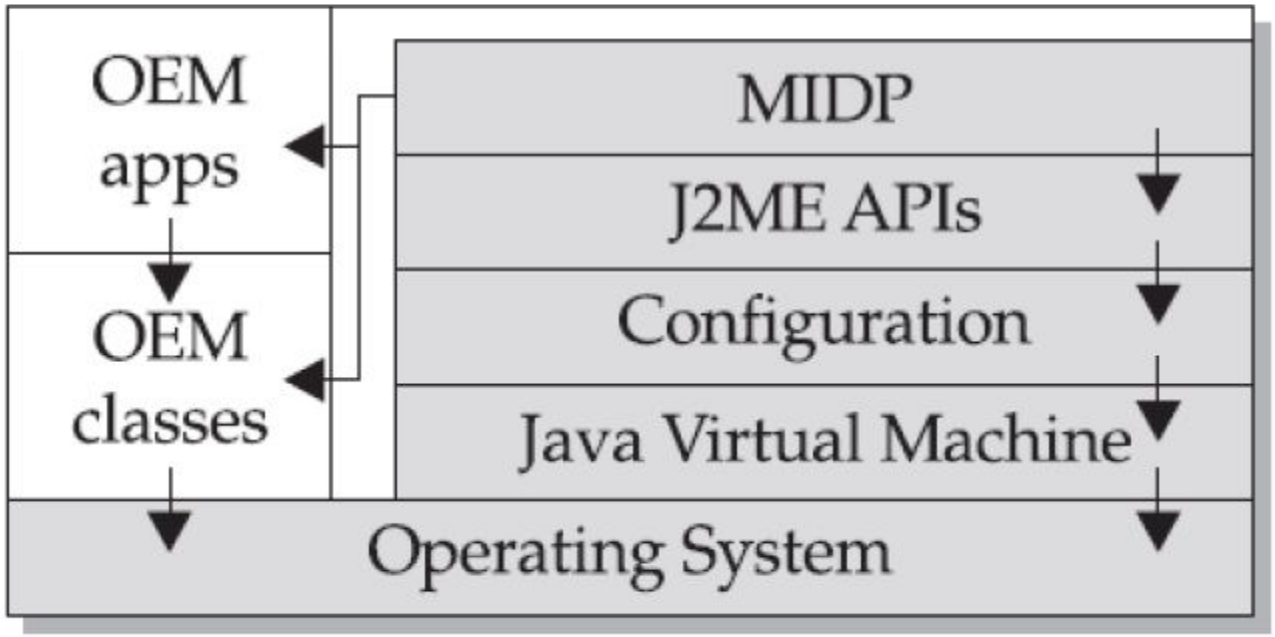
J2ME
Architecture and Development Environment

Jawaharlal Nehru Technological University Hyderabad

J2ME Architecture



Layers of the J2ME architecture

J2ME Architecture

J2ME architecture consists of layers located above the native operating system, collectively referred to as the **Connected Limited Device Configuration (CLDC).**

The J2ME architecture comprises three software layers.

- The first layer is the **configuration layer that includes the Java** Virtual Machine (JVM), which directly interacts with the native operating system. The configuration layer also handles interactions between the profile and the JVM.

- The second layer is the **profile layer, which consists of the minimum** set of application programming interfaces (APIs) for the small computing device.

- The third layer is the **Mobile Information Device Profile (MIDP).** The MIDP layer contains Java APIs for user network connections, persistence storage, and the user interface. It also has access to CLDC libraries and MIDP libraries.

A small computing device has two components supplied by the original equipment manufacturer (OEM). **These are classes and applications.**

- **OEM classes are used by the MIDP to access device-specific features** such as sending and receiving messages and accessing device-specific persistent data.

- **OEM applications are programs provided by the OEM, such as an** address book. OEM applications can be accessed by the MIDP.

Small Computing Device Requirements

a) Minimum resource requirements to run a J2ME application

First the device must have a minimum of **96 × 54 pixel display** that can handle bitmapped graphics and have a way for users to input information, such as a keypad, keyboard, or touch screen.

