Bab 9

Optimisasi

9.1 Tujuan

Setelah menyelesaikan bab ini, pelajar diharapkan menguasai :

- Mengetahui teknik yang berbeda dalam optimisasi aplikasi mobile

9.2 Optimisasi

Sebelum benar-benar melakukan setiap optimisasi pada program Anda, Anda seharusnya perlu memastikan bahwa package software anda memiliki kualitas yang baik. Anda perlu meletakkan optimisasi dalam agenda Anda. Beberapa teknik yang dibahas pada bab ini seharusnya dapat membantu dalam menghindari beberapa kesalahan pemrograman.

9.3 Eksekusi program

9.3.1 Gunakan StringBuffer sebagai pengganti String.

Anda perlu ingat bahwa pada Java, object String bersifat absolut atau abadi. Menggunakan method String menciptakan suatu object String terpisah. Perangkaian String yang sederhana menciptakan suatu object String ganda (kecuali jika String itu bersifat konstan dan kompiler cukup pandai untuk menggabungkan mereka pada proses compile berlangsung). Menggunakan StringBuffer tidak hanya mengoptimalkan runtime program Anda (lebih sedikit menimbulkan object runtime), itu juga mengoptimalkan pemakaian memori (lebih sedikit object String dibuat).

String	***StringBuffer***
String a, b, c; ... String message = "a=" + a + "\n" + "b=" + b + "\n" + "c=" + c + "\n";	**String a, b, c;** **...** **StringBuffer message = new StringBuffer(255);** **message.append("a=");** **message.append(a);** **message.append("\n");** **message.append("b=");**

String	*StringBuffer*
	message.append(b); **message.append("\n");** **message.append("c=");** **message.append(c);** **message.append("\n");**

9.3.2 Gunakan clipping area dalam menggambar

Menggunakan Graphics.setClip() akan mengurangi waktu eksekusi karena Anda hanya akan menggambar nomor-nomor yang optimal dari pixel-pixel di layar. Ingat, bahwa menggambar grafik pada layar meminta banyak terminologi pada waktu eksekusi. Mengurangi banyaknya pixel-pixel untuk digambar akan sangat mempengaruhi kinerja runtime program Anda.

```
Graphics g;
int x1, y1, x2, y2;
...

g.setClip(x1, y1, x2, y2);
g.drawString("JEDI", x, y, Graphics.TOP | Graphics.HCENTER);
// Operasi menggambar yang lainnya...
```

9.3.3 Hindari modifier yang sama

Menggunakan modifier yang sama mengambil sesuatu tanpa diduga pada kecepatan eksekusi program Anda karena hal tersebut menimbulkan beberapa ukuran tambahan sehingga itu tidak akan diakses secara bersamaan.

9.3.4 Lewatkan parameter sesedikit mungkin

Ketika memanggil suatu method, penerjemah akan mendorong semua parameter ke atas tumpukan eksekusi. Melewatkan banyak parameter akan mempengaruhi kecepatan eksekusi dan pemakaian Heap Memory.

9.3.5 Mengurangi pemanggilan method

Memanggil method menghabiskan Heap Memory dan waktu eksekusi. Lihat subbab sebelumnya.

9.3.6 Menunda semua inisialisasi

Untuk mempercepat awal permulaan aplikasi, tunda semua inisialisasi yang sangat besar sampai mereka dibutuhkan. Jangan meletakkan inisialisasi dalam konstruktor MIDlet atau

method startApp. Mempercepat waktu load sebuah aplikasi akan menambah penggunaan aplikasi Anda. Kebanyakan user akan meninggalkan aplikasi ketika aplikasi tersebut membutuhkan waktu yang lama untuk start up. Ingat bahwa waktu load aplikasi anda secara langsung mempengaruhi kesan pertama pengguna aplikasi Anda.

