

# Bab 2

## Memulai Pemrograman Mobile

### 2.1 Tujuan

In this section, we will be delving into writing, building, using the emulator and packaging J2ME applications. The Integrated Programming Environment that we will use is NetBeans 4.1 ([www.netbeans.org](http://www.netbeans.org)) and NetBeans Mobility Pack.

Pada bagian ini, kita akan menggali tentang menulis, membangun, menggunakan emulator dan melakukan packaging aplikasi J2ME. Integrated Programming Environment yang akan kita gunakan adalah NetBeans 4.1 ([www.netbeans.org](http://www.netbeans.org)) dan NetBeans Mobility Pack.

Setelah menyelesaikan bagian ini, siswa diharapkan mampu:

- Membuat MIDlet sederhana
- Membuat sebuah project di NetBeans
- Membuat sebuah MIDlet menggunakan NetBeans Mobility Pack
- Menjalankan MIDlet di emulator

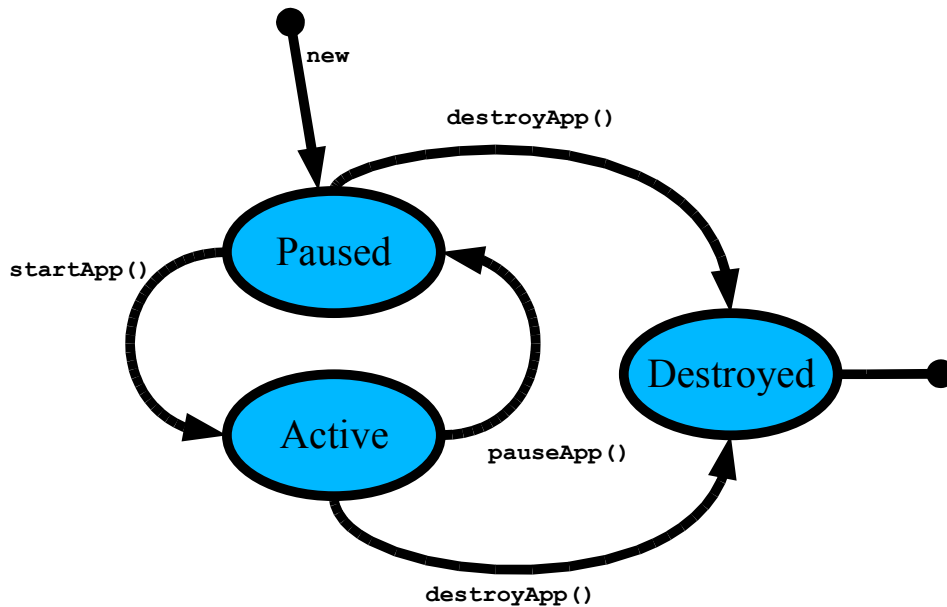
### 2.2 Pengenalan

IDE (Integrated Development Environment) adalah sebuah lingkungan pemrograman (programming environment) yang memiliki GUI builder, text atau code editor, compiler dan/atau interpreter dan debugger. Dalam hal ini, NetBeans Mobility Pack juga memiliki device emulator. Fasilitas ini bisa membuat kita melihat program kita pada device yang sesungguhnya.

### 2.3 "Hello, world!" MIDlet

Kita sudah mempelajari pada bagian sebelumnya tentang daur hidup MIDlet (MIDlet's life cycle). MIDlet mulai hidup ketika MIDlet dibuat oleh Application Management System (AMS) pada device.

Agar kita dapat membuat MIDlet, kita harus membuat subclass dari MIDlet class dari javax.microedition.midlet package. Kita juga harus melakukan override atau implement pada method: startApp(), destroyApp() dan pauseApp(). Method-method tersebut adalah method yang diperlukan oleh AMS untuk menjalankan dan mengontrol MIDlet.



Tidak seperti program Java pada umumnya dimana method main() hanya digunakan sekali pada jalannya program, method startApp() mungkin akan dipanggil lebih dari sekali dalam daur hidup MIDlet. Sehingga Anda diharuskan tidak membuat satu inialisasi code pada method startApp(). Daripada, anda dapat membuat MIDlet constructor dan melakukan inialisasi di situ.

Berikut ini adalah code program MIDP pertama kita:

```

/*
 * HelloMidlet.java
 *
 * Created on July 8, 2000, 9:00 AM
 */

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

/**
 *
 * @author JEDI Apprentice

```

```
* @version
*/
public class HelloMidlet extends MIDlet implements CommandListener {
    Display display;
    Command exitCommand = new Command("Exit", Command.EXIT, 1);
    Alert helloAlert;

    public HelloMidlet(){
        helloAlert = new Alert(
            "Hello MIDlet", "Hello, world!",
            null, AlertType.INFO
        );

        helloAlert.setTimeout(Alert.FOREVER);
        helloAlert.addCommand(exitCommand);
        helloAlert.setCommandListener(this);
    }

    public void startApp() {
        if (display == null){
            display = Display.getDisplay(this);
        }

        display.setCurrent(helloAlert);
    }

    public void pauseApp() {
    }

    public void destroyApp(boolean unconditional) {
    }

    public void commandAction(Command c, Displayable d){
        if (c == exitCommand){
            destroyApp(true);
            notifyDestroyed(); // Exit
        }
    }
}
```

```
    }  
}
```

Selanjutnya kita akan mempelajari MIDlet pertama kita, difokuskan pada line yang penting dari code diatas:

```
public class HelloMidlet extends MIDlet implements CommandListener {
```

Seperti yang sudah kita katakan sebelumnya, kita harus membuat subclass dari MIDlet untuk membuat MIDP program. Pada line ini, kita sudah membuat subclass dari MIDlet dengan memberikan turunan kelas induk dan menamakannya HelloMIDlet.

```
    Display display;  
    Command exitCommand = new Command("Exit", Command.EXIT, 1);  
    Alert helloAlert;
```

Line diatas ini adalah variabel properties dari MIDlet. Kita membutuhkan object Display (hanya ada satu display per MIDlet) untuk melakukan fungsi menggambar pada layar. exitCommand adalah perintah yang akan kita taruh pada layar agar kita dapat keluar dari program. Jika kita tidak memiliki perintah keluar, maka kita tidak memiliki cara untuk keluar dari MIDlet dengan benar.

```
public HelloMidlet() {  
    helloAlert = new Alert(  
        "Hello MIDlet", "Hello, world!",  
        null, AlertType.INFO  
    );  
  
    helloAlert.setTimeout(Alert.FOREVER);  
    helloAlert.addCommand(exitCommand);  
    helloAlert.setCommandListener(this);  
}
```

Constructor melakukan inialisasi dari object Alert. Kita akan mempelajari lebih lanjut dari Alert class pada bab berikutnya. Method addCommand() pada object Alert memberikan perintah "Exit" pada layar. Method setCommandListener() memberikan informasi kepada sistem untuk memberikan semua command events ke MIDlet.

```
public class HelloMidlet extends MIDlet implements CommandListener {
```

Code "implements CommandListener" adalah untuk command/key presses, sehingga program kita mampu *handle* "command" events. Jika kita melakukan implement CommandListener, kita harus membuat method `commandAction()`.

```
    public void commandAction(Command c, Displayable d){
        if (c == exitCommand){
            destroyApp(true);
            notifyDestroyed(); // Exit
        }
    }
}
```

`commandAction()` diatas hanya *handle* request untuk perintah "Exit". Method diatas akan menghentikan program menggunakan `notifyDestroyed()` jika perintah "Exit" dijalankan atau ditekan.

```
    public void startApp() {
        if (display == null){
            display = Display.getDisplay(this);
        }

        display.setCurrent(helloAlert);
    }
}
```

Code diatas adalah bagian awal dari program kita ketika program kita sudah siap untuk ditampilkan oleh AMS. Perlu diingat bahwa method `startApp()` mungkin / bisa dimasukkan lebih dari sekali seperti pada daur hidup MIDlet. Jika MIDlet berhenti / dihentikan, seperti bila ada telepon masuk, program akan masuk ke state berhenti (`pausedApp`). Jika panggilan sudah selesai AMS akan kembali ke program dan memanggil method `startApp()` lagi. Method `display.setCurrent()` memberikan informasi ke sistem bahwa kita menginginkan object Alert untuk dimunculkan ke layar. Kita dapat mendapat tampilan object dengan memanggil method statis `Display.getDisplay()`.