At least **128 KB of nonvolatile memory is necessary to run** Mobile Information Device (MID), and **8KB of nonvolatile memory is needed for storage of persistent application data.**

To run JVM, **32KB of volatile memory must be available.** The device must also provide two-way network connectivity.

b) Minimal hardware requirements for the native operating System

The native operating system must implement exception handling, process interrupts, be able to run the JVM, and provide schedule capabilities.

Run-Time Environment

- A MIDlet is a J2ME application designed to operate on an MIDP small computing device.
- A MIDlet is defined with at least a single class that is derived from **the javax.microedition.midlet.MIDlet abstract class**
- Developers commonly bundle related MIDlets into a MIDlet suite
- All MIDlets within a MIDlet suite are considered a group and must be installed and uninstalled as a group

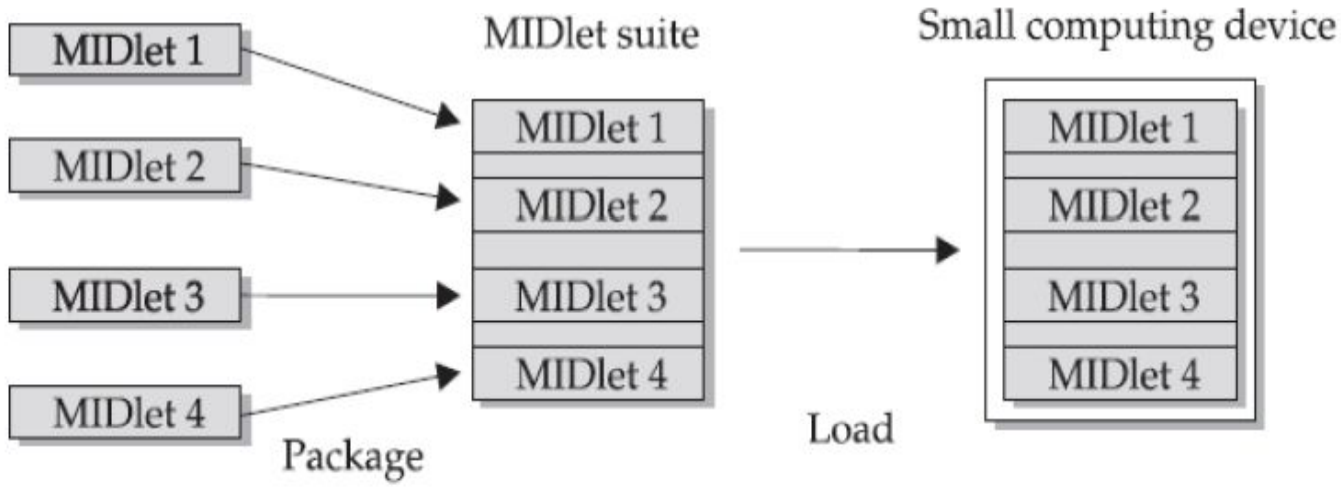
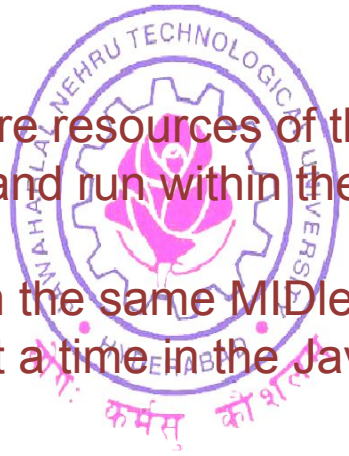


Figure 3-2. *MIDlets are packaged into MIDlet suites, which are loaded in a small computing device.*

Runtime Environment



- Members of a MIDlet suite share resources of the host environment and share the same instances of Java classes and run within the same JVM.

-This means if three MIDlets from the same MIDlet suite run the same class, only one instance of the class is created at a time in the Java Virtual Machine.

