2016

Modul Praktikum Pemrograman Berorientasi Obyek

LABORATORIUM PROGRAMMING

KELOMPOK KEAHLIAN PROGRAMMING

FAKULTAS ILMU TERAPAN

UNIVERSITAS TELKOM

DAFTAR PENYUSUN

Reza Budiawan, S.T., M.T.,

LEMBAR REVISI

No.	Keterangan Revisi	Tanggal Revisi Terakhir

LEMBAR PERNYATAAN

Saya yang bertanggung jawab di bawah ini:

Nama : Reza Budiawan, S.T., M.T.,

NIP : 14881329-1

Dosen PJMP : Pemrograman Berorientasi Obyek

Kelompok Keahlian : Programming

Menerangkan dengan sesungguhnya bahwa modul ini telah direview dan akan digunakan untuk pelaksanaan praktikum di Semester Gasal Tahun Ajaran 2016/2017 di Laboratorium Programming Fakultas Ilmu Terapan Universitas Telkom

Bandung, 25 Agustus 2016
Mengetahui,

Ketua Kelompok Keahlian	Dosen PJMP
Hariandi Maulid, S.T., M.sc.	Reza Budiawan, M.T.
NIP. 15781201-4	NIP. 14881329-1

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR PROGRAM

DAFTAR TABEL

Modul 0 : Running Modul

0.1 Tujuan

Setelah mengikuti Running Modul mahasiswa diharapkan dapat:

1. Memahami peraturan kegiatan praktikum.
2. Memahami Hak dan Kewajiban praktikan dalam kegiatan praktikum.
3. Memhami komponen penilaian kegiatan praktikum.

0.2 Peraturan Praktikum

1. Praktikum diampu oleh **Dosen Kelas** dan dibantu oleh **Asisten Laboratorium** dan **Asisten Praktikum**.
2. Praktikum dilaksanakan di Gedung FIT lantai 2 (**PROGRAMMING LAB**) sesuai jadwal yang ditentukan.
3. Praktikan wajib membawa **modul praktikum, dan alat tulis**.
4. Praktikan wajib mengisi **daftar hadir** dan **BAP praktikum** dengan bolpoin **bertinta hitam**.
5. Durasi kegiatan praktikum **D3 = 4 jam (200 menit)**.
    a. 15 menit untuk pengerjaan Tes Awal atau wawancara Tugas Pendahuluan
    b. 60 menit untuk penyampaian materi
    c. 125 menit untuk pengerjaan jurnal dan tes akhir
6. Jumlah **pertemuan praktikum**:
    - 10 kali di lab (praktikum rutin)
    - 3 kali di luar lab (terkait Tugas Besar dan/atau UAS)
    - 1 kali berupa presentasi Tugas Besar dan/atau pelaksanaan UAS
7. Praktikan **wajib hadir minimal 75%** dari seluruh pertemuan praktikum di lab.Jika total kehadiran kurang dari 75% maka nilai UAS/ Tugas Besar = 0.
8. Praktikan yang datang terlambat :
    - <= 30 menit : diperbolehkan mengikuti praktikum tanpa tambahan waktu Tes Awal
    - > 30 menit : tidak diperbolehkan mengikuti praktikum
9. Saat praktikum berlangsung, asisten praktikum dan praktikan:
    - Wajib menggunakan **seragam** sesuai aturan Institusi.
    - Wajib mematikan/ men-silent semua **alat komunikasi**(smartphone, tab, iPad, dsb).
    - Dilarang membuka **aplikasi yang tidak berhubungan** dengan praktikum yang berlangsung.
    - Dilarang mengubah **setting software maupun hardware** komputer tanpa ijin.
    - Dilarang **membawa makanan maupun minuman** di ruang praktikum.
    - Dilarang **memberikan jawaban ke praktikan lain** (pre-test, TP, jurnal, dan post-test).
    - Dilarang **menyebarkan soal pre-test, jurnal, dan post-test**.

- Dilarang **membuang sampah/sesuatu apapun** di ruangan praktikum.

10. Pelanggaran terhadap peraturan praktikum ini akan ditindak secara tegas secara berjenjang di lingkup Kelas, Laboratorium, Program Studi, Fakultas, hingga Institusi.

0.3 Penilaian Praktikum

1. Komponen penilaian praktikum:
   60% nilai permodul dan **40%** nilai Tugas Besar (atau UAS praktek)
2. Seluruh komponen penilaian beserta pembobotannya ditentukan oleh dosen **PJMP**
3. Penilaian per modul dilakukan oleh **asisten praktikum**, sedangkan nilai Tugas Besar/UAS diserahkan kepada **dosen kelas,** dilaporkan ke **PJMP.**
4. Baik praktikan maupun asisten tidak diperkenankan meminta atau memberikan **tugas tambahan** untuk perbaikan nilai.
5. Standar **indeks dan range nilai** ditentukan oleh dosen PJMP atas sepengetahuan Ketua Kelompok Keahlian

Modul 1 : Class & Object

1.1 Tujuan

Setelah mengikuti praktikum ini mahasiswa diharapkan dapat:

1. Memahami konsep class & object.
2. Dapat mengimplementasikan konsep class & object pada kasus sederhana.

1.2 Alat & Bahan

Alat & Bahan Yang digunakan adalah hardware perangkat PC beserta Kelengkapannya berjumlah 40 PC dengan Java SDK dan editor IDE (disarankan Netbeans) telah terinstall di masing-masing unit.

1.3 Dasar Teori

1.3.1 Class & Object

Object adalah gambaran dari entity, baik dunia nyata atau konsep dengan batasan-batasan dan pengertian yang tepat. Objek-objek ini kemudian juga dapat berupa gabungan dari beberapa objek yang lebih kecil. Sebagai contoh, lihatlah sebuah mobil. Mobil adalah sebuah objek dalam kehidupan nyata. Namun mobil sendiri merupakan gabungan beberapa objek yang lebih kecil seperti roda ban, mesin, jok, dan lainnya. Mobil sebagai objek yang merupakan gabungan dari objek yang lebih kecil dibentuk dengan membentuk hubungan antara objek-objek penyusunnya.

Kelas mendeskripsikan suatu objek. Dalam bahasa biologi, dapat dikatakan bahwa kelas adalah species, sedangkan objek merupakan individu. Kelas merupakan sebuah "blueprint" atau cetak biru yang sama untuk objek yang dibentuk dengan nilai yang berbeda-beda.

Kelas memiliki variabel yang disebut sebagai ***attribute*** dan *subroutine (a set of instructions designed to perform a frequently used operation*) yang biasa disebut ***method***. Dalam sudut pandang pemrograman, kelas digunakan untuk menciptakan suatu obyek. Atau dengan kata lain, kelas merupakan pembuat objek.

Pada class terdapat suatu *access modifier.* Hal ini berguna untuk menentukan tipe hak akses bagi sebuah attribute dan method:

Tabel 1: Daftar Access Modifier pada Java

Modifier	Class dan Interface	Method dan Variabel
Default (tak ada modifier) **friendly**	Tampak di Paketnya	Diwarisi oleh subclassnya di paket yang sama dengan classnya. Dapat diakses oleh method-method di class-class yang sepaket.
public	Tampak di manapun	Diwarisi oleh semua subclassnya. Dapat diakses dimanapun.
protected	Tidak dapat diterapkan	Diwarisi oleh semua subclassnya. Dapat diakses oleh method-method di class-

		class yang sepaket.
private	Tidak dapat diterapkan	Tidak diwarisi oleh subclassnya Tidak dapat diakses oleh class lain.

1.3.2 Method

Method dikenal juga sebagai suatu *function* dan *procedure*. Dalam OOP, method digunakan untuk memodularisasi program melalui pemisahan tugas dalam suatu class. Pemanggilan *method* menspesifikasikan nama *method* dan menyediakan informasi (parameter) yang diperlukan untuk melaksanakan tugasnya.

1.3.3 Keyword this

Di dalam Java terdapat suatu besaran referensi khusus yang disebut this, yang digunakan di dalam method yang dirujuk untuk objek yang sedang berlaku. Nilai this merujuk pada objek di mana method yang sedang berjalan dipanggil.

1.3.4 Konstruktor

Constructor atau konstruktor digunakan untuk melakukan inisialisasi variable-variabel instan class serta melakukan persiapan-persiapan yang diperlukan oleh suatu objek untuk dapat beroperasi dengan baik.

Format umum pendeklarasian dan pendefinisian *constructor* adalah :

a. Nama *constructor* sama dengan nama *class*.
b. Sebelum itu dapat diberi *access modifier* untuk mengatur *visibility constructor*.

Dalam suatu Class dapat lebih dari satu *constructor*, masing-masing harus mempunyai parameter yang berbeda sebagai penandanya. Hal seperti ini disebut ***overloading*** terhadap *constructor*.

1.3.5 Enkapsulasi

Enkapsulasi merupakan proses pemaketan objek beserta methodnya untuk menyembunyikan rincian implementasi dari pemakai/objek lainnya. Inti dari enkapsulasi atau pengkapsulan adalah ketidaktahuan apa yang ada dalam suatu objek dan bagaimana pengimplementasiannya. Yang dibutuhkan hanyalah apa kegunaan, bagaimana cara memakainya dan apa yang akan terjadi.

Dengan enkapsulasi, maka programmer tidak dapat mengakses suatu atribut yang dimiliki oleh suatu *class*. Kemampuan ini ditujukan untuk mendapatkan desain suatu *software* yang baik dan untuk keamanan *software* itu sendiri. Segala yang tidak perlu diketahui oleh yang lain, tidak perlu di-*publish*.

Salah satu implementasi dari enkapsulasi adalah adanya *setter* dan *getter* untuk suatu atribut dalam suatu kelas. Jika pada suatu kelas terdapat atribut a dan b, maka terdapat method setA-getA dan setB-getB. Bentuk lain dari enkapsulasi adalah memasukkan nilai atribut dengan menggunakan konstruktor.

1.4 Latihan

Latihan 1
Buatlah sebuah program untuk menentukan apakah sebuah bilangan termasuk ke dalam salah satu kriteria berikut: a. Positif Genap b. Negatif Genap c. Positif Ganjil d. Negatif Ganjil Gunakan prinsip class & object dengan sebuah konstruktor untuk mengeset bilangan.