NetBeans Mobility Pack secara otomatis membuat Java Application Descriptor (JAD) untuk program Anda. NetBeans Mobility Pack menaruh file JAD pada folder "dist" dari folder project. Berikut ini adalah contoh file JAD yang dibuat oleh NetBeans Mobility Pack:

```
MIDlet-1: HelloMidlet, , HelloMidlet
MIDlet-Jar-Size: 1415
MIDlet-Jar-URL: ProjectHello.jar
```

```
MIDlet-Name: ProjectHello
MIDlet-Vendor: Vendor
MIDlet-Version: 1.0
MicroEdition-Configuration: CLDC-1.1
MicroEdition-Profile: MIDP-2.0
```

Sekarang kita siap untuk meng*compile*, melakukan pemaketan (*package*) pada aplikasi MIDlet pertama kita.

## 2.4 Compilation dan Packaging MIDlets

Sebelum kita menggunakan integrated tools untuk meng*compile* dan melakukan *packaging* aplikasi MIDlet (MIDlet *suite*), kita akan mencoba menggunakan command line.

Aplikasi MIDlet biasanya dipaketkan ke dalam sebuah file yaitu file JAR. File ini adalah file terkompres, seperti file ZIP. Pada implementasinya, Anda dapat membuka file JAR menggunakan program dekompresor file ZIP.

Aplikasi MIDlet terdiri dari:

- File JAR
- File Java Application Descriptor (JAD)

File JAR memiliki:

- File class
- Manifest file describing the contents of the archive
- File manifest yang menjelaskan isi dari arsip
- Sumber: image/icon, video, data, dll. Digunakan oleh aplikasi

File manifest, manifest.mf adalah seperti file JAD. File ini digunakan oleh application manager dari device. Beberapa field yang diperlukan oleh file manifest adalah:

- MIDlet-Name
- MIDlet-Version
- MIDlet-Vendor
- MIDlet-<n> (dimana n adalah angka dari 1, untuk setiap MIDlet di file JAR)

- MicroEdition-Profile
- MicroEdition-Configuration

Selanjutnya kita meng*compile* file source java:

```
javac -bootclasspath C:\WTK23\lib\cldcapi11.jar;C:\WTK23\lib\midpapi20.jar *.java
```

Program Compiler Java, "javac", harus berada pada path Anda. Jika anda melihat error seperti "cannot find file" atau "not an executable", Anda bisa mengkonsultasikan dengan panduan instalasi untuk distribusi Java development kit Anda tentang bagaimana memasukkan executable PATH dari lokasi tools yang ada di Java.

Selanjutnya kita melakukan *pre-verify* dari file class:

```
preverify  
-classpath C:\WTK23\lib\cldcapi11.jar;C:\WTK23\lib\midpapi20.jar;.  
-d . HelloMidlet
```

Preverify sudah berada di wireless toolkit dari java.sun.com. Masukkan perintah ini pada sebuah baris.

Langkah terakhir adalah membuat file JAR tersebut:

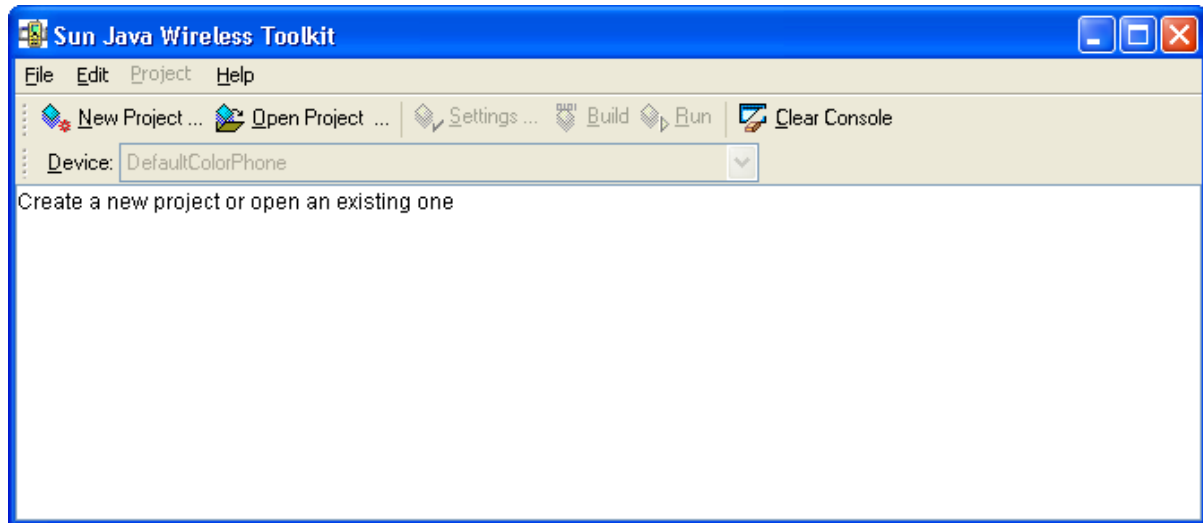
```
jar cvfm HelloMidlet.jar manifest.txt HelloMidlet.class
```

Program jar sudah berada di Java Development Kit, dan lokasinya harus dimasukkan pada executable path Anda. Perintah ini akan membuat file JAR dengan nama file HelloMidlet.jar. File manifest.txt namanya diganti dengan manifest.mf pada file JAR.

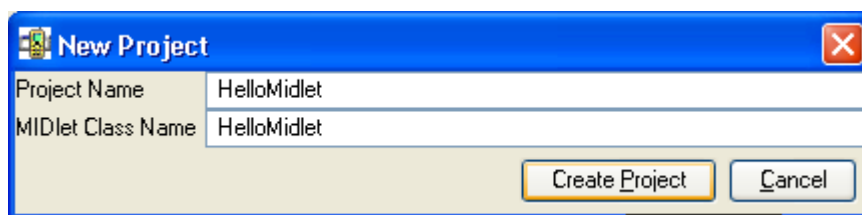
## 2.5 Menggunakan Sun Wireless Toolkit

Sekarang kita menggunakan Sun Wireless Toolkit untuk meng*compile* dan memaketkan aplikasi MIDlet / MIDlet suite (mengandung satu MIDlet)

