Legal Notice

Copyright © 2004 Nokia Corporation. All rights reserved.

Nokia and Nokia Connecting People are registered trademarks of Nokia Corporation. Java and all Java-based marks are trademarks or registered trademarks of Sun Microsystems, Inc. Other product and company names mentioned herein may be trademarks or trade names of their respective owners.

Disclaimer

The information in this document is provided “as is,” with no warranties whatsoever, including any warranty of merchantability, fitness for any particular purpose, or any warranty otherwise arising out of any proposal, specification, or sample. Furthermore, information provided in this document is preliminary, and may be changed substantially prior to final release. This document is provided for informational purposes only.

Nokia Corporation disclaims all liability, including liability for infringement of any proprietary rights, relating to implementation of information presented in this document. Nokia Corporation does not warrant or represent that such use will not infringe such rights.

Nokia Corporation retains the right to make changes to this specification at any time, without notice.

License

A license is hereby granted to download and print a copy of this specification for personal use only. No other license to any other intellectual property rights is granted herein.
Contents

1. Introduction ......................................................................................................... 5
   1.1 About this document .......................................................................................5
   1.2 What you should already know ....................................................................5
   1.3 What you will need .......................................................................................5
   1.4 Checking the Application Development Environment ..................................6
       1.4.1 Checking the paths for the Java compiler and preverifier tools ..............6
       1.4.2 Troubleshooting: PATH variable not set or not set correctly .............6
       1.4.3 Setting the PATH variable ....................................................................7

2. Environment and Tools ..................................................................................... 9
   2.1 Symbian OS 7.0s ..........................................................................................9
   2.2 Series 60 Platform ......................................................................................9
   2.3 Series 60 MIDP SDK for Symbian OS .........................................................10

3. Introduction to Java™ technology for Series 60 ............................................ 12
   3.1 Mobile Information Device Profile (MIDP) ................................................12
   3.2 Connected Limited Device Configuration (CLDC) ......................................12
   3.3 MIDlet .........................................................................................................12
   3.4 MIDlet lifecycle ...........................................................................................12
       3.4.1 startApp() Method ..............................................................................13
       3.4.2 pauseApp() Method ...........................................................................13
       3.4.3 destroyApp() Method .........................................................................13
   3.5 MIDP User Interface APIs ..........................................................................13
   3.6 MIDlet Suites and Application Descriptors .................................................13
   3.7 HelloWorld MIDlet .....................................................................................14

4. Creating an Application .................................................................................... 19
   4.1 Basics of creating a MIDlet for Series 60 ..................................................19
       4.1.1 Step 1: Building and running the Hello World MIDlet .......................19
       4.1.2 Step 2: Creating your own application from Hello World MIDlet ........20
       4.1.3 Step 3: Packaging the modified Hello World MIDlet .......................20
       4.1.4 Installing your application to your phone ............................................20
   4.2 Building and running the Hello World MIDlet .........................................20
       4.2.1 Compiling the HelloWorld MIDlet ...................................................20
       4.2.2 Preverifying the compiled HelloWorld MIDlet ...................................21
       4.2.3 Running the HelloWorld MIDlet in the emulator .............................21
   4.3 Creating your own application from the Hello World MIDlet ....................23
   4.4 Packaging the modified Hello World MIDlet ............................................25
       4.4.1 Creating the manifestation file ..........................................................25
       4.4.2 Creating the application JAR File ......................................................26
4.4.3 Creating the JAD file..............................................................27
4.5 Installing your application to your phone..............................28
   4.5.1 Example of how to install the MIDlet............................28
5. Moving On............................................................................30
6. Developer Resources..........................................................31
   6.1 SDK Help.....................................................................31
   6.2 Sun Java developer site..................................................31
   6.3 Symbian Developer Network.........................................31
7. Terms and Abbreviations....................................................33
1. Introduction

Thank you for installing the Series 60 MIDP Software Development Kit for Symbian OS (hereafter referred to as ‘SDK’). With the SDK, you can develop Java™ applications for Series 60 Platform smartphones on your PC.

The aim of this Getting Started Guide is to guide you through your first steps in developing a Java application for the Series 60 Platform with the SDK. With the help of the guide, you will learn the basics of the environment and the tools, after which you will create your first simple Java™ application for the Series 60 Platform. Once you have thus grasped the essentials of application implementation, you can move on to design and create more versatile applications on your own.

1.1 About this document

This guide is divided roughly into four main sections:

- The Introduction outlines the general structure and contents and structure of this guide as well as what you will need for application development.
- The Environment and Tools section provides a general overview of the environment in which you will develop applications as well as of the tools that you will use.
- The Introduction to Java™ technology for Series 60 section provides general information on Java technology and MIDlets that will come in handy when developing applications with the SDK.
- The Creating an Application section provides instructions on how to create an application for the Series 60 platform with the SDK. In this section you will create and modify a simple application by following step-by-step instructions that take you through the different phases involved in application creation.

In addition to the above, this guide also tells you where to find more detailed information on designing and creating applications of your own with the SDK.

