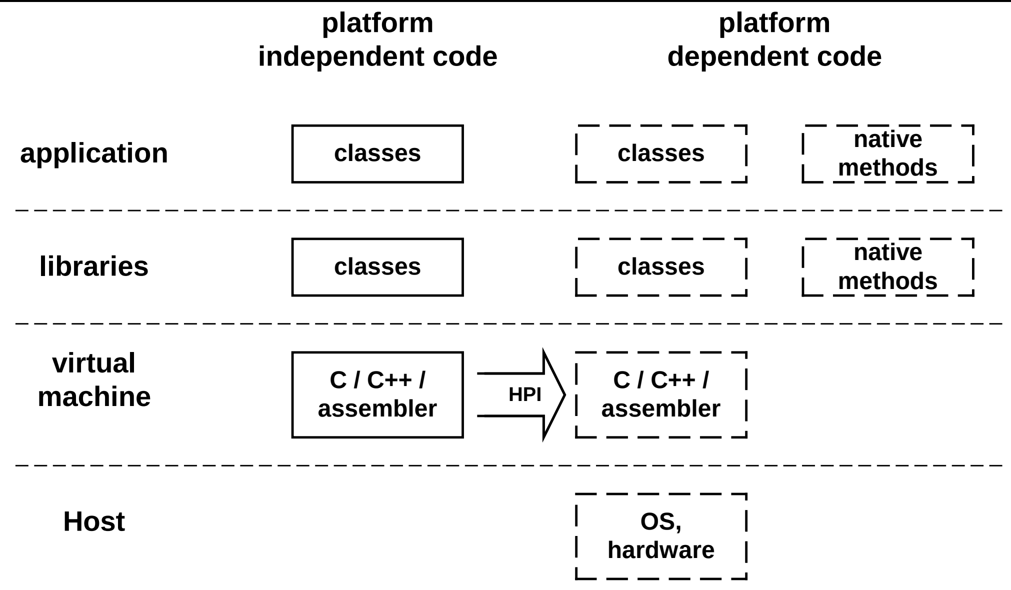

Figure 2: Java Runtime Environment Architecture

1.4 Java platforms

Java is available in several different forms, described below. In what follows it helps to keep in mind that a Java Application Environment (JAE) comprises a Java VM, APIs and support files, whereas a Java Development Kit (JDK) comprises an application environment, plus development tools. For some of the products listed below Sun Microsystems, Inc. provide a JDK and JAE, but others are only available as a JAE.
Information about all of these products is available at **http://java.sun.com/products/index.html.**
Enterprise Java is available in several versions: 1.0.2, 1.1 and 1.2 (or Java 2). Sun Microsystems, Inc. provide a JAE and JDK for each of these. Because of this, these versions of Java are often referred to as (for example) JDK 1.1, even though what is being described may actually be an application environment. The different versions of JDK 1.1 — running up to JDK 1.1.8 at the time of writing — are compatible with one another: JDK 1.1 and JDK 1.2 are not.
Personal Java is an application environment. It is distinguished from JDK 1.1 in that it includes only a subset of the JDK 1.1 APIs. The subset includes AWT and other classes that allow applets and applications to be supported. The Personal Java VM adheres to the same specification as the Enterprise Java VM's, although its implementation is tuned for resource-constrained devices.
Embedded Java is an application environment that is intended for devices having dedicated functionality and limitations on resources such as memory. The embedded Java APIs do not include AWT, so this application environment does not support applets. The intention is that the program to be run be provided as a ROM image.

2. Java on EPOC

2.1 Introduction

Symbian provides the following Java tools in EPOC Release 5.

1. the EPOC Runtime for Java is a Java Compatible™ runtime environment based on Sun Microsystems, Inc.'s JDK 1.1.4 that enables Java applets and applications to run on a Java-enabled EPOC device. The runtime supports 100% Pure Java applications and native methods.
2. the EPOC SDK for Java provides documentation and examples that explain how to develop and deploy Java applications on EPOC. It also provides tools that support native method development using JNI.

The EPOC Runtime for Java supports 100% Pure Java applications. It is not necessary to write any native code or to use any EPOC-specific APIs to run Java applications on EPOC. To write native code, the EPOC C++ SDK is required.
The EPOC Runtime for Java is integrated with the EPOC Web application. Applets in local and remote HTML files can be viewed in the browser.

Application deployment may involve using EPOC-specific tools. EPOC does not provide a command line to end users, so the platform-specific batch files typically used to deploy applications will need to be modified or replaced. This is explained further below.
Symbian does *not* provide:

- a Java Development Kit (JDK) for EPOC. Java developers should continue to use their preferred Java development environment.
- a JDBC driver for EPOC's native DBMS database format. However, the JDBC framework is supported, and can be used with pure Java database drivers.