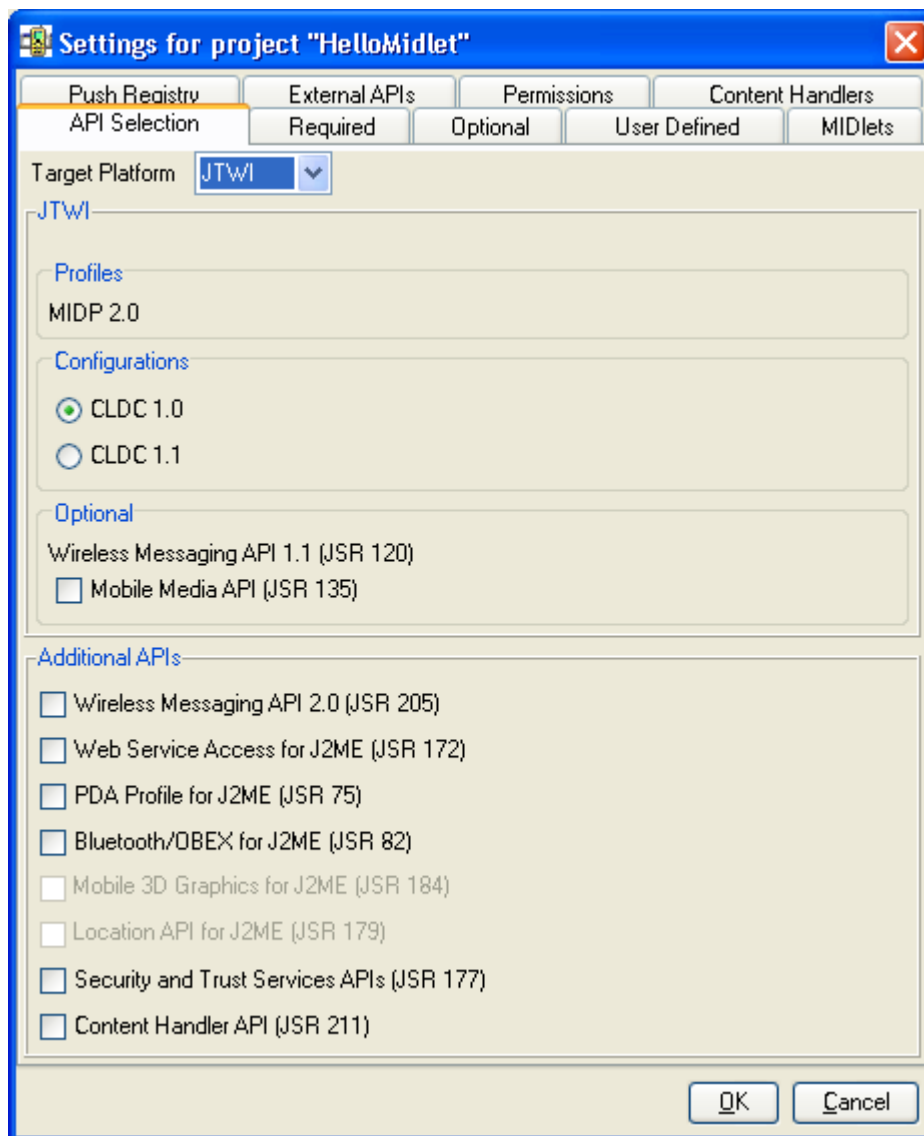
Buka ktoolbar (dari Wireless Toolkit distribution):



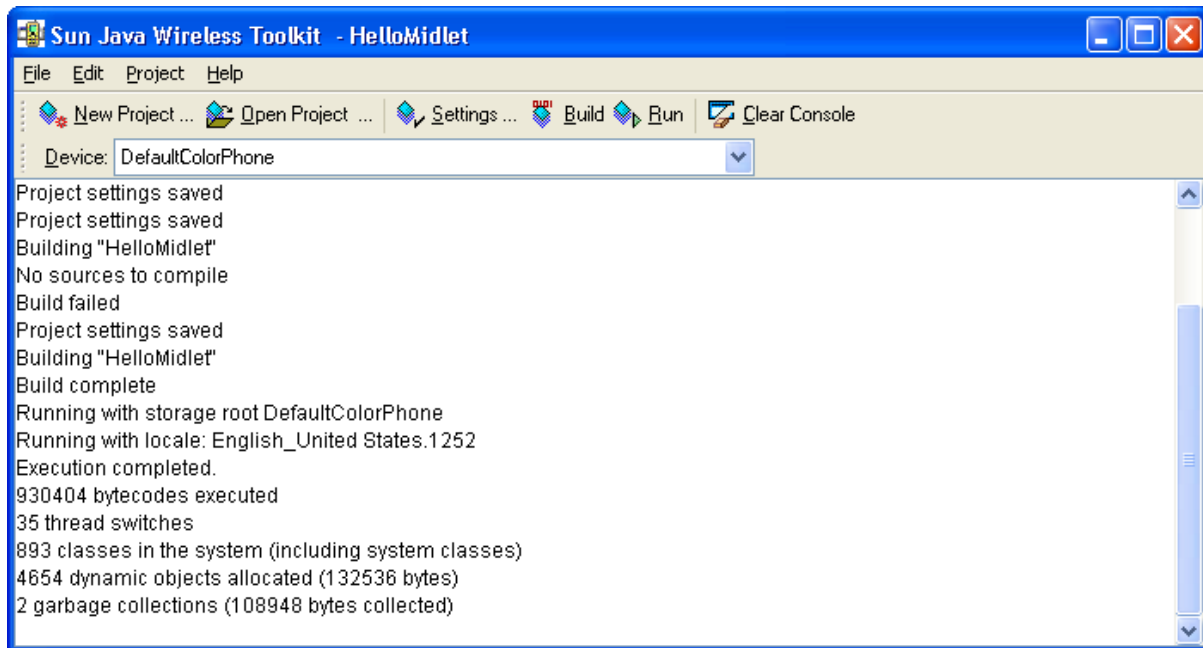
Buat sebuah project:



Pada Setting window, anda dapat merubah banyak pilihan-pilihan dari beberapa opsi konfigurasi untuk project Anda. Anda dapat memilih konfigurasi yang akan bekerja, package/API yang diperlukan, konfigurasi Push Registry dan yang lain. Untuk tujuan kita kali ini, kita akan menggunakan konfigurasi default project. Click "OK" untuk selesai membuat project.



Copy HelloMidlet.java kedalam direktori "src": Pada jendela ini berada di direktori: C:\WTK23\apps\HelloMidlet\src (dimana [C:\WTK23](#) adalah lokasi Anda menginstall wireless toolkit). Click "Build" dan "Run":