Latihan 2
Buatlah program untuk meng-generate deret dengan ketentuan terdapat suatu: a. Nilai awal b. Beda c. Jumlah kemunculan angka pada deret Program akan menghitung nilai rataan dari deret tersebut. Contoh: Masukkan jumlah kemunculan deret: 4 Deret: 2 5 8 11 Gunakan prinsip class & object dengan sebuah konstruktor untuk mengeset hal yang diketahui untuk menghasilkan deret.

Latihan 3
Conan merupakan anak SD penggemar novel detektif. Ia ingin membuat sebuah program untuk mendata setiap novel detektif yang ia miliki. Setiap novel memiliki judul, nama pengarang dan tahun terbit. Tapi Conan ingin mengetahui isi dari setiap novel, sehingga ia tahu deskripsi novel tersebut. Conan juga menginginkan informasi harga beli tercantum di program. Dikarenakan sewaktu-waktu ia ingin menjual kembali novelnya, terdapat mekanisme untuk menghitung harga jual novel. Rumus yang ia gunakan adalah “harga jual = harga beli – 20% * harga beli”. Conan menginginkan programnya dibuat menggunakan konsep OOP, menggunakan bahasa Java dengan terdapat konsep enkapsulasi di dalamnya. Bantulah Conan untuk merancang program yang akan ia buat. Tentukanlah atribut dan method terlibat, gambarkan class diagramnya. Buatlah 3 objek berdasarkan class tersebut (data bebas), dan tampilkan informasi dari setiap buku tersebut. Lalu, tampilkan harga total beli buku serta harga total buku jika dijual.

Modul 2 : Class & Object – Class Diagram

2.1 Tujuan

Setelah mengikuti praktikum ini mahasiswa diharapkan dapat:

1. Memahami konsep class & object.
2. Dapat mengimplementasikan konsep class & object pada kasus sederhana.
3. Mampu mentranslasikan class diagram menjadi kode program.

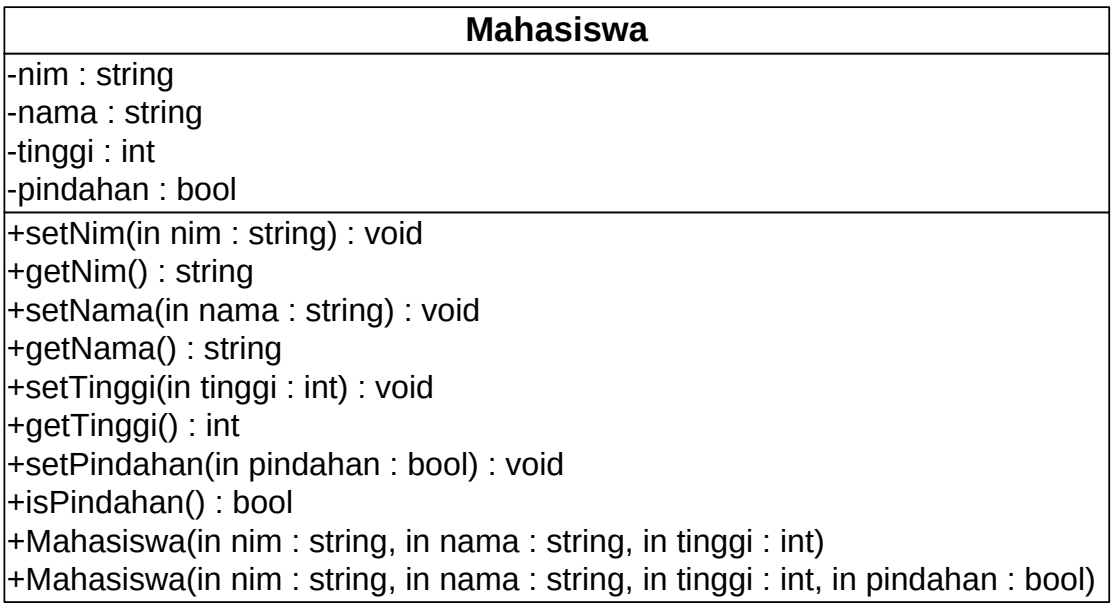
2.2 Alat & Bahan

Alat & Bahan Yang digunakan adalah hardware perangkat PC beserta Kelengkapannya berjumlah 40 PC dengan Java SDK dan editor IDE (disarankan Netbeans) telah terinstall di masing-masing unit.

2.3 Dasar Teori

2.3.1 Class & Object

Pada pembuatan perangkat lunak berbasis Object Oriented, terdapat mekanisme untuk merancang perangkat lunak tersebut menggunakan UML (Unified Modeling Language). Salah satu bentuk perancangan dari UML adalah **Class Diagram**. Contoh class diagram diperlihatkan pada gambar di bawah ini.



Gambar 1: Class Diagram dari Class Mahasiswa

Berdasarkan gambar di atas, terlihat bahwa class diagram merupakan sebuah bujur sangkar dengan 3 tingkatan, dengan konten di masing-masing tingkat sebagai berikut:

a) Tingkat pertama: nama Class
b) Tingkat kedua: nama atribut beserta tipe datanya
c) Tingkat ketiga: nama method beserta keterangannya

2.3.2 Penulisan Atribut

Atribut dituliskan dengan aturan berikut:

visibility name: type-expression= initial-value{ property-string}

Keterangan:

a) Visibility: private (-), protected(#), dan public (+)
b) Name: nama atribut
c) Type-expression: tipe data dari atribut
d) Initial-value: nilai awal atribut (jika ada)
e) Property-string: aturan perancangan lainnya (jika ada)

Jika sebuah atribut bersifat final (merupakan konstanta), maka pada class diagram ditulis dengan huruf kapital. Jika atribut bersifat static, berikan *underline*.

2.3.3 Penulisan Method

Method dituliskan dengan aturan berikut:

visibility name(parameter-list) : return-type-expression{ property-string}

Keterangan:

a) Visibility: private (-), protected(#), dan public (+)
b) Name: nama method
c) Parameter-list: penulisan parameter yang dilewatkan oleh method. Terdapat aturan lain untuk ini.
d) Return -ype-expression: tipe data yang dikembalikan oleh method
e) Property-string: aturan perancangan lainnya (jika ada)

Aturan penulisan parameter-list:

kind name: type-expression= default-value

Keterangan:

a) Kind: jenis parameter dari method. Terdapat 3 jenis parameter: in, out, inout. Khusus bahasa pemrograman Java, hanya ada tipe “in”.
b) Name: nama parameter yang dilewatkan
c) Type-expression: tipe data parameter yang dilewatkan
d) Default-value: nilai awal parameter (jika ada).

Class diagram “Mahasiswa” di atas memiliki 2 konstruktor. Dengan kata lain, overloading constructor terjadi pada class “Mahasiswa”. Sebuah class dapat memiliki lebih dari 1 konstruktor. Bahkan, sebuah class dapat tidak dituliskan konstruktornya. Secara default konstruktor kosong akan

terbentuk jika tidak dituliskan konstruktor secara eksplisit pada sebuah class. Bentuk dari konstruktor kosong:

Listing Program 1: Konstruktor Kosong

```
public NamaClass(){

}
```

Bentuk pengkodean dari class Mahasiswa adalah sebagai berikut:

Listing Program 2: Class Mahasiswa.java

```
public class Mahasiswa {
    private String nim;
    private String nama;
    private int tinggi;
    private boolean pindahan;

    public String getNim() {
        return nim;
    }

    public void setNim(String nim) {
        this.nim = nim;
    }

    public String getNama() {
        return nama;
    }

    public void setNama(String nama) {
        this.nama = nama;
    }

    public int getTinggi() {
        return tinggi;
    }

    public void setTinggi(int tinggi) {
        this.tinggi = tinggi;
    }

    public boolean isPindahan() {
        return pindahan;
    }

    public void setPindahan(boolean pindahan) {
        this.pindahan = pindahan;
    }

    public Mahasiswa(String nim, String nama, int tinggi) {
        this.nim = nim;
        this.nama = nama;
        this.tinggi = tinggi;
    }
```

```
    public Mahasiswa(String nim, String nama, int tinggi, boolean pindahan) {
        this(nim,nama,tinggi); //memanggil konstruktor 3 parameter
        this.pindahan = pindahan;
    }
}
```

Sebagai catatan, bentuk isPindahan merupakan hal yang sama dengan bentuk get lainnya. Hanya saja, untuk pengembalian tipe Boolean, “get” akan berubah menjadi “is”. Hal ini hanya bentuk kesepakatan.

Untuk membentuk objek dari class di atas dan menampilkan nilai objek, dapat dibentuk sebuah class baru dengan kandungan public static void main(String args[]) sebagai berikut:

Listing Program 3: Class MainMhs.java

```
public class MainMhs {
    public static void main(String[] args) {
        Mahasiswa m1 = new Mahasiswa("6701148000", "Angga", 166);
        m1.setPindahan(false);

        //menampilkan data
        System.out.println("Data Mahasiswa 1");
        System.out.println("NIM: "+m1.getNim());
        System.out.println("Nama: "+m1.getNama());
        System.out.println("Tinggi Badan: "+m1.getTinggi());
        if(m1.isPindahan()){
            System.out.println("Mahasiswa pindahan");
        }else{
            System.out.println("Mahasiswa reguler");
        }
    }
}
```

```
Output - Modul PBO Gasal1617 (run)
run:
Data Mahasiswa 1
NIM: 6701148000
Nama: Angga
Tinggi Badan: 166
Mahasiswa reguler
BUILD SUCCESSFUL (total time: 1 second)
```

Gambar 2: Hasil MainMhs.java

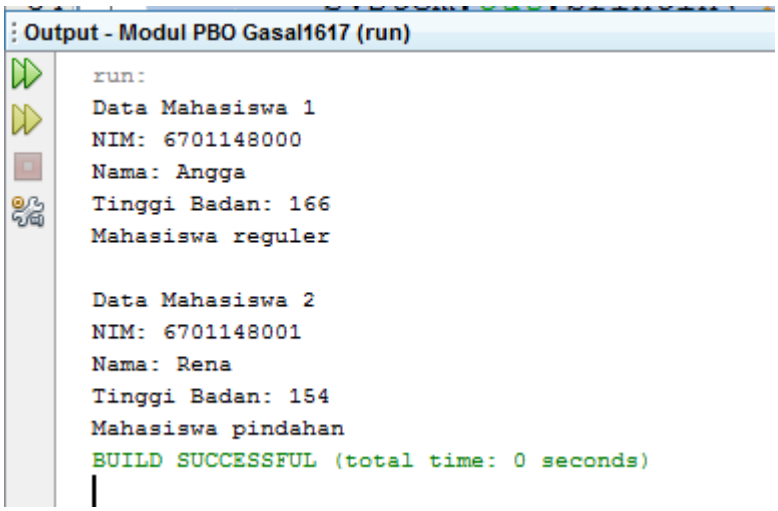
Class di atas membentuk sebuah objek dari class Mahasiswa bernama m1 melalui konstruktor dengan 3 parameter. Karena class Mahasiswa memiliki 4 atribut, sedangkan konstruktor hanya memberikan fasilitas 3 parameter (nim, nama, dan tinggi), maka data pindahan/tidak diberikan melalui set-nya.