1.2 What you should already know

Although this guide seeks to provide a general introduction to Java technology, it is recommended that you look into some more comprehensive introductions into Java. As the Java technology was created by Sun technologies Inc., a good place to find such sources is the java.sun.com (http://java.sun.com/) website, and its wireless pages (http://developers.sun.com/techtopics/mobility/) in particular.

In addition to the above, familiarity with the basics of wireless communication is also recommended.

1.3 What you will need

You should have the following tools and resources available on your PC before you can start to develop Java MIDP applications using the Series 60 MIDP SDK:

- Windows 2000 (SP3 or higher) or Windows XP (SP1a or higher)
- Java 2 Standard Edition (J2SE) SDK version 1.4.1_02 or higher
  The J2SE SDK provides the Java compiler and other tools used in application
development. You can download the J2SE SDK at http://java.sun.com/j2se
- The Series 60 MIDP SDK 2.1 installed on your PC
- A text editor, such as WordPad

1.4 Checking the Application Development Environment

When you start developing an application for the first time with the Series 60 MIDP SDK
2.1, you should first check that the development environment is set up properly. This
involves, for example, checking that the paths for the Java compiler and preverifier tool
are set correctly. This section provides instructions on how to test and troubleshoot the
needed path settings.

You can run the Java 2 SDK without setting the PATH variable. However, setting the
PATH variable allows you to run the executables (javac.exe, preverify.exe) from any
directory without having to specify the full path to the executable every time you run it.
See section Troubleshooting: PATH variable not set or not set correctly for detailed
instructions on how to set the path.

1.4.1 Checking the paths for the Java compiler and preverifier tools

1. In the Windows Start menu, open the Command Prompt by choosing Start >
   Programs > Accessories > Command Prompt
2. To check the Java compiler (javac.exe) settings, enter the following
   command in the Command Prompt:

   javac
   
   You should get the following output on the screen:

   Usage: javac <options> <source files> etc ... (output data)
   
   If you get an error message, see section 1.4.2 for more details.
3. To check the preverifier tool (preverify.exe), enter the following command in
   the Command Prompt:

   preverify
   
   You should get the following output on the screen:

   Usage: preverify [options] classnames|dirnames ...etc ...
            (output data)
   
   If you get an error message, see section 1.4.2 for more details.

1.4.2 Troubleshooting: PATH variable not set or not set correctly

When testing the application environment settings, you may get the following
error message:

The name specified is not recognized as an internal or external
command, operable program or batch file (Windows 2000, XP)

This means that Windows cannot locate the Java compiler (javac.exe) or the
preverifier tool (preverify.exe). See examples below on how to specify their
locations for Windows.
To specify the location of the Java compiler:

If you installed the Java 2 Software Development Kit in, for example, C:\jdk1.4, enter the following command in the Command Prompt:

C:\jdk1.4\bin\javac

To specify the location of the preverifier tool:

For example, C:\<SDK installation path>, enter the following command in the Command Prompt:

C:\Series_60_MIDP_SDK_2_1\bin\preverify

Note that in this case each time you compile or preverify an application, you have to precede your javac and preverifier commands with C:\jdk1.4\bin\ or C:\Series_60_MIDP_SDK_2_1\bin\, such as, C:\j2sdk1.4.2_<version>\bin\javac MyClass.java. To avoid this, update the PATH variable as instructed in the following section.

1.4.3 Setting the PATH variable

To avoid having to enter the full path every time you need to compile or preverify a MIDlet, it is useful to set the path permanently so it will persist even after rebooting. This is done by adding the full path of the directory to the PATH variable.

Typically, this full path is C:\j2sdk1.4.2_<version>\bin for the Java compiler and C:\Series_60_MIDP_SDK_2_1\bin for the preverifier.

Complete the following steps to set the path permanently on Microsoft 2000 and XP:

1. Choose Start > Settings > Control Panel > System
2. Go to the Advanced tab and select Environment Variables
3. Locate the Path variable in the User Variables or System Variables
   If you are not sure where to add the path, enter it as the last Path variable value listed in User Variables.
4. Select the Path variable and click Edit
5. Enter the path into the variable value field, for example, as follows:

   C:\j2sdk1.4.2_<version>\bin for the compiler
   C:\Series_60_MIDP_SDK_2_1\bin for the preverifier

Caution: Do not delete the existing path settings. Deleting the settings may affect the functioning of the system or other programs.
**Note:** The PATH can be a series of directories separated by semi-colons (;). Microsoft Windows looks for programs in the PATH directories in the order from left to right. You should have only one bin directory specified for the Java SDK at a time (those following the first one are ignored). If there is a path setting already specified, you can update it with the path given here.

6. **Click Set, OK or Apply**

The new path is now set for each time you open the Command Prompt.

You can verify the settings by completing the steps detailed in [Checking the paths for the Java compiler and preverifier tools](#).
2. Environment and Tools

This section provides basic information about the environment in which you will create applications with the SDK. Here you can also find information on the contents and structure of such an application, as well as a description of the SDK itself.

2.1 Symbian OS 7.0s

Symbian OS is an operating system optimized for mobile terminals, such as communicators and smart phones, and it is the underlying system upon which the Series 60 Platform has been built. The Symbian OS provides a reliable environment as it has been designed so that user data is never lost and the device running the operating system will never have to be rebooted.