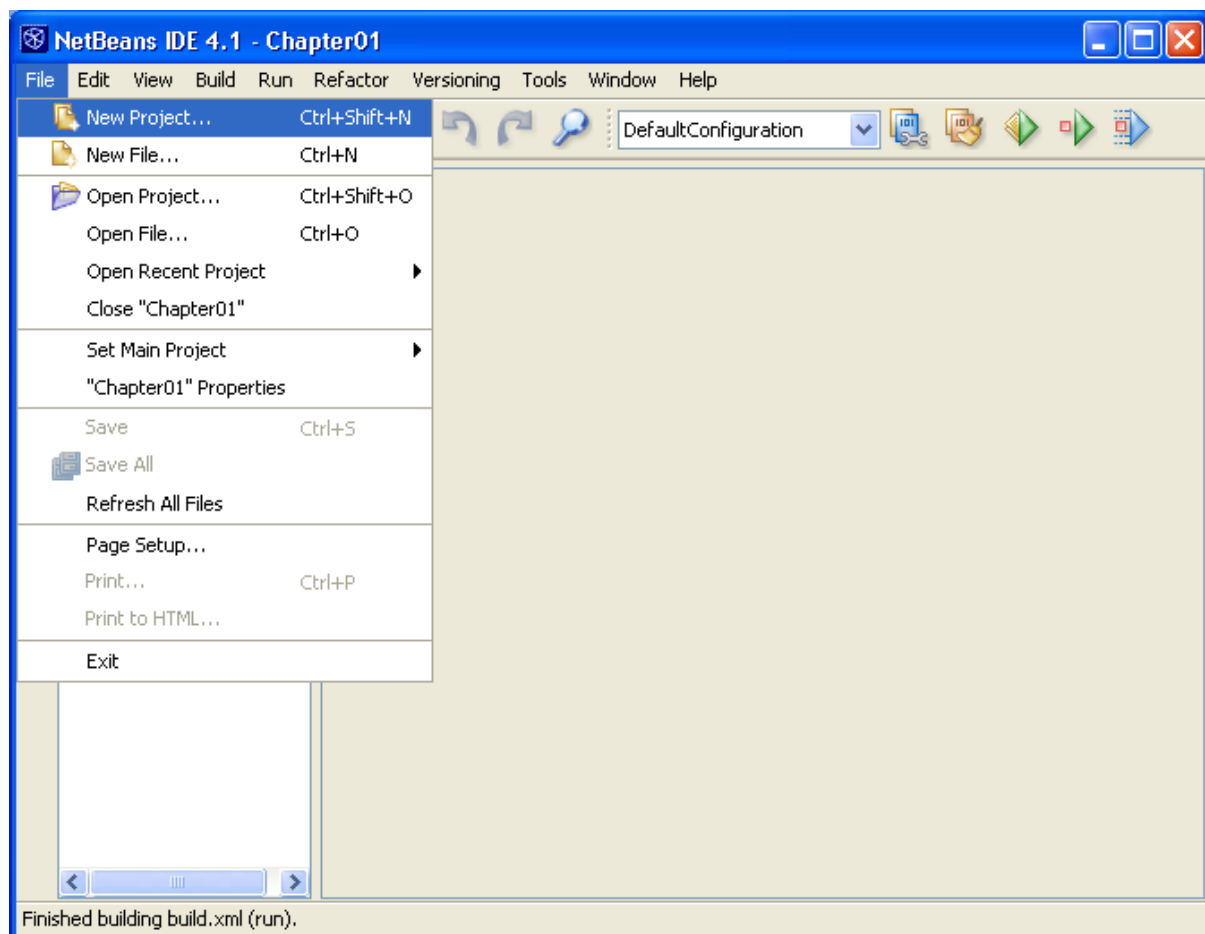




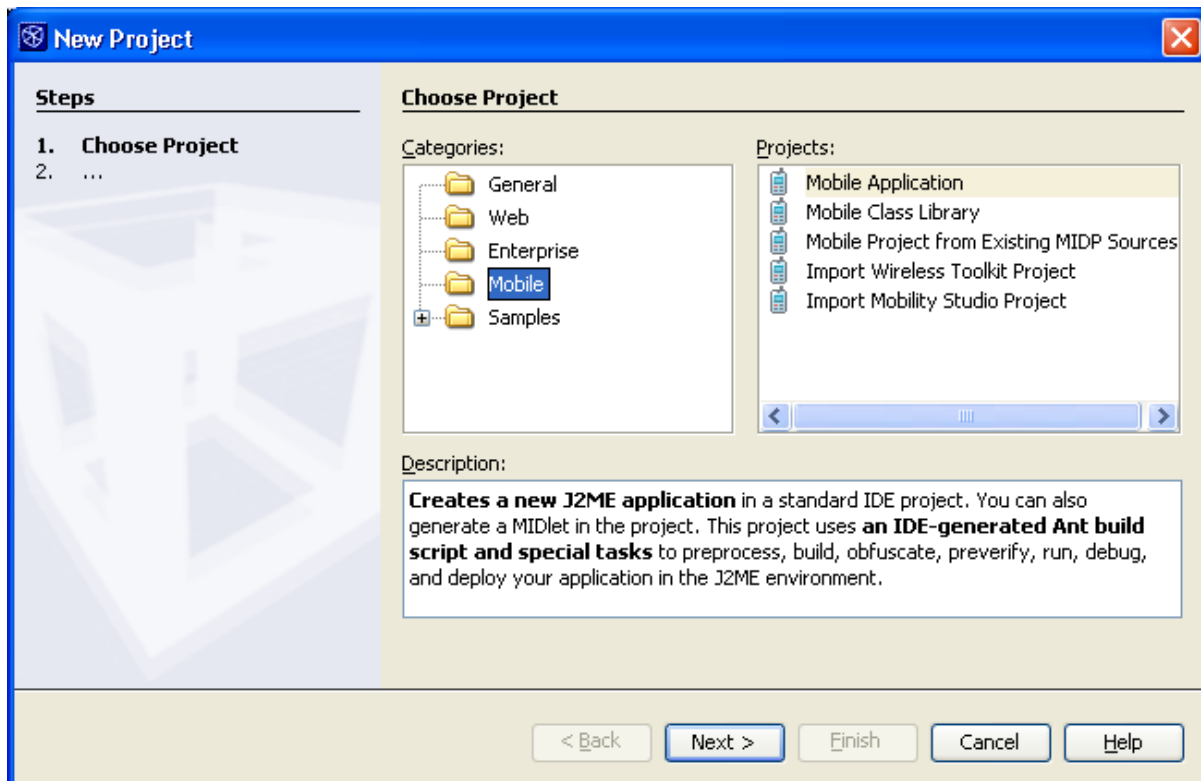
## 2.6 Menggunakan NetBeans Mobility Pack

Seperti yang telah dijelaskan pada awal bab ini tentang hal yang diperlukan, NetBeans 4.1 dan NetBeans Mobility Pack harus sudah *terinstall* di komputer Anda.