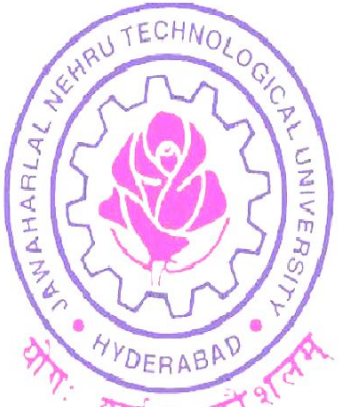
Inside the Java Archive File

Manifest File Attribute	Description
MIDlet-Name	MIDlet suite name.
MIDlet-Version	MIDlet version number.
MIDlet-Vendor	Name of the vendor who supplied the MIDlet.
MIDlet-n	Attribute per MIDlet. Values are MIDlet name, optional icon, and MIDlet class name.
MicroEdition-Profile	Identifies the J2ME profile that is necessary to run the MIDlet.
MicroEdition-Configuration	Identifies the J2ME configuration that is necessary to run the MIDlet.
MIDlet-Icon	Icon associated with MIDlet, must be in PNG image format (optional).
MIDlet-Description	Description of MIDlet (optional).
MIDlet-Info-URL	URL containing more information about the MIDlet.

Table 3-1. *Attributes of the Manifest File*

```
MIDlet-Name: Best MIDlet
MIDlet-Version: 2.0
MIDlet-Vendor: MyCompany
MIDlet-1: BestMIDlet, /images/BestMIDlet.png, Best.BestMIDlet
MicroEdition-Profile: MIDP-1.0
MicroEdition-Configuration: CLDC-1.0
```

Inside the Java Application Descriptor File

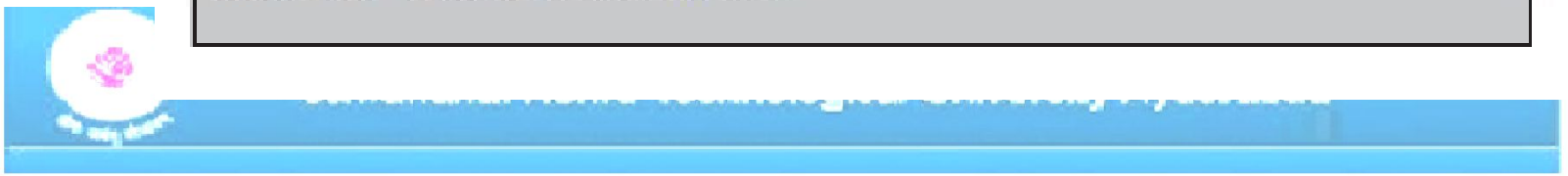


MIDlet-Name
MIDlet-Version
MIDlet-Vendor
MIDlet-n
MIDlet-Jar-URL

```
MIDlet-Name: Best MIDlet
MIDlet-Version: 2.0
MIDlet-Vendor: MyCompany
MIDlet-Jar-URL: http://www.mycompany.com/bestmidlet.jar
MIDlet-1: BestMIDlet, /images/BestMIDlet.png, Best.BestMIDlet
```

JAD File Attribute	Description
MIDlet-Name	MIDlet suite name.
MIDlet-Version	MIDlet version number.
MIDlet-Vendor	Name of the vendor who supplied the MIDlet.
MIDlet-*n*	Attribute per MIDlet. Values are MIDlet name, optional icon, and MIDlet class name.
MIDlet-Jar-URL	Location of the JAR file.
MIDlet-Jar-Size	Size of the JAR file in bytes (optional).
MIDlet-Data-Size	Minimum size (in bytes) for persistent data storage (optional).
MIDlet-Description	Description of MIDlet (optional).
MIDlet-Delete-Confirm	Confirmation required before removing the MIDlet suite (optional).
MIDlet-Install-Notify	Send installation status to given URL (optional).

Table 3-2. *Attributes for a JAD File*



MIDlet Programming

A MIDlet is a class that extends the MIDlet class and is the interface between application statements and the run-time environment, which is controlled by the application manager.

A MIDlet class must contain **three abstract methods that are called by** the application manager to manage the life cycle of the MIDlet.

