



JAVA untuk Pemula

Konsep dan tutorial pemrograman Java menggunakan Eclipse IDE

Instalasi, konsep OOP, syntax dasar, akses database, web programming (Vaadin), desktop programming (Swing GUI), mobile programming (Android)

ASWIAN EDITRI SUTRIANDI

Daftar Isi

1	Persiapan Pemrograman Java.....	4
1.1	Instalasi Software Pendukung.....	4
1.1.1	Instalasi JDK.....	4
1.1.2	Instalasi Eclipse.....	5
1.2	Memulai Eclipse.....	5
1.3	Membuat Project Java dalam Eclipse.....	5
1.4	Membuat Class 'Halo Java'.....	6
1.5	Menjalankan Program Java dari Eclipse.....	9
1.6	Aturan penamaan dalam Bahasa Pemrograman Java.....	10
1.7	Latihan.....	11
2	Class dan Objects.....	12
2.1	Konsep.....	12
2.2	Deklarasi Class pada Java.....	13
2.3	Deklarasi dan Instansiasi Object pada Java.....	13
2.4	Mengakses state dan behavior object.....	14
2.5	Latihan.....	14
3	Syntax Dasar.....	15
3.1	Variabel.....	15
3.1.1	Deklarasi variabel.....	15
3.1.2	Tipe Data Variabel.....	15
3.1.3	Inisialisasi Variabel.....	16
3.1.4	Variabel Instance, Class dan Lokal.....	16
3.2	Modifier.....	17
3.3	Method.....	19
3.4	Operator.....	20
3.4.1	Operator aritmatik pada Java.....	20
3.4.2	Operator relasional pada Java.....	20
3.4.3	Operator logika pada Java.....	21
3.5	Conditional (if & switch).....	21
3.5.1	Statement if.....	21
3.5.2	Statement switch.....	22
3.6	Loop (perulangan).....	22
3.6.1	Perintah while.....	22
3.6.2	Perintah for.....	23
3.7	Accessor dan Mutator (Getter & Setter).....	23
3.8	Latihan.....	24
4	Struktur Data Array dan Collection.....	26
4.1	Array.....	26
4.2	Collection.....	27
4.3	Latihan.....	28
5	Tutorial Akses Database dengan JDBC.....	30
5.1	Persiapan lingkungan pemrograman.....	30
5.1.1	Instalasi Database Server (MySQL).....	30
5.1.2	Menyiapkan JDBC driver untuk MySQL.....	30
5.1.3	Inisiasi database dan tabel.....	30
5.2	Pemrograman.....	30
5.2.1	Class Koneksi.....	30
5.2.2	Class Entitas.....	30
5.2.3	Class Data Access Object (DAO).....	30
6	Tutorial Desktop Application dengan Java GUI (Swing).....	31
7	Tutorial Web Application dengan Vaadin.....	32

8Tutorial Mobile Application (Android)	33
--	----

1 Persiapan Pemrograman Java

1.1 Instalasi Software Pendukung

Untuk dapat memulai pemrograman dengan bahasa Java, dua software yang pertama kali harus disiapkan adalah Java Development Kit (JDK) dan sebuah Integrated Development Environment (IDE) yang mendukung bahasa pemrograman Java.

JDK memungkinkan pengembangan aplikasi berbasis Java baik untuk desktop , server maupun embedded application. JDK dapat didownload gratis pada situs oracle.com sebagai pemilik lisensi bahasa pemrograman Java.



Gambar 1: Logo Java

IDE adalah software yang menyediakan fasilitas komprehensif yang memfasilitasi programmer dalam melakukan pengembangan software. Sebuah IDE dilengkapi dengan fasilitas *source code editor*, *build automation* dan *debugger*. Java memiliki banyak pilihan IDE yang tersedia secara gratis di Internet. Dua yang paling populer diantaranya adalah **Eclipse** dan **Netbeans**.

Dalam modul ini IDE yang digunakan adalah Eclipse, IDE ini tersedia secara gratis pada alamat <http://www.eclipse.org/downloads/>



Gambar 2: Logo Eclipse

1.1.1 Instalasi JDK

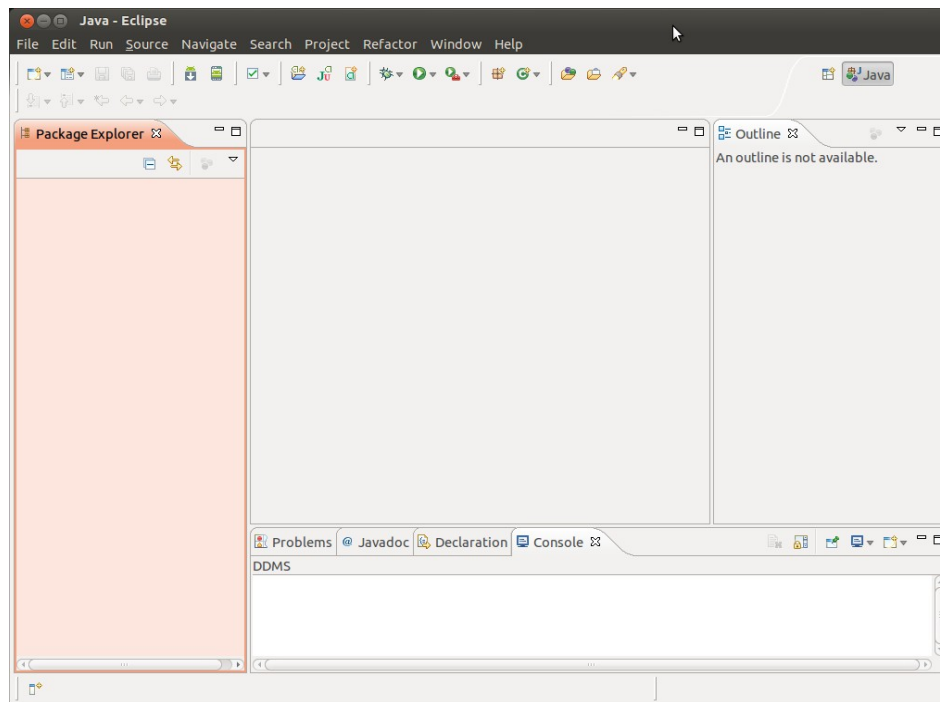
Instalasi JDK cukup dilakukan dengan mengikuti instruksi pada layar setelah file instalasi diklik. Anda dapat mengecek apakah komputer anda sebelumnya telah terinstall java atau tidak dengan menjalankan perintah **java -version** pada command line.

1.1.2 Instalasi Eclipse

Untuk sistem operasi windows file instalasi eclipse yang telah didownload biasanya tersedia dalam bentuk yang telah terkompresi. Cukup extract file tersebut pada folder yang anda inginkan misalkan **C:\Eclipse** dan anda sudah siap menggunakan IDE eclipse.

1.2 Memulai Eclipse

Jalankan eclipse dengan mengklik dua kali aplikasi Eclipse.exe (windows). Jika ini adalah kali pertama anda menjalankan eclipse, akan ada pertanyaan dimana anda akan menyimpan pekerjaan anda (workspace), pilih lokasi yang anda inginkan misalkan (<c:\eclipse-workspace>) dan checklist pilihan untuk menjadikan lokasi ini sebagai lokasi default.

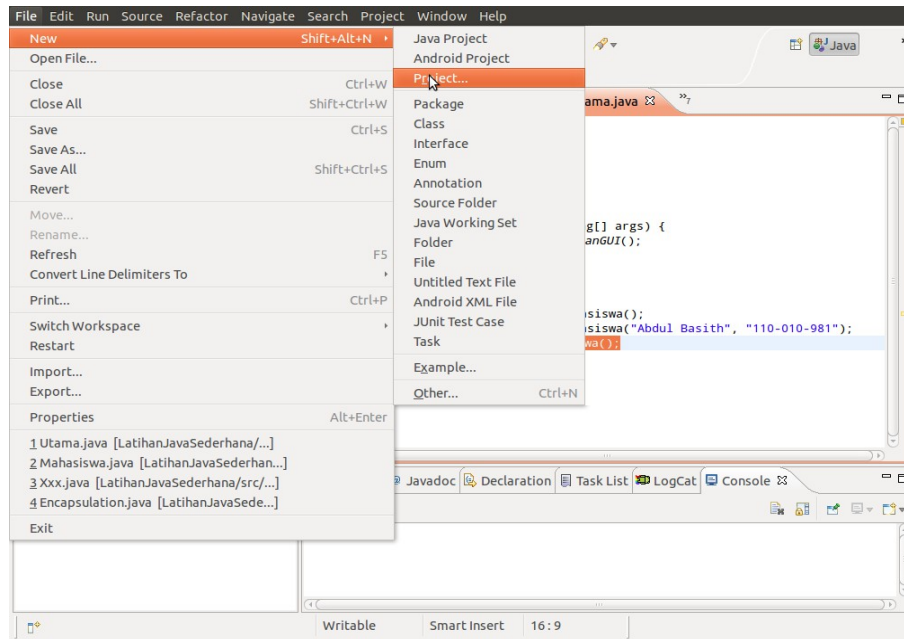


Gambar 3: Tampilan utama Eclipse IDE

Biasakan diri anda dengan tampilan utama Eclipse, perhatikan area-area utama seperti menu dan tool bar, package explorer, outline, problems dan console.

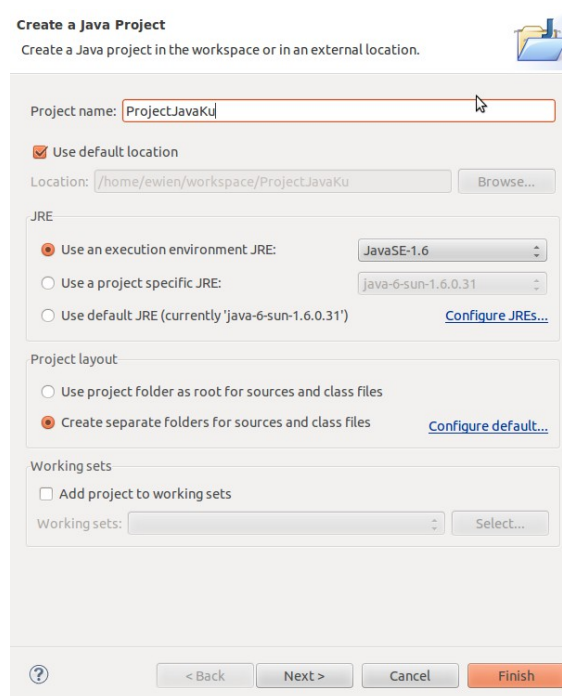
1.3 Membuat Project Java dalam Eclipse

Langkah selanjutnya adalah membuat project aplikasi java baru menggunakan eclipse. Pilih 'File – New – Project' pada eclipse kemudian pilih 'Java Project'. Jika sebelumnya anda pernah mengakses 'Java Project' dari menu 'File', pilihan tersebut selanjutnya akan muncul langsung tanpa perlu mengakses kempok menu 'Project' seperti pada screenshot dibawah.



Gambar 4: Membuat project baru

Pada layar selanjutnya anda akan diminta memberikan nama pada project yang akan anda buat. Isi nama project dengan nama yang anda inginkan kemudian klik 'Finish'.

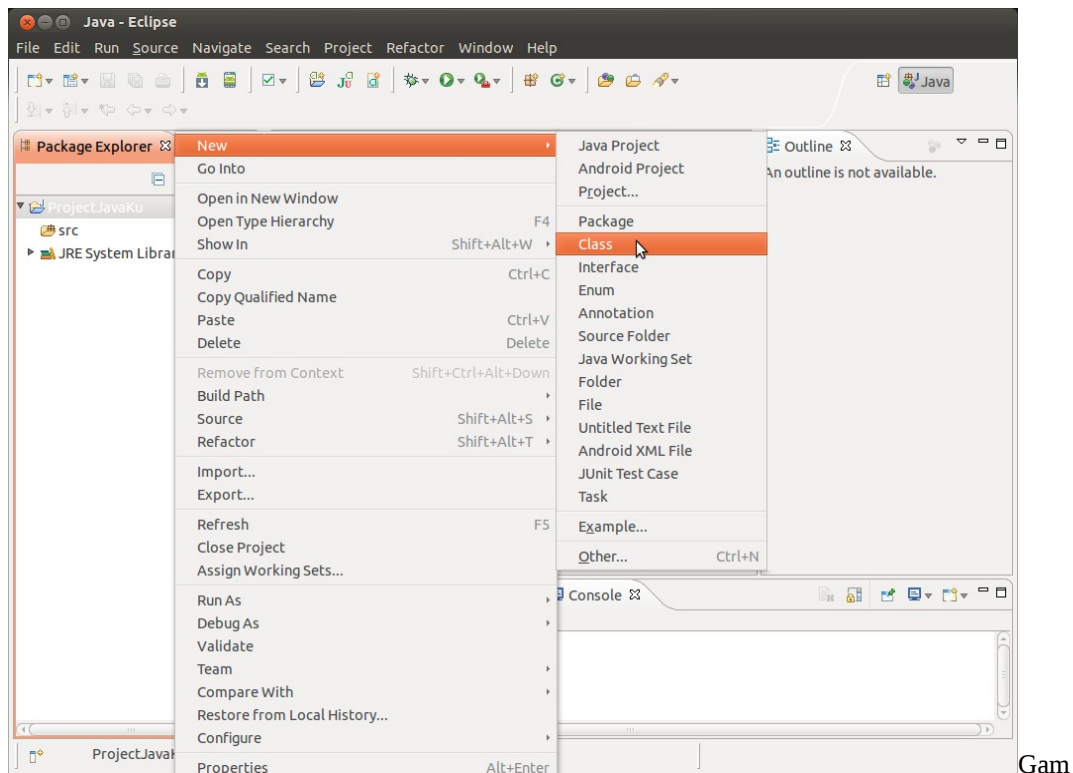


Gambar 5: Memberi nama project

1.4 Membuat Class 'Halo Java'

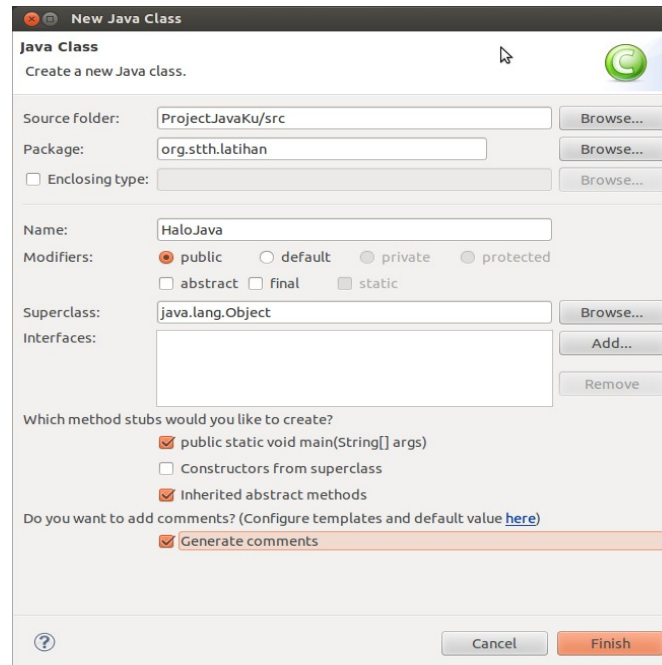
Setelah membuat project pertama, lanjutkan dengan membuat class. Klik kanan Project yang

sudah anda buat pada 'Package Explorer', pilih 'New' kemudian 'Class' seperti ditunjukkan pada screenshot di bawah ini.



bar 6: Membuat class baru

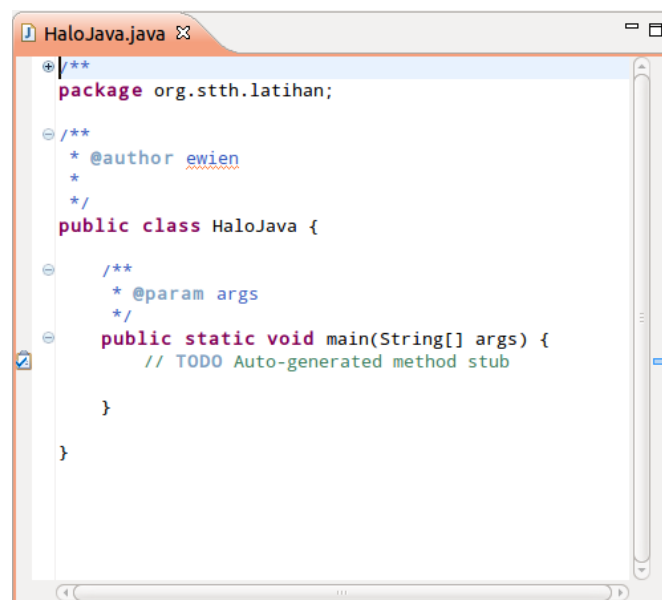
Langkah selanjutnya adalah memberi nama pada class yang anda buat. Aturan yang harus diikuti pada penamaan class adalah nama Class harus diawali dengan huruf besar dan tanpa spasi. Dianjurkan menggunakan huruf besar setiap memulai awal kata untuk class yang terdiri atas dua kata atau lebih. Pada contoh dibawah, nama class yang kita buat adalah 'HaloJava'



Gambar 7: Penamaan Class

Perhatikan bahwa pada contoh kita memilih untuk agar 'method stub' 'public static void main(String[] args)' untuk disertakan oleh eclipse secara otomatis. Method ini adalah method yang pertama kali dipanggil saat sebuah program Java dijalankan. Dengan kata lain, method ini harus tersedia pada class yang akan dipanggil pertama kali saat program yang kita buat dijalankan.

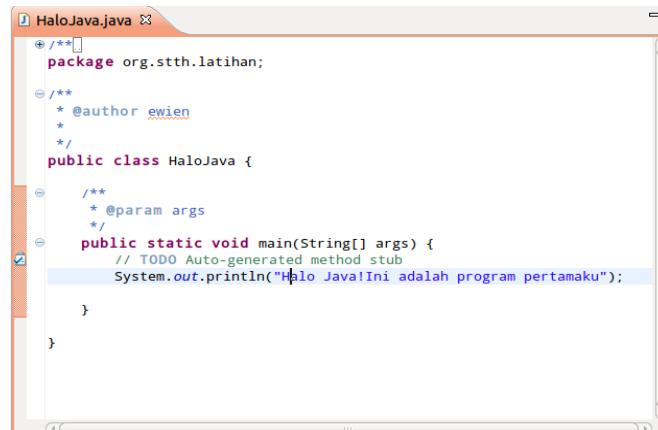
Setelah pengisian selesai, klik pada tombol 'Finish' dan di layar utama akan terlihat tampilan seperti screenshot dibawah ini.