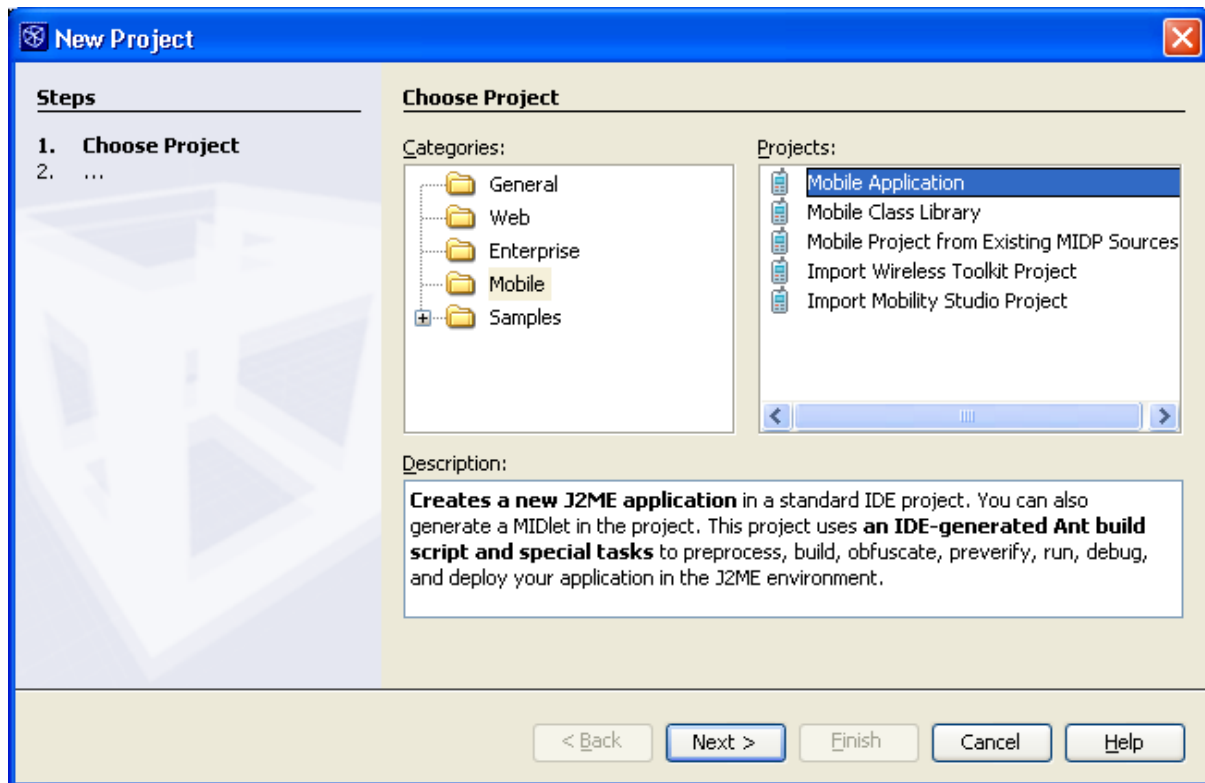
### Langkah 1: Membuat project baru



## Langkah 2: Memilih kategori "Mobile"

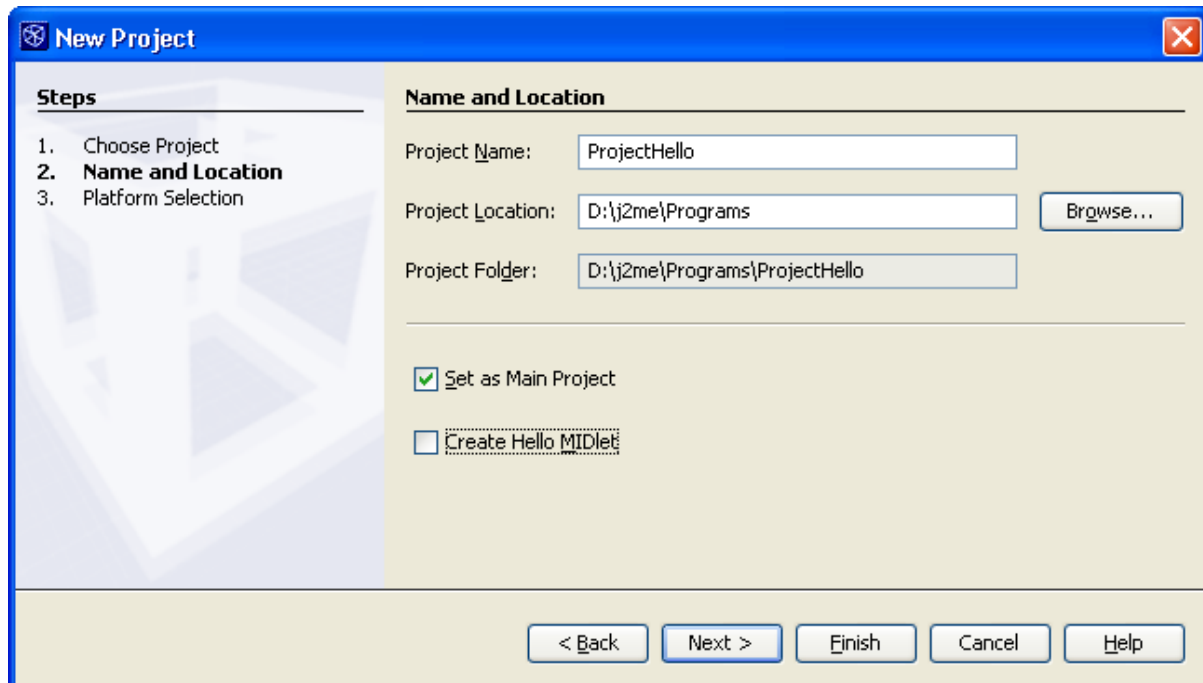


### Langkah 3: Memilih "Mobile Application"

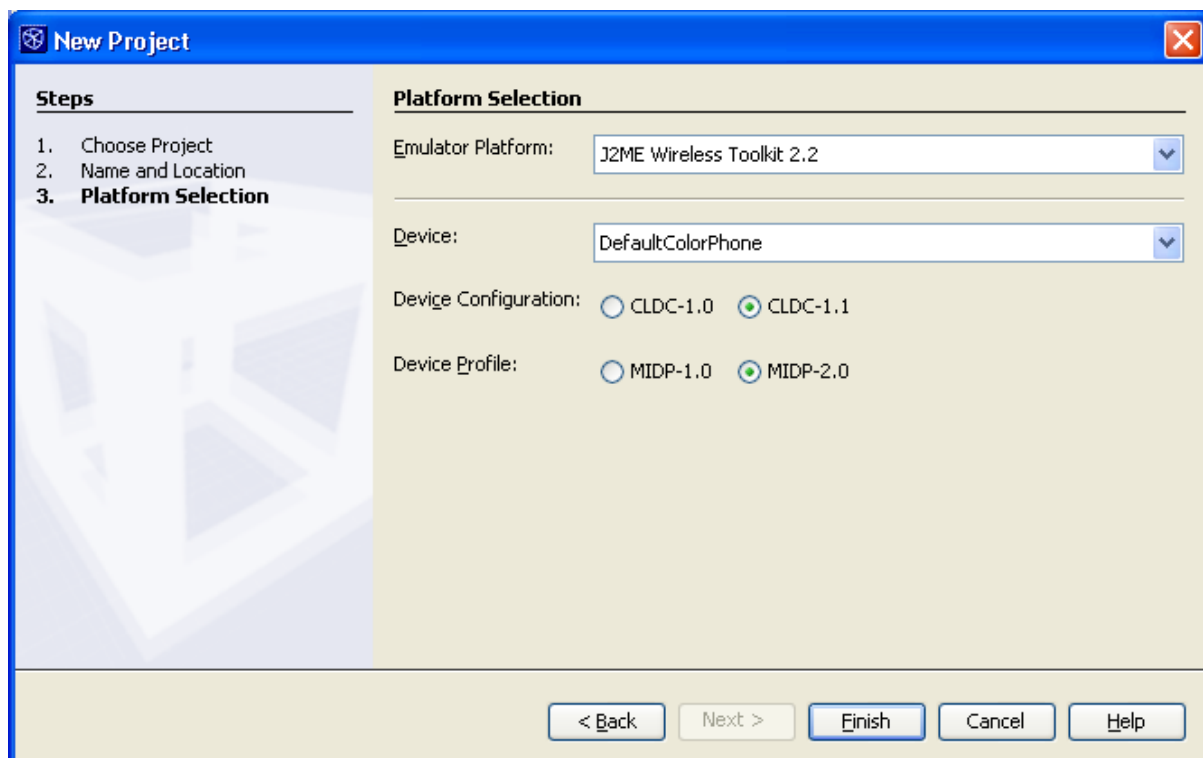


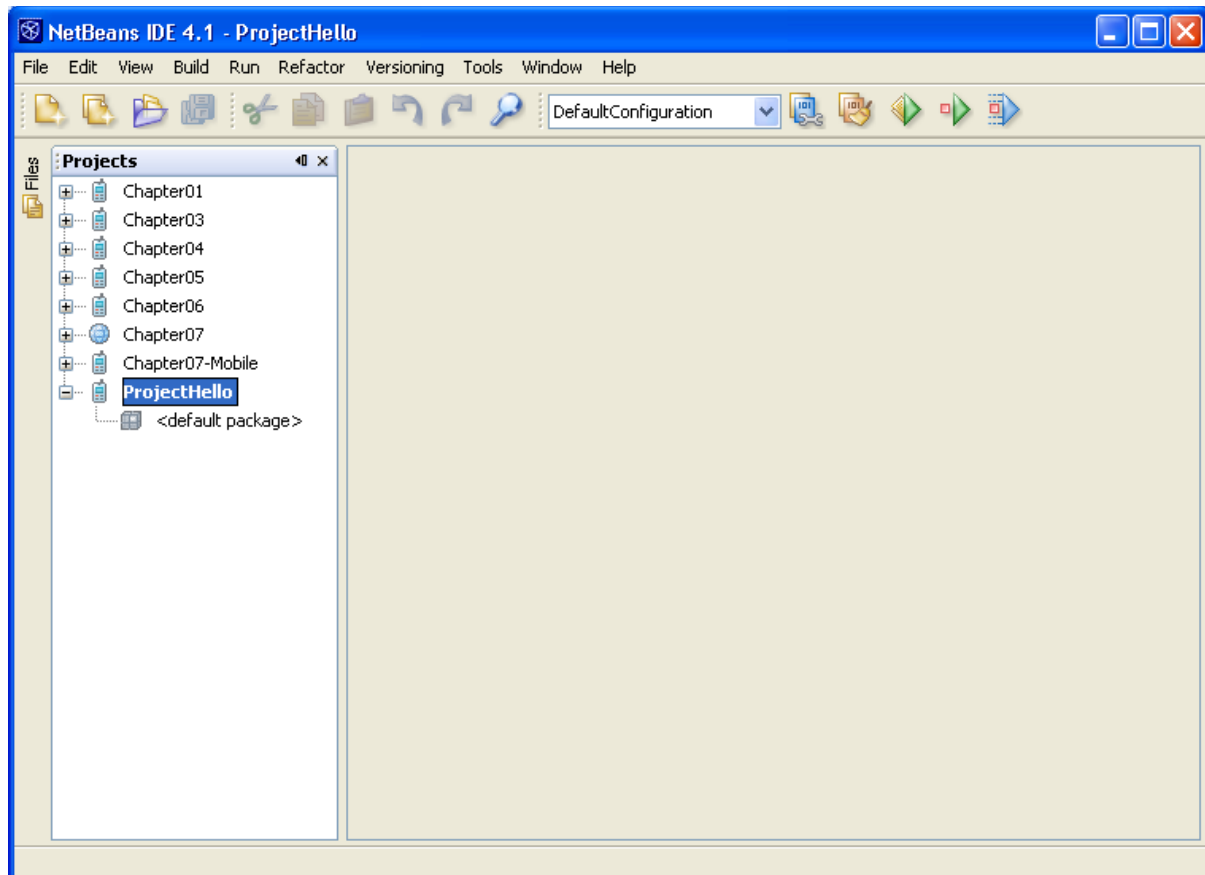
#### Langkah 4: Beri nama project dan tentukan lokasinya