Contoh lain dari pembentukan mahasiswa (m2) yang dibentuk melalui konstruktor 4 parameter dituliskan sebagai berikut.

Listing Program 4: MainMhs.java Modifikasi m2

```
public class MainMhs {
    public static void main(String[] args) {
        Mahasiswa m1 = new Mahasiswa("6701148000", "Angga", 166);
        m1.setPindahan(false);

        //menampilkan data
        System.out.println("Data Mahasiswa 1");
        System.out.println("NIM: "+m1.getNim());
        System.out.println("Nama: "+m1.getNama());
        System.out.println("Tinggi Badan: "+m1.getTinggi());
        if(m1.isPindahan()){
            System.out.println("Mahasiswa pindahan");
        }else{
            System.out.println("Mahasiswa reguler");
        }

        Mahasiswa m2 = new Mahasiswa("6701148001", "Rena", 154, true);

        //menampilkan data
        System.out.println();
        System.out.println("Data Mahasiswa 2");
        System.out.println("NIM: "+m2.getNim());
        System.out.println("Nama: "+m2.getNama());
        System.out.println("Tinggi Badan: "+m2.getTinggi());
        if(m2.isPindahan()){
            System.out.println("Mahasiswa pindahan");
        }else{
            System.out.println("Mahasiswa reguler");
        }
    }
}
```

Hasil run program:

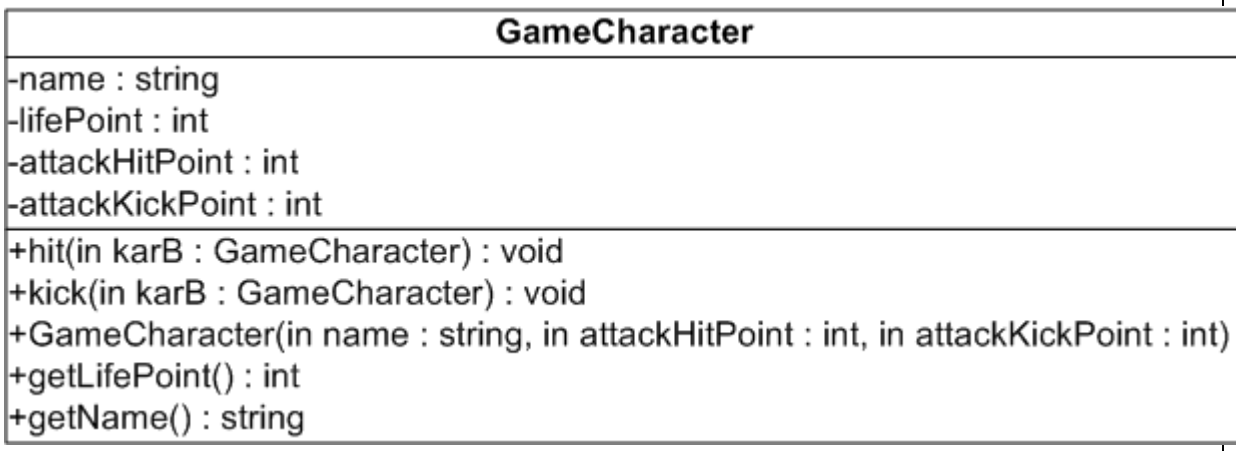


Gambar 3: Hasil MainMhs.java Modifikasi m2

2.4 Latihan

Latihan 1

Seorang developer ingin membuat sebuah simulasi fighting game menggunakan konsep PBO. Setelah menganalisis kebutuhan, class diagram yang dihasilkan adalah sebagai berikut:



Keterangan:

1. Atribut name merupakan identitas karakter
2. Atribut lifePoint merupakan atribut yang merepresentasikan tenaga dari tiap karakter. Nilai default dari lifePoint adalah 100.
3. Atribut attackHitPoint merupakan kekuatan yang diberikan ketika melakukan hit.
4. Atribut attackKickPoint merupakan kekuatan yang diberikan ketika melakukan kick
5. Method “hit” merupakan method untuk menendang lawan. Masukan dari method ini adalah objek GameCharacter lain. Efek dari method ini mengurangi lifePoint lawan sebanyak attackHitPoint yang dimiliki.
6. Method “kick” merupakan method untuk menendang lawan. Masukan dari method ini adalah objek GameCharacter lain. Efek dari method ini mengurangi lifePoint lawan sebanyak attackKickPoint yang dimiliki.
7. Konstruktor digunakan untuk menge-set nama, attackHitPoint, dan attackKickPoint sesuai parameter masukan. Selain itu, konstruktor juga melakukan set lifePoint sebesar nilai default.
8. Method getLifePoint dan getName merupakan method get bagi kedua atribut.
9. Tidak terdapat konstruktor kosong.

Berdasarkan skenario di atas, bantulah developer untuk mengkodekan class di atas. Lalu, pada class Main, berikan scenario berikut:

1. Buat sebuah objek dengan name: “Raiden”, attackHitPoint: 10, attackKickPoint: 20.
2. Buat sebuah objek dengan name: “Sub-Zero”, attackHitPoint: 5, attackKickPoint: 25
3. Objek Raiden memulai pertarungan dengan melakukan tendangan pada objek Sub-Zero.
4. Objek Sub-Zero melakukan perlawanan dengan menendang balik objek Raiden.
5. Objek Sub-Zero menyerang Raiden dengan pukulan berturut-turut sebanyak 3x (gunakan perulangan).
6. Pertarungan diakhiri oleh Raiden dengan melakukan tendangan beruntun 4x pada objek

Sub-Zero (gunakan perulangan).

7. Tampilkan lifePoint dari objek Raiden dan objek Sub-Zero. Bandingkan kedua lifePoint tersebut.
8. Tampilkan status kemenangan dari pertarungan kedua objek. Objek yang menang adalah objek yang memiliki lifePoint tertinggi di akhir pertarungan.

Latihan 2

Pecahan
-pembilang : int -penyebut : int
+getPembilang() : int +getPenyebut() : int +Pecahan(in pembilang : int, in penyebut : int) +tambah(in p : Pecahan) : Pecahan

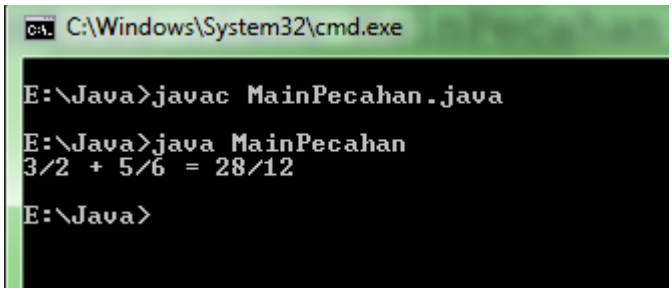
Diketahui sebuah class seperti di samping. Konstruktor memiliki 2 paramter untuk menge-set kedua atribut dari class Pecahan. Method tambah merupakan method untuk menambahkan sebuah objek Pecahan dengan objek Pecahan lainnya. Method ini merupakan function dengan nilai yang dikembalikan merupakan hasil penambahan kedua objek Pecahan dengan tipe data class Pecahan tersebut.

Berdasarkan informasi tersebut,

1. Kodekanlah class di samping beserta logika dari setiap method
2. Bentuklah sebuah Main class yang membentuk 2 objek sebagai berikut:

Objek 1: 3/2 Objek 2: 5/6

Setelah membentuk kedua objek, tambahkan kedua objek tersebut, dan tampilkan hasil penambahan kedua objek tersebut. Contoh tampilan program:



Modul 3 : Pewarisan

3.1 Tujuan

Setelah mengikuti praktikum ini mahasiswa diharapkan dapat:

1. Memahami konsep hubungan antar class Inheritance/Pewarisan.
2. Dapat mengimplementasikan konsep pewarisan class menggunakan bahasa pemrograman Java.

3.2 Alat & Bahan

Alat & Bahan Yang digunakan adalah hardware perangkat PC beserta Kelengkapannya berjumlah 40 PC dengan Java SDK dan editor IDE (disarankan Netbeans) telah terinstall di masing-masing unit.

3.3 Dasar Teori

3.3.1 Pewarisan/Inheritance

Inheritance atau pewarisan merupakan suatu cara untuk menurunkan suatu class yang lebih umum menjadi suatu class yang lebih spesifik. Inheritance adalah salah satu ciri utama suatu bahasa program yang berorientasi pada objek. Inti dari pewarisan adalah sifat reusable dari konsep object oriented. Setiap subclass akan "mewarisi" sifat dari superclass selama bersifat protected ataupun public.

Dalam inheritance terdapat dua istilah yang sering digunakan. Kelas yang menurunkan disebut kelas dasar (base class/super class), sedangkan kelas yang diturunkan disebut kelas turunan (derived class/sub class).Karakteristik pada super class akan dimiliki juga oleh subclassnya. Terdapat 2 bentuk pewarisan: single inheritance dan multiple inheritance. Bahasa pemrograman Java hanya mendukung single inheritance (1 super class memiliki 1-n subclass).

Pada class diagram, pewarisan digambarkan dengan sebuah garis tegas, dengan segitiga di ujungnya. Class yang dekat pada segitiga merupakan superclass, sedangkan class yang jauh dari segitiga merupakan subclass. Untuk membentuk sebuah subclass, keyword "extends" digunakan (lihat contoh pada sesi "Implementasi Pewarisan").

