Developing a 2D game in Java ME - Part 1

This article provides a quick introduction to the Java ME development tools, and provides a step-by-step guide to creating your first application, also known as a MIDlet. This is the first in a series of articles that covers all the basics of developing an application for mobile devices using Java ME, learning the main base libraries, classes, and methods available in Java ME.

Introduction

24 Aug 2008

So you want to develop applications for your mobile phone? Say hello to Java Micro Edition. Java ME combines a small Java virtual machine (VM) and a set of Java APIs for developing applications for mobile devices.

This article is the first in a series approaching all the basics of developing an application for mobile devices using Java ME. The final goal of these tutorials is to develop a small Arkanoid clone (http://en.wikipedia.org/wiki/Arkanoid) and, during the process, learn the base libraries, classes, and methods available in Java ME.

This article, after a quick introduction to the development tools, provides a step-by-step guide to creating your first Java ME application, also known as a MIDlet.

Development tools

The development environment you use has a big influence on your productivity.

After trying out a lot of different IDEs, I narrowed my options to the following two:

- Eclipse with the Mobile Tools for Java plug-in.
- Netbeans with the Mobility Pack.

Both IDEs are completely free of charge and have a strong community backing them up. They allow you to do all tasks you need to do to develop mobile applications.

Netbeans is very easy to install and configure and its interface is more intuitive than Eclipse, but it uses a lot of memory and it can get very slow when your project gets bigger.

Eclipse is a lot faster and has a smaller memory footprint. It has the same features as Netbeans, but its interface is more awkward to use and it requires more work for configuration.

So, what's the best for you? If you already have experience with Eclipse, its smaller memory footprint and lightning speed are a big advantage, but if you are a beginner with Java IDEs and mobile programming, Netbeans is a lot easier to start with.

Phone SDKs

Besides the Java IDE with mobility support, you need to download and install a special software, the phone SDK, that gives you access to the specific Java ME libraries.

Oracle Java ME SDK is available as a plugin to NetBeans and Eclipse IDEs. It can be downloaded from http://www.oracle.com/technetwork/java/javame/javamobile/download/sdk/index.html. Once downloaded, you can use the instructions given on that site to integrate it with your IDE. It includes both Java ME SDK CLDC emulator and Java ME SDK IMP-NG emulator.

Sun also has an emulator called Wireless Toolkit (WTK) that provides all the standard libraries. It is very fast, it has a lot of extra tools (Network Monitor, Profiler, SMS Emulator, etc.), and it can be customized to look and act like several phones. Oracle recommends using newer feature-rich Java ME SDK 3.0.

Besides these, it is also a good idea to have some SDKs from the manufacturers of the phones that you wish to support. These SDKs allow you to access any special libraries that those phones might use and often include software that allows you to deploy your applications to the phone.

For example newest Java SDK for Nokia phones can be found from link: http://www.developer.nokia.com/Develop/Java/Tools/

Your first MIDlet

After you have installed and configured your Java IDE and phone emulator, it is time to start coding.

All the Java ME MIDlets must extend the abstract MIDlet class found in the javax.microedition.midlet package, much like creating an applet by extending the java.applet.Applet class. The base entry point is the `startApp()` method.

(C) Copyright Nokia 2014. All rights reserved.

So lets create our first MIDlet, MyMidlet.

```
public class MyMidlet extends MIDlet{
  // invoked when the application starts and each time when it's resumed
  protected void startApp() throws MIDletStateChangeException {}

  // invoked when the MIDlet needs to be destroyed
  protected void destroyApp(boolean uncondicional) throws MIDletStateChangeException {}

  // invoked when the MIDlet needs to be paused. (Some phones ignore pauseApp().)
  protected void pauseApp() {}
}
```

If you create this class and run it in the emulator, you will have a fantastic blank screen!

In order to show content in your application, you need to use the Display class. This class controls what appears on the screen of the MIDlet. Each MIDlet has one Display and it's accessible through the Display static method `getDisplay()`.

To present something on the screen, you need to set a Displayable object using the `setCurrent()` method. One class that implements Displayable is the Alert class. This class shows a simple message on the screen.

With this information, let's change the `startApp()` method to something more useful.

```
Alert alert;

//entry point for the application
protected void startApp() throws MIDletStateChangeException {

  // creates alert
  alert = new Alert("Hello World");

  // shows alert in the screen.
  Display.getDisplay(this).setCurrent(alert);
}
```

If you run your MIDlet now, you will see a screen with "Hello World" like the one below:

That is a little better, but there is still no way to exit your MIDlet properly. You can, however, always press the end key. In order to have a proper exit, let's add some more lines of code to the startApp() method:

```
Command comExit;
[...]
protected void startApp() throws MIDletStateChangeException {
```


(C) Copyright Nokia 2014. All rights reserved.

```
[...]
  // create command
  comExit = new Command("Exit", Command.EXIT, 1);
  // add a command to exit the Midlet
  alert.addCommand(comExit);
  [...]
}
```

This shows you the exit command on the screen but if you click on it, it doesn't do anything because you need to add a CommandListener to your Alert in order to react to Command Actions:

```
public class MyMidlet extends MIDlet implements CommandListener{

  public void startApp(){
    [...]
    // adds a listener to the alert
    alert.setCommandListener(this);
  }

  public void commandAction(Command cmd, Displayable display) {
    // check what command was selected
    if (cmd == comExit) {
      notifyDestroyed();
    }
  }
}
```

With this change we finally have a complete functional MIDlet.

In the next lesson, we are going to look in more detail at the user interface elements such as the Alert, Display, and CommandListener classes. These elements define the menu interface for the Arkanoid clone.

Downloads

- Source Code
- Binaries

Go to Developing a 2D game in Java ME - Part 2

(C) Copyright Nokia 2014. All rights reserved.