(Hilangkan tanda pada "Create Hello MIDlet", kita akan membuat MIDlet kita sendiri nantinya)



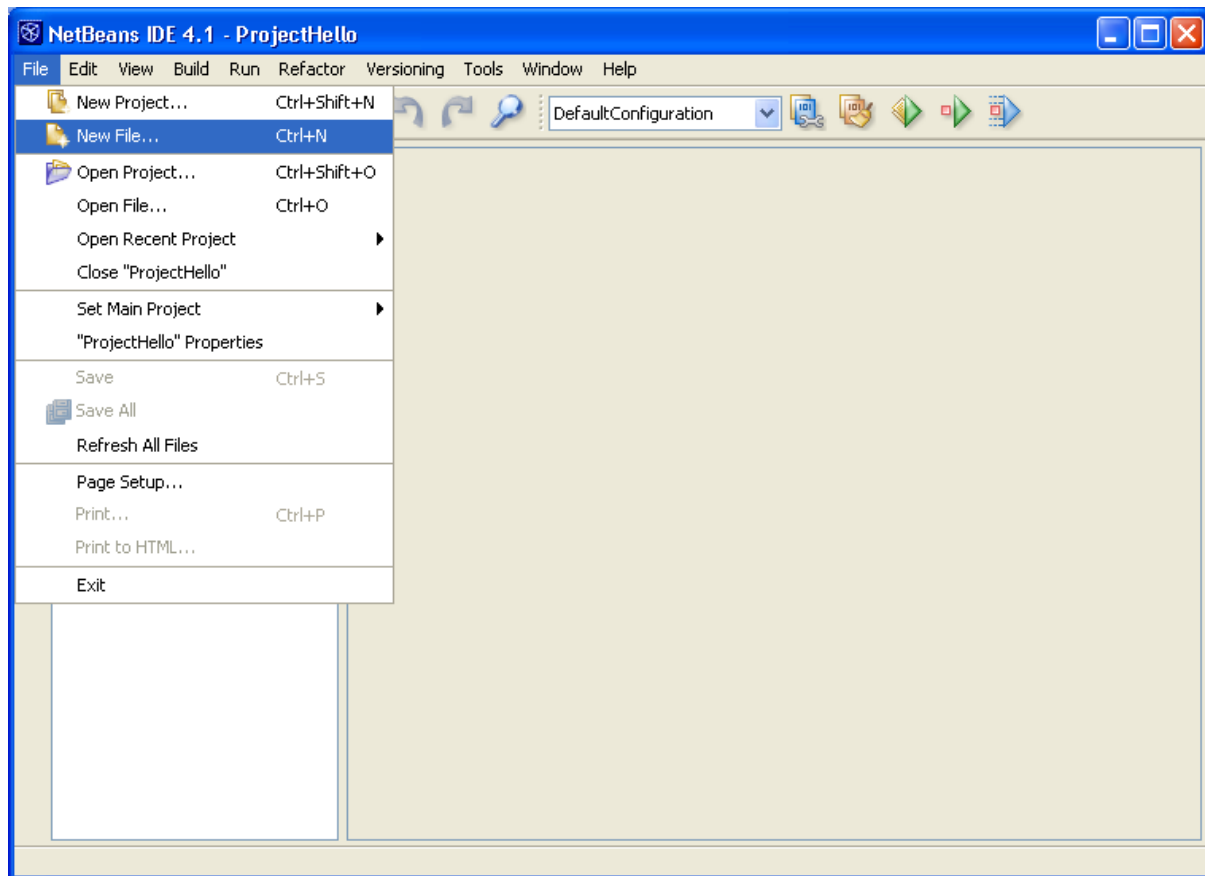
#### Step 5: Memilih Platform (optional)