3.3.2 Sifat Instansiasi & Pewarisan

Jika sebuah subclass diinstansiasi, maka konstruktor dari superclass juga akan dieksekusi untuk membentuk objek dari subclass. Contoh dapat dilihat pada kode di bawah ini:

Diketahui 2 buah class dituliskan seperti ini:

Listing Program 5: KelasSatu.java

```
public class KelasSatu {
    public KelasSatu(){
        System.out.println("Konstruktor Kelas Satu");
    }
}
```

Listing Program 6: KelasDua.java

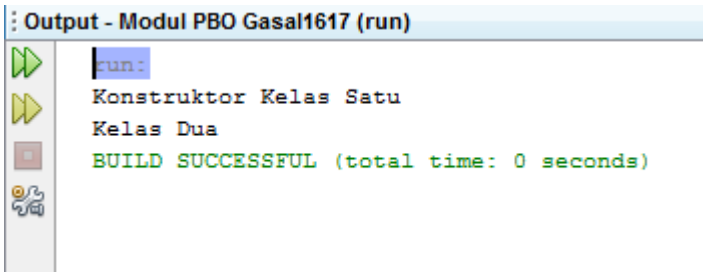
```
public class KelasDua extends KelasSatu{
    public KelasDua(){
        System.out.println("Kelas Dua");
    }
}
```

Ketika objek KelasDua dibentuk di class Main,

Listing Program 7: MainKelas.java

```
public class MainKelas {
    public static void main(String[] args) {
        KelasDua kd = new KelasDua();
    }
}
```

Hasil dari jalannya program adalah sebagai berikut:

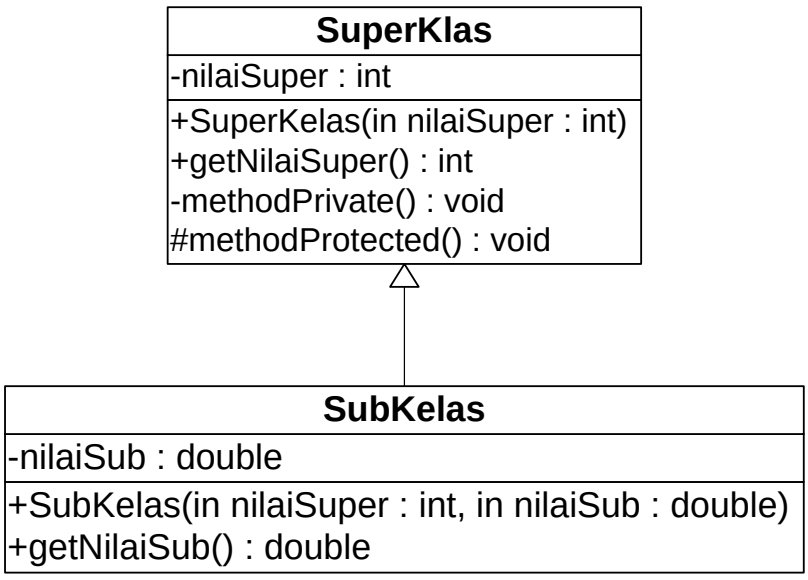


Gambar 4: Hasil MainKelas.java

Terlihat bahwa konstruktor superclass juga diakses ketika konstruktor subclass diakses.

3.3.3 Implementasi Pewarisan/Inheritance

Diketahui sebuah class diagram sebagai berikut:



Gambar 5: Class Diagram Inheritance/Pewarisan

Classes pada class diagram di atas dapat dituliskan kodenya sebagai berikut:

Listing Program 8: SuperKlas.java

```
public class SuperKlas {
    private int nilaiSuper;

    public SuperKlas(int nilaiSuper) {
        this.nilaiSuper = nilaiSuper;
    }

    public int getNilaiSuper() {
        return nilaiSuper;
    }

    private void methodPrivate(){
        System.out.println("Ini method private");
    }

    protected void methodProtected(){
        System.out.println("Ini method protected");
    }
}
```

Listing Program 9: SubKelas.java

```
public class SubKelas extends SuperKlas{
    private double nilaiSub;

    public SubKelas(int nilaiSuper, double nilaiSub) {
        super(nilaiSuper);
        this.nilaiSub = nilaiSub;
    }
}
```

Hal yang perlu diperhatikan, subclass dapat memanggil method dari superclass. Contoh modifikasi SubKelas:

Listing Program 10: SubKelas.java Modifikasi Penggunaan "super"

```
public class SubKelas extends SuperKlas{
    private double nilaiSub;

    public SubKelas(int nilaiSuper, double nilaiSub) {
        super(nilaiSuper);
        this.nilaiSub = nilaiSub;
    }

    public void methodSub(){
        super.methodProtected();
        System.out.println("Nilai Super: "+super.getNilaiSuper());
    }
}
```

Dengan jalannya pada Main class sebagai berikut:

Listing Program 11: Main.java Konsep Pewarisan

```
public class Main {
    public static void main(String[] args) {
        //bentuk objek superclass
        System.out.println("Objek superclass");
        SuperKlas sup = new SuperKlas(5);
        System.out.println("Nilai super: "+sup.getNilaiSuper());
        sup.methodProtected();

        //bentuk objek subclass
        System.out.println("\nObjek subclass");
        SubKelas sub = new SubKelas(10, 9.5);
        System.out.println("Pemanggilan method superclass dari objek subclass");
        System.out.println("Nilai super: "+sub.getNilaiSuper());
        sub.methodProtected();
        System.out.println("Pemanggilan method superclass dari sublcass");
        sub.methodSub();
    }
}
```

Berdasarkan kode di atas, terdapat hal yang harus diperhatikan:

1. Konstruktor subclass harus mengambil konstruktor dari super class dengan keyword super (lihat kode pada konstruktor subclass). Kasus berbeda akan terjadi jika super class tidak memiliki konstruktor eksplisit yang dituliskan. Sebagai informasi tambahan, kode akan error jika urutan kode pada konstruktor SubKlas dibalik penulisannya (pemanggilan super di bawah set nilai sub). Hal ini sesuai dengan sifat instansiasi pada subbab sebelumnya.
2. Method private tidak dapat dipanggil melalui main karena sifatnya yang private.
3. Method protected dan public dapat dipanggil dari class sub dengan keyword super atau this (lihat isi dari methodSub).
4. Method protected dan public dapat dipanggil melalui objek subclass walaupun tidak terdapat penulisan method tersebut pada class-nya (subclass). Hal ini juga berlaku bagi atribut.

3.3.4 Overriding

Pada konsep object oriented, dikenal istilah polimorfisme (banyak bentuk). Polimorfisme terbagi ke dalam 2 jenis: overloading dan overriding. Contoh overloading dapat dilihat pada Modul 1. Sedangkan overriding dapat terjadi ketika konsep pewarisan diterapkan. Overriding, sama seperti overloading, merupakan penulisan kembali sebuah method dengan algoritma (body method) yang berbeda.

Overriding terjadi di 2 class yang berbeda, dengan suatu class merupakan subclass dari class yang lainnya. Selain itu, terdapat suatu syarat lain, yaitu nama method dan parameter dari method tersebut harus sama. Hal ini berbeda dengan overloading yang memiliki sifat berlawanan dengan overriding (terjadi di class yang sama dengan nama method sama, tapi parameter berbeda). Contoh overriding terdapat pada modifikasi SubKelas berikut:

Listing Program 12: SubKelas.java Modifikasi Overriding

```
public class SubKelas extends SuperKlas{
    private double nilaiSub;

    public SubKelas(int nilaiSuper, double nilaiSub) {
        super(nilaiSuper);
        this.nilaiSub = nilaiSub;
    }

    public void methodSub(){
        super.methodProtected();
        System.out.println("Nilai Super: "+super.getNilaiSuper());
    }

    @Override
    protected void methodProtected() {
        System.out.println("Ini method protected dari SubKelas! Bukan SuperKlas");
    }

}
```

Overriding dimaksudkan agar class yang lebih spesifik (subclass) dapat menyesuaikan algoritma sesuai kebutuhannya. Contoh, ketika terdapat sebuah class "BangunDatar" dengan method "hitungLuas" yang memiliki perhitungan luas "panjang x lebar", maka sebuah "Segitiga" yang merupakan bagian dari "BangunDatar" harus menyesuaikan bentuk perhitungan luasnya. Karena perhitungan luas dari segitiga bukan "panjang x lebar", tapi "1/2 alas x tinggi". Inti dari overriding dan pewarisan adalah kemampuan untuk menggunakan algoritma dari sebuah class (reusable) dengan sifat dimungkinkan untuk mengubah algoritma menjadi lebih spesifik sesuai dengan yang dibutuhkan.