These abstract methods are

- **startApp()**
- **pauseApp(),**
- **destroyApp().**

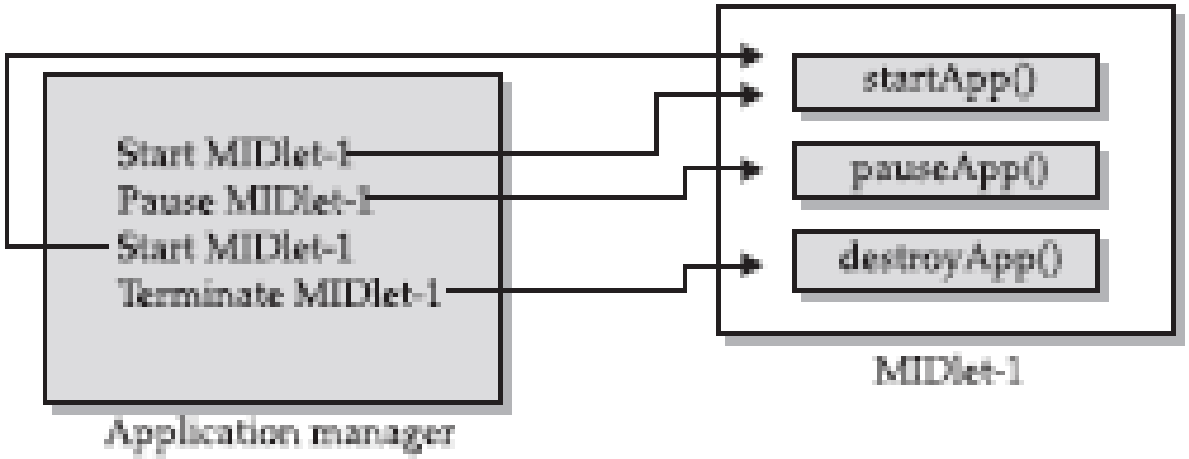
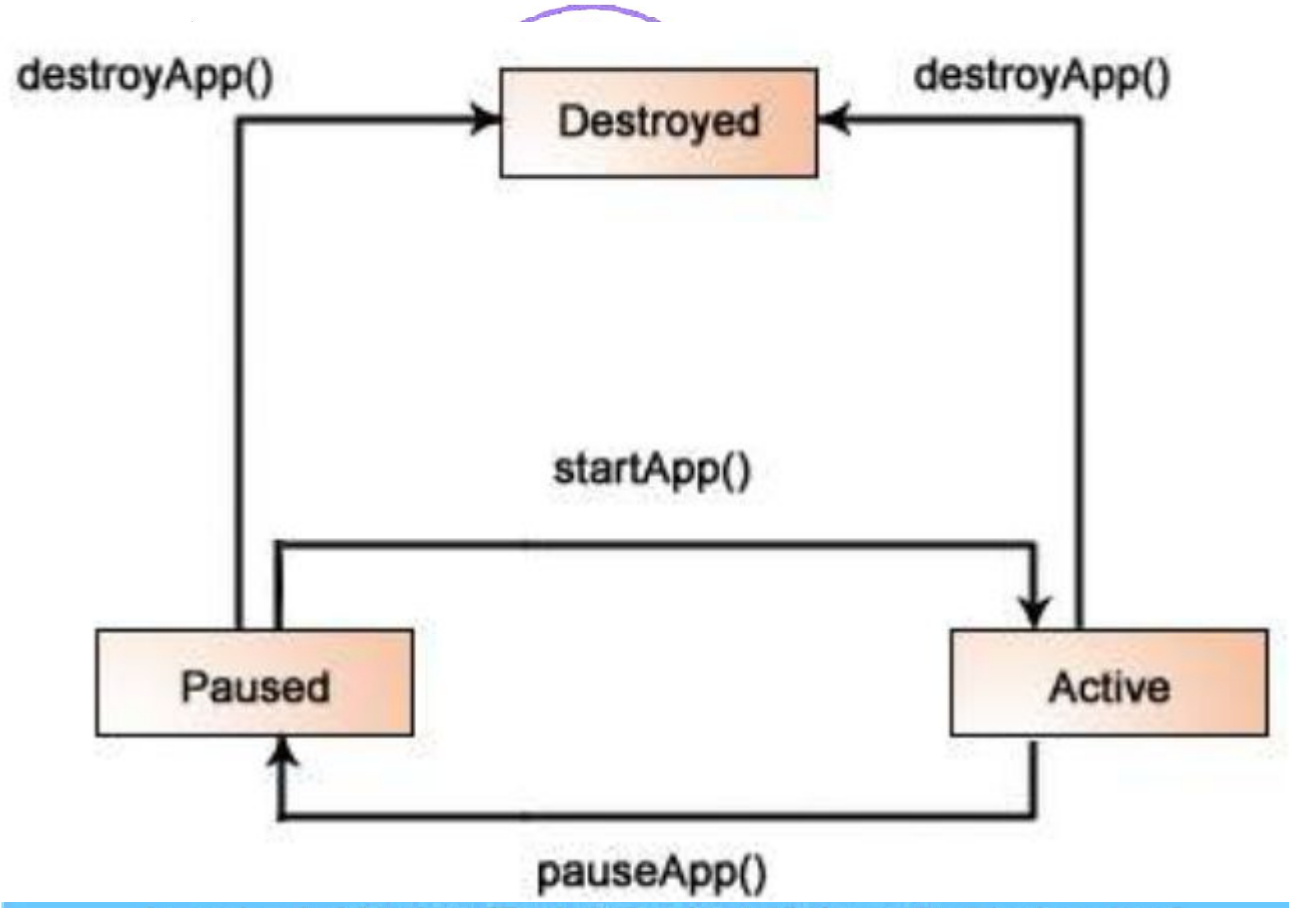