2.2 Architecture

The EPOC Runtime for Java is based on JDK 1.1.4 sources, from Sun Microsystems, Inc., though Symbian has incorporated bug fixes and other code changes, such as inclusion of the Euro symbol, from later JDK 1.1 releases. The architecture is described in §1.3. Porting the Java runtime to EPOC required Symbian to:

1. Implement the platform-dependent layer of the VM in C++ using native EPOC APIs.
2. Implement the class libraries that comprise JDK 1.1.4. A major component of this task was providing a port of the AWT classes to EPOC.
3. Extend tools provided by Sun Microsystems, Inc. that allow the Java class libraries to be placed in ROM.
4. Provide an optimised interpreter loop for the VM using assembler.

2.3 Java and ROM-based devices

Symbian has developed the EPOC operating system for use in Wireless Information Devices (WIDs). These devices are typically ROM-based: while they will have RAM or the equivalent, the operating system and primary software applications are provided in ROM.
This is a significant issue for a Java Runtime. When Java bytecodes are read by the class loader, they undergo processes of resolution and quickening that require the original bytecodes in the **.class** or **.jar** file to be modified. This requires the bytecodes to be loaded from ROM into RAM.
To enable the use of Java classes directly from ROM it is possible to specify that such classes be "pre-loaded" as part of the process of making the EPOC ROM image for a particular device.
For each class specified the pre-loader attempts to perform as many of the actions as possible that would be performed on the class were it to be loaded into the VM at run-time, and generates the run-time structures representing the class for use by the VM, ahead-of-time.
This results in the classes being available for use by the VM at runtime with little or none of the overhead which would result from loading each class individually at run-time, and enables them to be used directly from ROM.
When pre-loading a number of classes, the pre-loader will also attempt to share as many of the generated structures as possible thereby reducing the memory footprint of the classes within ROM.
It should be noted, however, that a consequence of pre-loading as it currently stands is that each class that is pre-loaded is effectively built-in to the VM and as such it cannot be overridden by another version of the class although such a class may be potentially visible to the VM at run-time. This has advantages — each such class is tamper-proof — and disadvantages — it is not possible to override the pre-loaded class with a newer version.

2.4 AWT

The Abstract Windowing Toolkit (AWT) is one of the core class libraries in the JDK 1.1. Porting the Java runtime environment to EPOC required Symbian to implement the AWT peer classes in terms of the native GUI widgets.
As a result, programs written to the AWT API will run on EPOC with the same look and feel as native EPOC applications written in C++.
The EPOC Runtime for Java is compatible with Swing, but due to the memory and CPU resources required by Swing, developers may prefer to use AWT.

3. Using Java on EPOC

3.1 Why use Java on EPOC?

Java adds a fourth development option to EPOC. As shown in Figure 3, EPOC also supports development of applications in C++, its native language, and OPL, a BASIC-like interpreted language. Additionally, EPOC supports development of PC-based connectivity and data synchronization applications using the Connectivity SDK.

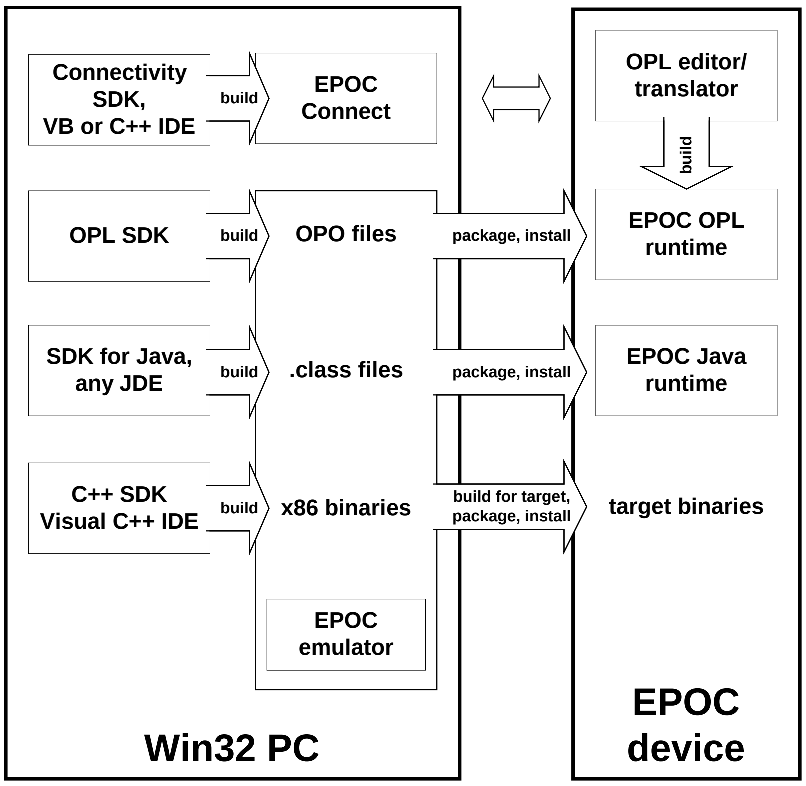

Figure 3: EPOC software development options

Java adds a popular and capable standard option for development of applications on EPOC devices, and for cooperation between these applications and applications based on other server systems.