3.3.5 Virtual Method Invocation

Ada kalanya, tipe objek yang dibuat tidak sesuai dengan konstruktor yang diakses. Ketika hal ini terjadi, maka konsep yang digunakan adalah virtual method invocation. Contoh:

Listing Program 13: Virtual Method Invocation

```
public class KelasSatu {
    int i = 10;
    public KelasSatu(){
        System.out.println("Konstruktor Kelas Satu");
    }
    public void methodA(){
        System.out.println("Method A dari KelasSatu");
    }
}
```

```
public class KelasDua extends KelasSatu{
    int i=8;
    public KelasDua(){
        System.out.println("Kelas Dua");
    }
```

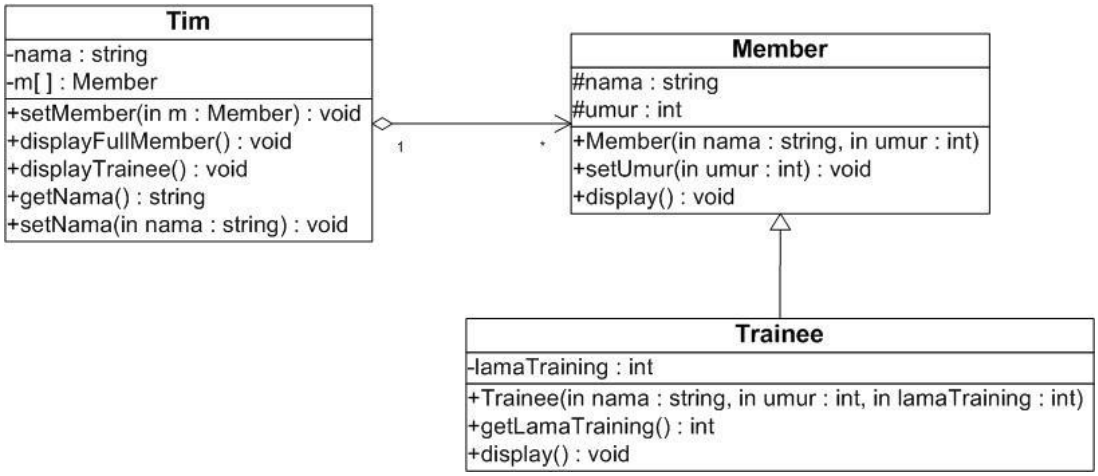
```
    @Override
    public void methodA(){
        System.out.println("MethodA dari KelasDua");
    }
}
```

```
public class MainKelas {
    public static void main(String[] args) {
        KelasSatu kd = new KelasDua();
        System.out.println("Nilai i: "+kd.i);
        kd.methodA();
    }
}
```

Perhatikan cara membentuk objek “kd”. Tipe objek yang digunakan adalah superclass, sedangkan konstruktor yang diakses adalah subclass. Ketika instansiasi dilakukan, objek akan terbentuk. Akan tetapi, ketika pemanggilan atribut, nilai atribut superclass akan menutupi nilai subclass, sedangkan untuk method akan mengikuti aturan dari subclass. Konsep ini berfungsi pada saat menjalankan real time scenario dari programming. Selain itu, konsep ini dimungkinkan terjadi karena adanya polimorfisme dari konsep *object oriented programming*.

3.4 Latihan

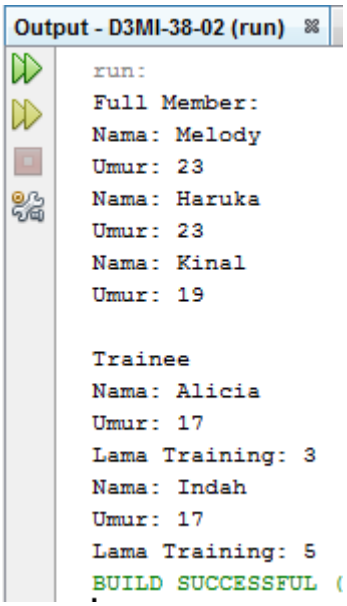
Kodekanlah class diagram dengan hubungan berikut di bawah bimbingan asisten/dosen praktikum.



Class Main dituliskan sebagai berikut:

```
public class Main {
    public static void main(String[] args) {
        Tim t = new Tim();
        t.setNama("Tim T");

        Member m1 = new Member("Melody", 23);
        Member m2 = new Member("Haruka", 23);
        Member m3 = new Member("Kinal", 19);
        Trainee t1 = new Trainee("Alicia", 17,
3);
        Trainee t2 = new Trainee("Indah", 17, 5);

        t.setMember(m1);
        t.setMember(m2);
        t.setMember(m3);
        t.setMember(t1);
        t.setMember(t2);

        System.out.println("Full Member:");
        t.displayFullMember();
        System.out.println("\nTrainee");
        t.displayTrainee();
    }
}
```

Hasil Akhir:



Keterangan: Method display pada class Member dan Trainee akan memperlihatkan nilai atribut dari masing-masing class. Method displayFullMember untuk menampilkan semua member yang tergabung dalam tim tertentu. Method displayTrainee untuk menampilkan semua trainee pada tim tertentu.

Modul 4 : Abstract

4.1 Tujuan

Setelah mengikuti praktikum ini mahasiswa diharapkan dapat:

1. Memahami bentuk special konsep hubungan antar class Inheritance/Pewarisan: Abstract class.
2. Dapat mengimplementasikan abstract class menggunakan bahasa pemrograman Java.

4.2 Alat & Bahan

Alat & Bahan Yang digunakan adalah hardware perangkat PC beserta Kelengkapannya berjumlah 40 PC dengan Java SDK dan editor IDE (disarankan Netbeans) telah terinstall di masing-masing unit.

4.3 Dasar Teori

4.3.1 Abstract Class

Kelas abstrak merupakan suatu bentuk khusus dari kelas di mana kelas tersebut tidak dapat diinstansiasi (dibentuk objeknya) dan digunakan hanya untuk diturunkan ke dalam bentuk kelas konkret atau kelas abstrak berikutnya. Walaupun demikian, penulisan konstruktor masih diperbolehkan pada abstract class. Kelas abstrak dideklarasikan menggunakan keyword abstract.

Di dalam kelas abstrak dapat dideklarasikan atribut dan method sama seperti concrete class (class yang telah dipelajari sebelumnya). Akan tetapi pada abstract class, dapat terkandung juga abstract method. Abstract method merupakan sebuah method yang tidak mempunyai algoritma (body method). Abstract method hanya sebatas deklarasi saja pada abstract class. Method yang bersifat abstract akan memastikan dirinya di-override oleh subclass dari abstract class tersebut. Pada class diagram, abstract method dan abstract class dituliskan dengan huruf miring.

4.3.2 Implementasi Abstract Class

Contoh kode dari abstract class:

Listing Program 14: LivingThing.java

```
public abstract class LivingThing {

    public void breath() {
        System.out.println("Living Thing breathing...");
    }

    public void eat() {
        System.out.println("Living Thing eating...");
    }

    public abstract void walk(); //merupakan penulisan abstract method
}
```

Class yang meng-extends abstract class di atas, harus meng-override method walk(). Jika hal ini tidak dilakukan, akan terdapat error pada kode yang di-compile.

Listing Program 15: Human.java Menggunakan Abstract Class LivingThing

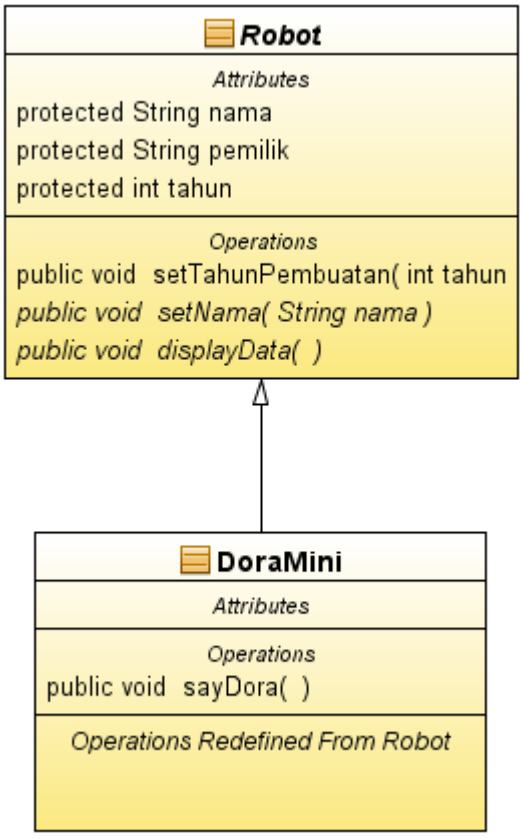
```
public class Human extends LivingThing {

    //implementasi dari method abstrak walk()
    @Override
    public void walk() {
        System.out.println("Human walks...");
    }
}
```

Hal yang perlu diperhatikan, setiap method abstract harus dituliskan pada abstract class. Akan tetapi, abstract class tidak harus mengandung abstract method. Sebagai informasi tambahan, abstract digunakan untuk memenuhi salah satu prinsip paradigm berorientasi objek: open-closed principle. Selain itu digunakan juga pada beberapa design pattern dari paradigm tersebut. Prinsip tersebut tidak dibahas pada mata kuliah PBO.

4.4 Latihan

Latihan 1

Diketahui diagram kelas sebagai berikut:



Tugas:

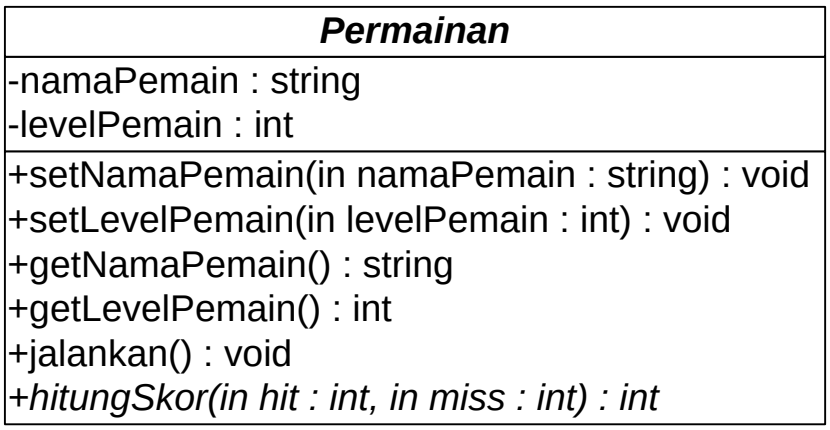
1. Ubahlah class diagram di atas ke dalam **bentuk standar UML class diagram**
2. Buatlah 2 kelas berdasarkan diagram kelas di atas (beserta kelas Main).

Keterangan tambahan:

a) Kelas Robot adalah kelas abstrak
b) Method "sayDora" akan menampilkan "Halo, Saya Dora Mini" di layar
c) Method displayData akan menampilkan setiap nilai dari atribut yang dimiliki ke layar
d) Method setNama dan displayData adalah method abstract

Latihan 2

Seorang analyst membuat aplikasi simulasi permainan. Analyst membuat class diagram dengan abstract class sebagai berikut:



Deskripsi:

a) Atribut namaPemain dan levelPemain menyimpan nama dan level pemain.
b) Nilai levelPemain berkisar 1-100 dengan ketentuan:
    a. 1-20: normal
    b. 21-80: medium
    c. 81-100: hard
c) Method jalankan() akan menjalankan skenario permainan (set nama dan level pemain, mengeluarkan data tersebut, dan menghitung skor pemain).
d) Method hitungSkor merupakan abstract method.