Figure 3-3. *The application manager calls methods of a MIDlet.*

MIDlet life cycle



MIDlet structure

```
public class BasicMIDletShell extends MIDlet
{
public void startApp()
{
}
public void pauseApp()
{
}
public void destroyApp( boolean unconditional)
{
}
}
```

The startApp() is called by the application manager when the MIDlet is started and contains statements that are executed each time the application begins execution

The pauseApp() is called before the application manager temporarily stops the MIDlet. The application manager restarts the MIDlet by recalling the startApp() method.

The destroyApp() is called prior to the termination of the MIDlet by the application manager.

The MIDP API classes used by the MIDlet to interact with the user & Handle data management.

User interactions are managed by user interface MIDP API classes.

These APIs enable a developer to display screens of data and prompt the user to respond with an appropriate command

The **command causes the MIDlet to execute one of three routines:**
a. Perform a computation,
b. Make a network request/ display another screen.

The **data-handling MIDP API classes enable the developer to perform** four kinds of data routines:

a. write and read persistent data,
b. store data in data types,
c. receive data from and send data to a network,
d. Interact with the small computing device's input/output features.

Java Language for J2ME

MIDlet cannot use any floating-point data types or calculations

Small computing device are too scarce to process the finalize() method

JVM for small computing devices requires a custom class loader

You cannot group threads

JVM uses class file verification this process is replaced with a two-step process

- preverification
- MIDlet class is load

JVM for small computing devices requires a custom class loader that is supplied by the device manufacturer

The number of error-handling exceptions are trimmed

System Classes		
java.lang.Class	java.lang.Runtime	java.lang.System
java.lang.Object	java.lang.String	java.lang.Thread
java.lang.Runnable	java.lang.StringBuffer	java.lang.Throwable
Data Type Classes		
java.lang.Boolean	java.lang.Character	java.lang.Long
java.lang.Byte	java.lang.Integer	java.lang.Short
Collection Classes		
java.util.Enumeration	java.util.Stack	
java.util.Hashtable	java.util.Vector	
Input/Output Classes		
java.io.ByteArrayInputStream	java.io.DataOutputStream	java.io.PrintStream
java.io.ByteArrayOutputStream	java.io.InputStream	java.io.Reader
java.io.DataInput	java.io.InputStreamReader	java.io.Writer
java.io.DataInputStream	java.io.OutputStream	
java.io.DataOutput	java.io.OutputStreamWriter	
Calendar and Time Classes		
java.util.Calendar	java.util.Date	java.util.TimeZone