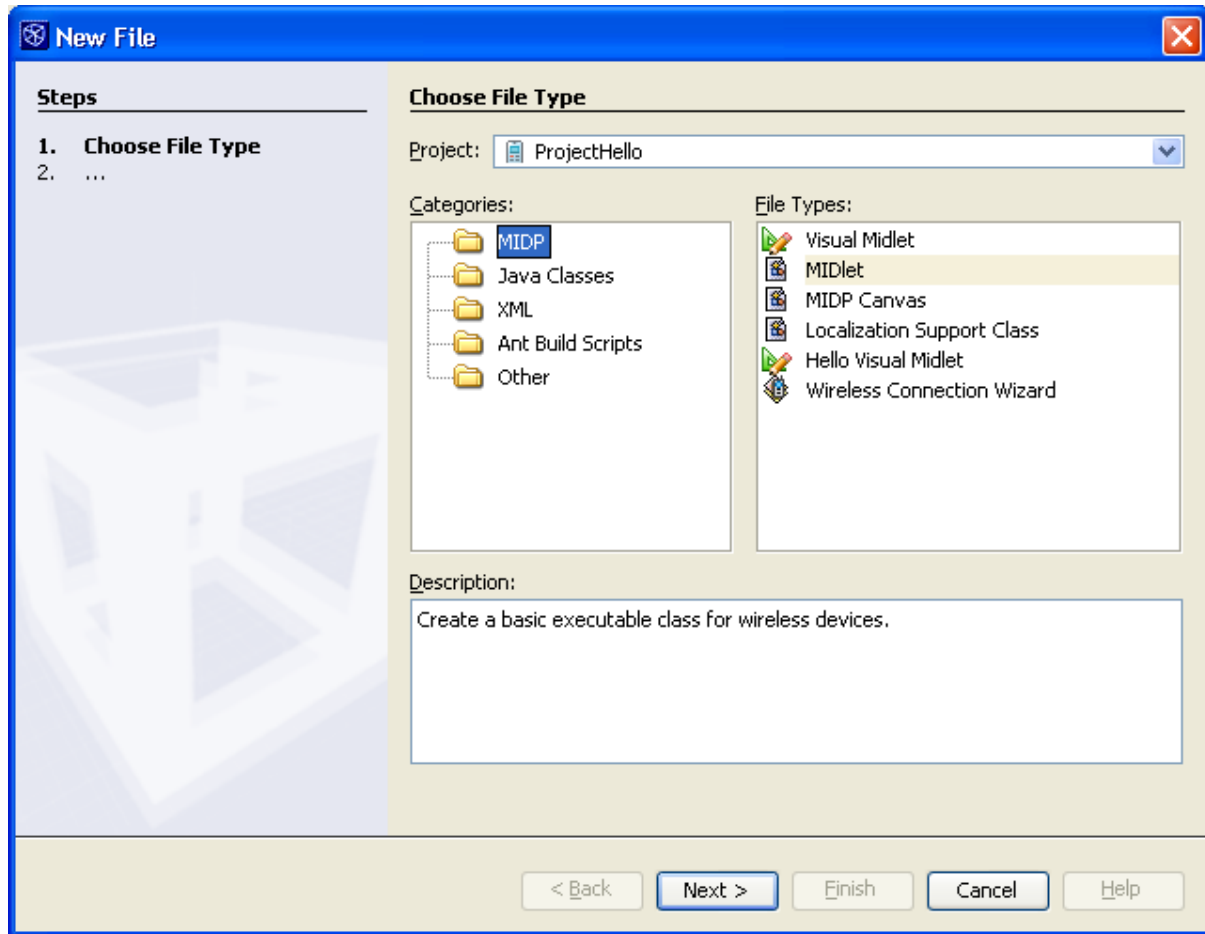


*Gambar 1: Mobile Project yang baru dibuat (NetBeans Mobility Pack)*

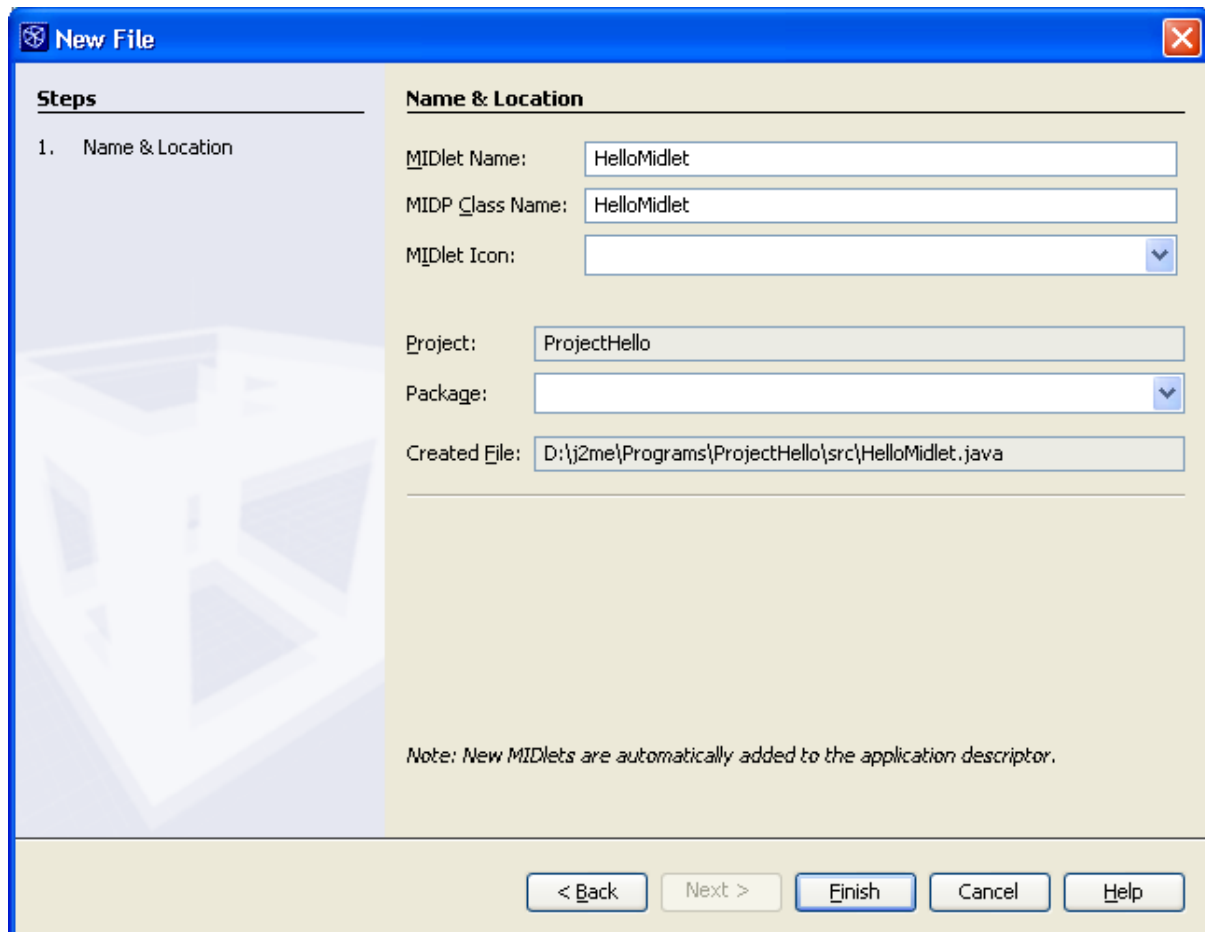
## Langkah 6: Membuat sebuah MIDlet baru



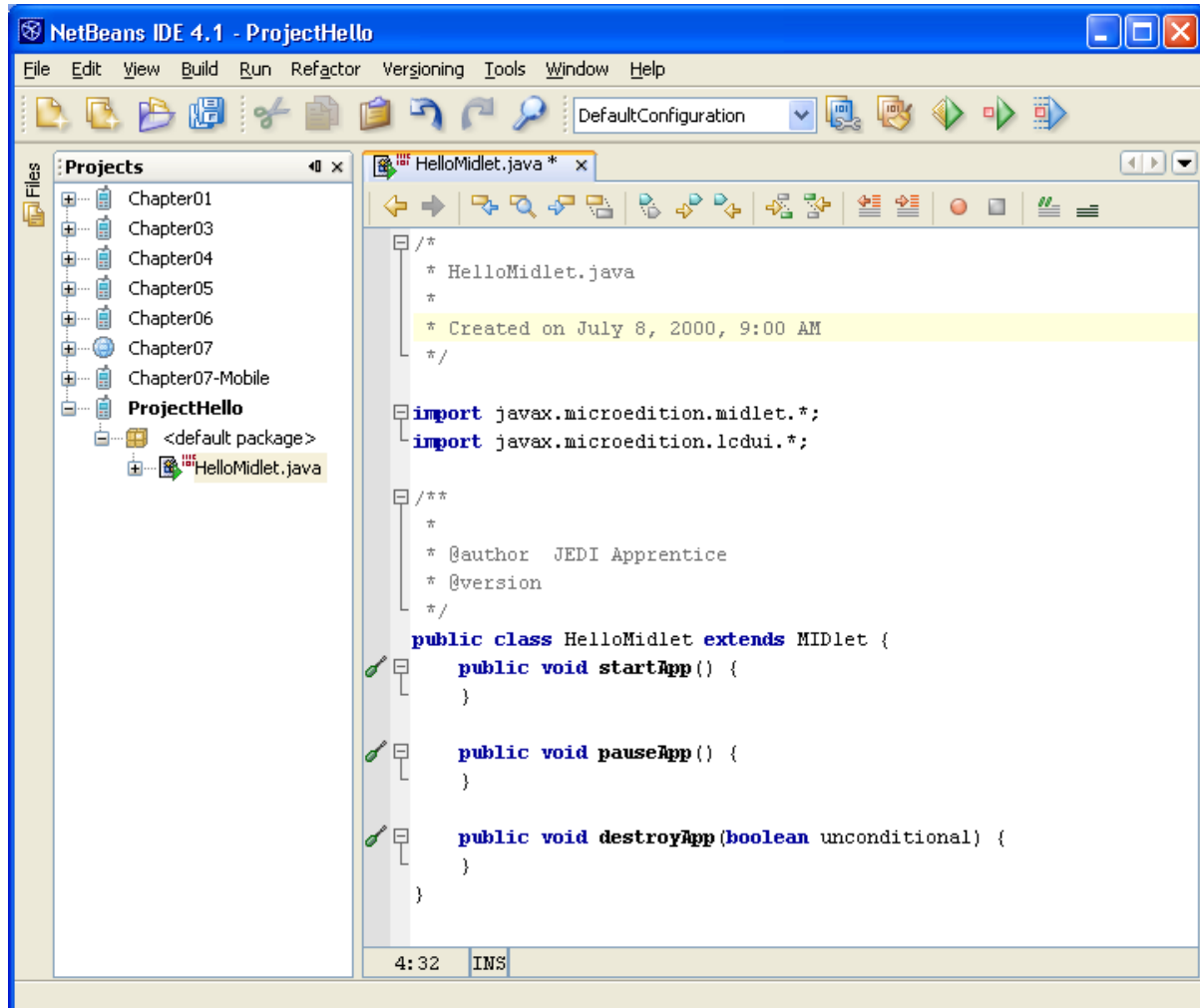
### Langkah 7: Memilih MIDP "Category" dan MIDlet "File Type"