Gambar 8: Kode program pada class HaloJava

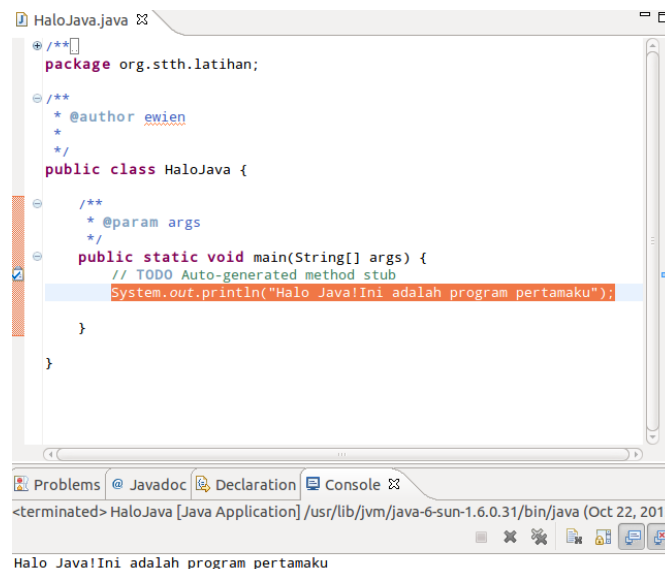
1.5 Menjalankan Program Java dari Eclipse

Setelah berhasil membuat class pertama, selanjutnya anda dapat mencoba menjalankan program sederhana yang masih berisi satu class Java. Sebelumnya, tambahkan baris `System.out.println("Halo Java!Ini adalah program pertamaku");` seperti pada screenshot di dalam method `main(String[] args)` yang ada pada class anda. Baris program yang ditambahkan berisi perintah untuk menampilkan tulisan di dalam tanda kutip pada console. Perhatikan agar huruf besar/kecil dan tanda baca identik dengan contoh, tulisan yang diapit tanda kutip dapat anda modifikasi sesuai keinginan anda.



Gambar 9: Menampilkan tulisan pada console

Kemudian klik kanan pada class anda dan pilih 'Run as' 'Java Application'. Jika anda melakukannya dengan benar, pada console akan muncul tulisan seperti pada screenshot berikut.



Gambar 10: Output pada console saat program dijalankan

1.6 Struktur Program Java Sederhana

1.7 Membiasakan Diri Menggunakan Eclipse

1.8 Aturan penamaan dalam Bahasa Pemrograman Java

Dalam pemrograman Java terdapat beberapa aturan penamaan yang harus diketahui programmer sebelum memulai menulis programnya. Dengan menaati aturan ini, akan membuat program menjadi lebih mudah dipahami (*readability*) baik oleh programmer yang menuliskannya atau programmer lain yang suatu saat membaca program yang ia buat:

- ▯ Pemrograman Java *Case-Sensitive*, artinya memperhatikan besar kecil huruf yang digunakan, kode 'Mahasiswa' dan 'mahasiswa' akan merujuk kepada dua hal yang berbeda.

- ▯ Nama package untuk class ditulis dengan huruf kecil (*lowercase*) dan diisi dengan alamat domain organisasi tempat programmer bernaung. Hal ini untuk menghindari adanya bentrok nama Class bila programmer menggunakan library dari berbagai tempat. Contoh penamaan:

- ▯ **package** com.betokngoncer.gamedengklak

- ▯ **package** org.stth.siakademik

- ▯ Nama class harus dimulai dengan huruf besar, jika terdiri atas lebih dari satu kata, maka setiap awal kata dimulai dengan huruf besar. Tata cara penulisan ini disebut juga *Camel Case*. Contoh penamaan :

- ▯ **class** Mahasiswa

- ▯ **class** MataKuliah

- ▯ Nama interface juga menggunakan *Camel Case* namun biasanya dengan diselipkan huruf "I" besar pada awal nama. Contoh:

- ▯ **interface** IComparable

- ▯ **interface** IEnumerable

- ▯ Nama method dan variabel harus dimulai dengan huruf kecil. Jika variabel atau method terdiri atas lebih dari maka setiap awal kata dimulai dengan huruf besar. Tata cara penulisan ini juga dikenal dengan nama *Mixed Case*. Contoh

- ▯ **int** nilaiUjian

- ▯ **char** nilaiAkhir

□ `String getNim()`

□ `void setAlamat()`

□ Konstanta ditulis menggunakan huruf besar (uppercase). Contoh

□ `static final double PI=3.14;`

1.9 Latihan



Buatlah sebuah project java baru menggunakan eclipse dan beri nama 'Latihan Java'

Buatlah sebuah class dengan nama 'ProgramKecilKu'.

Lengkapi class tersebut dengan method yang diperlukan agar dapat dijalankan sebagai sebuah aplikasi Java.

Isi method tersebut agar saat dijalankan ia menampilkan tulisan “Jika tulisan ini tampil di console, maka aku telah lulus bab pertama modul belajar Java”

2 Class dan Objects pada Java

2.1 Konsep

Konsep dasar dari pemrograman berorientasi objek adalah konsep *Class* and *Object*.

Object/Objek memiliki state dan behaviors. State atau juga sering disebut atribut/property/field adalah data yang melekat dan membentuk identitas suatu objek. Behavior adalah representasi perilaku objek bersangkutan.

Class adalah cara pandang terhadap klasifikasi/tipe dari objek. Class merepresentasikan struktur objek (state dan behavior) yang menjadi anggota dari class tertentu. **Class dapat juga disebut sebagai blueprint dari sebuah objek.**

Tabel berikut berisi contoh class beserta state dan behaviornya.

Class	State/Atribut/Property	Behaviour/Method
Mahasiswa	Nama, NIM, tanggal lahir, tahun masuk, jurusan	Mengambil mata kuliah, mendapat nilai, cuti, bayar SPP, lulus
Mata kuliah	Kode, judul, SKS, semester ajar, jurusan, dosen pengajar	Dibuka, ditutup
Lampu	Menyala	Nyalakan, matikan
Sepeda Motor	Merk, tipe, cc, warna, jarak tempuh	Starter, gas, kopling, pindah gigi, berhenti

Tabel berikut berisi contoh objek dari class Mahasiswa

State & behaviour	Nilai
State	
NIM	23507033
Nama	Nico Robin
Tanggal lahir	27-Juli-1986
Tahun masuk	2003
Jurusan	Teknik Informatika
Behaviour	
Mengambil matakuliah (matakuliah)lulus	
Mendapat nilai (matakuliah, nilai)	
Cuti (tanggalmulai,tanggalberhenti)	
Bayarspp(semester)	
Lulus()	

2.2 Deklarasi Class pada Java

Berikut adalah contoh sebuah class yang diimplementasikan dalam baris kode program Java. Dalam pemrograman berbahasa Java state diterjemahkan menjadi **variabel** sedangkan behavior diterjemahkan menjadi **method**. Pada contoh tersebut class Mahasiswa memiliki variabel **nama**, **nim**, **jurusan**, dan **thnMasuk**. Class Mahasiswa tersebut juga dilengkapi dengan method **lulus()** serta method **tampilkanDataMahasiswa()** untuk menampilkan nama, nim dan jurusan pada console.

```
public class Mahasiswa {  
  
    String nama;  
    String nim;  
    String jurusan;  
    int thnMasuk;  
    //Konstruktor  
    public Mahasiswa(){  
  
    }  
    public Mahasiswa(String nama, String nim) {  
        super();  
        this.nama = nama;  
        this.nim = nim;  
  
    }  
    void lulus(){  
  
    }  
    void tampilkanDataMahasiswa(){  
        System.out.println(nama);  
        System.out.println(nim);  
        System.out.println(jurusan);  
    }  
}
```

2.3 Constructor



Pertanyaan pengantar:

Jika Class merupakan blueprint sebuah object, bagaimana sebuah object dapat dibuat menggunakan blueprint tersebut?

Object pada Java merupakan instansiasi dari sebuah class. Sebuah objek dapat dibuat dengan memanggil constructor kelas tersebut. Setiap class dapat memiliki lebih dari satu jenis constructor.

Pada contoh kode class Mahasiswa sebelumnya, terdapat dua constructor yakni :

```
public Mahasiswa(){  
  
    }
```

```

public Mahasiswa(String nama, String nim) {
    super();
    this.nama = nama;
    this.nim = nim;
}

```

Berdasarkan dua constructor tersebut objek mahasiswa dapat dibuat dengan dua cara seperti contoh dibawah yakni menggunakan constructor tanpa parameter (siswa1) dan menggunakan parameter nama dan nim (siswa2).

Pada contoh class Utama dibawah siswa1 dan siswa2 merupakan object dari class Mahasiswa. Dua object tersebut kemudian diinstansiasi dengan menggunakan constructor yang berbeda. Object siswa1 diinstansiasi dengan constructor pertama sedangkan object siswa2 menggunakan constructor kedua yang menerima parameter nama dan nim siswa dengan tipe data string.

```

package org.stth.latihan;

public class Utama {

    public static void main(String[] args) {
        Mahasiswa siswa1;
        siswa1 = new Mahasiswa();
        System.out.println("nama siswa1 adalah: "+siswa1.nama);
        Mahasiswa siswa2;
        siswa2 = new Mahasiswa("Abdul Basith", "110-010-981");
        System.out.println("nama siswa2 adalah "+siswa2.nama);
    }
}

```

2.4 Mengakses variabel dan method object

Variabel dan method sebuah object dapat diakses dengan syntax **object.namaVariabel** dan **object.namaMethod()**. Dapat tidaknya sebuah variabel dan method object diakses oleh object lain bergantung dari tipe **access modifier** dari variabel dan method object tersebut (akan dipelajari pada bab 3.2). Bagian yang dihighlight pada kode berikut menunjukkan cara mengakses variabel dan method object siswa2.

```

package org.stth.latihan;

public class Utama {

    public static void main(String[] args) {
        Mahasiswa siswa1;
        siswa1 = new Mahasiswa();
        System.out.println("nama siswa1 adalah: "+siswa1.nama);
        Mahasiswa siswa2;
        siswa2 = new Mahasiswa("Abdul Basith", "110-010-981");
        System.out.println("nama siswa2 adalah "+siswa2.nama);
        siswa2.tampilkanDataMahasiswa();
    }
}

```

2.5 Latihan



Buatlah sebuah class MataKuliah dalam program Java dengan spesifikasi sebagai berikut :

1. Class MataKuliah memiliki state : kodeMatakuliah, namaMatakuliah dan sks.
2. Buatlah dua constructor untuk class tersebut. Constructor pertama tanpa parameter dan constructor kedua dengan parameter seluruh state class MataKuliah .
3. Buatlah method pada class MataKuliah tersebut untuk menuliskan kode, nama dan sks mata kuliah pada console

3 Syntax Dasar

3.1 Variabel

Variabel pada pemrograman adalah wadah penyimpanan nilai dengan tipe data tertentu. Setiap wadah dapat menampung jenis data tertentu sesuai dengan peruntukannya. Tipe data tersebut dapat berupa bilangan integer, desimal, karakter dan jenis lainnya. Saat program membuat sebuah variabel, operating system pada komputer mencadangkan lokasi pada memori untuk menampung variabel tersebut.



Petunjuk

Analogi (perumpamaan) untuk mengilustrasikan variabel adalah keranjang buah. Keranjang mewakili variabel dan buah mewakili tipe data yang dapat ditampung variabel tersebut. Keranjang tersebut kemudian dapat diisi dengan buah-buahan yang mewakili nilai dari variabel tersebut.

3.1.1 Deklarasi variabel

Sebelum dapat menggunakan variabel, variabel tersebut harus dideklarasikan terlebih dahulu. Sintaks deklarasi variabel pada Java dapat dilihat pada contoh berikut

```
int bil; // deklarasi variabel bil bertipe data int (integer sederhana)
char huruf; // deklarasi variabel huruf bertipe data char (character)
float bilDes; // deklarasi variabel bilDes bertipe data float (bilangan desimal)
String kata; // deklarasi variabel kata bertipe data class String
Mahasiswa siswa; // deklarasi variabel siswa bertipe data class Mahasiswa
```

3.1.2 Tipe Data Variabel

Java mengenal dua jenis tipe data yakni tipe data primitif dan tipe data objek (referensi). Contoh pertama hingga ketiga pada deklarasi variabel sebelumnya (int, char, float) adalah contoh variabel bertipe data primitif, sedangkan String dan Mahasiswa adalah tipe data object.

Variabel bertipe data primitif

Tipe data primitif adalah tipe data sederhana yang disediakan *built in* oleh bahasa pemrograman. Pada Java terdapat 8 jenis tipe data primitif. Jika tidak diinisialisasi, variabel dengan tipe data primitif akan terisi dengan nilai *default* seperti pada tabel berikut.

Tipe	Deksripsi	Default
byte	Integer 8 bit, dari -128 s/d 127	0
short	Integer 16 bit, dari -32.768 s/d 32.767	0
int	Integer 32 bit, dari -2.147.483.648 s/d 2.147.483.647	0
long	Integer 64 bit, dari -9.223.372.036.854.775.808 s/d 9.223.372.036.854.775.807	0

float	Desimal dengan 1 angka di belakang koma	0.0f
double	Desimal dengan 2 angka di belakang koma	0.0d
boolean	Bernilai logika benar (true) atau salah (false)	FALSE
char	Bernilai karakter	'\u0000' atau 0

Variabel bertipe data objek/referensi

Tipe data objek adalah tipe data yang digunakan untuk mengakses objek. Tipe data objek disebut juga tipe data referensi karena ia merujuk pada (*refer to*) class, array atau interface. Jika tidak diinisialisasi, maka variabel dengan tipe data objek tidak akan berisi nilai tertentu atau kosong (null).

3.1.3 Inisialisasi Variabel

Untuk variabel bertipe data primitif, variabel tersebut akan otomatis terisi nilai default saat dideklarasikan. Namun bila programmer ingin menginisiasi nilai untuk variabel tersebut, maka programmer dapat menggunakan sintaks seperti dibawah ini :

Contoh 1

```
int bil = 1; //deklarasi sekaligus inisialisasi variabel bertipe integer bil
dengan nilai 1
```

Contoh 2

```
int bil; //deklarasi variabel
bil = 1; //inisialisasi variabel yang telah dideklarasikan
```

Untuk variabel bertipe data referensi, inisialisasi dapat dilakukan dengan memanggil *constructor* kelas bersangkutan dengan syntax “new”. Contoh pernggunaan constructor telah diperlihatkan pada bab 2.3 mengenai object pada Java. Contoh berikut menyajikan dua gaya deklarasi dan inisialisasi variabel berupa objek siswa1 dan siswa2 dari Class Mahasiswa.

```
Mahasiswa siswa1 = new Mahasiswa();
Mahasiswa siswa2;
siswa2 = new Mahasiswa();
```



Petunjuk

*Siswa1 dan siswa2 kemudian disebut sebagai **objek** atau **instance** dari class Mahasiswa*

3.1.4 Variabel Instance, Class dan Lokal

Berdasarkan posisi dan jenis deklarasinya, variabel dapat dibedakan menjadi tiga jenis yakni variabel instance, variabel class dan variabel lokal.

<pre>class Lingkaran { final static double PI=3.14; double jarijari;</pre>	<pre>variabel instance : jarijari variabel class : PI variabel lokal : keliling dan</pre>
--	---

<pre> double getKeliling(){ double keliling; keliling = 2 * jarijari * PI; return keliling; } double getLuas(){ double luas; luas = PI * jarijari * jarijari; return luas; } </pre>	luas
--	------

Variabel instance merupakan variabel yang menjadi milik objek yang merupakan instance dari class tertentu. Ini berarti bahwa setiap objek dari class tersebut dapat memiliki nilai variabel instance yang berbeda.

Pada contoh kode class Lingkaran di atas yang merupakan variabel class adalah 'jarijari'. Artinya, misalkan terdapat dua objek Lingkaran a dan b, nilai 'jarijari' a dan b dapat berbeda.

Variabel class di sisi lain adalah milik class yang bersangkutan. Ini berarti bahwa setiap objek dari class tersebut akan berbagi nilai variabel class yang sama. Untuk mendeklarasikan variabel class, ditambahkan modifier static sebelum tipe data. Variabel ini umumnya digunakan untuk mendeklarasikan konstanta.

Pada contoh kode class Lingkaran di atas, yang merupakan variabel instance adalah 'PI'. Artinya, misalkan terdapat dua objek Lingkaran a dan b, nilai PI tetap sama.

Variabel lokal adalah variabel yang digunakan di dalam blok method atau *constructor*. Variabel ini tidak dapat digunakan di tempat lain selain pada method atau constructor yang mengandung variabel tersebut.

Pada contoh kode *class* Lingkaran di atas, terdapat dua method yakni getKeliling() dan getLuas(). Pada method getKeliling() terdapat variabel lokal 'keliling' sedangkan pada variabel lokal pada method getLuas() adalah 'luas'.

3.2 Modifier

Java mengenal dua jenis modifier yakni *access modifier* dan *non-access modifier*.

Dengan menggunakan modifier, programmer mengatur dari mana dan kapan sebuah *class*, variabel atau *method* dan *constructor* dapat dipanggil. Modifier digunakan dengan menyertakan keyword jenis modifier saat deklarasi *class*, variabel, *method* dan *constructor* bersangkutan.

Access modifier pada Java adalah fasilitas yang memungkinkan

programmer melakukan *encapsulation/data-hiding* pada variabel atau method tertentu. Java mengenal empat jenis access modifier yakni *public*, *protected*, *private* dan tanpa *access modifier*.