Table 3-3. *J2ME Support Classes*

Utility Classes		
java.lang.Math	java.util.Random	
Exception Classes		
java.io.EOFException	java.lang.ClassNotFoundException	java.lang.NegativeArraySizeException
java.io.InterruptedIOException	java.lang.Exception	java.lang.NullPointerException
java.io.IOException	java.lang.IllegalAccessException	java.lang.NumberFormatException
java.io.UnsupportedEncodingException	java.lang.IllegalArgumentException	java.lang.RuntimeException
java.io.UTFDataformatException	java.lang.IllegalMonitorStateException	java.lang.SecurityException
java.lang.ArithmeticException	java.lang.IllegalThreadStateException	java.lang.StringIndexOutOfBoundsException
java.lang.ArrayIndexOutOfBoundsException	java.lang.IndexOutOfBoundsException	java.util.emptyStackException
java.lang.ArrayStoreException	java.lang.InstantiationException	java.util.NoSuchElementException
java.lang.ClassCastException	java.lang.InterruptedException	
Error Classes		
java.lang.Error	java.lang.OutOfMemoryError	java.lang.VirtualMachineError
Internationalization		
java.io.InputStreamReader	java.ioOutputStreamWriter	

Table 3-3. *J2ME Support Classes* (continued)

J2ME Software Development Kits

Once the Java development kit is installed, place the c:\jdk\bin directory, or whatever directory you selected for the Java development kit, on the PATH environment variable (see “Setting the Path inWindows” sidebar). This enables you to invoke the Java compiler from anywhere on your computer

Setting the Path in Windows

Windows 2000 and Windows NT

1. Choose System from the Control Panel.
2. Select Environment or Advanced/Environment.
3. Locate the PATH environment variable.
4. Enter the directory at the end of the path. Be sure to separate entries with a semicolon.

Windows 98 and Windows 95

1. Select Start.
2. Select Run.
3. Enter **sysedit**.
4. Select OK.
5. Locate the autoexec.bat dialog box.
6. Add the directory to the PATH environment variable.

J2ME Software Development Kits

Install the CLDC once the Java development kit is installed. Unzip the downloaded CLDC files from the java.sun.com web site onto the d:\j2me directory (J2ME_HOME) on your computer.

Next, download and unzip the MIDP file. Be sure to use \j2me as the directory for the MIDP file

create two environment variables. These are CLASSPATH and MIDP_HOME. The CLASSPATH environment variable identifies the path to be searched whenever a class is invoked

Set the CLASSPATH to
d:\j2me\midp1.0.3fcs\classes;.

Set the MIDP_HOME environment variable to
d:\j2me\midp1.0.3fcs

Hello World J2ME Style

```
package greeting;
import javax.microedition.midlet.*;
import javax.microedition.lcdui.*;
public class HelloWorld extends MIDlet implements CommandListener
{
  private Display display ;
  private TextBox textBox ;
  private Command quitCommand;
  public void startApp()
  {
    display   = Display.getDisplay(this);
    quitCommand = new Command("Quit", Command.SCREEN, 1);
    textBox   = new TextBox("Hello World", "My first MIDlet", 40, 0);
    textBox .addCommand(quitCommand);
    textBox .setCommandListener(this);
    display .setCurrent(textBox );
  }
```

Hello World J2ME Style

```
    public void pauseApp()
    {
    }
    public void destroyApp(boolean unconditional)
    {
    }
    public void commandAction(Command choice, Displayable displayable)
    {
      if (choice == quitCommand)
      {
        destroyApp(false);
        notifyDestroyed();
      }
    }
}
```