Tugas:

a) Buatlah kode berdasarkan abstract class di atas.
b) Buatlah 2 classes lain yang menggunakan abstract class di atas dengan deskripsi:
    a. "PermainanArcade" dengan aturan hitung skor: jumlah hit x 3 – jumlah miss x 1;
    b. "PermainanStrategy" dengan aturan hitung skor: jumlah hit x 5;

Perhatikan bahwa algoritma hitungSkor ditentukan oleh subclass, bukan superclass.

c) Buktikan bahwa abstract method memastikan bahwa method tersebut di-override oleh subclass.
d) Buktikan bahwa objek dari abstract class tidak dapat dibentuk.

Modul 5 : Interface

5.1 Tujuan

Setelah mengikuti praktikum ini mahasiswa diharapkan dapat:

1. Memahami bentuk special konsep hubungan antar class Inheritance/Pewarisan: Interface.
2. Dapat mengimplementasikan interface menggunakan bahasa pemrograman Java.

5.2 Alat & Bahan

Alat & Bahan Yang digunakan adalah hardware perangkat PC beserta Kelengkapannya berjumlah 40 PC dengan Java SDK dan editor IDE (disarankan Netbeans) telah terinstall di masing-masing unit.

5.3 Dasar Teori

5.3.1 Interface

Interface adalah prototype kelas yang berisi definisi konstanta dan deklarasi method (hanya nama method tanpa definisi kode programnya). Dalam sebuah interface:

a) Semua atribut adalah public dan final (semua atribut bertindak sebagai konstanta). Dapat juga menambahkan modifier static.
b) Semua method adalah abstract dan public
c) Tidak boleh ada deklarasi konstruktor

Interface digunakan untuk menyatakan spefisikasi fungsional beberapa kelas secara umum. Dengan adanya interface, Java menyediakan sebuah fitur untuk keperluan pewarisan jamak (Multiple inheritance). Pada konsep paradigm pemrograman berorientasi objek, interface dimaksudkan untuk memenuhi open/closed principle. Hal tersebut tidak dibahas di mata kuliah Pemrograman Berorientasi Objek.

Hubungan penggunaan interface dari sebuah class dilambangkan dengan garis putus-putus dengan segitiga terdapat di salah satu garis tersebut. Interface melekat ke segitiga dari simbol hubungan tersebut. Deklarasi dari interface menggunakan keyword “interface” sebagai pengganti “class”. Class yang menggunakan interface menggunakan keyword “implements”.

5.3.2 Contoh Implementasi Interface

Listing Program 16: Interface Relation.java

```
public interface Relation {
    public boolean isGreater(Object a, Object b);
    public boolean isLess(Object a, Object b);
    public boolean isEqual(Object a, Object b);
}
```

Perhatikan bahwa interface hanya memiliki method tanpa body method. Hal ini berbeda dengan abstract class yang masih memungkinkan memiliki concrete method (method dengan body method).

Listing Program 17: Line.java Menggunakan Interface Relation

```
public class Line implements Relation {

    private double x1;
    private double x2;
    private double y1;
    private double y2;



    public Line(double x1, double x2, double y1, double y2) {
        this.x1 = x1;
        this.x2 = x2;
        this.y1 = y1;
        this.y2 = y2;
    }

    public double getLength() {
        double length = Math.sqrt((x2 - x1) * (x2 - x1) + (y2 - y1) * (y2 - y1));
        return length;
    }

    public boolean isGreater(Object a, Object b) {
        double aLen = ((Line) a).getLength();
        double bLen = ((Line) b).getLength();
        return (aLen > bLen);
    }

    public boolean isLess(Object a, Object b) {
        double aLen = ((Line) a).getLength();
        double bLen = ((Line) b).getLength();
        return (aLen < bLen);
    }

    public boolean isEqual(Object a, Object b) {
        double aLen = ((Line) a).getLength();
        double bLen = ((Line) b).getLength();
        return (aLen == bLen);
    }
}
```

Perhatikan bahwa class Line meng-implements interface Relation. Hal ini menyebabkan setiap method yang dituliskan pada interface Relation harus di-override di class Line.

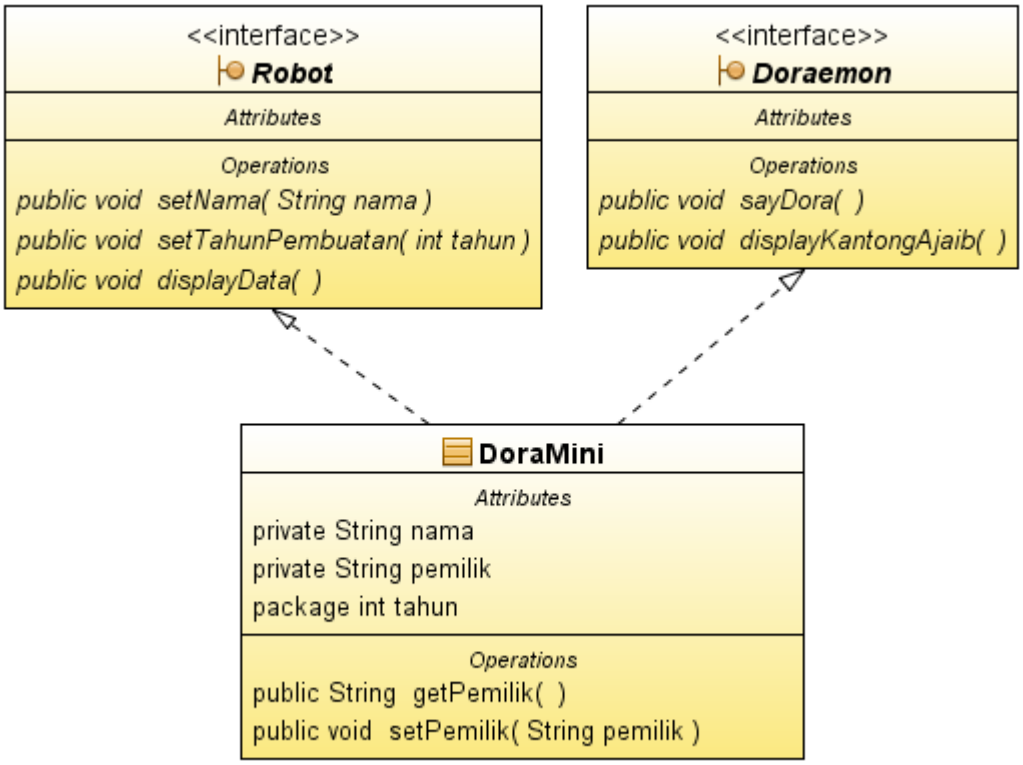
Salah satu bentuk penggunaan dari Interface yaitu pada pemrograman GUI Java. Pemberian aksi pada komponen swing (tombol atau yang lain) harus dilakukan dengan meng-implements salah satu Listener yang tersedia.

5.4 Latihan

Latihan 1
Tulislah dan jelaskan perbedaan antara abstract class dan interface

Latihan 2

Diketahui class diagram sebagai berikut:



Keterangan:

a) Method “sayDora” akan menampilkan “Halo, Saya Dora Mini” di layar
b) Method “dispKantungAjaib” akan menampilkan “Saya juga seperti Doraemon yang memiliki kantung ajaib”
c) Method displayData akan menampilkan setiap nilai dari atribut yang dimiliki ke layar

Tugas: Buatlah 3 kelas berdasarkan diagram kelas di atas (dan juga class Main untuk membentuk objeknya).

Modul 6 : Swing Component

6.1 Tujuan

Setelah mengikuti praktikum ini mahasiswa diharapkan dapat:

1. Mampu membuat aplikasi sederhana menggunakan komponen swing.
2. Mampu menggunakan Matisse Builder sebagai editor GUI pada pemrograman visual Java.
3. Mampu menambahkan aksi pada aplikasi yang dibuat.

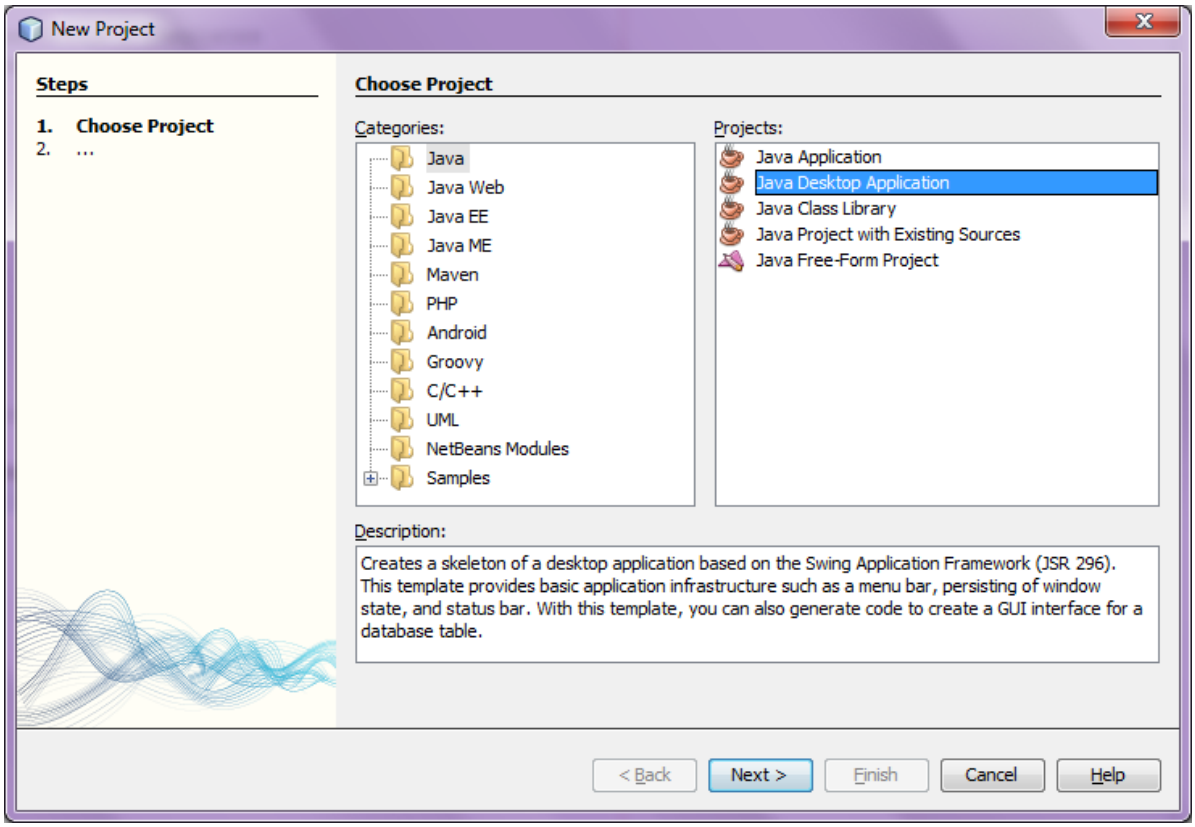
6.2 Alat & Bahan

Alat & Bahan Yang digunakan adalah hardware perangkat PC beserta Kelengkapannya berjumlah 40 PC dengan Java SDK dan editor IDE (disarankan Netbeans) telah terinstall di masing-masing unit.