Pada level class, *access modifier* yang dapat digunakan adalah *public* dan tanpa *access modifier* dengan penggunaan sebagai berikut :

1. Dengan menggunakan *access modifier public*, *class* dapat diakses dari mana saja
2. Tanpa *access modifier*, *class* hanya bisa diakses dari package yang sama

Pada level member (variabel, method dan constructor), access modifier dapat digunakan sebagai berikut:

1. Untuk modifier *public* dan tanpa modifier, berperilaku sama dengan level class

2. Dengan menggunakan modifier private, akses dapat dilakukan dari dalam class bersangkutan
3. Dengan menggunakan modifier protected, akses dapat dilakukan dari dalam class, subclass dan package yang sama

Non-access modifier pada Java disediakan untuk berbagai keperluan antara lain:

1. Modifier static untuk deklarasi variabel dan method static yang menjadi milik class bersangkutan (lihat tipe variabel class pada 3.1.4).
2. Modifier final untuk finalisasi implementasi class, method dan variabel
3. Modifier abstract untuk membuat abstract class dan method
4. Modifier synchronized dan volatile digunakan untuk thread

Contoh class Lingkaran berikut adalah modifikasi dari class pada bab 3.1.4, perhatikan penambahan modifier pada variabel, method dan constructor yang ada. Pada class baru ini, variabel jarihari tidak akan bisa diakses dari luar class karena menggunakan modifier private. Nilai jarihari kemudian dapat diisi dengan memanggil constructor Lingkaran.

```
public class Lingkaran {
    final static double PI=3.14;
    private double jarihari;

    public Lingkaran(double jarihari) {
        this.jarihari = jarihari;
    }

    public double getKeliling(){
        double keliling;
        keliling = 2 * jarihari * PI;
        return keliling;
    }

    public double getLuas(){
        double luas;
        luas = PI * jarihari * jarihari;
        return luas;
    }
}
```

Pelajari dampaknya dengan mencoba kode dibawah ini

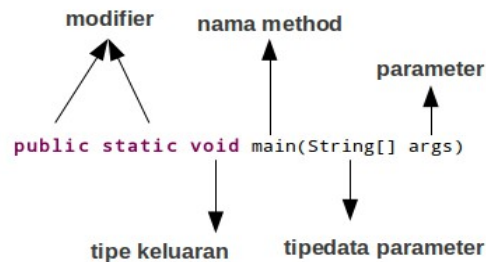
```
public class CobaModifier {

    public static void main(String[] args){
        Lingkaran a = new Lingkaran(3.0);
        System.out.println("Luas lingkaran :" + a.getLuas());
        System.out.println("Keliling lingkaran :" + a.getKeliling());
    }
}
```

3.3 Method

Sebuah method pada java adalah sebuah blok kode pada class berisi kumpulan perintah yang digabung untuk melakukan operasi tertentu.

Struktur deklarasi sebuah method dapat dilihat pada ilustrasi berikut ini



Gambar 11: Struktur deklarasi method

Beberapa poin aturan terkait method :

- ▢ Method dibuat dengan mixed case, diawali dengan huruf kecil pada kata pertama kemudian huruf besar pada awal kata berikutnya
- ▢ Method dapat dibuat dengan atau tanpa parameter masukan
- ▢ Method dapat dibuat dengan atau tanpa keluaran tertentu, method tanpa keluaran tertentu dideklarasikan dengan menuliskan 'void' sebagai tipe data keluarannya (void=kosong)
- ▢ Blok method dimulai dengan karakter '{' dan ditutup dengan karakter '}'
- ▢ Modifier sebuah method dapat dilihat pada sub bab 3.2 mengenai modifier
- ▢ Method dipanggil dengan syntax objek.namaMethod(), jika method tersebut tersebut memiliki modifier static maka ia dapat dipanggil dari Namaclass.namaMethod()

Berikut adalah beberapa contoh method dengan struktur yang berbeda

```
public static void tulisKeConsole(String kalimat) {  
    System.out.println(kalimat);  
}
```

Method bersifat static, tanpa keluaran, menerima masukan parameter string untuk dituliskan ke layar

```
public static int pilihYangBesar(int a, int b){  
    if (a>b){  
        return a;  
    } else {  
        return b;  
    }  
}
```

Method bersifat static, menerima parameter masukan a dan b dan keluaran dengan tipe data int

```
public double getLuas(){
    double luas;
    luas = PI * jarijari * jarijari;
    return luas;
}
```

Method tanpa parameter, dengan keluaran bertipe data double

```
public void predikatLulus(Mahasiswa siswa){
    if (siswa.lulus()){
        System.out.println(siswa.nama + " lulus dari STTH");
        if (siswa.ipk >=3.5){
            System.out.println(" dengan predikat Cum Laude");
        }
    } else {
        System.out.println(siswa.nama + " belum lulus dari STTH");
    }
}
```

Method tanpa keluaran, menerima masukan berupa object Mahasiswa

3.4 Operator

Java mengenal beberapa jenis operator, pada kode-kode program berikut akan didemonstrasikan cara penggunaan operator aritmatik, operasional dan logika pada Java.

3.4.1 Operator aritmatik pada Java

Kode dibawah ini mendemonstrasikan penggunaan operator aritmatik (+,-,/,%,++,--) pada Java

```
static void operatorAritmatik(){
    int a = 35;
    int b = 10;
    System.out.println(a + b); //operator penambahan
    System.out.println(a - b); //operator pengurangan
    System.out.println(a / b); //operator pembagian,
    System.out.println(a * b); //operator perkalian
    System.out.println(a % b); //operator modulus, sis da ri ha sil
    pembagian
    a++; //operator menambahkan 1 pada variabel
    System.out.println(a);
    b--; //operator mengurangi 1 pada nilai variabel
    System.out.println(b);
}
```

Hasil pada console bila kode di atas dijalankan

```
45
25
3
350
5
36
9
```

3.4.2 Operator relasional pada Java

Kode dibawah ini mendemonstrasikan penggunaan operator relasional (+,-,/,%,++,--) pada Java

```

static void operatorRelasional(){
    int a = 35;
    int b = 10;
    System.out.println(a == b); //operator sama dengan
    System.out.println(a != b); //operator tidak sama dengan
    System.out.println(a > b); //operator lebih besar
    System.out.println(a < b); //operator lebih kecil
    System.out.println(a >= b); //operator lebih besar sama dengan
    System.out.println(a <= b); //operator lebih kecil sama dengan
}

```

Hasil pada console bila kode di atas dijalankan

```

false
true
true
false
true
false

```

3.4.3 Operator logika pada Java

Kode dibawah ini mendemonstrasikan penggunaan operator logika (&&,||,!) pada Java

```

static void operatorLogika(){
    boolean a=true;
    boolean b=false;
    System.out.println(a && b); //operator AND
    System.out.println(a || b); //operator OR
    System.out.println(a && !b); //operator NOT (!)
}

```

Hasil pada console bila kode di atas dijalankan

```

false
true
true

```

3.5 Conditional (if & switch)

Dua statement conditional/pengambilan keputusan pada bahasa pemrograman Java adalah statement 'if' dan statement 'switch'.

3.5.1 Statement if

Statement if menggunakan evaluasi terhadap sebuah ekspresi boolean (true/false). Bila ekspresi boolean tersebut bernilai true, maka kode di dalam blok selanjutnya akan dieksekusi. Statement if juga dapat dilengkapi dengan statement else yang berfungsi menyediakan kode program untuk dieksekusi bila ekspresi boolean tersebut bernilai false. Kode dibawah mendemostrasikan penggunaan statement if dan if else.

```

static void demoIfStatement(){
    boolean b = true;
    if (b) {
        System.out.println("b bernilai true");
    }
}

```

```

static void demoIfElseStatement(){

```

```

int a=10;
int b=20;
if (a >= b){
    System.out.println("a lebih besar sama dengan b");
} else {
    System.out.println("a lebih kecil dari b");
}
}

```

Jika kode dijalankan, apa output yang tertulis pada console?

3.5.2 Statement switch

Statement switch digunakan untuk menguji sebuah sama-tidaknya nilai variabel tertentu terhadap sebuah daftar nilai. Tiap nilai dalam daftar disebut sebuah 'case', dan variabel dimaksud akan diuji terhadap case yang ada secara berurutan.

Kode berikut mendemonstrasikan cara penggunaan statement switch

```

static void demoSwitchStatement(){
    char nilai = 'A';
    switch (nilai) {
        case 'E':
            System.out.println("Anda belum lulus");
            break;
        case 'D':
            System.out.println("Anda belum lulus");
            break;
        case 'C':
            System.out.println("Anda lulus dengan nilai cukup");
            break;
        case 'B':
            System.out.println("Anda lulus dengan nilai baik");
            break;
        case 'A':
            System.out.println("Selamat, anda mendapatkan grade nilai
terbaik");
            break;
        default:
            System.out.println("Nilai anda tidak valid");
            break;
    }
}

```

Jika kode dijalankan, apa output yang tertulis pada console?

3.6 Loop (perulangan)

Perulangan pada Java dapat dilakukan dengan dua cara yakni menggunakan perintah *while* dan *for*.

3.6.1 Perintah while

Perintah while memiliki dua variasi yakni while dan do-while. Pada do-while, perintah di dalam blok do dijalankan sekali sebelum ekspresi boolean yang ada dievaluasi. Dua contoh berikut mendemonstrasikan penggunaan variasi tersebut.

```
static void demoWhile(){
    int c = 0;
    while (c<10){
        System.out.println(c);
        c++;
    }
}
```

```
static void demoDoWhile(){
    int c = 9;
    do {
        System.out.println(c);
        c++;
    } while (c>10);
}
```

Jika kode dijalankan, apa output yang tertulis pada console?

3.6.2 Perintah for

Perintah for memiliki beberapa variasi dan mendukung perulangan pada tipe data array dan collection. Kode sederhana perulangan menggunakan perintah for dapat dilihat pada kode berikut, sedangkan kode perulangan dengan variasi lainnya akan dibahas pada bab mengenai array.

```
static void demoFor(){
    for (int i = 0; i < 10; i++) {
        System.out.println(i);
    }
}
```

Jika kode dijalankan pada console akan tertulis bilangan 0 s/d 9

3.7 Accessor dan Mutator (Getter & Setter)

Salah satu kelebihan dari Java sebagai sebuah bahasa pemrograman berbasis object adalah fasilitas encapsulation. Fasilitas ini disediakan melalui access modifier (private/protected/public) yang dapat disematkan pada class, variabel ataupun method.

Contoh penerapan standar penggunaan access modifier adalah pada method **accessor** dan **mutator** atau lazim disebut **getter** dan **setter**.

Pada standar ini variabel sebuah object tidak diperkenankan untuk diakses maupun diubah secara langsung. Akses terhadap nilai variabel object harus dilakukan melalui method getter sedangkan pengubahan nilai dilakukan melalui setter. Untuk mengontrol hal ini, seluruh variabel pada class object bersangkutan dideklarasikan dengan access modifier private.

Dengan penggunaan getter/setter kita dapat mengontrol nilai yang diberikan terhadap variabel bersangkutan agar sesuai dengan aturan tertentu yang ingin kita terapkan.

Perhatikan contoh pada kode class Lingkaran berikut

```
package org.stth.latihan;

public class Lingkaran {
    final static double PI=3.14;
    private double jarijari;
```



```

public Lingkaran(double jarijari) {
    this.jarijari = jarijari;
}

public double getJarijari() {
    return jarijari;
}

public void setJarijari(double jarijari) {
    if (jarijari < 0){
        this.jarijari = 0;
    } else {
        this.jarijari = jarijari;
    }
}

public double getKeliling(){
    double keliling;
    keliling = 2 * jarijari * PI;
    return keliling;
}

public double getLuas(){
    double luas;
    luas = PI * jarijari * jarijari;
    return luas;
}
}

```

Perhatikan pada tipe data variabel jarijari dan method setJarijari. Tipe data double memungkinkan nilai negatif, sedangkan jari-jari sebuah lingkaran tidak mungkin bernilai negatif. Tambahan kode yang diberikan pada method setJarijari mengontrol agar jari-jari selalu bernilai nol atau positif.



Petunjuk

Eclipse menyediakan fasilitas untuk membuat getter dan setter secara otomatis. Klik kanan pada sebarang tempat pada class editor, pilih source, kemudian pilih generate getter & setter

3.8 Latihan



Buatlah sebuah class NilaiMataKuliah dalam program Java dengan spesifikasi sebagai berikut :

1. Class NilaiMataKuliah memiliki state/variabel : mataKuliah dengan tipe data object Matakuliah, mahasiswa dengan tipe data object Mahasiswa, thnAmbil dengan tipe data int dan nilai dengan tipe data Char
2. Beri modifier private pada seluruh state tersebut
3. Lengkapi class dengan getter & setter untuk seluruh state tersebut
4. Lengkapi setter nilai (setNilai) dengan validasi agar nilai yang dapat diterima program hanya A,B,C,D,E selain dari nilai itu program secara otomatis mengisi nilai E

- | | |
|--|--|
| | <ol style="list-style-type: none">5. Lengkapi class dengan sebuah method <code>isLulus()</code> yang menghasilkan nilai <code>true/false</code> berdasarkan isi dari state nilai6. Lengkapi class dengan sebuah method <code>getBobot()</code> yang menghasilkan nilai <code>double</code> berisi bobot nilai mata kuliah (jika A=4, jika B=3, C=2, D=1, lainnya=0) |
|--|--|

4 Struktur Data Array dan Collection



Pertanyaan pengantar:

Jika mahasiswa mengikuti lebih dari satu mata kuliah, tipe data apa yang dapat digunakan pada variabel daftar mata kuliah yang diikuti oleh mahasiswa tersebut?

Untuk menyimpan lebih dari satu object pada sebuah variabel, java menyediakan tipe data **array** dan **collection**.

4.1 Array

Untuk keperluan yang sederhana, programmer dapat memilih tipe data array. Array dapat digunakan untuk koleksi object dengan jumlah yang tetap. Berdasarkan keterbatasan ini array hanya cocok digunakan jika jumlah object yang disimpan pada array tersebut yang tidak memiliki kemungkinan bertambah. Contoh berikut menyajikan cara menggunakan array, dari deklarasi hingga looping ke dalam array bersangkutan.

Penting untuk diingat bahwa pada bahasa pemrograman Java, indeks sebuah array dimulai dari angka 0.

```
package org.stth.latihan;

public class CobaArray {

    static void arrayString(){
        //deklarasi dan instansiasi array
        String[] lampuLaluLintas;
        lampuLaluLintas = new String[3];
        //isi array
        lampuLaluLintas[0]="Hijau";
        lampuLaluLintas[1]="Kuning";
        lampuLaluLintas[2]="Merah";
        //output isi array ke console
        for (String string : lampuLaluLintas) {
            System.out.println(string);
        }
    }

    static void arrayIntDuaDimensi(){
        int[][] papancatur;
        papancatur = new int[8][8];
        for (int i = 0; i < 8; i++) {
            for (int j = 0; j < 8; j++) {
                System.out.println("Kolom "+ i+ " dan baris "+ j);
            }
        }
    }

    public static void main(String[] args){
        arrayString();
        arrayIntDuaDimensi();
    }
}
```

```
}
```

4.2 Collection

Jika array memiliki kelemahan dari sisi jumlah kandungan object yang tidak bisa ditambah, tidak demikian halnya dengan collection. Dengan collection, programmer tidak perlu mendefinisikan ukuran seperti pada array. Jumlah object yang dikandung collection dapat mengembang sesuai kebutuhan.

Java mengenal 3 kelompok utama tipe data collection yakni **list**, **set** dan **map**. ketiga jenis collection ini dapat mengembang sesuai jumlah data yang dimasukkan ke dalamnya.

- List adalah kelompok tipe data collection yang dapat diperlakukan sebagaimana array biasa. List mengijinkan object yang sama untuk ditambahkan dua kali (duplikat). List dapat diakses menggunakan indeks posisi object pada list bersangkutan. Tipe data yang termasuk dalam kelompok list antara lain: ArrayList dan LinkedList
- Set tidak mengijinkan adanya duplikasi isi dan object di dalamnya tidak dapat diakses menggunakan indeks seperti pada List. Tipe data yang termasuk dalam kelompok set adalah HashSet dan LinkedHashSet dan TreeSet. TreeSet adalah jenis set khusus yang isinya terurut berdasarkan urutan alaminya (angka, karakter), namun dengan performa lebih lambat.
- Map menggunakan key untuk mengakses object/value tertentu dan dapat mengandung duplikat. Tipe data yang termasuk dalam kelompok map adalah HashMap, TreeMap, LinkedHashMap dan IdentityHashMap

Contoh class berikut memperlihatkan bagaimana cara menggunakan ArrayList, HashMap dan TreeSet. Jalankan class tersebut dan perhatikan urutan munculnya output list pada console.

```
package org.stth.latihan;

import java.util.ArrayList;
import java.util.HashMap;
import java.util.HashSet;
import java.util.Iterator;
import java.util.Map;
import java.util.Map.Entry;
import java.util.Set;
import java.util.TreeSet;

public class CobaListSetMap {

    static void contohList(){
        //deklarasi dan instansiasi array list
        ArrayList<Mahasiswa> daftarMahasiswa;
        daftarMahasiswa = new ArrayList<Mahasiswa>();
        //mengisi daftar dengan 30 object mahasiswa
        for (int i = 0; i < 30; i++) {
            Mahasiswa s = new Mahasiswa();
            s.nama = "Mahasiswa ke-"+(i+1);
            daftarMahasiswa.add(s);
        }
        //loop ke dalam array list menggunakan for
```

```

        for (Mahasiswa mahasiswa : daftarMahasiswa) {
            System.out.println(mahasiswa.nama);
        }
    }
    static void contohMap(){
        HashMap<String, String> kabkota;
        kabkota = new HashMap<String, String>();
        //isi dengan singkatan sbg key, nama panjang sebagai value
        kabkota.put("Mtr", "Kotamadya Mataram");
        kabkota.put("Lobar", "Kabupaten Lombok Barat");
        kabkota.put("Loteng", "Kabupaten Lombok Tengah");
        kabkota.put("Lotim", "Kabupaten Lombok Timur");
        //loop ke dalam hashmap menggunakan iterator
        Iterator iter = kabkota.entrySet().iterator();
        while (iter.hasNext()) {
            Map.Entry isi = (Map.Entry) iter.next();
            System.out.println(isi.getKey() + " : " + isi.getValue());
        }
    }

    static void contohSet(){
        //deklarasi dan inisiasi TreeSet dengan isi String
        Set<String> s = new TreeSet<String>();
        //penambahan object ke dalam treeset
        s.add("Brokoli");
        s.add("Dendeng");
        s.add("Pelecing");
        s.add("Capcay");
        s.add("Ares");
        s.add("Beberok");
        //loop dan mencetak isi TreeSet ke console
        for (String string : s) {
            System.out.println(string);
        }
    }

    public static void main(String[] args){
        contohList();
        contohMap();
        contohSet();
    }
}

```

4.3 Latihan



Buatlah sebuah Java Project baru dengan judul “Latihan Collection”
 Buatlah/Siapkan class Mahasiswa (contoh bab 1) MataKuliah (Latihan bab 2)
 dan class NilaiMataKuliah (latihan bab 3) pada project tersebut
 Modifikasi class Mahasiswa dengan spesifikasi sebagai berikut :

1. State/Variabel: nim, nama, jurusan dan angkatan dan daftar nilai mata kuliah dalam ArrayList (misal: ArrayList<NilaiMatakuliah> nilai)
2. Buatlah method yang dapat menghitung IPK mahasiswa berdasarkan

	nilai pada variabel nilai mata kuliah yang anda buat Insiasi sebuah object mahasiswa dengan 3 jenis mata kuliah beserta nilainya kemudian tampilkan IPK mahasiswa tersebut pada console
--	---

5 Tutorial Akses Database dengan JDBC

Bab ini berisi tutorial akses database menggunakan Java Database Connectivity (JDBC). Di akhir tutorial, anda akan dapat membuat program yang terkoneksi dengan database server. Contoh pada tutorial ini menggunakan database MySQL, bila anda sudah biasa menggunakan RDBMS, anda dapat menggunakan tutorial ini dengan menggunakan RDBMS lain dengan mengganti driver JDBC sesuai database yang anda gunakan dan parameter koneksi pada class ConnectionFactory.