2.2 Series 60 Platform

The Series 60 Platform is a complete package of applications, user interface and development tools built upon Symbian OS technology. It has been designed to run on various models of different manufacturer’s devices. In other words, an application developed for the Series 60 Platform will run smoothly on all devices that run the same platform.

The Series 60 User Interface (UI) has been specifically designed for easy, one-handed use. From the user’s point of view, probably the most important feature of the platform is its user interface: A large color screen (176x208 pixels) and the various input keys (two “soft keys”, five-way navigator and several dedicated keys). The principle of one-handed use and the large color screen also enables application developers to present attractive content and provide easy navigation. It is easy for any user with experience of mobile phones to grasp the basic idea of the UI.

Figure 1: The Series 60 Platform User Interface

There are also a variety of applications to be found on the Series 60 Platform. Important amongst these are the advanced smartphone telephony applications, Personal Information Management (PIM) applications (for example, Phonebook, Calendar, Photo Album, etc.), messaging, browsing, e-mail and an installation engine that allows the user to add or remove applications to and from the platform, either via PC Connectivity or “Over-the-Air” downloads.
The Series 60 Platform also provides support for applications written in the Java programming language. The Java platform implemented on Series 60 devices is the Java 2 Platform Mobile Edition (J2ME™), which is designed for small mobile devices, such as smartphones.

Any Java application can easily be delivered over the Internet, or any network, without operating system or hardware platform compatibility issues. Java technology components run on any kind of compatible device that supports the Java platform. Note, however, that using the Nokia-specific Nokia UI API included in the SDK affects the portability of your application.

For more detailed information about Java technology supported in the Series 60 Platform, please refer to section Introduction to Java™ technology for Series 60, below.

2.3 Series 60 MIDP SDK for Symbian OS

The Series 60 MIDP SDK is a development tool that lets you develop, preview and install Java applications to any Series 60 smartphone. For this purpose, the SDK provides a wide range of tools, Application Programming Interfaces (APIs), libraries and documentation to facilitate application development. Development is PC hosted and you can view and test applications on an emulator provided with the SDK.

Once you have downloaded and installed the SDK, the following components and functionality will be installed on your PC.

**API files**

The SDK includes the needed J2ME Mobile Information Device Profile (MIDP) API files that you need in developing Java applications for Series 60 smartphones. These include, for example, the Bluetooth API, Mobile Media API and Wireless Messaging API.

The role of APIs in MIDP applications and their development is covered in more detail in Introduction to Java™ technology for Series 60 below.

**Series 60 emulator**

The emulator enables you to view and test applications on your PC before installing them to a target device. It provides a graphical user interface with the needed phone functionality to test your application:

![Figure 2: The emulator](image-url)
The emulator simulates the operation of an application on a target device so accurately that application development can be done even before the required hardware, that is, the smartphone, is available.

Using the emulator is covered in more detail in section Running the HelloWorld MIDlet in the emulator, below.

User documentation

The SDK Help contains information and instructions needed in using the SDK. This includes Java docs, which provide needed reference information on the APIs used, as well as developer guides, which provide useful information, instructions and tips on application development.
3. **Introduction to Java™ technology for Series 60**

This section provides basic information on the Java technology. It is recommended that you familiarize yourself with these basics before moving on to creating your own Java application with the SDK, as described in Creating an Application below. Here we will go through the basics of the application environment, what an application consists of and how it works.

3.1 Mobile Information Device Profile (MIDP)

The Mobile Information Device Profile (MIDP) is a set of Java APIs that together with the Connected Limited Device Configuration (CLDC) provides a complete J2ME application runtime environment for mobile information devices (MIDs). The MIDP specifications (JSR-00118) define minimum hardware, software, networking, application requirements, as well as standard system APIs for devices that support MIDP.

MIDP is intended for “connected” devices such as smartphones, which are characterized by having limited Central Processing Unit (CPU), memory, keyboard, and display capabilities.


3.2 Connected Limited Device Configuration (CLDC)

Connected Limited Device Configuration (CLDC) defines the Java language features and the core libraries of the Java Virtual Machine (JVM).

3.3 MIDlet

Java applications developed with the SDK are called MIDlets. A MIDlet is an application, which has been written for the Java 2 Micro Edition (J2ME) Mobile Information Device Profile (MIDP).

A simple example of a MIDlet is the HelloWorld example application provided with the SDK. In section Creating an Application, you will build, personalize and run the HelloWorld MIDlet.

3.4 MIDlet lifecycle

MIDlets are designed to co-exist with other applications on a Mobile Information Device (MID). The MIDlet lifecycle is pivotal in enabling this co-existence. Basically, a MIDlet can be in one of the following states: paused, active, or destroyed.

The Application Management Software (AMS) controls the state of a MIDlet by directing a MIDlet to start, pause, or destroy itself. It constructs a MIDlet and communicates changes in its state by calling the following methods, which every MIDlet must implement: startApp(), pauseApp(), and destroyApp().