9.3.7 Gunakan array sebagai pengganti collection

Mengakses Array lebih cepat daripada menggunakan vektor

9.3.8 Menggunakan variabel lokal

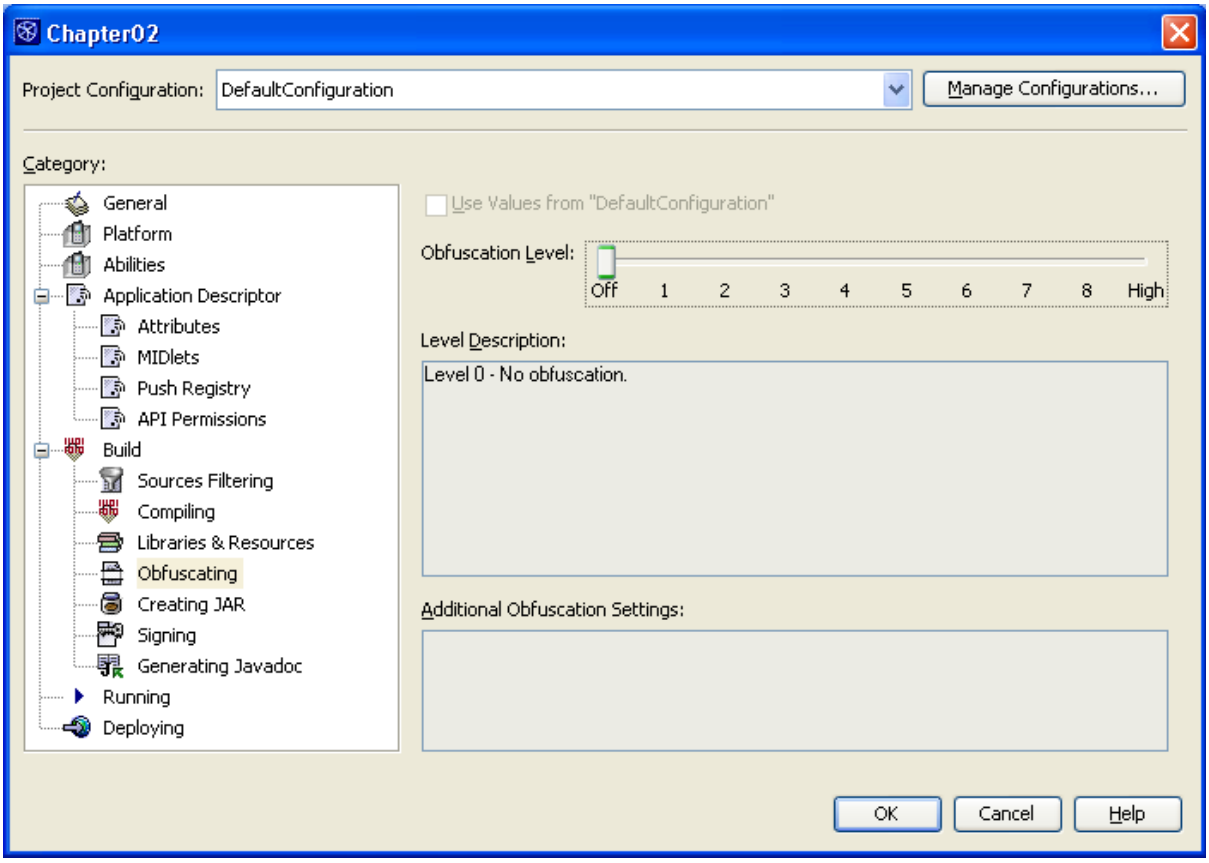
Hal tersebut lebih cepat mengakses variabel lokal daripada mengakses variabel instance.