Tutorial ini memanfaatkan latihan-latihan pada bab sebelumnya, terutama latihan pada bab 2, 3 dan 4. Program pada tutorial ini akan membaca dan menyimpan data mahasiswa, mata kuliah dan nilai mata kuliah yang diambil mahasiswa pada database.

5.1 Menyiapkan Database

Bagian ini berisi tutorial menyiapkan sebuah database server (MySQL) sebagai target koneksi program yang akan kita bangun beserta database dan tabel-tabel untuk menampung data kita.

5.1.1 Instalasi Database Server (MySQL)

Jika laptop/komputer anda sudah terinstall MySQL/RDBMS lain, bagian ini dapat dilewati.

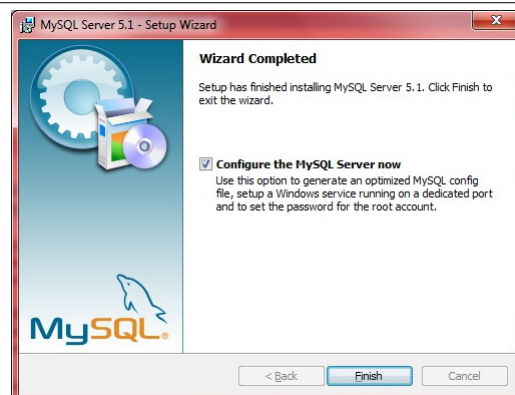
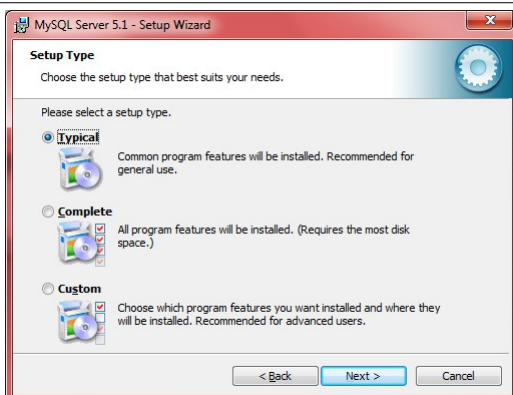
Siapkan file instalasi MySQL Server dan MySQL Workbench/

MySQL Community Server terbaru dapat didownload secara gratis pada alamat : <http://dev.mysql.com/downloads/mysql/>

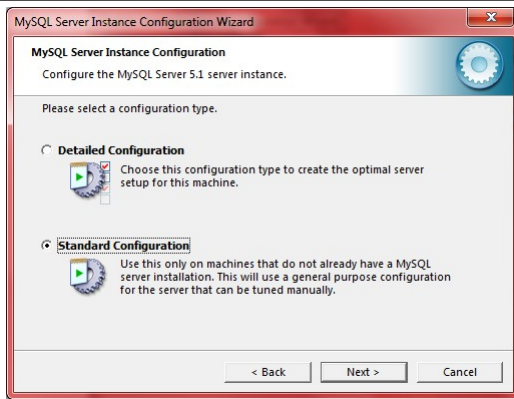
MySQL Workbench bisa didapatkan pada alamat

<http://dev.mysql.com/downloads/workbench/>

Lakukan instalasi MySQL server seperti layaknya instalasi program biasa dengan mengklik file instalasi yang anda miliki, ikuti perintah yang ada pada layar. Jika anda bingung saat installer menawarkan pilihan instalasi, anda dapat menggunakan screenshots berikut sebagai panduan.



Checklist pilihan untuk mengkonfigurasi MySQL server



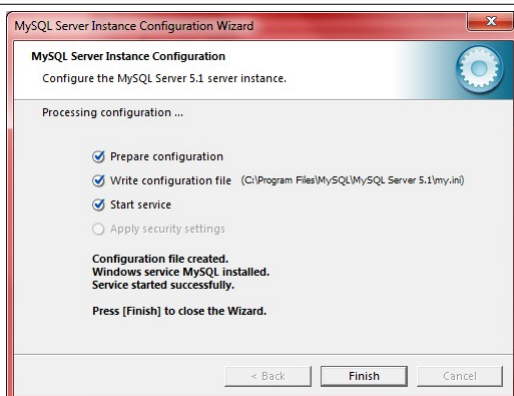
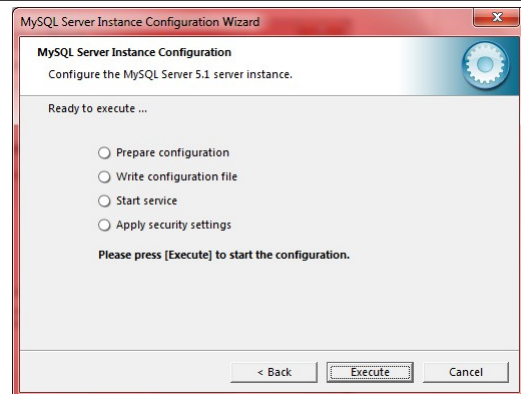
Pilih tipe konfigurasi **Standard**



Klik **Execute** untuk langkah terakhir instalasi



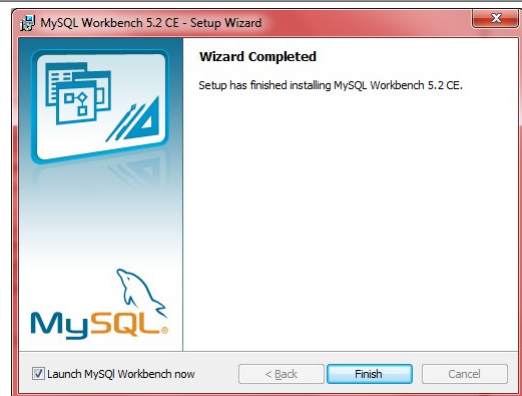
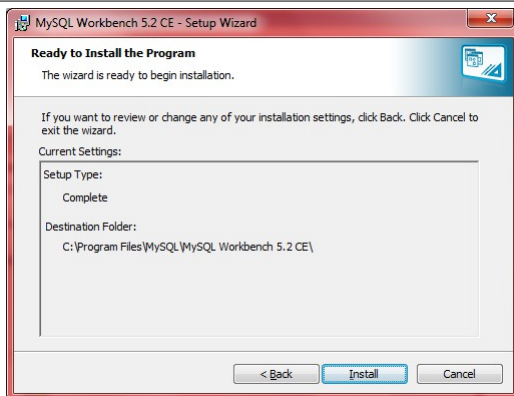
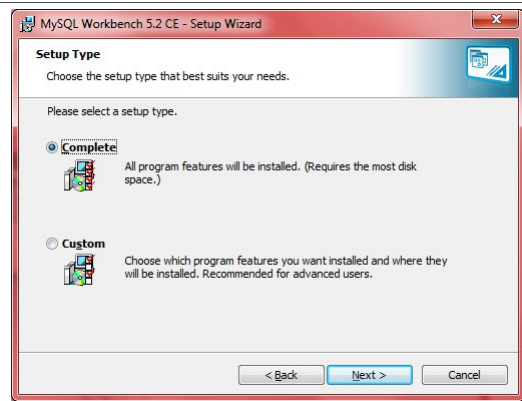
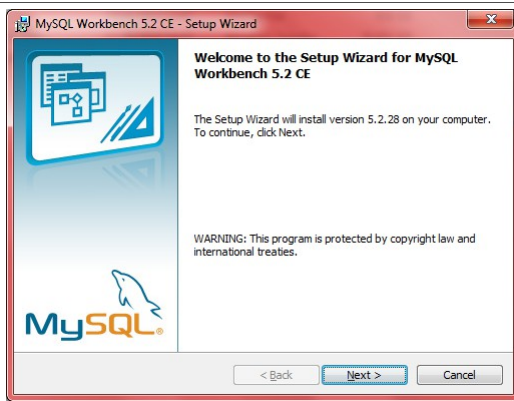
Berikan password untuk user root, ingat/tulis password yang anda isi



Jika layar ini tampil berarti komputer anda telah terinstall MySQL

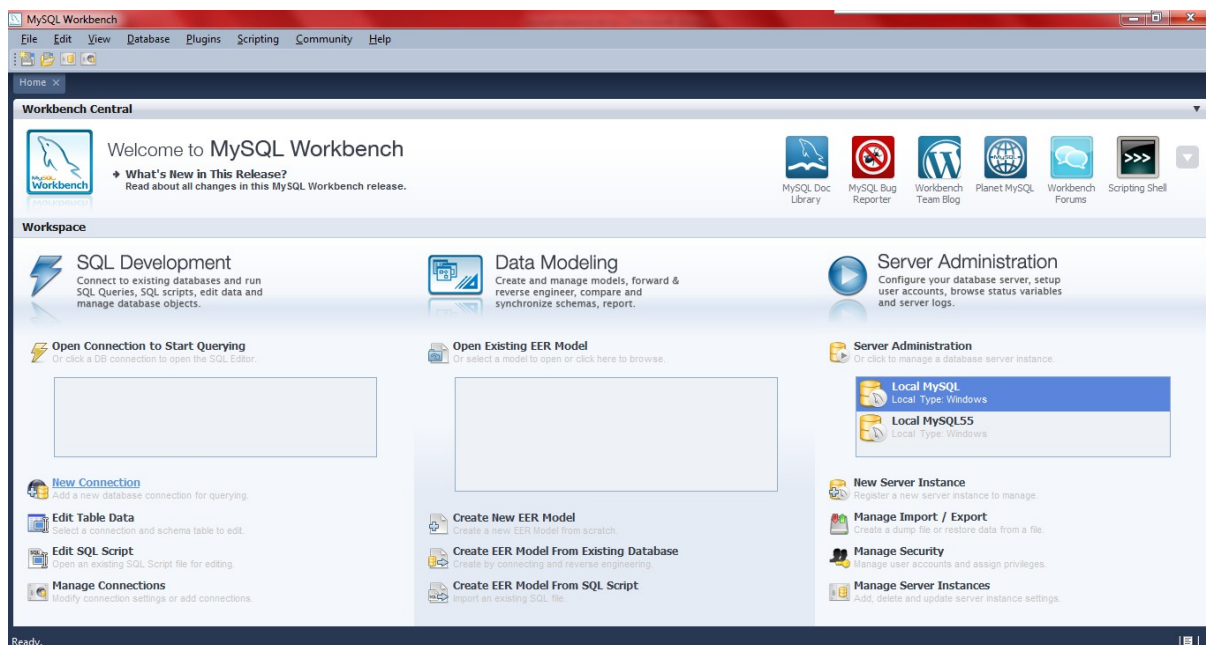
5.1.2 Instalasi MySQL Workbench

Langkah selanjutnya adalah menginstall MySQL workbench sebagai GUI Tool untuk mengelola database MySQL kita. Klik dua kali pada file instalasi dan gunakan screenshot berikut sebagai panduan saat installer menawarkan pilihan instalasi.

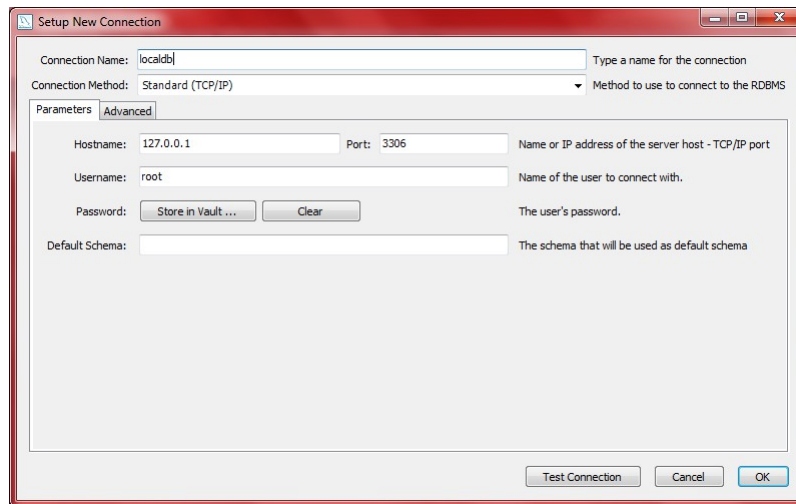


Kemudian kita akan mencoba koneksi database server yang telah terinstall pada komputer menggunakan MySQL workbench.

Jalankan MySQL Workbench, sampai anda menjumpai layar berikut

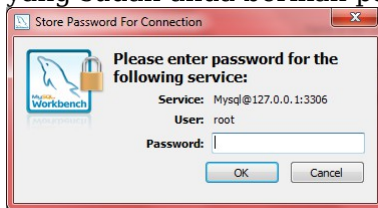


Perhatikan kolom pertama (SQL Development), klik pada link 'New Connection', akan muncul form untuk setup koneksi baru

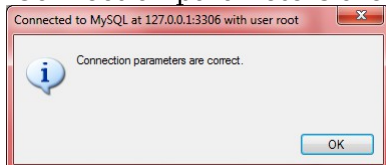


Berikut adalah panduan untuk melengkapi isian pada form koneksi baru

1. Isi 'Connection Name' dengan nama koneksi, misalnya **localdb**
2. Pilih **Standard (TCP/IP)** sebagai metode koneksi (biasanya sudah terpilih secara default)
3. Isi hostname dengan **127.0.0.1** dan port **3306** (biasanya sudah terisi secara default)
4. Isi username dengan **root** (biasanya sudah terisi secara default)
5. Klik pada tombol 'Store in Vault' dan isi form yang muncul dengan password database yang sudah anda berikan pada instalasi MySQL sebelumnya kemudian klik 'OK'



6. Klik pada tombol 'Test Connection', jika isian anda benar akan muncul notifikasi 'Connection parameters are correct', kemudian 'klik OK'.



7. Klik 'OK' untuk menyimpan koneksi ini, anda akan dibawa kembali ke layar utama Workbench.

Jika seluruh langkah ini dapat anda jalankan, berarti database server MySQL pada komputer anda siap digunakan.

5.1.3 Desain Tabel pada Database

Pada tahap ini kita akan melakukan merancang tabel-tabel database untuk menyimpan data mahasiswa, mata kuliah dan nilai mata kuliah.



Petunjuk

Tabel dibentuk dari field-field dengan tipe data tertentu, setiap field mewakili satu jenis data. Data sebuah tabel disebut record, setiap record memiliki nilai atas field-field yang dimiliki tabel tersebut.

Tabel untuk menyimpan data mahasiswa.

Untuk menyimpan data mahasiswa kita akan membuat tabel bernama sama yakni tabel mahasiswa. Variabel pada Class Mahasiswa yang kita miliki sebelumnya (bab 2.2) akan kita konversi menjadi field-field pada tabel tersebut.

Variabel pada class Mahasiswa dari bab 2.2 antara lain:

- nim,
- nama,
- jurusan,
- tahun masuk

Selain keempat variabel tersebut, kita akan menambahkan satu variabel baru yakni 'id' pada Class Mahasiswa untuk kita jadikan 'primary key' pada tabel Mahasiswa.

Kita bisa saja menjadikan 'nim' sebagai primary key pada tabel Mahasiswa, namun praktik seperti ini tidak direkomendasikan karena kita tidak bisa menjamin bahwa aturan mengenai pengkodean nim mahasiswa tidak akan berubah. Di sisi lain primary key haruslah kebal terhadap perubahan agar tidak diperlukan banyak penyesuaian pada level kode program saat terdapat perubahan aturan.

Tabel berikut memperlihatkan konversi variabel dari class Mahasiswa menjadi field pada tabel Mahasiswa pada database

Variabel class Mahasiswa pada kode Program	Field tabel Mahasiswa pada Database
id (long)	id (int), primary key, auto increment
nim (String)	nim (varchar)
nama (String)	nama (varchar)
jurusan (String)	jurusan (varchar)
thnMasuk (int)	thnmasuk (int)

Ilustrasi penyimpanan data Mahasiswa pada tabel mahasiswa dapat dilihat pada tabel berikut

record	field				
	id	nim	nama	jurusan	thnmasuk
	1	1210991	Dende Raisah	Teknik Informatika	2012
	2	1210992	Menggep el Fanshury	Teknik Informatika	2012
	3	1210993	Mahdan Rengget	Teknik Informatika	2012

Dengan cara yang sama seperti pada class Mahasiswa maka konversi variabel dari class MataKuliah menjadi tabel matakuliah pada database menjadi

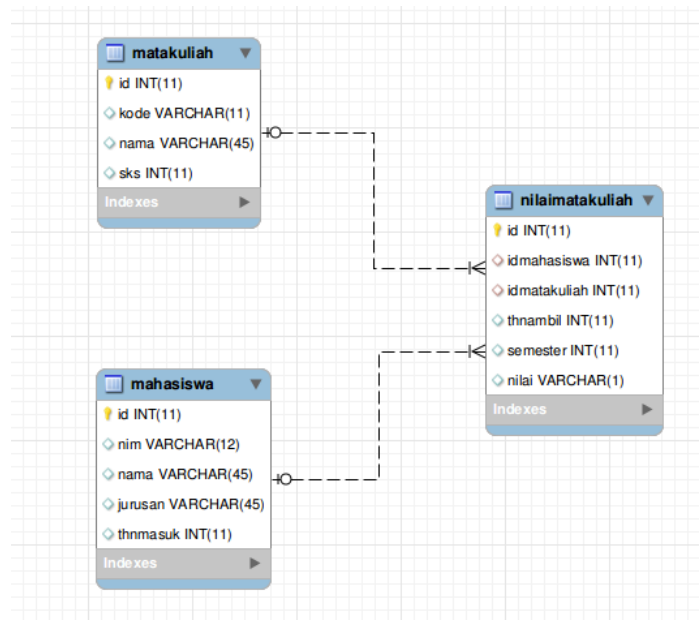
Variabel class MataKuliah pada kode	Field tabel matakuliah pada Database
--	---

Program	
id (long)	id (int), primary key
kode (String)	kode (varchar)
nama (String)	nama (varchar)
sks (int)	sks (int)

Untuk class NilaiMataKuliah, terdapat tipe variabel objek berupa mahasiswa dan matakuliah, kita cukup mengkonversinya menjadi primary key dari tabel mahasiswa dan primary key dari tabel matakuliah yakni kolom id.