All MIDlets start out in the paused state. When a user starts an application, the AMS calls the startApp() Method of the MIDlet in question, after which the MIDlet is in active state (that is, the application is opened).
3.4.1 startApp() Method

When a user opens a Java application on his or her device, the AMS calls the startApp() method. In this case, the MIDlet is newly constructed with the method. In the case of such a newly constructed MIDlet, the startApp() method calls Display.getDisplay(this).setCurrent(startup_screen) to invoke the startup screen of the MIDlet (that is, application) in question.

The startApp() method is also invoked by the AMS when a MIDlet is returned from the paused state. In this case, the startApp() method will typically need to restore the state it was in before it was paused.

3.4.2 pauseApp() Method

When a user pauses (but does not close) an application, the AMS calls the pauseApp() method. In this case, the application is paused and it is no longer displayed in the User Interface (UI) to the user. However, any threads or timers created by the MIDlet in question will continue to run unless the user stops these manually. These threads keep on running by default, because the application will need to restore its state when it is reactivated with the startApp() method.

3.4.3 destroyApp() Method

When a user closes an application, the AMS calls the destroyApp() method. This method instructs the MIDlet to release all the resources it has reserved and to close down as soon as possible. In effect, this means that the application in question closes all input and output streams, terminates all threads, and cancels all timers. Once the destroyApp() method has been called, the MIDlet ceases to access the display, in other words, the application is no longer displayed in the UI to the user.

3.5 MIDP User Interface APIs

The MIDP User Interface (UI) has a high-level and low-level API.

The high-level API classes Alert, Form, List, and TextBox are extensions of the abstract class Screen. These are designed to provide abstractions and components that are highly portable across different Mobile Information Devices (MIDs), as the application takes care of aspects such as drawing, font characteristics, navigation, and scrolling. A particular device's implementation of these classes performs the adaptation to its hardware and native UI look and feel.

The low-level API is based on use of the abstract class Canvas. In comparison with the high-level APIs, this class allows applications to have more direct control of the UI by permitting greater control of what is drawn on the display and reception of low-level keyboard events.

3.6 MIDlet Suites and Application Descriptors

A MIDlet suite is a Java archive (JAR file), which contains one or more MIDlets. All MIDlets are deployed to devices as suites. MIDlets in the same suite share the same execution environment (virtual machine) and can interact with one another.

Each MIDlet suite JAR file has an associated Application Descriptor (JAD file) used to describe its contents. The application descriptor MIME type is text/vnd.sun.j2me.app-descriptor. The file extension of such an application descriptor must be .jad. The AMS uses the descriptor to manage each MIDlet included
in the suite, that is, to verify that a MIDlet is suited for execution on the device before loading the MIDlet suite JAR file.

3.7 HelloWorld MIDlet

In this section the general structure and design of the simple HelloWorld application is described. In Creating an Application we will study the MIDlet in more detail, and you will personalize it.

The HelloWorld application is a simple MIDlet included with the SDK as an example application. When opened, the application displays a 'Hello World' message on the display of the device. The application is closed by selecting the Exit command of the right softkey.

Figure 3: Hello World application
The structure of the HelloWorld application is depicted below:

Figure 4: Structure of the HelloWorldMIDlet

The HelloWorldMIDlet class extends the javax.microedition.midlet.MIDlet class. It uses the MIDP high-level API javax.microedition.lcdui.TextBox class to enter and edit text, in this case, 'Hello World'.

The TextBox class extends the abstract class Screen, which, in turn, extends the abstract class Displayable. The TextBox UI component has a string contents area that allows users to enter and edit text.

The HelloWorldMIDlet class also implements the CommandListener interface used to receive user input from the command area.

The functionality of the HelloWorldMIDlet is the following: The startApp() method sets the MIDlet’s current Display to the HelloWorldMIDlet object, if a current display does not already exist. If the HelloWorldMIDlet object receives an Exit command from the user interface, it calls the commandAction() method of the HelloWorldMIDlet. The HelloWorldMIDlet then handles the MIDlet state transition to a Destroyed state. See below for the HelloWorldMIDlet.java example code.

```java
/* Copyright (c) 2003, Nokia. All Rights Reserved */
package com.series60.examples.lcdui.helloworld;

import javax.microedition.midlet.*;
```
import javax.microedition.lcdui.*;

/**
 * This class illustrates the implementation of a simple MIDlet that
 * displays a "Hello World" message to the screen
 * <p>
 * This class extends the class javax.microedition.midlet.MIDlet. It
 * creates a TextBox object and displays it on screen.
 * <p>
 * Additionally this MIDlet provides a way for the user to exit the
 * MIDlet
 * 
 * @author Nokia
 * @version 1.0
 */

public class HelloWorldMIDlet extends MIDlet implements
CommandListener {

    /** create a hello world string */
    public final static String HELLO_STRING = "Hello World";

    /** create a default value for the priority of a command */
    private final static int DEFAULT_COMMAND_PRIORITY = 1;

    /** the exit button command */
    private final static Command EXIT_COMMAND = new Command("Exit",
        Command.EXIT, DEFAULT_COMMAND_PRIORITY);

    /**
     * Creates the new text box, then set it to
     * current active component on screen
     */
    public void startApp() {
        /*
         * Create a textbox
         * titled HelloWorldMIDlet
         */