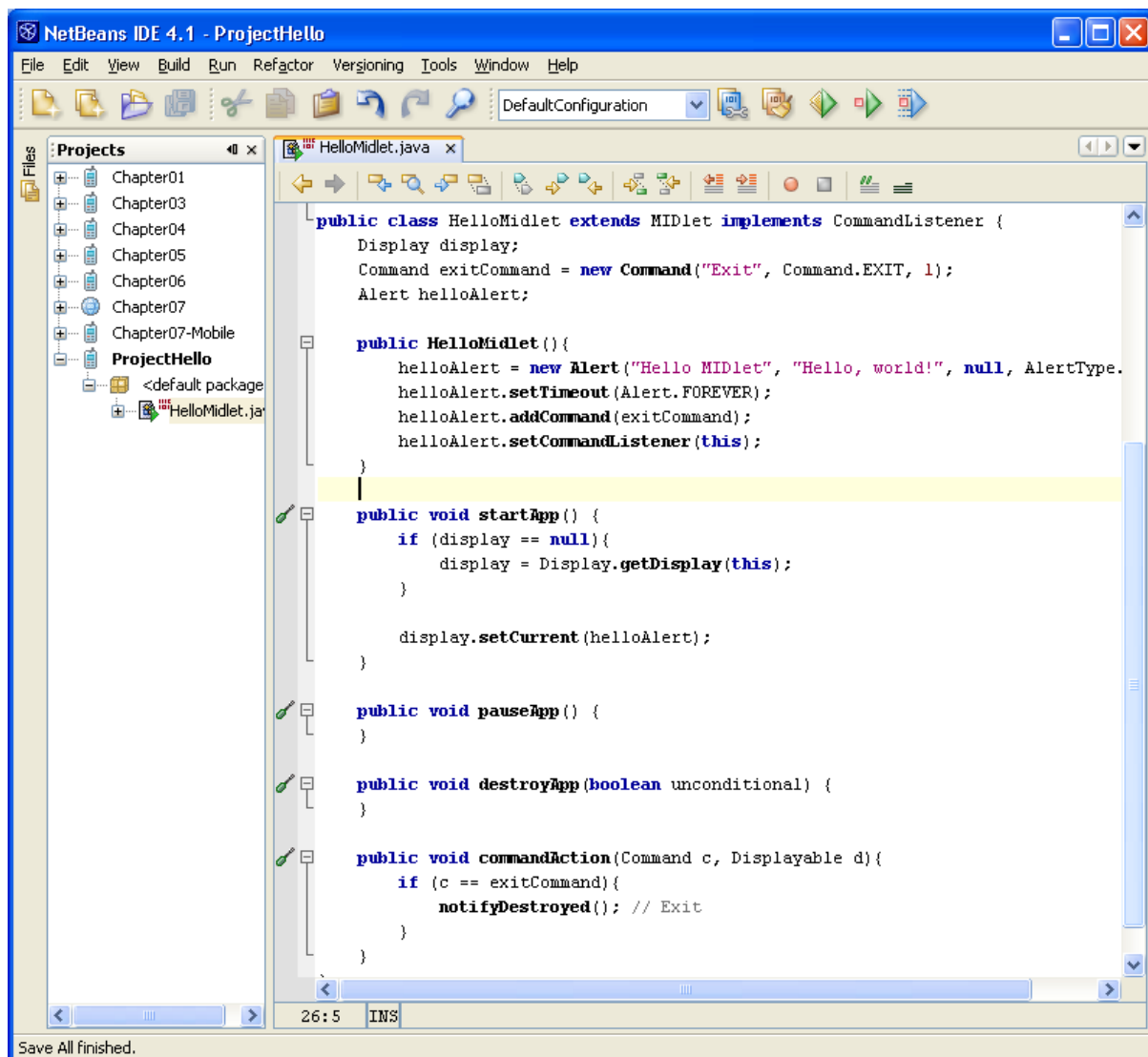
### Langkah 8: Memberi nama MIDlet



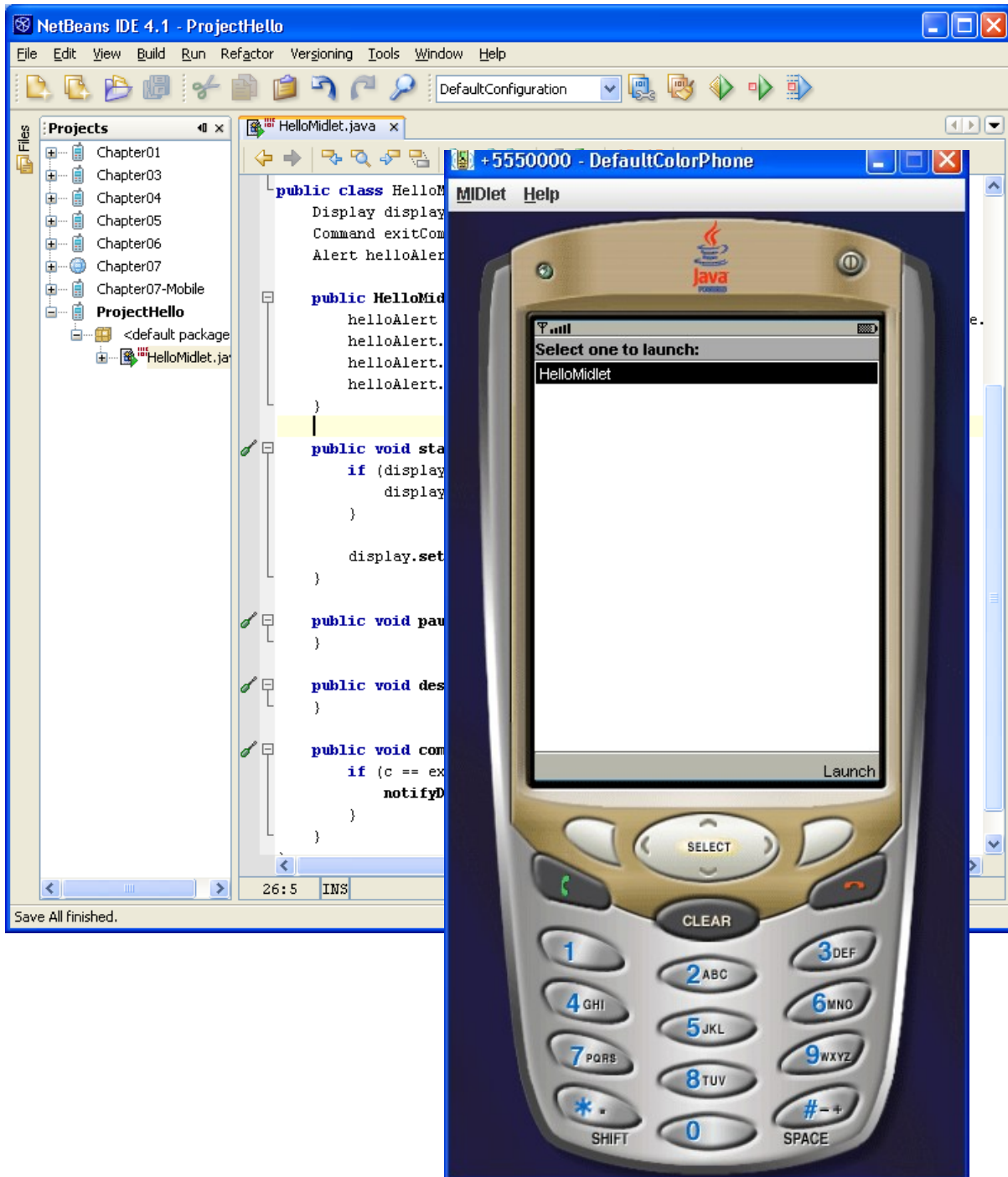
## Langkah 9



Gambar 2: Membuat MIDlet secara otomatis membuat method MIDlet yang diperlukan

**Langkah 10: Mengganti code yang dibuat secara otomatis dengan code program kita.**

### Langkah 11: Mengcompile dan Menjalankan (Run) MIDlet di Emulator



## Langkah 12: Menjalankan MIDlet kita di Emulator





Gambar 3: Hello World MIDlet

## **2.7 Latihan**

### ***2.7.1 Multiple MIDlets dalam satu MIDlet suite***

Tambahkan satu MIDlet baru pada project "ProjectHello". Perlu anda catat bahwa NetBeans Mobility Pack secara otomatis menambahkan MIDlet baru pada aplikasi file JAD ketika anda menggunakan "New File..." Wizard.

### ***2.7.2 Multiple MIDlets dalam satu MIDlet suite menggunakan Wireless Toolkit***

Gunakan Sun Wireless Toolkit untuk menambahkan MIDlet baru pada aplikasi MIDlet anda.