Variabel class NilaiMataKuliah pada kode Program	Field tabel nilaimatakuliah pada Database
id (long)	id (int), primary key
mahasiswa (Mahasiswa)	idmahasiswa (int), foreign key
matakuliah (MataKuliah)	idmatakuliah (int), foreign key
thnAmbil (int)	thnambil (int)
semester (int)	semester (int)
nilai (char)	nilai (varchar)

Jika dipresentasikan dalam sebuah ERD (Entity Relationship Diagram), maka relasi antar tabel kita akan terlihat seperti diagram dibawah ini



Relasi antara tabel mahasiswa, matakuliah dan nilaimatakuliah



Petunjuk

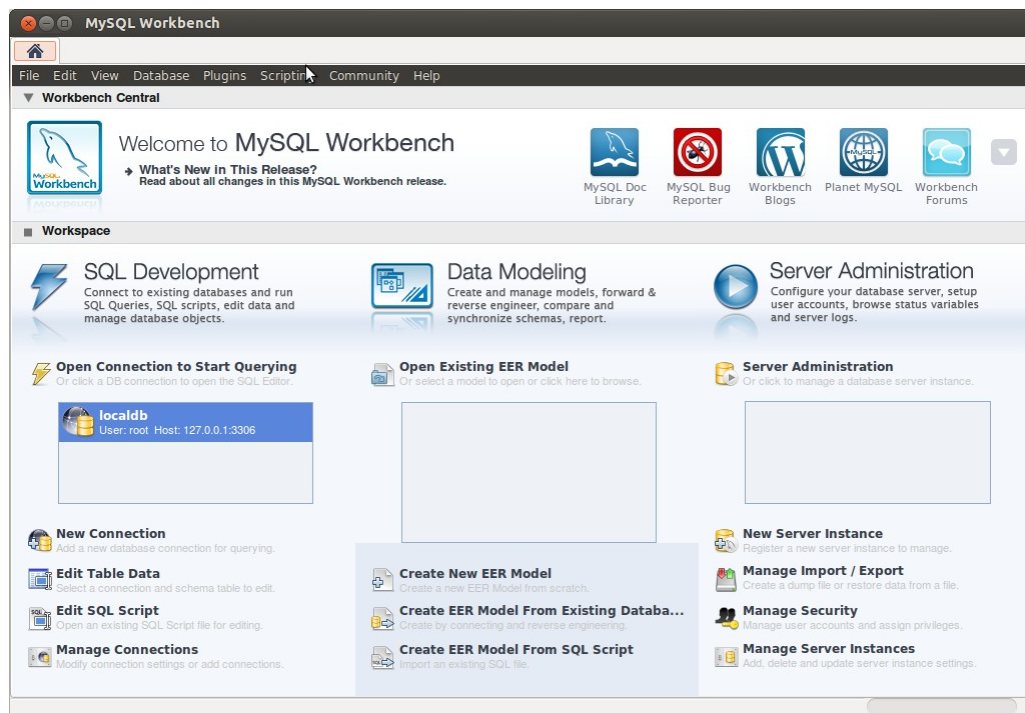
Entity Relationship Diagram (ERD) adalah sebuah diagram yang menggambarkan keterkaitan antara tabel-tabel pada sebuah database.

5.1.4 Inisiasi Database dan Data Dummy

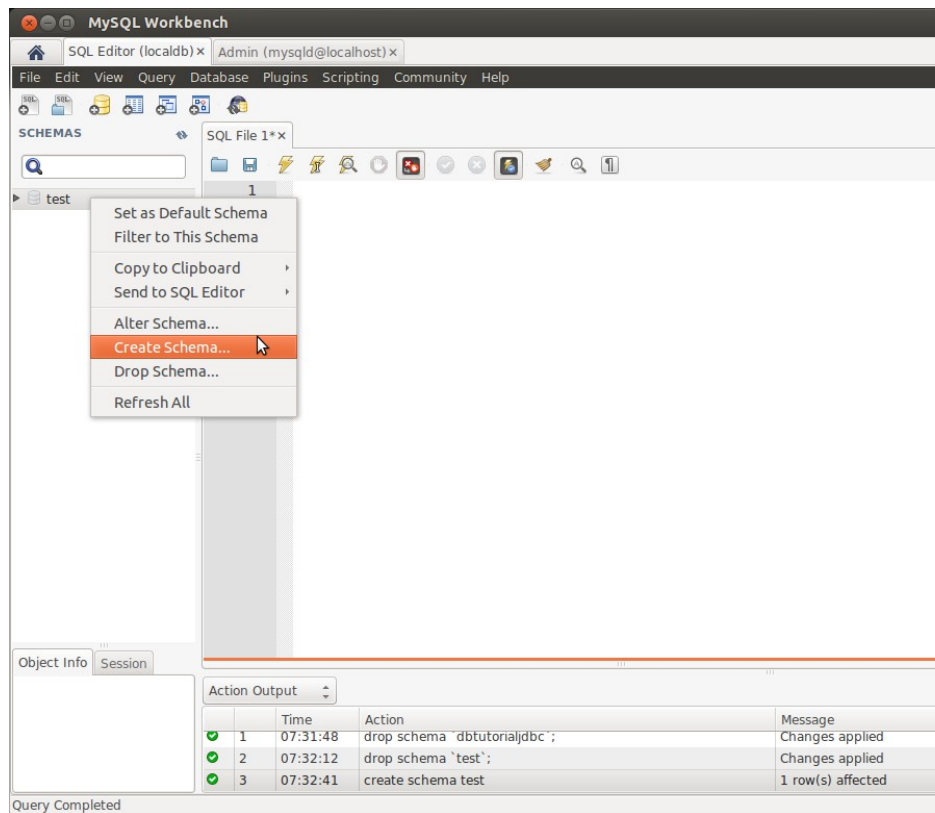
Pada titik ini kita sudah memiliki desain tabel untuk menyimpan data-data kita. Langkah selanjutnya adalah menginisiasi database dan tabel sesuai desain yang telah kita buat dengan menggunakan MySQL workbench.

Pada tutorial ini disediakan dua pilihan untuk melakukan hal tersebut. Pertama, menggunakan MySQL workbench GUI untuk membuat tabel dan melakukan entry data. Jika memilih menggunakan cara ini, anda akan dapat memahami lebih baik konsep tabel, field, record dan key pada database. Kedua, menggunakan dua SQL script yang telah disiapkan sebelumnya. Cara ini lebih cepat karena anda tinggal menyalin script dimaksud dan mengeksekusinya di layar SQL editor.

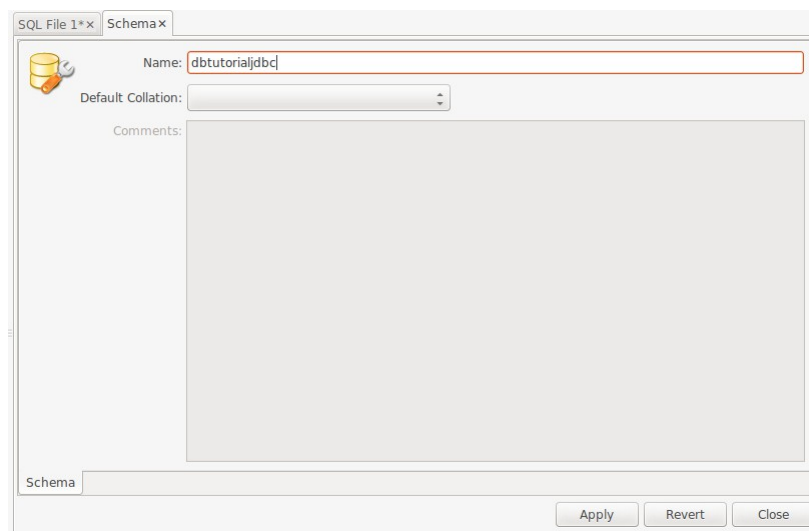
Cara pertama, menggunakan workbench GUI dapat dilakukan dengan langkah-langkah sebagai berikut.



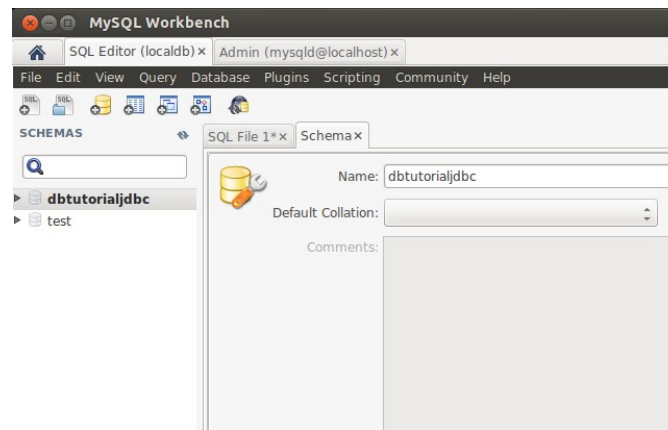
Jalankan SQL workbench kemudian klik dua kali pada koneksi yang telah kita buat pada bab sebelumnya (bab 5.1.2)



Klik kanan pada database test pada daftar SCHEMAS, kemudian pilih create schemas

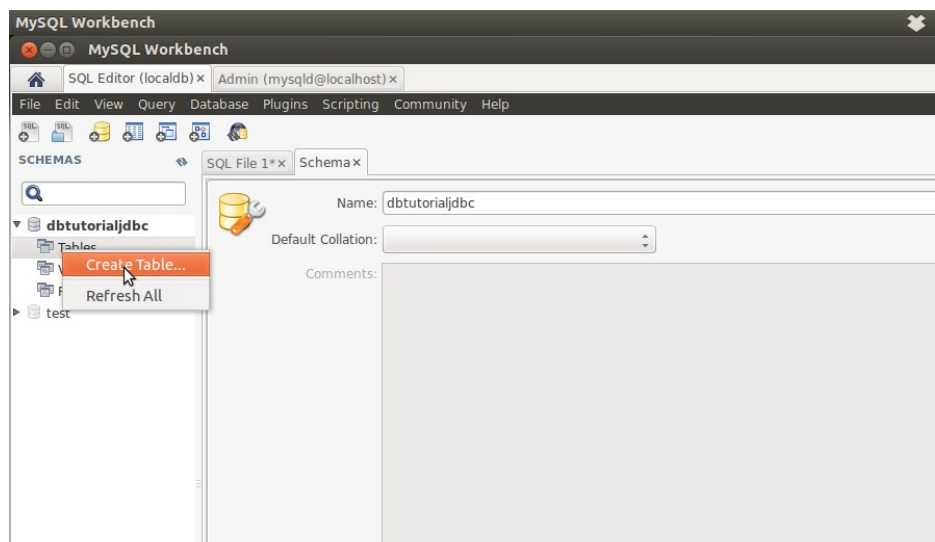


Beri nama 'dbtutorialjdb' pada nama skema kemudian klik apply dua kali (untuk konfirmasi)



Perhatikan bahwa pada tahap ini kita sudah memiliki schema baru yakni 'dbtutorialjdbc'

Selanjutnya kita akan membuat tabel untuk menampung data Mahasiswa, MataKuliah dan NilaiMataKuliah sesuai dengan desain yang telah kita buat pada bab 5.1.3.



Telusuri item pada skema 'dbtutorialjdbc', kemudian klik kanan pada 'table' dan pilih 'Create Table'

SQL File 1*x Table: mahasiswa x

Name: Schema: dbtutorialjdbc

Collation: latin1 - default collation Engine: InnoDB

Comment:

Column Name	Datatype	PK	NN	UQ	BIN	UN	ZF	AI	Default
id	INT(11)	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	
nim	VARCHAR(12)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL
nama	VARCHAR(45)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL
jurusan	VARCHAR(45)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL
thnmasuk	INT(11)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL

Column Details

Collation: *Table Default*

Comment:

Columns Indexes Foreign Keys Triggers Partitioning Options

Apply Revert Close

Isi form untuk membuat tabel mahasiswa dengan detail seperti pada gambar, kemudian klik 'Apply' dua kali

Ulangi langkah-langkah pembuatan tabel Mahasiswa untuk membuat tabel Matakuliah dan Nilaimatakuliah. Gunakan dua screenshot berikut sebagai panduan.

SQL File 1*x Table: mahasiswa x Table: matakuliah x

Name: Schema: dbtutorialjdbc

Collation: latin1 - default collation Engine: InnoDB

Comment:

Column Name	Datatype	PK	NN	UQ	BIN	UN	ZF	AI	Default
id	INT(11)	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	
kode	VARCHAR(11)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL
nama	VARCHAR(45)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL
sks	INT(11)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL

Column Details

Collation: *Table Default*

Comment:

Columns Indexes Foreign Keys Triggers Partitioning Options

Apply Revert Close

Isi form untuk membuat tabel matakuliah dengan detail seperti pada gambar, kemudian klik 'Apply' dua kali

SQL File 1*x Table: nilaimatakuliah x

Name: nilaimatakuliah Schema: dbtutorialjdbc

Collation: latin1 - default collation Engine: InnoDB

Comment:

Column Name	Datatype	PK	NN	UQ	BIN	UN	ZF	AI	Default
id	INT(11)	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	
idmahasiswa	INT(11)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL
idmatakuliah	INT(11)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL
thnambil	INT(11)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL
semester	INT(11)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL
nilai	VARCHAR(1)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL

Column Details

Collation: *Table Default*

Comment:

Columns Indexes Foreign Keys Triggers Partitioning Options

Apply Revert Close

Isi form untuk membuat tabel nilaimatakuliah dengan detail seperti pada gambar, kemudian klik 'Apply' dua kali

MySQL Workbench

SQL Editor (localdb) x

File Edit View Query Database Plugins Scripting Community Help

SCHEMAS

dbtutorialjdbc

- Tables
 - mahasiswa
 - matakuliah
 - nilaimatakuliah**
- Views
- Routines
- test

SQL File 1*x Table: nilaimatakuliah x

Name: nilaimatakuliah

Collation: latin1 - default collation

Comment:

Column Name	Datatype
id	INT(11)
idmahasiswa	INT(11)
idmatakuliah	INT(11)
thnambil	INT(11)
semester	INT(11)
nilai	VARCHAR(1)

Column Details

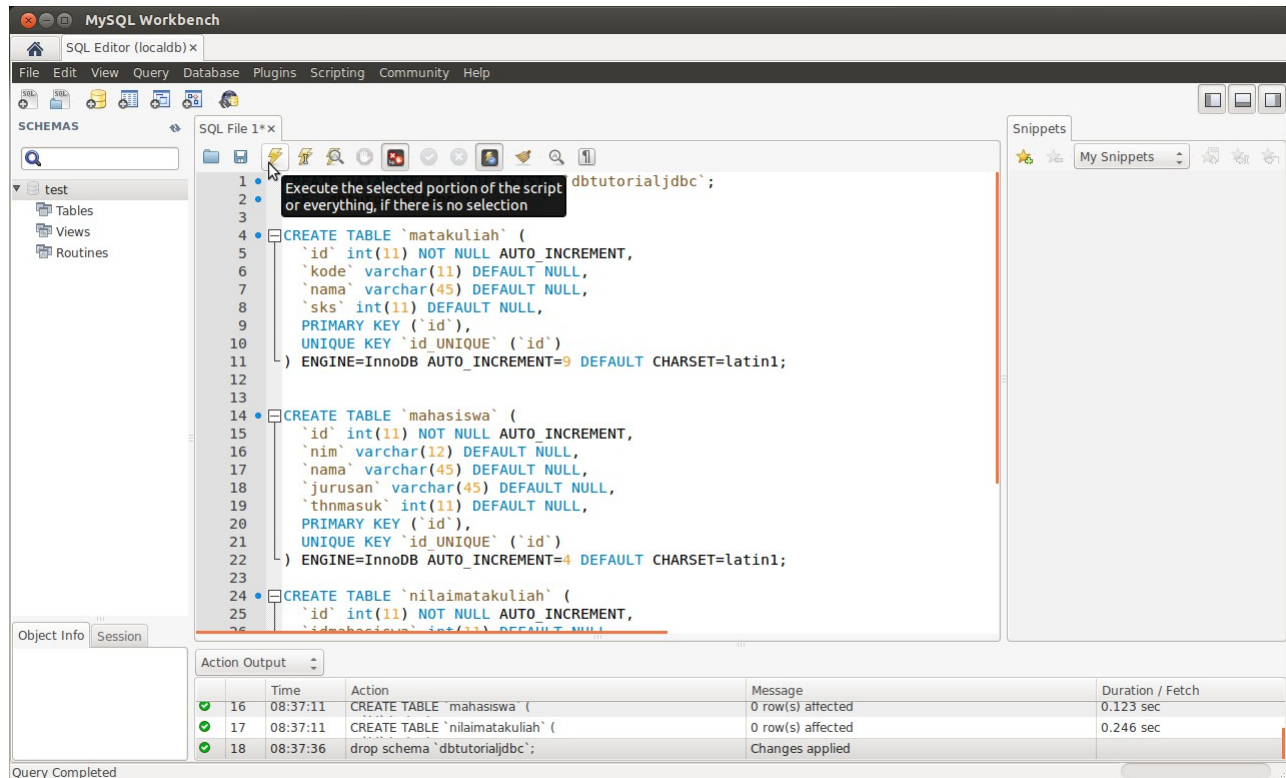
Collation: *Table Default*

Comment:

Columns Indexes Foreign Keys Triggers Partitioning Options

Perhatikan bahwa jika tahap sebelumnya sukses, kita akan memiliki tiga tabel pada skema dbtutorialjdbc

Cara kedua, menggunakan script dapat dilakukan dengan langkah-langkah sebagai berikut.