6.3 Dasar Teori

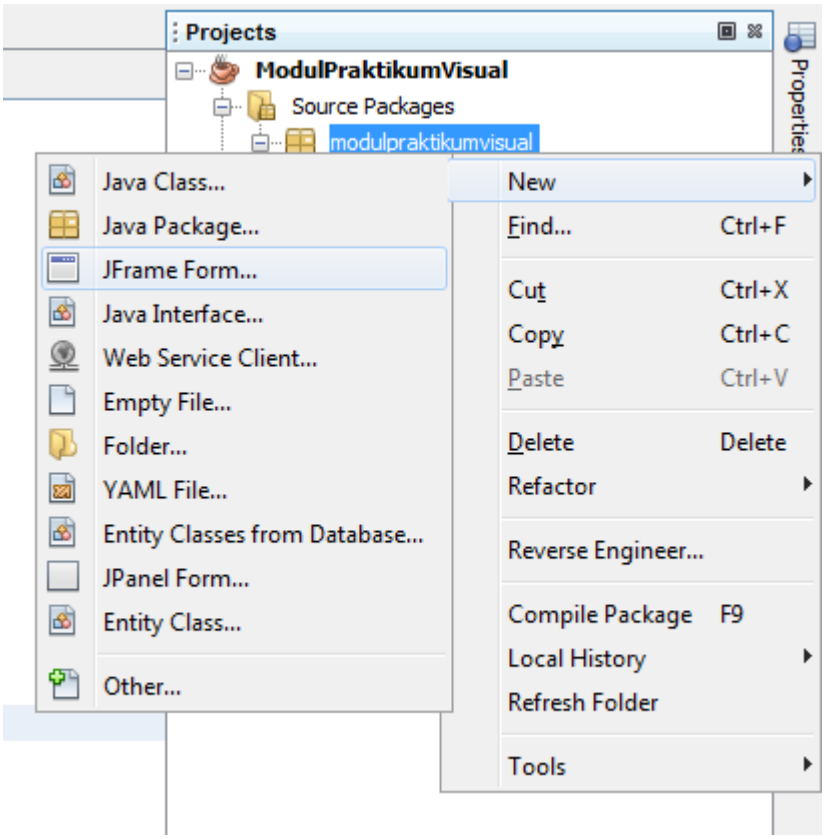
Netbeans memiliki sebuah project bernama Project Matisse untuk membuat sebuah builder GUI dengan basis swing menggunakan bahasa pemrograman java. Swing GUI builder ini membantu para programmer untuk membangun sebuah aplikasi desktop karena dapat membangun GUI secara visual dan bukan hanya sekedar text-based code. Dengan melakukan drag-and-drop komponen swing ke top level container-nya, sebuah aplikasi gui menggunakan bahasa java sudah dapat dibangun.

Untuk menggunakan matisse builder dapat memilih aplikasi “Java Desktop Application” pada saat pembuatan project baru.



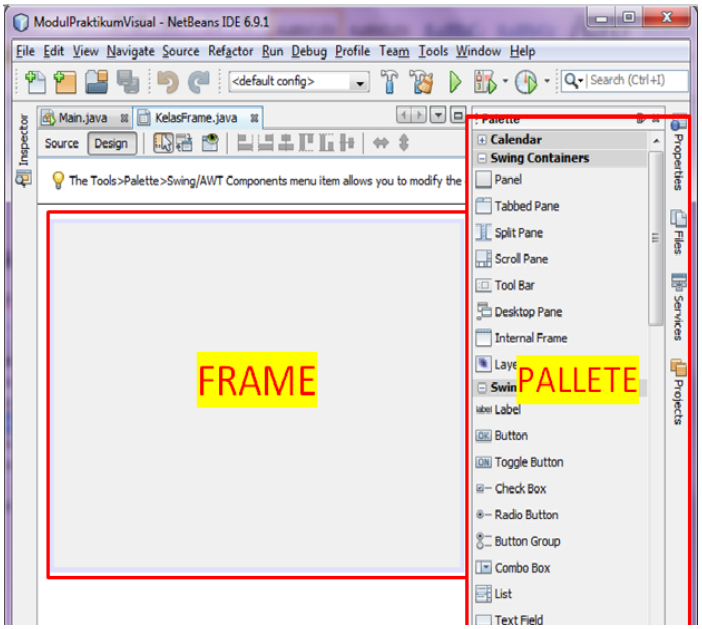
Gambar 6: Java Desktop Application

Atau dapat memilih “Java Application” dan menambahkan “JFrame Form” pada project tersebut



Gambar 7: Java Application

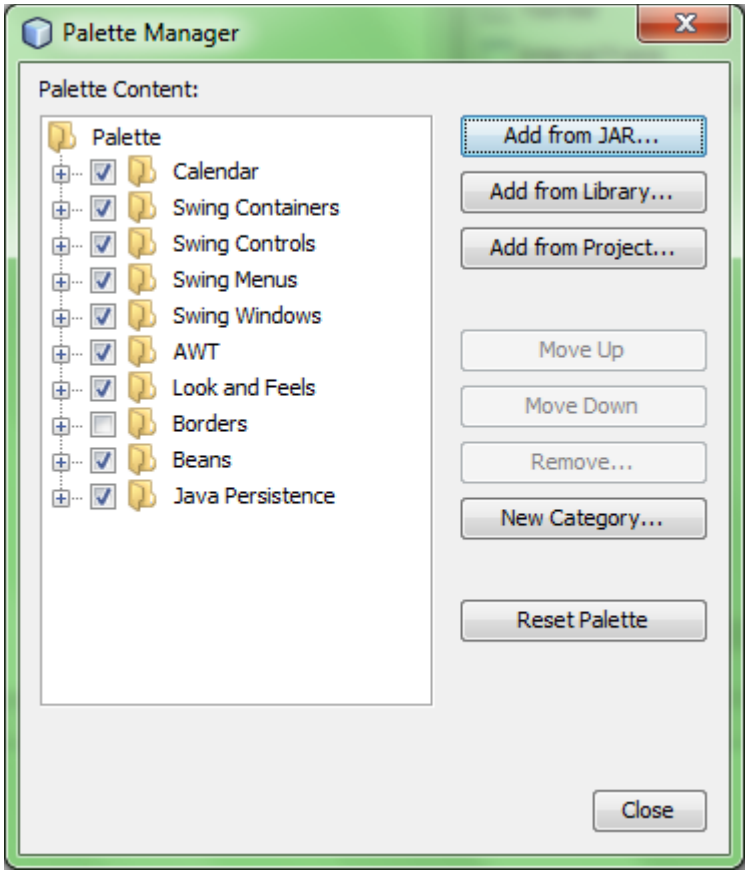
Hal ini akan menciptakan satu kelas java yang meng-extends JFrame yang merupakan container tertinggi dari sebuah aplikasi java. Matisse Builder/GUI editor ini terdiri dari beberapa bagian, 2 diantaranya yaitu bagian frame dan palette.



Gambar 8: Frame & Palette Matisse Builder

Bagian frame merupakan sebuah class java yang meng-extends class dari komponen swing, yaitu JFrame. JFrame merupakan top-level-container pada paket swing. Bagian frame ini layaknya sebuah kanvas yang dapat diisi komponen lain dari paket swing, container ataupun komponen umum gui seperti button, textfield dan lainnya. Palette merupakan tempat peletakan komponen swing yang bisa ditambahkan ke sebuah frame. Penambahannya dilakukan dengan cara drag-and-drop.

Sedangkan, palette merupakan bagian dari matisse builder yang berisikan komponen swing/awt (komponen pembentuk gui menggunakan bahasa java). Komponen ini bisa ditambahkan pada bagian frame yang telah dibentuk sebelumnya. Palette dapat ditambahkan komponennya dari luar yang dibangun oleh pihak ketiga (third-party). Caranya, klik kanan di bagian palette, pilih palette manager. Klik "Add From Jar" untuk menambahkan library yang dibangun oleh pihak ketiga. Contoh penggunaannya pada saat menambahkan date picker ke komponen palette. Atau menambahkan komponen swing yang belum tercantum ke default palette (contohnya, Border).