9.4 Ukuran JAR

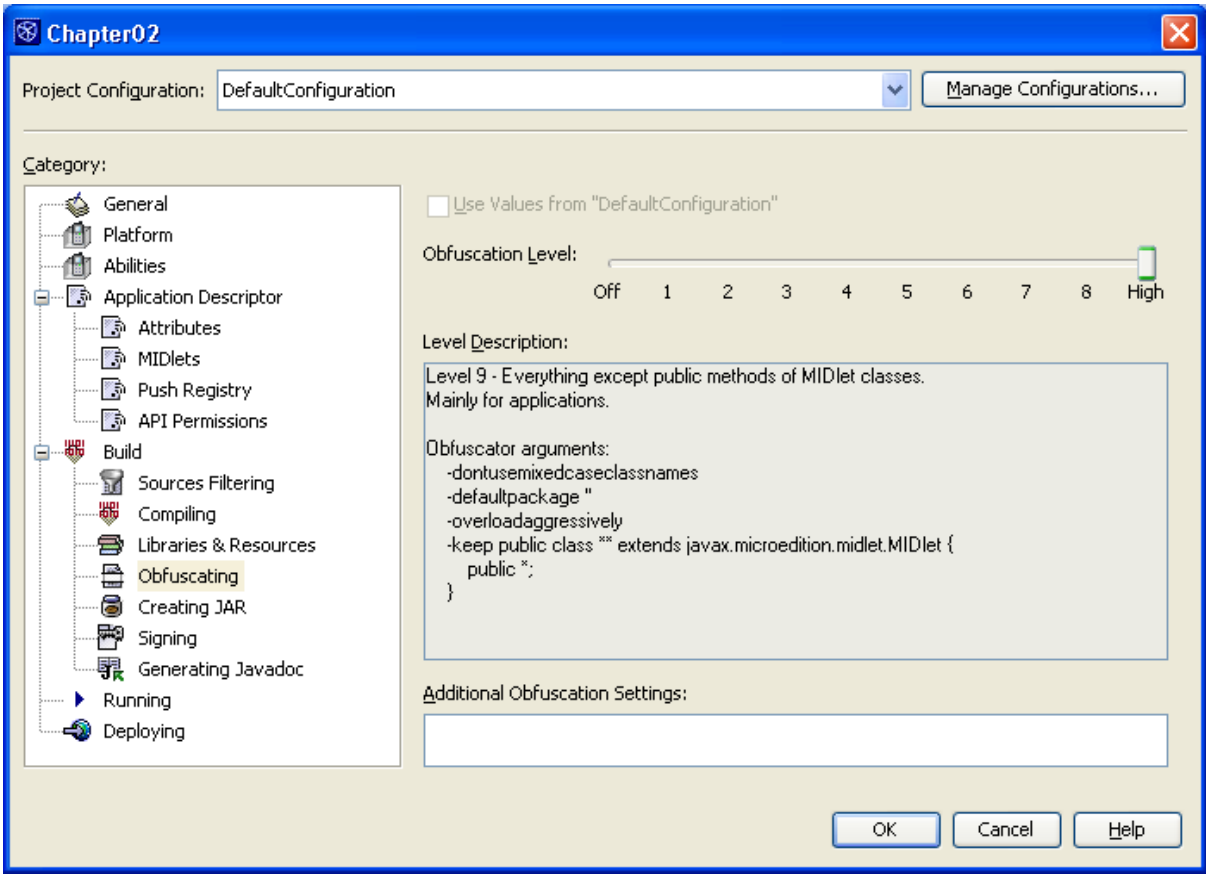
9.4.1 Gunakan Obfuscator

Tujuan utama obfuscator adalah untuk mengacak file class yang dikompile sehingga sulit untuk di decompile. Tetapi proses obfuscator juga mengurangi ukuran sebuah aplikasi. Salah satu method yang digunakan oleh obfuscator adalah memberi nama baru pada class menjadi sebuah nama. Karena obfuscator melakukan hal ini berdasar kepada modifier dari method-method. Jika method memiliki private atau protected modifier, lalu itu dapat diasumsikan aman ketika method ini tidak akan digunakan oleh package lainnya dan oleh karena itu dapat diberi nama baru kembali.

Netbeans Mobility Pack datang dengan satu obfuscator. Dia tidak diaktifkan sebagai default. Buka tab property dan klik pada cabang "Obfuscating" :