Compiling Hello World

javac -d d:\j2me\tmp_classes -target 1.1 -bootclasspath d:\j2me\midp1.0.3fcs\classes HelloWorld.java

preverify -d d:\j2me\classes -classpath d:\j2me\midp1.0.3fcs\classes d:\j2me\tmp_classes

preverify -d d:\j2me\classes d:\j2me\tmp_classes

Running Hello World

midp -classpath d:\j2me\classes greeting.HelloWorld

Deploying Hello World

Manifest file


```
MIDlet-1: HelloWorld, , greeting.HelloWorld


MIDlet-Name: Hello World
MIDlet-Version: 1.0
MIDlet-Vendor: Jim
MIDlet-1: HelloWorld, /greeting/myLogo.png, greeting.HelloWorld
MicroEdition-Configuration: CLDC-1.0
MicroEdition-Profile: MIDP-1.0
```

jar -cfvm d:\j2me\midlets\HelloWorld.jar manifest.txt -C d:\j2me\classes greeting

Hello World J2ME Style

JAD file

```
MIDlet-Name: Hello World
MIDlet-Version: 1.0
MIDlet-Vendor: Jim
MIDlet-Description: My First MIDlet suite
MIDlet-1: HelloWorld, /greeting/myLogo.png, greeting.HelloWorld
MIDlet-Jar-URL: HelloWorld.jar
MIDlet-Jar-Size: 1428
```

midp -classpath HelloWorld.jar -Xdescriptor HelloWorld.jad

Multiple MIDlets in a MIDlet Suite

Multiple MIDlets are distributed in a single MIDlet suite

The new MIDlet is called GoodbyeWorld and is shown in next Slide

Multiple MIDlets in a MIDlet Suite

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

public class GoodbyeWorld extends MIDlet implements CommandListener
{
private Display display ;
private TextBox textBox ;
private Command quitCommand;
public void startApp()
{
display = Display.getDisplay(this);
quitCommand = new Command("Quit", Command.SCREEN, 1);
textBox = new TextBox("Goodbye World", "My second MIDlet", 40, 0);
textBox .addCommand(quitCommand);
textBox .setCommandListener(this);
display .setCurrent(textBox );
}
```

Multiple MIDlets in a MIDlet Suite

```
public void pauseApp()
{
}
public void destroyApp(boolean unconditional)
{
}
public void commandAction(Command choice, Displayable displayable )
{
if (choice == quitCommand)
{
destroyApp(false);
notifyDestroyed();
}
}
}
```

Multiple MIDlets in a MIDlet Suite

Compile both the HelloWorld.java and GoodbyeWorld.java files by entering the following command at the command line.

```
javac -d d:\j2me\tmp_classes -target 1.1 –bootclasspath d:\j2me\midp1.0.3fcs\classes *.java
```

Preverify these files by entering the following command at the command line:

```
preverify -d d:\j2me\classes -classpath d:\j2me\midp1.0.3fcs\classes d:\j2me\tmp_classes
```

Create the HelloWorld.jar file by entering the following command. Make sure that the j2m/src/greeting directory is the current directory.

```
jar -cfvm d:\j2me\midlets\HelloWorld.jar manifest.txt -C d:\j2me\classes greeting
```

To run the J2ME application

```
midp -classpath HelloWorld.jar -Xdescriptor HelloWorld.jad
```

Multiple MIDlets in a MIDlet Suite

Manifest file

MIDlet-Name: Hello World

MIDlet-Version: 1.0

MIDlet-Vendor: Jim

MIDlet-1: HelloWorld, /greeting/myLogo.png, greeting.HelloWorld

MIDlet-2: GoodbyeWorld, /greeting/myLogo.png, greeting.GoodbyeWorld

MicroEdition-Configuration: CLDC-1.0

MicroEdition-Profile: MIDP-1.0

Multiple MIDlets in a MIDlet Suite

JAD File

MIDlet-Name: Hello World

MIDlet-Version: 1.0

MIDlet-Vendor: Jim

MIDlet-Description: My First MIDlet suite

MIDlet-1: HelloWorld, /greeting/myLogo.png, greeting.HelloWorld

MIDlet-2: GoodbyeWorld, /greeting/myLogo.png, greeting.GoodbyeWorld

MIDlet-Jar-URL: HelloWorld.jar

MIDlet-Jar-Size: 4048