* the initial value "Hello World"
* max input the length of "Hello World"
* and uneditable

```
TextBox textBox = new TextBox("HelloWorldMIDlet",
HELLO_STRING, HELLO_STRING.length(), TextField.UNEDITABLE);

/** Add the exit command to the TextBox */
textBox.addCommand(EXIT_COMMAND);

/** Set this instance to be the event handler */
textBox.setCommandListener(this);

/** Set the display to show the textbox */
Display.getDisplay(this).setCurrent(textBox);
}

/**
 * Performs no operation, because there are no background
 * activities or record stores to be closed.
 */
public void pauseApp() {
}

/**
 * Performs no operation because there is nothing which is not
 * handled
 * by the garbage collector.
 * *
 * @param unconditional ignored.
 */
public void destroyApp(boolean unconditional) {
}

/**
 * This is performed when a button is pressed
public void commandAction(Command cmd, Displayable source) {
    if (cmd == EXIT_COMMAND) {
        destroyApp(false);
        notifyDestroyed();
    }
}

Example code: HelloWorldMIDlet.java
4. Creating an Application

In this section we will go through the steps that are needed in creating a MIDP application, a MIDlet, for the Series 60 Platform with the SDK. The idea is, that by the end of this section you will have your own MIDlet running on a Series 60 Platform target device, that is, your own smartphone.

**Note:** The application development described in this guide is done with a minimal development environment, that is, by using a simple editor (Windows Notepad) and command prompts.

More robust development environments that you might want to use include, for example, Borland JBuilder 9. Notice that before starting to use these in connection with SDK, you will need to configure them so that you can run the Series 60 Emulator (included with the SDK) with them.

For instructions on how to configure the Series 60 Emulator for the development tools mentioned above, see the *Series 60 MIDP SDK 2.1 Emulator Guide* in the SDK Help.

For instructions on how to use the above mentioned development environment, please refer to the Help documentation provided with the tool in question.

4.1 Basics of creating a MIDlet for Series 60

In this section we will go through the main tasks involved in creating a MIDP application (that is, a MIDlet) with the SDK. These steps will then be covered in more detail in the following sections.

**Note:** The task sequence in this section presumes that you have already installed the SDK and that the prerequisites for installation and use outlined in section *What you will need* and *Checking the Application Development Environment* above have been met. It is also recommended that you have a Series 60 Platform target device to which you can upload the application created in this section. For more detailed information on installation and related requirements, please refer to the *Installation Guide*.

4.1.1 Step 1: Building and running the Hello World MIDlet

The SDK comes with several example MIDP applications that demonstrate various possible applications that you can develop for the Series 60 Platform. Thus, a good way to familiarize yourself with how to use the SDK is to build and run an example application. This will not only guide you through the basic steps of building an application, but will also introduce you to the emulator and its functionality.

In the *Building and running the Hello World MIDlet* section below you will build a simple interactive *Hello World* MIDlet and run it in the emulator. With the emulator you can see how the application would run on a real phone.
Although this application is very basic in that it displays only a simple string on the phone screen, it can be easily adapted to make more complex applications. In fact, you will personalize it in the next step below.

4.1.2 Step 2: Creating your own application from Hello World MIDlet

In the Creating your own application from the Hello World MIDlet section below you will go through the basic process of creating a MIDlet by editing the Hello World

After building an application and running it in the emulator, typically the next phase in application development is testing and debugging before uploading it to your phone. This is sometimes referred to as the ‘edit/compile/build cycle’.

4.1.3 Step 3: Packaging the modified Hello World MIDlet

Before installing the MIDlet to your phone you will first need to package it for installation. This involves creating a manifest file, application JAR File and application JAD file.

4.1.4 Installing your application to your phone

Although a simple procedure as such, installing a MIDlet to a Series 60 phone varies depending on the specific smartphone model. The Installing your application to your phone section will, nevertheless, provide general instructions on how to install a MIDlet to your phone as well as an example on how to do this by using IrDA (infrared).

4.2 Building and running the Hello World MIDlet

This section describes how compile, preverify and run the simple HelloWorld example MIDlet.

4.2.1 Compiling the HelloWorld MIDlet

To compile the HelloWorld example MIDlet, do the following:

1. In the Windows Start menu, open the Command Prompt by choosing Start > Programs > Accessories > Command Prompt.

2. In the Command Prompt, use the mkdir command to create a directory into which the Java compiler should output the classes created in compiling the MIDlet.

Enter the following command in the Command Prompt:

`mkdir C:\tmpclasses`

3. In the Command Prompt, use the cd command to switch to the `s60examples\src\` subdirectory of the directory where you installed the SDK.

Enter, for example, the following command in the Command Prompt:

`cd C:\Series_60_MIDP_SDK_2_1\s60examples\src`
4. Compile the Hello World MIDlet by entering the following command in the Command Prompt:

```
javac -d C:\tmpclasses -classpath C:\Series_60_MIDP_SDK_2_1\lib\*.zip; C:\Series_60_MIDP_SDK_2_1\s60examples\src\com\series60\examples\lcdui\helloworld\HelloWorldMIDlet.java
```

The HelloWorld MIDlet has now been compiled to the tmpclasses directory. Next, you will need to preverify the classes that have thus been created.

### 4.2.2 Preverifying the compiled HelloWorld MIDlet

To allow more efficient memory usage on the target device, checking the code for correctness is done offline by using the preverifier tool. The preverifier checks that the classes are correct and inserts additional information into the class files. The K Virtual Machine looks for this information and uses it for a much simpler verification based on the data collected by the preverifier.

To preverify the HelloWorld MIDlet that you have compiled, do the following:

1. In the Windows Start menu, open the Command Prompt by choosing Start > Programs > Accessories > Command Prompt.
2. In the Command Prompt, use the mkdir command to create a directory into which the preverifier tool should output the classes created in compiling the MIDlet.

   Enter the following command in the Command Prompt:

   ```
mkdir C:\classes
```

3. In the Command Prompt, use the cd command to switch to the \bin subdirectory of the directory where you installed the SDK.

   Enter, for example, the following command in the Command Prompt:

   ```
cd C:\Series_60_MIDP_SDK_2_1\bin
```

4. Preverify the Hello World MIDlet classes by entering the following command in the Command Prompt:

   ```
   preverify -d c:\classes -classpath C:\Series_60_MIDP_SDK_2_1\lib\*.zip C:\tmpclasses
   ```

Now that you have compiled and preverified the HelloWorld MIDlet, you can run and view it in the emulator.

### 4.2.3 Running the HelloWord MIDlet in the emulator

To run the HelloWord MIDlet in the emulator, do the following:

1. In the Windows Start menu, open the Command Prompt by choosing Start > Programs > Accessories > Command Prompt.
2. In the Command Prompt, use the `cd` command to switch to the `bin\` subdirectory of the directory where you installed the SDK.

   Enter, for example, the following command in the Command Prompt:
   
   ```
   cd C:\Series_60_MIDP_SDK_2_1\bin
   ```

3. Run the Hello World MIDlet in the emulator by entering the following command in the Command Prompt:

   ```
   emulator -classpath C:\classes com.series60examples.lcdui.helloworld.HelloWorldMIDlet
   ```

   The emulator appears on the screen and displays the Hello World string:

   ![Image of Hello World application in the emulator]

   **Figure 5: Hello World application in the emulator**

   To simulate pressing buttons on a real device, click the corresponding keys on the emulator with the mouse. Note that the **Close** and **Exit** commands also close the emulator.

   Selecting the button takes you to the emulator's content display screen, the Applications view, from where you can explore the emulator and its functionality. The emulator works like a real Series 60 device: click the keys in the emulator as would press the keys on a real device. Use the five-way navigation key to select applications and open them by clicking **OK** or **Select** under the left soft key. Exiting applications happens typically by clicking the right soft key.

   In this section you have compiled and preverified the Hello World MIDlet, and also seen what the **Hello World** application would look like in a real phone.

   The next step would be to install it to a target device, but before doing this you will first personalize and tailor the application. These procedures are covered in the next section.
4.3 Creating your own application from the Hello World MIDlet

In this section you will tailor the Hello World MIDlet into a new, personalized application of your own by editing the source code of the MIDlet. After editing the code, you will compile and view it in the emulator as described in the previous section.

In order to be able to edit the source code, you need to open it in an editor. To open the source in a WordPad editor, do the following:

1. In the Windows Start menu, open the Command Prompt by choosing Start > Programs > Accessories > Command Prompt.

2. In the Command Prompt, use the cd command switch to the s60examples\src\com\series60\examples\lcdui\helloworld subdirectory of the directory where you installed the SDK. Enter, for example, the following command:

   `cd C:\Series_60_MIDP_SDK_2_1\s60examples\src\com\series60\examples\lcdui\helloworld`

3. Open the HelloWorld MIDlet source file in the WordPad editor by entering the following command:

   `write HelloWorldMIDlet.java`

   The HelloWorldMIDlet.java source file opens in WordPad:

   ![Figure 6: HelloWorld MIDlet in WordPad editor](image)

4. To change the text displayed in on the screen in the HelloWorld application, you first need to locate it in the source file. Select Edit > Find... from the menu bar (or press CTRL + F). The Find dialog box is displayed:
5. In the **Find** dialog box, enter the text `HELLO_STRING = "Hello World"` in the **Find what** field and click the **Find Next** button.

   The cursor will go to the ‘Hello World’ string in the HelloWorldMIDlet.java source file:
   ```java
   public final static String HELLO_STRING = "Hello World";
   ```

6. Close the **Find** dialog box by clicking **Cancel**.

7. Edit the source code, for example, as follows:
   ```java
   public final static String HELLO_STRING = "Just Java it!";
   ```

8. To save your changes, select **File > Save** and click **Yes** to confirm the command.

9. Compile, preverify and run your new MIDlet as described in section **Building and running the Hello World MIDlet**.

---

**Note:** As you have already created the `tmpclasses` and `classes` directories, you can skip the `mkdir` commands when compiling, preverifying and running your MIDlet.
Once you have compiled and preverified the new MIDlet, you can run it on the emulator. The emulator should now display the following:

![Image of the emulator displaying the MIDlet](image)

Figure 8: Just Java It! application

Once you have tested the application on the emulator and you are satisfied with it, the next step is to package your MIDlet as described in the following section.

4.4 Packaging the modified Hello World MIDlet

Before installing your application to a target device, you need to package the MIDlet. This involves creating a manifestation file, Java archive (JAR) file and Java application descriptor (JAD) file. You can use any text editor for editing the files.

4.4.1 Creating the manifestation file

1. In the Windows Start menu, open the Command Prompt by choosing Start > Programs > Accessories > Command Prompt.
2. Change to the classes subdirectory where the preverifier created the verified class files. Enter, for example, the following command:
   ```
   cd C:\classes
   ```
3. Create the MANIFEST.MF file by entering, for example, the following command:
   ```
   edit MANIFEST.MF
   ```
   The manifest file opens in the editor:
4. Enter the following information into the manifest file that you are creating:

   Manifest-Version: 1.0
   MicroEdition-Configuration: CLDC-1.0
   MIDlet-Name: HelloWorldMIDlet
   MIDlet-Vendor: Nokia
   MIDlet-1: HelloWorldMIDlet, ,
             com.nokia.examples.lcdui.helloworld.HelloWorldMIDlet
   MIDlet-Version: 1.0
   MicroEdition-Profile: MIDP-2.0

   The MANIFEST.MF file should now look like this in the editor:

   Figure 10: MANIFEST.MF file with needed information

5. Select File > Save to save the edited MF file.
6. Select File > Exit to close the editor.

4.4.2 Creating the application JAR File

1. In the Windows Start menu, open the Command Prompt by choosing Start > Programs > Accessories > Command Prompt.
2. Change to the classes subdirectory where the preverified class files are located and where you created the MANIFEST.MF file.
   Enter, for example, the following command:
cd C:\classes

3. Create a JAR file named HelloWorldMIDlet.jar which includes the manifest file, the MIDlet classes and the resource files. Enter, for example, the following command:

```bash
jar -cmf MANIFEST.MF HelloWorldMIDlet.jar com\series60\examples\lcdui\helloworld\*.class
```

The JAR tool creates the file HelloWorldMIDlet.jar to the classes subdirectory.

### 4.4.3 Creating the JAD file

1. In the Windows Start menu, open the Command Prompt by choosing Start > Programs > Accessories > Command Prompt.
2. Change to the classes subdirectory by entering, for example, the following command:

```bash
cd C:\classes
```
3. Create a JAD file named HelloWorldMIDlet.jad by entering, for example, the following command:

```bash
edit HelloWorldMIDlet.jad
```

**Note:** Make sure that you use the extension .jad in the file.

The manifest file opens in the editor.

4. Enter the following information into the JAD file that you are creating:

```plain
MIDlet-Version: 1.0
MIDlet-Vendor: Nokia
MIDlet-Jar-URL: HelloWorldMIDlet.jar
MicroEdition-Configuration: CLDC-1.0
MIDlet-1: HelloWorldMIDlet,,
com.nokia.examples.lcdui.helloworld.HelloWorldMIDlet
MicroEdition-Profile: MIDP-2.0
MIDlet-Jar-Size: 1276
MIDlet-Name: HelloWorldMIDlet
```

**Note:** Make sure that you enter the exact byte size of the HelloWorldMIDlet.jar file as the MIDlet-Jar-Size.
The HelloWorldMIDlet.jad file should now look like this in the editor:

![Image of HelloWorldMIDlet.jad file]

Figure 11: HelloWorldMIDlet.jad file with needed information

5. Select File > Save to save the edited JAD file.
6. Select File > Exit to close the editor.

Now that you have created the package MIDlet, you can install the application to the target device, that is, your smartphone.

4.5 Installing your application to your phone

Installing the MIDlet involves transferring the .jad and .jar files which you created in the previous section to your phone. Follow the instructions provided in your phone’s user manual on how to install new applications as the installation procedure varies depending on the device. The steps below provide an example on how to install a MIDlet by using IrDA (infrared).

**Note:** Installing the MIDlet requires that your phone supports the open J2ME MIDP platform.

You must also have built-in IrDA hardware and Infrared and Wireless Link installed on your computer. If you do not have built-in IrDA, see your computer manual for more details on how to install an Infrared and Wireless Link.

4.5.1 Example of how to install the MIDlet

1. Make sure that the infrared ports of the sending and receiving devices are pointing at each other and that there are no obstructions between the devices.

2. Activate the infrared port of your phone to receive data via an infrared connection by selecting, for example, Menu > Connectivity > Infrared. The infrared connection status icon is displayed on your phone for the duration of the connection.

3. Open the Windows Explorer by selecting Start > Programs > Accessories > Windows Explorer on your computer.
4. Go to the classes\ subdirectory of the directory where you created the JAR and JAD files HelloWorldMIDlet.jar and HelloWorldMIDlet.jad.

5. Select the files HelloWorldMIDlet.jar and HelloWorldMIDlet.jad.

6. Right-click and select **Send To > Infrared Recipient**

   The installation files are transferred to your phone from a computer via infrared. Your phone should receive the installation files as a message attachment.

7. Go to the messaging folder in your phone and open the HelloWorldMIDlet.jad message.

8. If requested, confirm the installation command.

   A notification is displayed when the installation is complete.

9. Go to the content display and start the HelloWorldMIDlet Java application view.

   The HelloWorldMIDlet starts.
5. Moving On

In personalizing and implementing the Hello World application as instructed above, you basically edited a fairly simple existing example application. The aim of this exercise has been to provide you with an insight into the basics of application development. The end result is, hopefully, that you have grasped the basic features and possibilities of the SDK, the emulator, and the general outline of the steps involved in application development.

In personalizing the Hello World, you have only scratched the surface of the various possibilities of application development with the SDK. In order to start developing more complex applications, you will first need a deeper understanding of both Java technology and Series 60 Platform features.

The Developer Guides section of the SDK Help provides useful material for learning more about Java application development with the Series 60 SDK. We recommend that you start with the following documents:

- **Series 60 Platform Basics**