Salin script berikut pada SQL Editor terpisah kemudian eksekusi dengan menekan icon petir atau menu Query-Execute.

```

CREATE DATABASE IF NOT EXISTS `dbtutorialjdbc`;
USE `dbtutorialjdbc`;

CREATE TABLE `matakuliah` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `kode` varchar(11) DEFAULT NULL,
  `nama` varchar(45) DEFAULT NULL,
  `sks` int(11) DEFAULT NULL,
  PRIMARY KEY (`id`),
  UNIQUE KEY `id_UNIQUE` (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=9 DEFAULT CHARSET=latin1;

CREATE TABLE `mahasiswa` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `nim` varchar(12) DEFAULT NULL,
  `nama` varchar(45) DEFAULT NULL,
  `jurusan` varchar(45) DEFAULT NULL,
  `thnmasuk` int(11) DEFAULT NULL,
  PRIMARY KEY (`id`),
  UNIQUE KEY `id_UNIQUE` (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=4 DEFAULT CHARSET=latin1;

CREATE TABLE `nilaimatakuliah` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `idmahasiswa` int(11) DEFAULT NULL,
  `idmatakuliah` int(11) DEFAULT NULL,
  `nilai` float(10,2) DEFAULT NULL,
  PRIMARY KEY (`id`),
  UNIQUE KEY `id_UNIQUE` (`id`),
  FOREIGN KEY (`idmahasiswa`) REFERENCES `mahasiswa` (`id`),
  FOREIGN KEY (`idmatakuliah`) REFERENCES `matakuliah` (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=1 DEFAULT CHARSET=latin1;

```

```

`thnambil` int(11) DEFAULT NULL,
`semester` int(11) DEFAULT NULL,
`nilai` varchar(1) DEFAULT NULL,
PRIMARY KEY (`id`),
UNIQUE KEY `id_UNIQUE` (`id`),
KEY `fk_nilaimatakuliah_1` (`idmahasiswa`),
KEY `fk_nilaimatakuliah_2` (`idmatakuliah`),
CONSTRAINT `fk_nilaimatakuliah_1` FOREIGN KEY (`idmahasiswa`) REFERENCES
`mahasiswa` (`id`) ON DELETE NO ACTION ON UPDATE NO ACTION,
CONSTRAINT `fk_nilaimatakuliah_2` FOREIGN KEY (`idmatakuliah`) REFERENCES
`matakuliah` (`id`) ON DELETE NO ACTION ON UPDATE NO ACTION
) ENGINE=InnoDB AUTO_INCREMENT=13 DEFAULT CHARSET=latin1;

```

Bila script ini selesai dijalankan, hasilnya akan sama dengan cara sebelumnya yakni kita akan memiliki schema 'dbtutorialjdbc' dengan tiga tabel yakni mahasiswa, matakuliah dan nilaimatakuliah.

Inisiasi Data Dummy

Langkah selanjutnya adalah mengisi tabel yang telah kita buat dengan records data. Seperti langkah sebelumnya, ada dua pilihan cara untuk mengisi tabel pada tutorial ini.







Cara pertama adalah cara manual. Cara ini membutuhkan waktu yang agak lama namun dapat memberikan pemahaman lebih mengenai konsep tabel dan data di dalamnya (records). Anda bisa mengisinya secara manual dengan mengklik kanan pada tabel yang akan diisi kemudian pilih 'edit table data'. Cara pengisiannya seperti saat mengisi data pada aplikasi spreadsheet (MS Excel, calc, dll).

Gunakan screenshot pada tabel-tabel berikut sebagai panduan untuk mengisi data ketiga tabel yang ada pada skema 'dbtutorialjdbc', jangan lupa menekan tombol apply setelah anda selesai melakukan pengisian.

SQL File 2 * x

mahasiswa x

Filter:


 Edit:    Export:  Autosize: 


#	id	nim	nama	jurusan	thnmasuk
1	1	01210991	Dende Raisah	Teknik Informatika	2012
2	2	01210992	Menggep el Fanshury	Teknik Informatika	2012
3	3	01210993	Mahdan Rengget	Teknik Informatika	2012
*	NULL	NULL	NULL	NULL	NULL


SQL File 2*x


matakuliah x


Filter:




Edit: 











Export: 

Autosize: 

#	id	kode	nama	sks
1	1	AP	Algoritma Pemrograman	2
2	2	PBO	Pemrograman Berorientasi Objek	3
3	3	PCS	Pemrograman Client Server	3
4	4	PTI	Pengantar Teknologi Informasi	2
*	NULL	NULL	NULL	NULL

SQL File 2*x nilaimatakuliah x						
Filter:  Edit:    Export:  Autosize: 						
#	id	idmahasiswa	idmatakuliah	thnambil	semester	nilai
1	1	1	1	2012	1	B
2	2	1	2	2012	1	B
3	3	1	3	2010	1	A
4	4	1	4	2010	2	A
5	5	2	1	2012	1	A
6	6	2	2	2012	1	A
7	7	2	3	2011	1	A
8	8	2	4	2011	2	B
9	9	3	1	2012	1	C
10	10	3	2	2012	1	B
11	11	3	3	2011	1	A
12	12	3	4	2011	2	B
*	NULL	NULL	NULL	NULL	NULL	NULL

Cara kedua adalah menggunakan script. Cara ini jauh lebih cepat dan cocok bagi anda yang telah memiliki pengetahuan dasar database.

Salin dan eksekusi script berikut pada SQL editor.

```
USE `dbtutorialjdbc`;
/*
 * inisiasi data tabel mahasiswa
 */
INSERT INTO `mahasiswa` VALUES (1,'01210991','Dende Raisah','Teknik Informatika',2012);
INSERT INTO `mahasiswa` VALUES (2,'01210992','Menggep el Fanshury','Teknik Informatika',2012);
INSERT INTO `mahasiswa` VALUES (3,'01210993','Mahdan Rengget','Teknik Informatika',2012);

/*
 * inisiasi data tabel matakuliah
 */
INSERT INTO `matakuliah` VALUES (1,'AP','Algoritma Pemrograman',2);
INSERT INTO `matakuliah` VALUES (2,'PBO','Pemrograman Berorientasi Objek',3);
INSERT INTO `matakuliah` VALUES (3,'PCS','Pemrograman Client Server',3);
INSERT INTO `matakuliah` VALUES (4,'PTI','Pengantar Teknologi Informasi',2);

/*
 * inisiasi data tabel nilaimatakuliah
 */
INSERT INTO `nilaimatakuliah` VALUES (1,1,1,2012,1,'B');
INSERT INTO `nilaimatakuliah` VALUES (2,1,2,2012,1,'B');
INSERT INTO `nilaimatakuliah` VALUES (3,1,3,2010,1,'A');
INSERT INTO `nilaimatakuliah` VALUES (4,1,4,2010,2,'A');
INSERT INTO `nilaimatakuliah` VALUES (5,2,1,2012,1,'A');
INSERT INTO `nilaimatakuliah` VALUES (6,2,2,2012,1,'A');
INSERT INTO `nilaimatakuliah` VALUES (7,2,3,2011,1,'A');
INSERT INTO `nilaimatakuliah` VALUES (8,2,4,2011,2,'B');
```

```
INSERT INTO `nilaimatakuliah` VALUES (9,3,1,2012,1,'C');
INSERT INTO `nilaimatakuliah` VALUES (10,3,2,2012,1,'B');
INSERT INTO `nilaimatakuliah` VALUES (11,3,3,2011,1,'A');
INSERT INTO `nilaimatakuliah` VALUES (12,3,4,2011,2,'B');
```

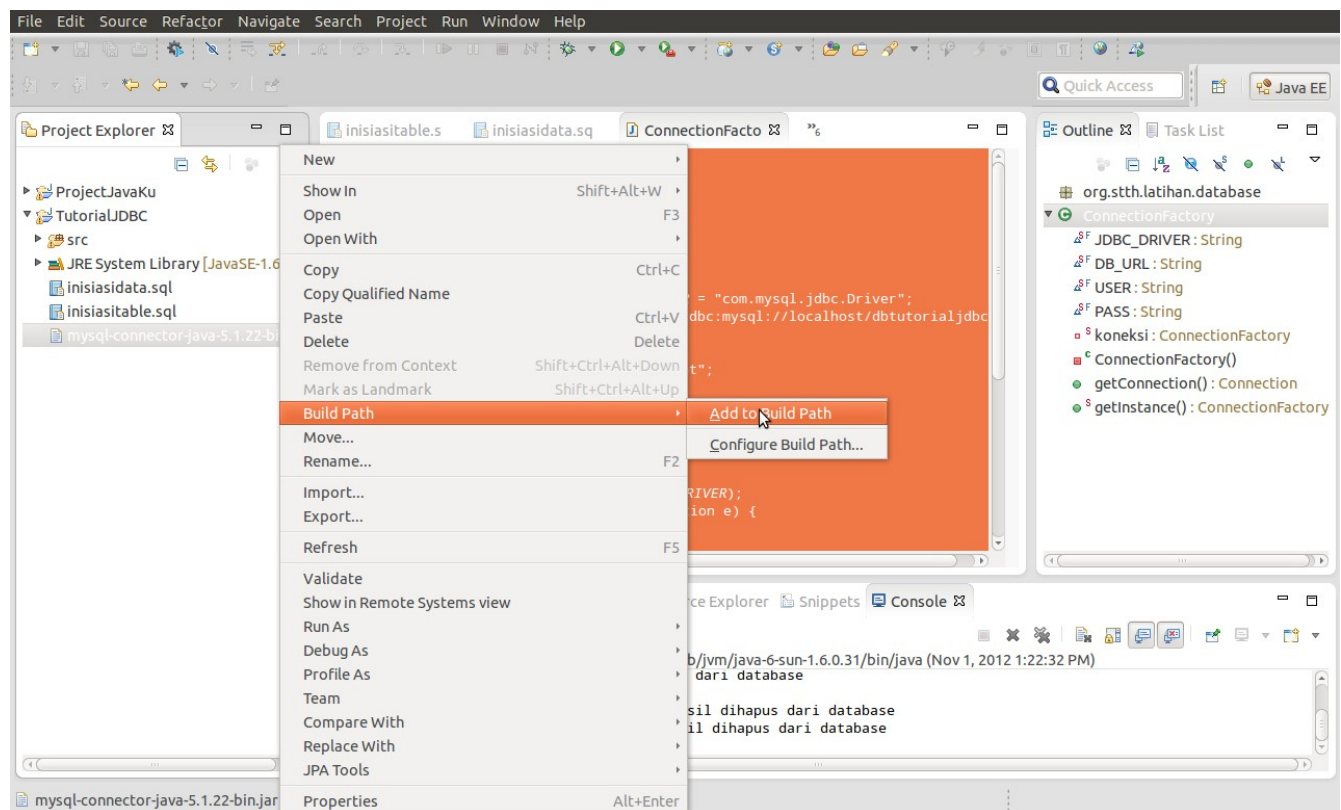
Jika langkah di atas telah selesai anda lakukan, maka kita siap menuju tahap selanjutnya yakni programming.

5.2 Pemrograman

Buat sebuah project menggunakan Eclipse, beri nama project tersebut 'TutorialJDBC' kemudian ikuti langkah-langkah berikut.

5.2.1 Menambahkan library JDBC driver untuk MySQL pada project

Siapkan JDBC Driver untuk MySQL, download file jar resmi dari situs mySQL pada alamat <http://dev.mysql.com/downloads/connector/j/>



Cara menambahkan library JDBC ke dalam build path

Copy file tersebut pada explorer dan tempatkan/paste pada project anda menggunakan eclipse. Selanjutnya tambahkan pada build path dengan cara klik kanan pada file tersebut pada eclipse, pilih Build Path – Add to Build Path

5.2.2 Class Koneksi

Untuk menginisiasi koneksi ke database server yang telah kita siapkan, buatlah class ConnectionFactory dengan kode seperti dibawah ini. Sesuaikan parameter database (DB_URL, USER, PASS) dengan konfigurasi database anda.

```

package org.stth.latihan.database;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;

public class ConnectionFactory {

    static final String JDBC_DRIVER = "com.mysql.jdbc.Driver";
    static final String DB_URL = "jdbc:mysql://localhost/dbtutorialjdbc";

    // Database credentials
    static final String USER = "root";
    static final String PASS = "";

    private static ConnectionFactory koneksi = null;

    private ConnectionFactory() {
        try {
            Class.forName(JDBC_DRIVER);
        } catch (ClassNotFoundException e) {
            e.printStackTrace();
        }
    }

    public Connection getConnection() throws SQLException {
        Connection conn = DriverManager.getConnection(DB_URL, USER, PASS);
        return conn;
    }

    public static ConnectionFactory getInstance() {
        if (koneksi == null) {
            koneksi = new ConnectionFactory();
        }
        return koneksi;
    }
}

```

5.2.3 Class Entitas

Class entitas adalah class yang berasosiasi dengan data (record) yang kita simpan ke dalam database. Pada kasus dalam tutorial ini kita memiliki tiga class entitas yakni class Mahasiswa, MataKuliah dan NilaiMataKuliah. Berikut adalah kode untuk ketiga class tersebut.

Class Mahasiswa

```

package org.stth.latihan.database;

import java.util.ArrayList;

public class Mahasiswa {

    private long id;
    private String nama;
    private String nim;
    private String jurusan;
}

```

```

    private int thnMasuk;
    private ArrayList<NilaiMataKuliah> daftarNilaiKuliah= new
ArrayList<NilaiMataKuliah>();

    public Mahasiswa(String nama, String nim, String jurusan, int tahunmasuk) {
        super();
        this.nama = nama;
        this.nim = nim;
        this.jurusan = jurusan;
        this.thnMasuk = tahunmasuk;
    }
    public Mahasiswa() {

    }

    public long getId() {
        return id;
    }
    public void setId(long id) {
        this.id = id;
    }
    public String getNama() {
        return nama;
    }
    public void setNama(String nama) {
        this.nama = nama;
    }
    public String getNim() {
        return nim;
    }
    public void setNim(String nim) {
        this.nim = nim;
    }
    public String getJurusan() {
        return jurusan;
    }
    public void setJurusan(String jurusan) {
        this.jurusan = jurusan;
    }
    public int getTahunmasuk() {
        return thnMasuk;
    }
    public void setTahunmasuk(int tahunmasuk) {
        this.thnMasuk = tahunmasuk;
    }
    public ArrayList<NilaiMataKuliah> getNilaiKuliah() {
        return daftarNilaiKuliah;
    }
    public void setNilaiKuliah(ArrayList<NilaiMataKuliah> daftarNilaiKuliah) {
        this.daftarNilaiKuliah = daftarNilaiKuliah;
    }
    public Double getIPK(){
        double ipk=0;
        double kumulatif = 0;
        int jumlahSKS=0;
        for (NilaiMataKuliah n : daftarNilaiKuliah) {
            kumulatif = kumulatif + n.getBobot();

```

```

        jumlahSKS = jumlahSKS + n.getMataKuliah().getSks();
    }
    if (kumulatif>0){
        ipk = kumulatif/jumlahSKS;
    }
    return ipk;
}
}

```

Class MataKuliah

```

package org.stth.latihan.database;

public class MataKuliah {
    private long id;
    private String kode;
    private String nama;
    private int sks;

    public MataKuliah(){

    }
    public MataKuliah(String kode, String nama, int sks) {
        super();
        this.kode = kode;
        this.nama = nama;
        this.sks = sks;
    }
    public long getId() {
        return id;
    }
    public void setId(long id) {
        this.id = id;
    }

    public String getKode() {
        return kode;
    }
    public void setKode(String kode) {
        this.kode = kode;
    }
    public String getNama() {
        return nama;
    }
    public void setNama(String nama) {
        this.nama = nama;
    }
    public int getSks() {
        return sks;
    }
    public void setSks(int sks) {
        this.sks = sks;
    }
}

```


Class NilaiMataKuliah

```
package org.stth.latihan.database;

public class NilaiMataKuliah {
    public final static int SEMESTER_GANJIL=1;
    public final static int SEMESTER_GENAP=2;
    public final static int SEMESTER_PENDEK=3;
    private int id;
    private Mahasiswa mahasiswa;
    private MataKuliah mataKuliah;
    private int thnAmbil;
    private int semester;
    private char nilai;

    public NilaiMataKuliah(){

    }
    public NilaiMataKuliah(Mahasiswa mahasiswa, MataKuliah mataKuliah,
        int thnAmbil, char nilai) {
        super();
        this.mahasiswa = mahasiswa;
        this.mataKuliah = mataKuliah;
        this.thnAmbil = thnAmbil;
        this.nilai = nilai;
    }

    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }

    public Mahasiswa getMahasiswa() {
        return mahasiswa;
    }

    public void setMahasiswa(Mahasiswa mahasiswa) {
        this.mahasiswa = mahasiswa;
    }

    public MataKuliah getMataKuliah() {
        return mataKuliah;
    }

    public void setMataKuliah(MataKuliah mataKuliah) {
        this.mataKuliah = mataKuliah;
    }

    public int getThnAmbil() {
        return thnAmbil;
    }
}
```

```

    public void setThnAmbil(int thnAmbil) {
        this.thnAmbil = thnAmbil;
    }

    public int getSemester() {
        return semester;
    }

    public void setSemester(int semester) {
        this.semester = semester;
    }

    public char getNilai() {
        return nilai;
    }

    public void setNilai(char nilai) {
        if (nilai=='A' || nilai=='B' || nilai=='C' || nilai=='D'){
            this.nilai = nilai;
        } else {
            this.nilai = 'E';
        }
    }

    public double getBobot(){
        switch (this.nilai) {
            case 'A':
                return 4.0;
            case 'B':
                return 3.0;
            case 'C':
                return 2.0;
            case 'D':
                return 1.0;
            default:
                return 0;
        }
    }

    public boolean isLulus(){
        if (nilai=='A' || nilai=='B' || nilai=='C'){
            return true;
        }
        return false;
    }
}

```

5.2.4 Class Data Access Object (DAO)

Setelah kita membuat class untuk keperluan koneksi ke database (ConnectionFactory) dan class untuk entitas (Mahasiswa, MataKuliah dan NilaiMataKuliah), selanjutnya kita akan membuat class yang akan menjadi jembatan antara data (record) pada database dengan object pada kode program kita. Class ini disebut dengan Data Access Object (DAO). Class DAO yang kita buat bertanggungjawab menyediakan fasilitas untuk menyimpan, membaca, mengubah dan menghapus data yang ada pada tabel-tabel yang kita miliki.

Setiap entitas yang kita miliki akan memiliki DAO masing-masing. Ini berarti kita akan memiliki 3 class DAO masing-masing untuk entitas Mahasiswa, MataKuliah dan NilaiMataKuliah. Class-class

DAO tersebut nantinya akan kita beri nama MahasiswaDAO, MataKuliahDAO dan NilaiMataKuliahDAO.