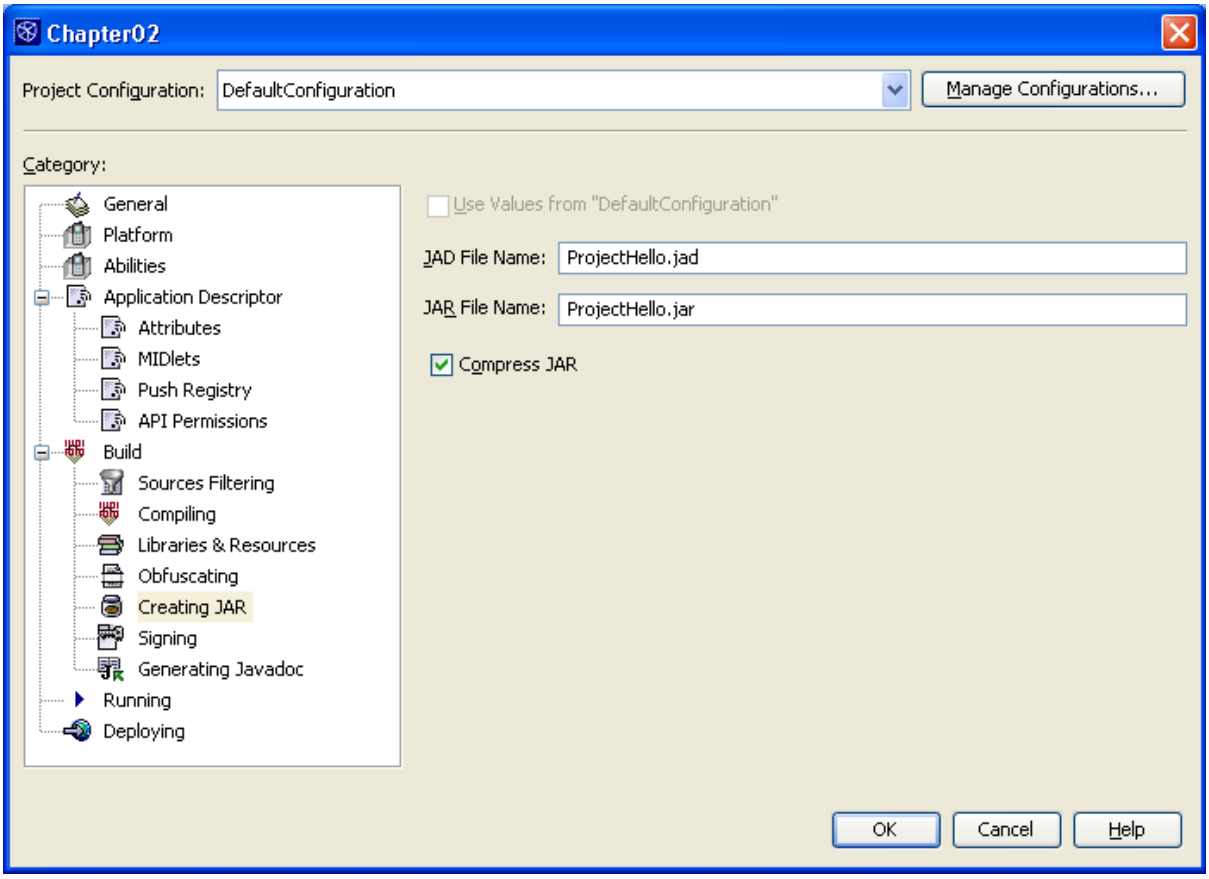
Ada sepuluh tingkat obfuscation, dari tanpa obfuscation sampai ke obfuscation yang paling agresif.



9.4.2 Memadatkan file JAR Anda

Pastikan bahwa sebelum mendistribusikan aplikasi Anda, Anda memampatkan file akhir JAR untuk distribusi. Sebuah file JAR adalah sebuah arsip ZIP, dan suatu arsip ZIP mempunyai beberapa tingkat tekanan (termasuk tanpa tekanan). NetBeans Mobility Pack tidak mendukung tingkat tekanan.

Untuk mengatur pilihan tekanan JAR, buka halaman properti dari aplikasi dan pilih cabang "Creating JAR". Centang radio box "Compress JAR" untuk memampatkan file JAR proyek Anda. Jangan lupa untuk membangun kembali proyek Anda.



9.4.3 Hindari membuat class yang tidak perlu

Ini akan tampak berlawanan untuk prinsip berorientasi object, tapi apakah Anda mengetahui bahwa suatu class kosong yang sederhana seperti ini :

```
public class EmptyClass {
    public EmptyClass(){}
}
```

akan dikompile menjadi file class dengan ukuran file sebesar 250kb (tanpa dimampatkan)?

Anda dapat mencoba mengkompile class kosong ini dan buktikan sendiri. Netbeans Mobility Pack menyimpan package file JAR di dalam folder distribusi dibawah folder proyek. Anda dapat merubah nama file .jar menjadi file .zip dan buka dengan program ZIP favorit Anda untuk melihat ukuran dari file class yang Anda kompile.

9.4.4 Hindari membuat interfaces

Teknik ini berkaitan dengan teknik sebelumnya. Memiliki banyak class dan interfaces akan menambahkan lebih ukuran file (kilobytes) dalam aplikasi Anda.

9.4.5 Hindari inner dan anonymous class

Sama seperti diatas. Inner class adalah semua class yang sama. Anonymous class mungkin tidak memiliki nama, tetapi mereka mengambil ruang yang sama untuk definisi class.

9.4.6 Gunakan satu Listener untuk object yang ganda

Ini akan mengurangi banyaknya class dalam aplikasi Anda. Buatlah MIDlet Anda mengimplementasikan CommandListener interface sehingga membantu anda memangkas package Anda oleh satu class (Dimana mengurangi 250 + byte).

9.4.7 Gunakan package default (package tanpa nama)

Didalam permintaan kita untuk package berukuran kecil, memendekkan (atau tidak menggunakan) nama package tersebut mendukung pengurangan byte.