  Provides basic information on the Series 60 platform, Symbian OS and the related technologies.

- **Developer Platform 2.0 for Series 60: Designing Java Applications**
  Provides more detailed information on Java application development, the UI classes, high and low-level UI APIs, graphics, timers and threads, as well as overviews of networking MIDlets, persistent storage, Mobile Media API and the Push Registry.
6. Developer Resources

6.1 SDK Help

The SDK Help provides information and instructions that needed when working with the Series 60 MIDP Software Development Kit (SDK) to create new Series 60 applications with Java. The Help contains the needed documentation on Application Programming Interfaces (API) and example applications as well as a variety of developer guides to facilitate application development.

6.2 Sun Java developer site

The Java Developer Home website at http://developers.sun.com/index.html provides an abundance of information and support on Java technology. The wireless pages (http://developers.sun.com/techtopics/mobility/), also provided by Sun technologies Inc., is the definitive resource for developers seeking to create, test, certify, and market Java mobility applications.

6.3 Symbian Developer Network

The Symbian Developer Network (Symbian DevNet) is a primary resource for access to the technical and commercial information and resources developers need to succeed in the wireless space. The Symbian Developer Network is Symbian’s developer service organization, providing developers using Symbian OS with support and information to help them produce well-written and stable applications for Symbian OS devices.

Symbian DevNet provides:

- Discussion forums, code resources, and tips about Symbian OS development
- A portal to other such resources and technical and commercial services on Symbian DevNet partner sites
- A newsletter on what's new and hot on the Symbian DevNet Web site and across all our partner sites
- The Symbian OS Knowledgebase that provides up-to-the-minute information about Symbian OS such as:
  - Answers to frequently asked questions
  - Defect reports and the corresponding solutions
  - Useful code snippets illustrating Symbian OS techniques
  - Other development information that is not available in the Software Development Kit documentation

The Symbian Developer Network provides both free public discussion forums at:
www.symbian.com/developer/support.html

Professional support forums for subscribing members at:
The Symbian Developer Network Web site includes a technical library that can be found at:
www.symbian.com/developer/techlib/index.html
## 7. Terms and Abbreviations

<table>
<thead>
<tr>
<th>Term or abbreviation</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>API</td>
<td>Application Programming Interface. System function libraries and other facilities that enable developers to produce applications that are suitable for a particular platform or device.</td>
</tr>
<tr>
<td>Canvas</td>
<td>A low-level UI component of LCDUI.</td>
</tr>
<tr>
<td>CLDC</td>
<td>Connected Limited Device Configuration. A J2ME configuration used in MIDP.</td>
</tr>
<tr>
<td>Displayable</td>
<td>An LCDUI object that has the capability of being placed on display.</td>
</tr>
<tr>
<td>IDE</td>
<td>Integrated Development Environment</td>
</tr>
<tr>
<td>JAD</td>
<td>Java Application Descriptor. The .jad file includes information about the MIDlet.</td>
</tr>
<tr>
<td>JAM</td>
<td>Java Application Manager. A MIDP system component used for managing MIDP applications on a mobile device.</td>
</tr>
<tr>
<td>JAR</td>
<td>Java Archive. A single file (.jar) format that includes the required class and resource files for one or several applications.</td>
</tr>
<tr>
<td>KVM</td>
<td>Kilobyte Virtual Machine. A virtual machine used in MIDP</td>
</tr>
<tr>
<td>LCDUI</td>
<td>Limited Capability Device User Interface. An UI toolkit in MIDP.</td>
</tr>
<tr>
<td>MIDP</td>
<td>Mobile Information Device Profile. Specifications that define minimum hardware, software, networking, application requirements, as well as standard system APIs for devices that support MIDP. MIDP is intended for “connected” devices such as smartphones, which are characterized by having limited Central Processing Unit (CPU), memory, keyboard, and display capabilities.</td>
</tr>
<tr>
<td>Screen</td>
<td>A subclass of the Displayable class. The Screen is an abstract superclass for all high-level UI components in MIDP LCDUI.</td>
</tr>
<tr>
<td>SDK</td>
<td>Software Development Kit</td>
</tr>
<tr>
<td>UI</td>
<td>User Interface</td>
</tr>
</tbody>
</table>