Setiap class akan bertanggung jawab untuk menyediakan fasilitas :

1. Insert, menyimpan object baru menjadi record baru pada database
2. Update, menyimpan perubahan nilai object pada database
3. Load, membaca record pada database kemudian mengubahnya menjadi object, fungsi load dapat dibuat beberapa versi, misalnya : load berdasarkan nilai field tertentu, load keseluruhan, dsb.
4. Delete, menghapus record pada database

Untuk lebih memahami fasilitas tersebut, akan disajikan cuplikan kode dari class MataKuliahDAO. Kita akan membahas setiap kode dan perannya pada class DAO yang kita buat.

```
Connection connection = null;
PreparedStatement ptmt = null;
ResultSet resultSet = null;
```

Ketiga baris ini adalah baris deklarasi variabel utama yang akan digunakan pada class DAO yang kita miliki. Variabel connection adalah variabel yang berisi koneksi ke database. Variabel ptmt merupakan variabel untuk menampung prepared statement, perintah berupa SQL query yang dieksekusi pada database. Variabel terakhir yakni resultSet merupakan variabel untuk menampung hasil eksekusi SQL query pada database.

```
private Connection getConnection() throws SQLException {
    Connection conn;
    conn = ConnectionFactory.getInstance().getConnection();
    return conn;
}
```

Method ini bertugas menyiapkan koneksi dengan memanfaatkan class ConnectionFactory. Pada class DAO kita, method ini akan digunakan untuk mengisi variabel connection.

```
public void insert(MataKuliah matkul) {
    try {
        String queryString = "INSERT INTO matakuliah(kode, nama, sks)
VALUES(?,?,?)";
        connection = getConnection();
        ptmt = connection.prepareStatement(queryString);
        ptmt.setString(1, matkul.getKode());
        ptmt.setString(2, matkul.getNama());
        ptmt.setInt(3, matkul.getSks());
        ptmt.executeUpdate();
    } catch (SQLException e) {
        e.printStackTrace();
    } finally {
        try {
            if (ptmt != null)
                ptmt.close();
            if (connection != null)
                connection.close();
        } catch (SQLException e) {
            e.printStackTrace();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

```

    }

}

```

Method ini bertugas menyimpan objek MataKuliah menjadi record baru pada database. Perhatikan bahwa SQL Query yang dieksekusi melalui prepared statement yang ada adalah SQL Query bertipe Insert. Tanda '?' pada sintaks utama Insert diisi dengan nilai konversi dari object MataKuliah yang akan disimpan yakni kode, nama dan nilai sks. Nilai id tidak diisi karena tipe id pada tabel matakuliah bertipe *auto increment* yang berarti ia terisi otomatis dengan nilai yang terus meningkat (+1). Method ini dibungkus dengan blok try-catch-final untuk menangkap bila ada terjadi error dengan tipe SQLException, dengan blok final berisi perintah untuk mencek dan menutup koneksi

```

public void update(MataKuliah matkul) {
    try {
        String queryString = "update matakuliah set kode=?, nama=?, sks=?
where id=?";
        connection = getConnection();
        ptmt = connection.prepareStatement(queryString);
        ptmt.setString(1, matkul.getKode());
        ptmt.setString(2, matkul.getNama());
        ptmt.setInt(3, matkul.getSks());
        ptmt.setLong(4, matkul.getId());
        ptmt.executeUpdate();
    } catch (SQLException e) {
        e.printStackTrace();
    } finally {
        try {
            if (ptmt != null)
                ptmt.close();
            if (connection != null)
                connection.close();
        } catch (SQLException e) {
            e.printStackTrace();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
}

```

Method ini bertugas mengupdate record matakuliah tertentu pada database. Perintah ini dilakukan pada program bila variabel object MataKuliah yang sebelumnya diloat dari database mengalami perubahan nilai sehingga padanannya pada tabel matakuliah juga perlu diubah nilainya. Perhatikan bahwa SQL Query yang dieksekusi melalui prepared statement yang ada adalah SQL Query bertipe Update. Pada query ini id dari object matakuliah disertakan sebagai parameter karena ia merupakan primary key record yang akan diupdate.

```

public void delete(MataKuliah matkul) {
    try {
        String queryString = "delete from matakuliah where id=?";
        connection = getConnection();
        ptmt = connection.prepareStatement(queryString);
        ptmt.setLong(1, matkul.getId());
    }
}

```

```

        ptmt.executeUpdate();
    } catch (SQLException e) {
        e.printStackTrace();
    } finally {
        try {
            if (ptmt != null)
                ptmt.close();
            if (connection != null)
                connection.close();
        } catch (SQLException e) {
            e.printStackTrace();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
}

```

Method ini bertugas menghapus record matakuliah tertentu pada database. Perhatikan bahwa SQL Query yang dieksekusi melalui prepared statement yang ada adalah SQL Query bertipe Delete. Pada query ini id dari object matakuliah disertakan sebagai parameter karena ia merupakan primary key record yang akan dihapus.

```

public MataKuliah getById(int id) {
    try {
        String queryString = "select kode,nama,sks,id from matakuliah where
id=?";
        connection = getConnection();
        ptmt = connection.prepareStatement(queryString);
        ptmt.setInt(1, id);
        ResultSet rs = ptmt.executeQuery();
        while (rs.next()) {
            MataKuliah mk = new MataKuliah();
            mk.setKode(rs.getString(1));
            mk.setNama(rs.getString(2));
            mk.setSks(rs.getInt(3));
            mk.setId(rs.getLong(4));
            return mk;
        }
    } catch (SQLException e) {
        e.printStackTrace();
    } finally {
        try {
            if (ptmt != null)
                ptmt.close();
            if (connection != null)
                connection.close();
        } catch (SQLException e) {
            e.printStackTrace();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
    return null;
}

```

```
}
```

Method ini bertugas meload record matakuliah tertentu dari database berdasarkan id matakuliah. Tipe keluaran dari method ini adalah sebuah object MataKuliah. Perhatikan bahwa SQL Query yang dieksekusi melalui prepared statement yang ada adalah SQL Query bertipe Select. Hasil dari eksekusi berupa ResultSet kemudian ditransformasi menjadi object MataKuliah yang kemudian menjadi keluaran method ini.

```
public MataKuliah getByKode(String kode) {  
    try {  
        String queryString = "select kode,nama,sks,id from matakuliah where  
kode=?";  
        connection = getConnection();  
        ptmt = connection.prepareStatement(queryString);  
        ptmt.setString(1, kode);  
        ResultSet rs = ptmt.executeQuery();  
        while (rs.next()) {  
            MataKuliah mk = new MataKuliah();  
            mk.setKode(rs.getString(1));  
            mk.setNama(rs.getString(2));  
            mk.setSks(rs.getInt(3));  
            mk.setId(rs.getLong(4));  
            return mk;  
        }  
    } catch (SQLException e) {  
        e.printStackTrace();  
    } finally {  
        try {  
            if (ptmt != null)  
                ptmt.close();  
            if (connection != null)  
                connection.close();  
        } catch (SQLException e) {  
            e.printStackTrace();  
        } catch (Exception e) {  
            e.printStackTrace();  
        }  
    }  
    return null;  
}
```

Method ini bertugas meload record matakuliah tertentu dari database berdasarkan kode matakuliah. Tipe keluaran dari method ini adalah sebuah object MataKuliah. Perhatikan bahwa SQL Query yang dieksekusi melalui prepared statement yang ada adalah SQL Query bertipe Select. Hasil dari eksekusi berupa ResultSet kemudian ditransformasi menjadi object MataKuliah yang kemudian menjadi keluaran method ini.

```
public ArrayList<MataKuliah> getAll() {  
    ArrayList<MataKuliah> listMatkul = new ArrayList<MataKuliah>();  
    try {  
        String queryString = "select kode,nama,sks,id from matakuliah order
```

```

by kode";

        connection = getConnection();
        ptmt = connection.prepareStatement(queryString);
        ResultSet rs = ptmt.executeQuery();
        while (rs.next()) {
            MataKuliah mk = new MataKuliah();
            mk.setKode(rs.getString(1));
            mk.setNama(rs.getString(2));
            mk.setSks(rs.getInt(3));
            mk.setId(rs.getLong(4));
            listMatkul.add(mk);
        }
    } catch (SQLException e) {
        e.printStackTrace();
    } finally {
        try {
            if (ptmt != null)
                ptmt.close();
            if (connection != null)
                connection.close();
        } catch (SQLException e) {
            e.printStackTrace();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
    return listMatkul;
}

```

Method ini bertugas meload seluruh record matakuliah dari database dan mengeluarkannya dalam bentuk ArrayList.

SQL Query yang dieksekusi melalui prepared statement yang ada adalah SQL Query bertipe Select. Hasil dari eksekusi berupa ResultSet kemudian ditransformasi menjadi daftar object MataKuliah dalam ArrayList yang kemudian menjadi keluaran method ini.

Bentuk akhir dari class MataKuliahDAO dapat dilihat pada kode berikut ini. Buatlah class yang sama pada project anda.

```

package org.stth.latihan.database;

import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.ArrayList;

public class MataKuliahDAO {
    Connection connection = null;
    PreparedStatement ptmt = null;
    ResultSet resultSet = null;

    private Connection getConnection() throws SQLException {
        Connection conn;
        conn = ConnectionFactory.getInstance().getConnection();
        return conn;
    }
}

```

```

    }

    public void insert(MataKuliah matkul) {
        try {
            String queryString = "INSERT INTO matakuliah(kode, nama, sks)
VALUES(?,?,?)";
            connection = getConnection();
            ptmt = connection.prepareStatement(queryString);
            ptmt.setString(1, matkul.getKode());
            ptmt.setString(2, matkul.getNama());
            ptmt.setInt(3, matkul.getSks());
            ptmt.executeUpdate();
        } catch (SQLException e) {
            e.printStackTrace();
        } finally {
            try {
                if (ptmt != null)
                    ptmt.close();
                if (connection != null)
                    connection.close();
            } catch (SQLException e) {
                e.printStackTrace();
            } catch (Exception e) {
                e.printStackTrace();
            }
        }
    }
}

public void update(MataKuliah matkul) {
    try {
        String queryString = "update matakuliah set kode=?, nama=?, sks=?
where id=?";
        connection = getConnection();
        ptmt = connection.prepareStatement(queryString);
        ptmt.setString(1, matkul.getKode());
        ptmt.setString(2, matkul.getNama());
        ptmt.setInt(3, matkul.getSks());
        ptmt.setLong(4, matkul.getId());
        ptmt.executeUpdate();
    } catch (SQLException e) {
        e.printStackTrace();
    } finally {
        try {
            if (ptmt != null)
                ptmt.close();
            if (connection != null)
                connection.close();
        } catch (SQLException e) {
            e.printStackTrace();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
}

```



```

public void delete(MataKuliah matkul) {
    try {
        String queryString = "delete from matakuliah where id=?";
        connection = getConnection();
        ptmt = connection.prepareStatement(queryString);
        ptmt.setLong(1, matkul.getId());
        ptmt.executeUpdate();
    } catch (SQLException e) {
        e.printStackTrace();
    } finally {
        try {
            if (ptmt != null)
                ptmt.close();
            if (connection != null)
                connection.close();
        } catch (SQLException e) {
            e.printStackTrace();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

public MataKuliah getById(int id) {
    try {
        String queryString = "select kode,nama,sks,id from matakuliah where
id=?";
        connection = getConnection();
        ptmt = connection.prepareStatement(queryString);
        ptmt.setInt(1, id);
        ResultSet rs = ptmt.executeQuery();
        while (rs.next()) {
            MataKuliah mk = new MataKuliah();
            mk.setKode(rs.getString(1));
            mk.setNama(rs.getString(2));
            mk.setSks(rs.getInt(3));
            mk.setId(rs.getLong(4));
            return mk;
        }
    } catch (SQLException e) {
        e.printStackTrace();
    } finally {
        try {
            if (ptmt != null)
                ptmt.close();
            if (connection != null)
                connection.close();
        } catch (SQLException e) {
            e.printStackTrace();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
    return null;
}

```

```

    }

    public MataKuliah getByKode(String kode) {

        try {
            String queryString = "select kode,nama,sks,id from matakuliah where
kode=?";

            connection = getConnection();
            ptmt = connection.prepareStatement(queryString);
            ptmt.setString(1, kode);
            ResultSet rs = ptmt.executeQuery();
            while (rs.next()) {
                MataKuliah mk = new MataKuliah();
                mk.setKode(rs.getString(1));
                mk.setNama(rs.getString(2));
                mk.setSks(rs.getInt(3));
                mk.setId(rs.getLong(4));
                return mk;
            }

        } catch (SQLException e) {
            e.printStackTrace();
        } finally {
            try {
                if (ptmt != null)
                    ptmt.close();
                if (connection != null)
                    connection.close();
            } catch (SQLException e) {
                e.printStackTrace();
            } catch (Exception e) {
                e.printStackTrace();
            }
        }
        return null;
    }

    public ArrayList<MataKuliah> getAll() {
        ArrayList<MataKuliah> listMatkul = new ArrayList<MataKuliah>();
        try {
            String queryString = "select kode,nama,sks,id from matakuliah order
by kode";

            connection = getConnection();
            ptmt = connection.prepareStatement(queryString);
            ResultSet rs = ptmt.executeQuery();
            while (rs.next()) {
                MataKuliah mk = new MataKuliah();
                mk.setKode(rs.getString(1));
                mk.setNama(rs.getString(2));
                mk.setSks(rs.getInt(3));
                mk.setId(rs.getLong(4));
                listMatkul.add(mk);
            }
        } catch (SQLException e) {
            e.printStackTrace();
        } finally {
            try {

```

```

        if (ptmt != null)
            ptmt.close();
        if (connection != null)
            connection.close();
    } catch (SQLException e) {
        e.printStackTrace();
    } catch (Exception e) {
        e.printStackTrace();
    }
}
return listMatkul;
}
}

```

Dengan cara yang sama class DAO untuk entitas Mahasiswa dapat kita susun sebagai berikut. Salin kode bersangkutan pada project anda.

```

package org.stth.latihan.database;

import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.ArrayList;

public class MahasiswaDAO {
    private Connection connection = null;
    private PreparedStatement ptmt = null;

    private Connection getConnection() throws SQLException {
        Connection conn;
        conn = ConnectionFactory.getInstance().getConnection();
        return conn;
    }

    public void insert(Mahasiswa mahasiswa) {
        try {
            String queryString = "INSERT INTO mahasiswa(nim, nama, jurusan,
thnmasuk) VALUES(?,?,?,?)";
            connection = getConnection();
            ptmt = connection.prepareStatement(queryString);
            ptmt.setString(1, mahasiswa.getNim());
            ptmt.setString(2, mahasiswa.getNama());
            ptmt.setString(3, mahasiswa.getJurusan());
            ptmt.setInt(4, mahasiswa.getTahunmasuk());
            ptmt.executeUpdate();
        } catch (SQLException e) {
            e.printStackTrace();
        } finally {
            try {
                if (ptmt != null)
                    ptmt.close();
                if (connection != null)
                    connection.close();
            }
        }
    }
}

```

```

        } catch (SQLException e) {
            e.printStackTrace();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

}

}

public void update(Mahasiswa mahasiswa) {
    try {
        String queryString = "update mahasiswa set nim=?, nama=?,
jurusan=?, thnmasuk=? where id=?";
        connection = getConnection();
        ptmt = connection.prepareStatement(queryString);
        ptmt.setString(1, mahasiswa.getNim());
        ptmt.setString(2, mahasiswa.getNama());
        ptmt.setString(3, mahasiswa.getJurusan());
        ptmt.setInt(4, mahasiswa.getTahunmasuk());
        ptmt.setLong(5, mahasiswa.getId());
        ptmt.executeUpdate();
    } catch (SQLException e) {
        e.printStackTrace();
    } finally {
        try {
            if (ptmt != null)
                ptmt.close();
            if (connection != null)
                connection.close();
        } catch (SQLException e) {
            e.printStackTrace();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

}

public void delete(Mahasiswa mahasiswa) {
    try {
        String queryString = "delete from mahasiswa where id=?";
        connection = getConnection();
        ptmt = connection.prepareStatement(queryString);
        ptmt.setLong(1, mahasiswa.getId());
        ptmt.executeUpdate();
    } catch (SQLException e) {
        e.printStackTrace();
    } finally {
        try {
            if (ptmt != null)
                ptmt.close();
            if (connection != null)
                connection.close();
        } catch (SQLException e) {
            e.printStackTrace();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

```

```

        }

    }

}

public ArrayList<Mahasiswa> getAll() {
    ArrayList<Mahasiswa> listMahasiswa = new ArrayList<Mahasiswa>();
    try {
        String queryString = "select id,nim,nama,jurusan,thnmasuk from
mahasiswa order by nim";
        connection = getConnection();
        ptmt = connection.prepareStatement(queryString);
        ResultSet rs = ptmt.executeQuery();
        while (rs.next()) {
            Mahasiswa s = new Mahasiswa();
            s.setId(rs.getLong(1));
            s.setNim(rs.getString(2));
            s.setNama(rs.getString(3));
            s.setJurusan(rs.getString(4));
            s.setTahunmasuk(rs.getInt(5));
            listMahasiswa.add(s);
        }
    } catch (SQLException e) {
        e.printStackTrace();
    } finally {
        try {
            if (ptmt != null)
                ptmt.close();
            if (connection != null)
                connection.close();
        } catch (SQLException e) {
            e.printStackTrace();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
    return listMahasiswa;
}

public Mahasiswa getByNim(String nim) {

    try {
        String queryString = "select id,nim,nama,jurusan,thnmasuk from
mahasiswa where nim=?";
        connection = getConnection();
        ptmt = connection.prepareStatement(queryString);
        ptmt.setString(1, nim);
        ResultSet rs = ptmt.executeQuery();
        while (rs.next()) {
            Mahasiswa s = new Mahasiswa();
            s.setId(rs.getLong(1));
            s.setNim(rs.getString(2));
            s.setNama(rs.getString(3));
            s.setJurusan(rs.getString(4));
            s.setTahunmasuk(rs.getInt(5));
            return s;
        }
    }
}

```

```

    } catch (SQLException e) {
        e.printStackTrace();
    } finally {
        try {
            if (ptmt != null)
                ptmt.close();
            if (connection != null)
                connection.close();
        } catch (SQLException e) {
            e.printStackTrace();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
    return null;
}

public Mahasiswa getById(int id) {

    try {
        String queryString = "select id,nim,nama,jurusan,thnmasuk from
mahasiswa where id=?";
        connection = getConnection();
        ptmt = connection.prepareStatement(queryString);
        ptmt.setInt(1, id);
        ResultSet rs = ptmt.executeQuery();
        while (rs.next()) {
            Mahasiswa s = new Mahasiswa();
            s.setId(rs.getLong(1));
            s.setNim(rs.getString(2));
            s.setNama(rs.getString(3));
            s.setJurusan(rs.getString(4));
            s.setTahunmasuk(rs.getInt(5));
            return s;
        }

    } catch (SQLException e) {
        e.printStackTrace();
    } finally {
        try {
            if (ptmt != null)
                ptmt.close();
            if (connection != null)
                connection.close();
        } catch (SQLException e) {
            e.printStackTrace();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
    return null;
}
}

```

Sedangkan DAO untuk entitas NilaiMataKuliah dapat kita susun sebagai berikut. Salin kode bersangkutan pada class yang sama pada project anda.