9.4.8 Batasi penggunaan dari initializer static

Menggunakan inisialisasi static seperti ini :

```
int[] tones = { 64, 63, 65, 76, 45, 56, 44, 88 };
```

Akan dikompile oleh kompiler Java menjadi pernyataan berikut :

```
tones[0] = 64;
tones[1] = 63;
tones[2] = 65;
tones[3] = 76;
tones[4] = 45;
tones[5] = 56;
tones[6] = 44;
tones[7] = 88;
```

Contoh ini menggambarkan hanya delapan anggota array. Bayangkan jika inisialisasi ratusan nilai menggunakan statemen terpisah. Hal tersebut akan menjadikan overhead pada ukuran aplikasi Anda.

Sebagai salah satu alternatif, Anda dapat menggunakan method `getResourceAsStream()` untuk mendapatkan nilai dari sebuah file atau menggunakan single string untuk menyimpan nilai array Anda.

9.4.9 Menggabungkan gambar ke dalam satu file

Memampatkan gambar lebih baik ketika di-kelompokkan menjadi satu file gambar. Karena memampatkan format gambar (contohnya PNG) adalah lebih spesifik untuk gambar daripada memampatkan method pengarsipan JAR. Ada teknik-teknik untuk mendapatkan gambar yang

spesifik dari sebuah gambar yang besar yaitu dengan memotongnya.

9.4.10 Bereksperimen dengan memampatkan gambar

method tekanan(compressing) tidak diciptakan sama. Beberapa mungkin memampatkan lebih baik pada beberapa jenis gambar tetapi kadang memiliki rasio yang rendah dalam memampatkan jenis gambar yang lain. Pilih sebuah format gambar yang dapat meningkatkan rasio pemampatan gambar Anda. Terkadang, rasio pemampatan juga dipengaruhi oleh software pengolah gambar yang anda gunakan. Cobalah bereksperimen dengan berbagai macam jenis software pengolah gambar untuk mendapatkan ukuran gambar yang lebih baik.

9.4.11 Gunakan class yang belum diinstal

Gunakan semua class yang bisa diterapkan yang tersedia pada platform yang anda gunakan. Buatlah class Anda sendiri yang tidak akan menambah ukuran aplikasi Anda, tetapi juga mengurangi stabilitas aplikasi Anda.

9.5 Jaringan

9.5.1 Gunakan thread yang terpisah

Gunakan thread yang terpisah untuk jaringan Anda yang berfungsi untuk menghindari screen lockups.

9.5.2 Memampatkan data jaringan

Menggunakan data yang dimampatkan untuk mengurangi lalu lintas jaringan dari aplikasi Anda. Hal ini akan membutuhkan client dan server Anda untuk menggunakan protokol dan method pemampatan yang sama.

Memampatkan XML akan memberikan rasio yang lebih baik karena data XML terwakili dalam suatu format teks.

9.5.3 Mengurangi lalu lintas jaringan

Karena komunikasi jaringan semakin lambat dan mahal, cobalah sebisa mungkin untuk memasukkan beberapa perintah kedalam satu permintaan jaringan. Ini akan mengurangi overhead yang dikenakan oleh protokol jaringan.

9.6 Penggunaan Memori

9.6.1 Gunakan struktur data ringkas

Gunakan struktur data memory yang sering digunakan. Array jarang bisa diwakili dengan cara lain tanpa mengkonsumsi jumlah yang sama dari memory.

Ada tradeoff ketika mengoptimalkan untuk ukuran dan kecepatan. Menggunakan struktur data komplek akan mempengaruhi kecepatan eksekusi program Anda.

9.6.2 Membebaskan object yang tidak terpakai untuk garbage collection

Membebaskan object yang tak terpakai untuk garbage collection layar, koneksi jaringan, rekaman RMS. Menentukan variabel untuk menunjuk kepada object yang tak terpakai menjadi null dan akan memberi isyarat kepada garbage collector bahwa object ini aman untuk tidak di-load dari memory.

9.6.3 Jangan sering menggunakan layar on-the-fly

Tidak sering menggunakan object Screen (seperti Help dan about screen)on-the-fly akan banyak bebaskan banyak kebutuhan Anda yang menumpuk pada memory. Meski Anda harus membayar harga yaitu loading yang lambat untuk screen tertentu. Layar ini akan diduga menimbun pada heap memory sementara mereka tidak digunakan untuk membantu dalam penghematan memory

```
public void commandAction(Command c, Displayable d) {
    if (c == helpCommand) {
        display.setCurrent(new HelpForm());
    }
}
```

9.7 Latihan

9.7.1 Ide optimisasi lainnya.

Diskusikan ide-ide tentang optimisasi lainnya yang ada pada pikiran Anda atau teknik yang telah Anda kembangkan.