Java provides a pervasive platform for users and developers. Java programs and applications can run on a wide range of desktop and laptop computers. Developers working in Java can target many different devices with one program, and as the range of EPOC devices grows they will be able to target the whole range with their programs.

Developers can use standard desktop IDEs to prototype and develop Java programs for all platforms, including EPOC WIDs.

WIDs especially are ideal targets for further extending access to Java applications so that users can carry the same applications away from their desktop, and even their laptop or notebook computer.

WIDs allow developers of value-added software to sell this software even more widely; they are particularly good clients for enterprise software with which Java is already successful

Java will allow, in a natural way, for the downloading of dynamically changing information and program content to EPOC devices.

Java provides an open standards based development environment for EPOC developers. Symbian is committed to the Java platform as defined by Sun Microsystems, Inc.

Java is a first-class object oriented programming language. The Java platform is widely understood, and there are many skilled Java developers and programmers around the world. Java is thus a very productive environment for developers and provides a rapid-time-to-market approach.

The movement towards Java being incorporated into standards such as MExE, which will be important in the WID arena, means that Java already has an important position in plans for future development work.

3.2 Java applications on EPOC

Because the EPOC Runtime for Java is a Java Compatible implementation of the JDK 1.1 application environment, there is no requirement for the developer to use custom APIs or native code. Applications written for the JDK 1.1 environment on another platform will work on EPOC provided the device characteristics have been taken into account. The most important of these are:

1. The screen size will be significantly smaller than that in a desktop Java environment.
2. The screen may not support colours.
3. There will probably not be a mouse, and on some EPOC implementations not even a touch screen or pointing device of any kind.
4. Memory will be limited, as will the computing power of the CPU.

5. The AWT implementation is not identical to desktop environments: buttons, check boxes, scroll bars, etc, may be of slightly different sizes and have slightly different behaviour, as allowed by the AWT specification.

What will be required is that applications provided as **.class**, **.zip** or **.jar** files be packaged using the EPOC **.sis** file format, so that they can be installed on the EPOC device. This is discussed in more detail in §3.4.

3.3 Using the EPOC SDK for Java

Java developers will need to continue using their preferred Java development environment, whether this be Sun Microsystems, Inc.'s JDK 1.1 or a commercial Java IDE.

To facilitate development of Java applications to run on EPOC, Symbian provides the EPOC SDK for Java. This SDK comprises:

- An EPOC emulator that runs on Windows NT or 9x
- A Java Runtime Environment that runs on the EPOC emulator
- Documentation, including background information on EPOC for Java developers
- Examples and demo applets and applications
- Tools for native development
- Tools that support deployment of Java applications on EPOC

Typically, the following steps are required to get a Java application running on EPOC.

1. Write the application using the Java development environment of your choice
2. Debug the application using JDB or your JDE's debugging facilities
3. Transfer the **.html**, **.class**, **.zip** or **.jar** file(s) to the Windows directories used by the EPOC emulator.
4. Use a Windows command shell to run your application or applet on the EPOC emulator's JVM
5. Debug
6. Package your application for deployment on an EPOC device.

3.4 Application deployment

Deployment deals with issues you must address to get an application running on a particular OS. For example, to have your Java application installed on Windows, you must worry about batch files, Start Menus, and so on. Even though you have written a 100% Pure Java application, at this stage you have to deal with the OS.

EPOC is no different. Each EPOC device has a mechanism that allows the end user to launch applications. On a Psion Series 5, for example, users launch third party applications from the Extras bar. Users do not have access to a command line on EPOC devices.

The EPOC SDK for Java provides tools that allow you to package your application into a **.sis** file, EPOC's native installation package format, for installation on an EPOC device in the same manner as C++ or OPL applications. The process does not require any native C++ programming.

If you are deploying an applet, just use the web browser.

For more information

Sun Microsystems, Inc.'s website contains much useful introductory and reference information about Java. The following are especially useful:

- the Java homepage at **http://java.sun.com/**
- the Java documentation pages at **http://java.sun.com/docs/**
- the Java tutorial at **http://java.sun.com/docs/books/tutorial/**

Symbian licenses, develops and supports the EPOC operating system, providing leading software, user interfaces, application frameworks and development tools for Wireless Information Devices such as Communicators and Smartphones. Symbian is based in London, with offices worldwide. See **http://www.symbian.com/** for more technical papers, information about Symbian, and information about EPOC.

Trademarks and acknowledgements