```

package org.stth.latihan.database;

import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.ArrayList;

public class NilaiMataKuliahDAO {
    Connection connection = null;
    PreparedStatement ptmt = null;
    ResultSet resultSet = null;

    private Connection getConnection() throws SQLException {
        Connection conn;
        conn = ConnectionFactory.getInstance().getConnection();
        return conn;
    }

    public void insert(NilaiMataKuliah nmk) {
        try {
            String queryString = "INSERT INTO
nilaimatakuliah(idmahasiswa,idmatakuliah, thnambil, semester,nilai)
VALUES(?,?,?,?,?)";
            connection = getConnection();
            ptmt = connection.prepareStatement(queryString);
            ptmt.setLong(1, nmk.getMahasiswa().getId());
            ptmt.setLong(2, nmk.getMataKuliah().getId());
            ptmt.setInt(3, nmk.getThnAmbil());
            ptmt.setInt(4, nmk.getSemester());
            ptmt.setString(5, String.valueOf(nmk.getNilai()));
            ptmt.executeUpdate();
        } catch (SQLException e) {
            e.printStackTrace();
        } finally {
            try {
                if (ptmt != null)
                    ptmt.close();
                if (connection != null)
                    connection.close();
            } catch (SQLException e) {
                e.printStackTrace();
            } catch (Exception e) {
                e.printStackTrace();
            }
        }
    }

    public void update(NilaiMataKuliah nmk) {
        try {
            String queryString = "update nilaimatakuliah set nimmahasiswa=?,
kodematakuliah=?, thnambil=?, semester=?, nilai=?" +
                                "where id=?";
            connection = getConnection();

```

```

        ptmt = connection.prepareStatement(queryString);
        ptmt.setString(1, nmk.getMahasiswa().getNim());
        ptmt.setString(2, nmk.getMataKuliah().getKode());
        ptmt.setInt(3, nmk.getThnAmbil());
        ptmt.setInt(4, nmk.getSemester());
        ptmt.setString(5, String.valueOf(nmk.getNilai()));
        ptmt.setInt(6, nmk.getId());
        ptmt.executeUpdate();
    } catch (SQLException e) {
        e.printStackTrace();
    } finally {
        try {
            if (ptmt != null)
                ptmt.close();
            if (connection != null)
                connection.close();
        } catch (SQLException e) {
            e.printStackTrace();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

public void delete(NilaiMataKuliah nmk) {
    try {
        String queryString = "delete from nilaimatakuliah where
id=?";

        connection = getConnection();
        ptmt = connection.prepareStatement(queryString);
        ptmt.setInt(1, nmk.getId());
        ptmt.executeUpdate();
    } catch (SQLException e) {
        e.printStackTrace();
    } finally {
        try {
            if (ptmt != null)
                ptmt.close();
            if (connection != null)
                connection.close();
        } catch (SQLException e) {
            e.printStackTrace();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

public ArrayList<NilaiMataKuliah> getAll() {
    ArrayList<NilaiMataKuliah> listNmk = new
ArrayList<NilaiMataKuliah>();
    try {
        String queryString = "select
id,idmahasiswa,idmatakuliah,thnambil,semester,nilai from nilaimatakuliah";

```



```

        connection = getConnection();
        ptmt = connection.prepareStatement(queryString);
        ResultSet rs = ptmt.executeQuery();
        while (rs.next()) {
            MahasiswaDAO mhdao = new MahasiswaDAO();
            MataKuliahDAO mkdao = new MataKuliahDAO();
            Mahasiswa mh = mhdao.getById(rs.getInt(2));
            MataKuliah mk = mkdao.getById(rs.getInt(3));
            NilaiMataKuliah nmk = new NilaiMataKuliah();
            nmk.setId(rs.getInt(1));
            nmk.setMahasiswa(mh);
            nmk.setMataKuliah(mk);
            nmk.setThnAmbil(rs.getInt(4));
            nmk.setSemester(rs.getInt(5));
            nmk.setNilai(rs.getString(6).charAt(0));
            listNmk.add(nmk);
        }
    } catch (SQLException e) {
        e.printStackTrace();
    } finally {
        try {
            if (ptmt != null)
                ptmt.close();
            if (connection != null)
                connection.close();
        } catch (SQLException e) {
            e.printStackTrace();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
    return listNmk;
}

public ArrayList<NilaiMataKuliah> getByMahasiswa(Mahasiswa mh) {
    ArrayList<NilaiMataKuliah> listNmk = new
ArrayList<NilaiMataKuliah>();
    try {
        String queryString = "select
id,idmahasiswa,idmatakuliah,thnambil,semester,nilai from nilaimatakuliah where
idmahasiswa=?";
        connection = getConnection();
        ptmt = connection.prepareStatement(queryString);
        ptmt.setLong(1, mh.getId());
        ResultSet rs = ptmt.executeQuery();
        while (rs.next()) {
            MahasiswaDAO mhdao = new MahasiswaDAO();
            MataKuliahDAO mkdao = new MataKuliahDAO();
            mh = mhdao.getById(rs.getInt(2));
            MataKuliah mk = mkdao.getById(rs.getInt(3));
            NilaiMataKuliah nmk = new NilaiMataKuliah();
            nmk.setId(rs.getInt(1));
            nmk.setMahasiswa(mh);
            nmk.setMataKuliah(mk);
            nmk.setThnAmbil(rs.getInt(4));
            nmk.setSemester(rs.getInt(5));
            nmk.setNilai(rs.getString(6).charAt(0));
            listNmk.add(nmk);
        }
    }
}

```

```

    }
    } catch (SQLException e) {
        e.printStackTrace();
    } finally {
        try {
            if (ptmt != null)
                ptmt.close();
            if (connection != null)
                connection.close();
        } catch (SQLException e) {
            e.printStackTrace();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
    return listNmk;
}

public ArrayList<NilaiMataKuliah> getByMataKuliah(MataKuliah mk) {
    ArrayList<NilaiMataKuliah> listNmk = new
ArrayList<NilaiMataKuliah>();
    try {
        String queryString = "select
id,idmahasiswa,idmatakuliah,thnambil,semester,nilai from nilaimatakuliah where
idmatakuliah=?";
        connection = getConnection();
        ptmt = connection.prepareStatement(queryString);
        ptmt.setLong(1, mk.getId());
        ResultSet rs = ptmt.executeQuery();
        while (rs.next()) {
            MahasiswaDAO mhdao = new MahasiswaDAO();
            MataKuliahDAO mkdao = new MataKuliahDAO();
            Mahasiswa mh = mhdao.getById(rs.getInt(2));
            mk = mkdao.getById(rs.getInt(3));
            NilaiMataKuliah nmk = new NilaiMataKuliah();
            nmk.setId(rs.getInt(1));
            nmk.setMahasiswa(mh);
            nmk.setMataKuliah(mk);
            nmk.setThnAmbil(rs.getInt(4));
            nmk.setSemester(rs.getInt(5));
            nmk.setNilai(rs.getString(6).charAt(0));
            listNmk.add(nmk);
        }
    } catch (SQLException e) {
        e.printStackTrace();
    } finally {
        try {
            if (ptmt != null)
                ptmt.close();
            if (connection != null)
                connection.close();
        } catch (SQLException e) {
            e.printStackTrace();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
    return listNmk;
}

```

```

    }
    public NilaiMataKuliah getByMahasiswaAndMatakuliah(Mahasiswa mh, MataKuliah
mk) {

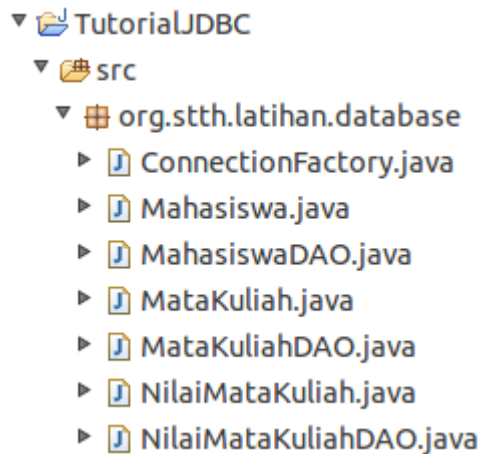
        try {
            String queryString = "select
id,idmahasiswa,idmatakuliah,thnambil,semester,nilai from nilaimatakuliah where
idmahasiswa=? and idmatakuliah=?";
            connection = getConnection();
            ptmt = connection.prepareStatement(queryString);
            ptmt.setLong(1, mh.getId());
            ptmt.setLong(2, mk.getId());
            ResultSet rs = ptmt.executeQuery();
            while (rs.next()) {
                MahasiswaDAO mhdao = new MahasiswaDAO();
                MataKuliahDAO mkdao = new MataKuliahDAO();
                mh = mhdao.getById(rs.getLong(2));
                mk = mkdao.getById(rs.getLong(3));
                NilaiMataKuliah nmk = new NilaiMataKuliah();
                nmk.setId(rs.getInt(1));
                nmk.setMahasiswa(mh);
                nmk.setMataKuliah(mk);
                nmk.setThnAmbil(rs.getInt(4));
                nmk.setSemester(rs.getInt(5));
                nmk.setNilai(rs.getString(6).charAt(0));
                return nmk;
            }

            } catch (SQLException e) {
                e.printStackTrace();
            } finally {
                try {
                    if (ptmt != null)
                        ptmt.close();
                    if (connection != null)
                        connection.close();
                } catch (SQLException e) {
                    e.printStackTrace();
                } catch (Exception e) {
                    e.printStackTrace();
                }
            }
            return null;
        }
    }
}

```

5.3 Mencoba program

Pada tahap ini, kita sudah membahas class-class utama yakni class untuk koneksi, class entitas dan class DAO untuk masing-masing entitas. Sebelum melanjutkan ke tahap menguji kode program yang telah kita buat, pastikan bahwa class-class dimaksud telah ada anda salin ke project anda. Gunakan daftar class pada gambar berikut sebagai panduan.



Gambar 12: Daftar class pada tutorial akses database menggunakan JDBC

Jika anda telah memastikan bahwa class-class tersebut sudah ada pada project anda, buatlah sebuah class baru dengan nama 'TestDAO'. Class ini akan kita gunakan untuk mencoba koneksi ke database dan menguji kode yang telah kita bangun.

Skenario pengujian :

Tabel-tabel pada database yang telah kita buat telah masing-masing telah memiliki record data. Tabel mahasiswa memiliki 3 record, tabel matakuliah memiliki 4 record dan tabel nilaimatakuliah yang merupakan perkawinan antara tabel mahasiswa dan matakuliah telah dilengkapi dengan 12 data nilai mahasiswa (3 X 4).

Kita akan membuat kode program yang akan menambahkan satu record mahasiswa, satu mata kuliah baru dan satu record nilai mata kuliah, kemudian menampilkan ipk seluruh mahasiswa yang ada di dalam database.

Berikut adalah cuplikan kode untuk pengujian dimaksud.

```
public static void main(String[] args) {
    tambahDataMahasiswa();
    tambahDataMataKuliah();
    tambahDataNilaiMataKuliah();
    cetakIPK();
    bersihkanData();
}
```

Ini adalah blok utama pengujian yang akan memanggil method lain secara berurutan. Perhatikan bahwa pada akhir pengujian kita melakukan pembersihan data yang dimaksudkan untuk membuang data-data yang kita ciptakan pada database selama pengujian (mengembalikan database ke posisi sebelum pengujian)

```
static void tambahDataMahasiswa(){
    MahasiswaDAO mahasiswaDAO = new MahasiswaDAO();
    //siapkan data mahasiswa baru
    Mahasiswa m = new Mahasiswa("Udin Gaul", "01210995", "Teknik
Informatika", 2010);
    //simpan data ke dalam database
    mahasiswaDAO.insert(m);
    //ambil kembali data dari database
    m = mahasiswaDAO.getByNim("01210995");
    //cek hasil pada console
```

```

        System.out.println(m.getId()+":"+m.getNama());
    }

```

Method ini membuat object Mahasiswa kemudian menyimpannya ke dalam database, data yang sama kemudian di load ulang untuk pengecekan.

```

static void tambahDataMataKuliah(){
    MataKuliahDAO mataKuliahDAO = new MataKuliahDAO();
    //siapkan data mata kuliah baru
    MataKuliah mk = new MataKuliah("PWB", "Pemrograman Berbasis Web", 3);
    //simpan data ke dalam database
    mataKuliahDAO.insert(mk);
    //ambil data dari database
    mk = mataKuliahDAO.getByKode("PWB");
    //cek hasil pada console
    System.out.println(mk.getId()+":"+mk.getNama());
}

```

Method ini membuat object MataKuliah kemudian menyimpannya ke dalam database, data yang sama kemudian di load ulang untuk pengecekan.

```

static void tambahDataNilaiMataKuliah(){
    MahasiswaDAO mahasiswaDAO = new MahasiswaDAO();
    Mahasiswa m = mahasiswaDAO.getByNim("01210995");
    MataKuliahDAO mataKuliahDAO = new MataKuliahDAO();
    MataKuliah mk = mataKuliahDAO.getByKode("PWB");
    NilaiMataKuliahDAO nilaiMataKuliahDAO = new NilaiMataKuliahDAO();
    //siapkan data nilai mata kuliah baru
    NilaiMataKuliah nmk = new NilaiMataKuliah(m, mk, 2012, 'A');
    //simpan ke dalam database
    nilaiMataKuliahDAO.insert(nmk);
    //ambil dari database
    nmk = nilaiMataKuliahDAO.getByMahasiswaAndMatakuliah(m, mk);
    //cek hasil pada console
    System.out.println(nmk.getId()+":"+nmk.getMahasiswa().getNama()+":"
        +nmk.getMataKuliah().getNama()
        +":"+nmk.getNilai());
}

```

Method ini membuat object NilaiMataKuliah dari object Mahasiswa dan NilaiMataKuliah sebelumnya dan menyimpannya ke dalam database. Load kemudian ulang dilakukan untuk memastikan data tersebut telah tersimpan.

```

static void cetakIPK(){
    MahasiswaDAO mahasiswaDAO = new MahasiswaDAO();
    NilaiMataKuliahDAO nmkDAO = new NilaiMataKuliahDAO();
    ArrayList<Mahasiswa> ms = new ArrayList<Mahasiswa>();
    ms = mahasiswaDAO.getAll();
    for (Mahasiswa mahasiswa : ms) {
        ArrayList<NilaiMataKuliah> listNilai =
nmkDAO.getByMahasiswa(mahasiswa);
        mahasiswa.setNilaiKuliah(listNilai);
        System.out.println(mahasiswa.getNim()+" "+mahasiswa.getNama()+"
memiliki IPK "+mahasiswa.getIPK());
    }
}

```

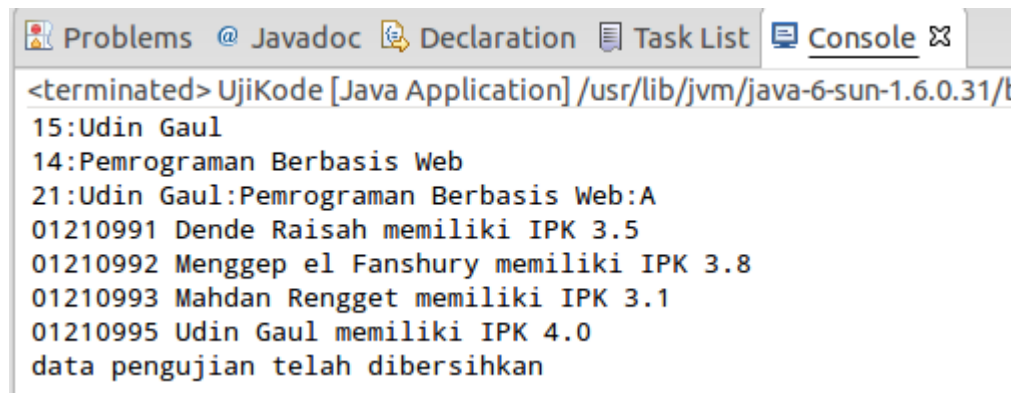
```
}  
}
```

Method ini mencetak data seluruh mahasiswa pada database sekaligus menampilkan IPK-nya, termasuk untuk mahasiswa yang kita masukkan pada method sebelumnya.

```
static void bersihkanData(){  
    MahasiswaDAO mahasiswaDAO = new MahasiswaDAO();  
    MataKuliahDAO mataKuliahDAO = new MataKuliahDAO();  
    NilaiMataKuliahDAO nmkDAO = new NilaiMataKuliahDAO();  
    Mahasiswa m = mahasiswaDAO.getByNim("01210995");  
    MataKuliah mk = mataKuliahDAO.getByKode("PWB");  
    NilaiMataKuliah nmk = nmkDAO.getByMahasiswaAndMatakuliah(m, mk);  
    nmkDAO.delete(nmk);  
    mahasiswaDAO.delete(m);  
    mataKuliahDAO.delete(mk);  
    System.out.println("data pengujian telah dibersihkan");  
}
```

Method ini membersihkan seluruh data yang telah kita ciptakan selama pengujian

Gabung method tersebut dalam sebuah Class dan jalankan. Jika console tidak menampilkan error dan hasilnya sama dengan output berikut, maka anda telah sukses membuat program yang terhubung dengan sebuah database server.



```
<terminated> UjiKode [Java Application] /usr/lib/jvm/java-6-sun-1.6.0.31/t  
15:Udin Gaul  
14:Pemrograman Berbasis Web  
21:Udin Gaul:Pemrograman Berbasis Web:A  
01210991 Dende Raisah memiliki IPK 3.5  
01210992 Menggep el Fanshury memiliki IPK 3.8  
01210993 Mahdan Rengget memiliki IPK 3.1  
01210995 Udin Gaul memiliki IPK 4.0  
data pengujian telah dibersihkan
```

Gambar 13: Hasil standar pengujian kode untuk mengakses database

5.4 Menggunakan database lain

Jika anda menggunakan database selain MySQL maka anda hanya perlu mengganti driver JDBC dengan driver yang sesuai dengan database anda, kemudian mengganti parameter url database dan nama class JDBC yang digunakan pada class ConnectionFactory.

Berikut adalah beberapa alamat di internet yang dapat anda gunakan untuk referensi bila anda kebetulan menggunakan database selain MySQL.

1. Oracle, <http://www.oracle.com/technetwork/database/enterprise-edition/jdbc-112010-090769.html>
2. Microsoft SQL Server, <http://msdn.microsoft.com/en-us/sqlserver/aa937724.aspx>
3. PostgreSQL, <http://jdbc.postgresql.org/>

6 Tutorial Desktop Application dengan Java GUI (Swing)

7 Tutorial Web Application dengan Vaadin

8 Tutorial Mobile Application (Android)