Gambar 9: Palette Manager

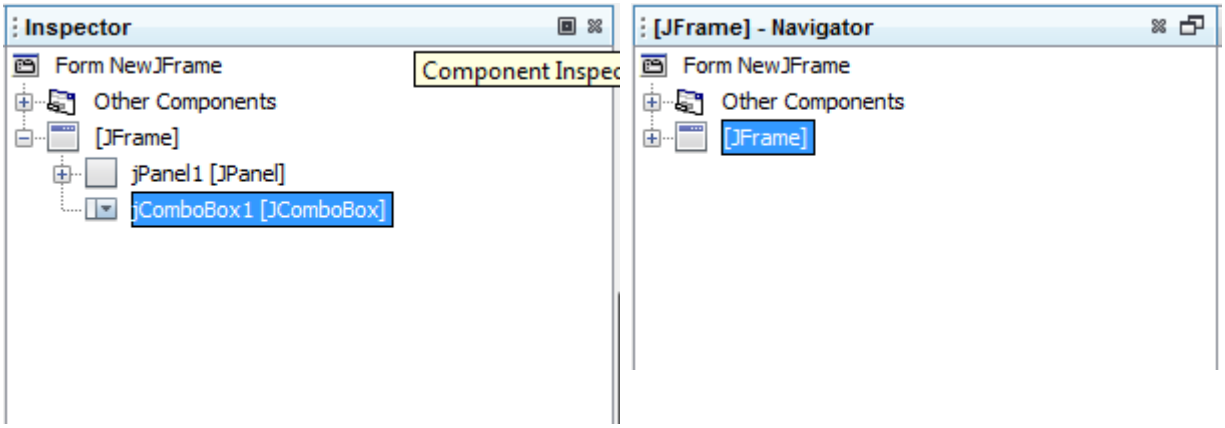
Palette Default terbagi ke dalam beberapa kategori, beberapa diantaranya adalah sebagai berikut:

a) Swing container: merupakan container pada pemrograman gui menggunakan bahasa java. Biasa digunakan untuk windowing (cara/bentuk menampilan aplikasi ke user)
b) Swing control: kategori yang menyimpan komponen swing yang penting seperti label untuk membuat tulisan, button untuk membuat tombol, combo box untuk menambahkan menu pull/drop down, dan lainnya

c) Swing menu: kategori untuk menambahkan menu yang terdapat pada bagian atas suatu window/frame. Menu memiliki hirarki tersendiri. Hirarki menu biasanya dituliskan sebagai "*Menu Bar → Menu → Menu Item | Menu Check Box | Menu Radio Button*". Pop-up menu digunakan untuk menambahkan menu "klik-kanan".

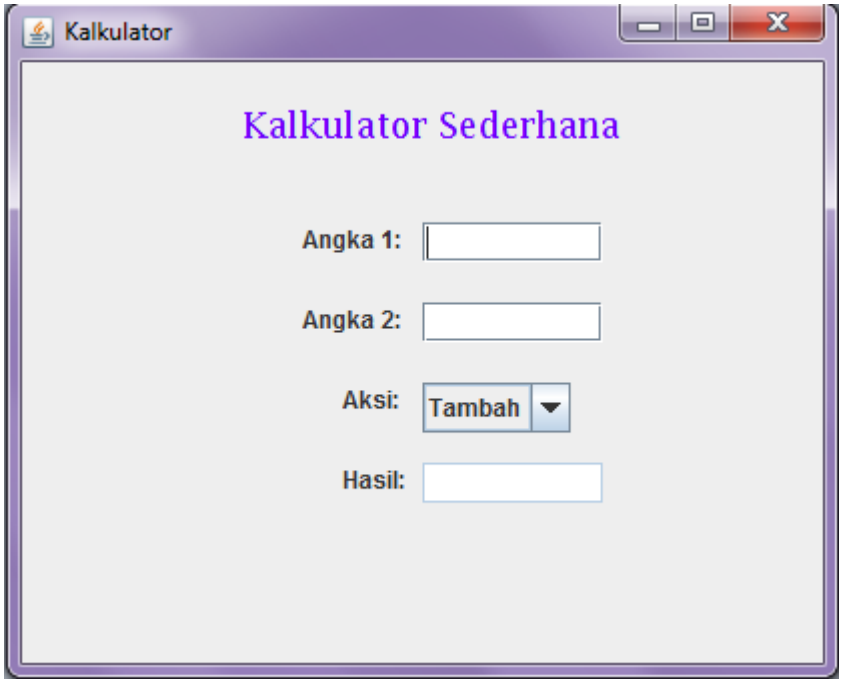
Menambahkan komponen pada matisse builder dilakukan dengan melakukan penarikan komponen yang ada di palette ke bagian frame. Pengaturan peletakan komponen di frame juga dilakukan secara visual. Saat melakukan drag-and-drop, sebuah objek dari kelas tertentu dari package swing akan dibentuk dan ditambahkan ke frame. Untuk beberapa komponen, terdapat modifikasi agar dapat digunakan sesuai keinginan. Hal-hal yang bisa diubah terdapat di window properties.

Untuk melihat susunan hirarki gui, bisa dilihat di window inspector (Netbeans 6.9.x ke bawah) atau di window navigator (Netbeans 7.0 ke atas).



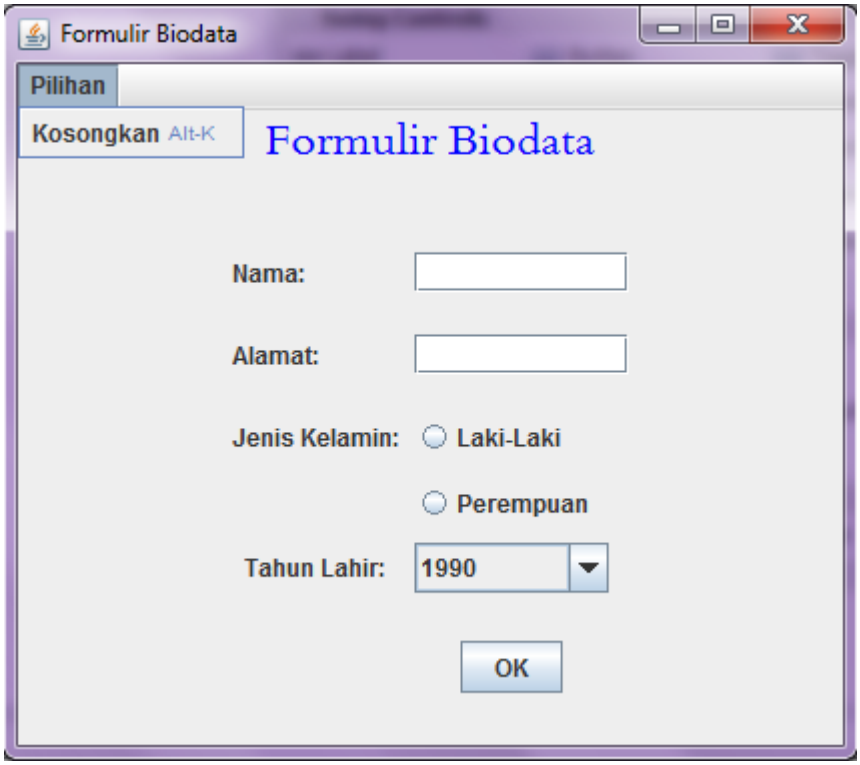
Gambar 10: Inspector & Navigator

6.4 Latihan

Latihan 1
Buatlah sebuah GUI dengan mockup tampilan sebagai berikut:    Keterangan: a) Masukkan komponen dalam panel; Jangan memasukkan langsung komponen dalam frame b) Gunakan Absolute Layout c) Field hasil tidak dapat diubah dan berwana biru d) Hasil memiliki 2 angka belakang koma e) Frame muncul di tengah screen f) Setelah memilih drop down, hasil akan langsung terisi dan menampilkan 2 angka di belakang koma

Latihan 2

Buatlah sebuah GUI dengan mockup tampilan sebagai berikut:



Keterangan:

a) Masukkan komponen dalam panel; Jangan memasukkan langsung komponen dalam frame
b) Gunakan Gridbag Layout
c) Terdapat menu dengan shortcut sesuai huruf pertama masing-masing menu ditambah ALT.
d) Untuk pilihan di atas, setelah menekan ALT+K, akan menampilkan tampilan kosong semua field pada formulir biodata
e) Gunakan Mnemonic pada tombol dengan huruf pertama pada tombol tersebut
f) Frame muncul di tengah screen
g) Setelah menekan “OK” muncul option dialog yang menampilkan semua input
h) Jika ada yang salah satu textfield yang kosong, maka program menampilkan pesan kesalahan

Modul 7 : JTable

7.1 Tujuan

Setelah mengikuti praktikum ini mahasiswa diharapkan dapat:

1. Mampu membuat aplikasi sederhana menggunakan komponen swing.
2. Mampu menggunakan Matisse Builder sebagai editor GUI pada pemrograman visual Java.
3. Mampu menambahkan aksi pada aplikasi yang dibuat.

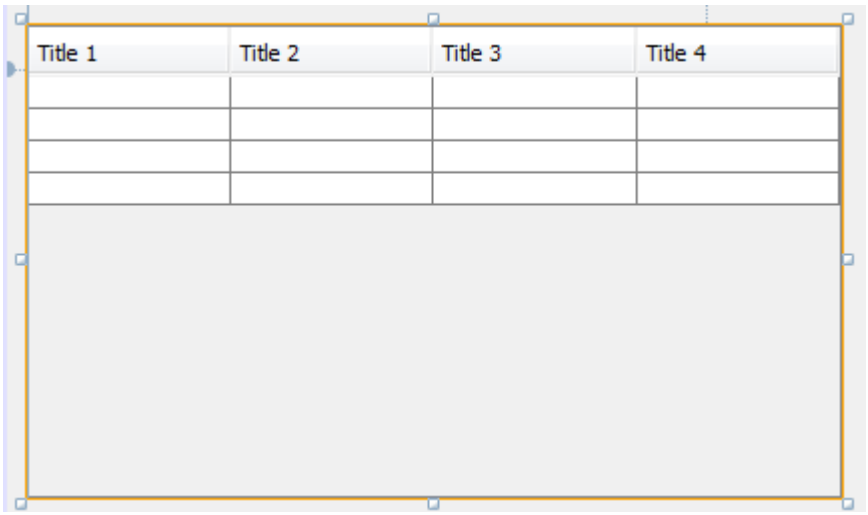
7.2 Alat & Bahan

Alat & Bahan Yang digunakan adalah hardware perangkat PC beserta Kelengkapannya berjumlah 40 PC dengan Java SDK dan editor IDE (disarankan Netbeans) telah terinstall di masing-masing unit.

7.3 Dasar Teori

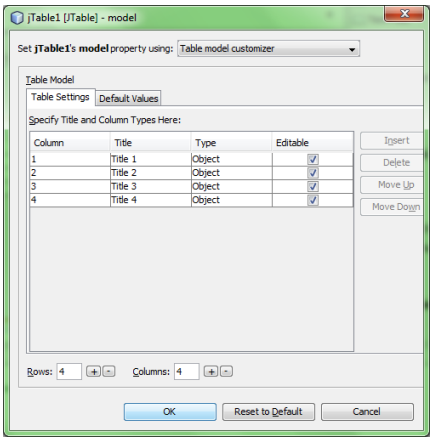
7.3.1 JTable

Sama seperti komponen swing lain, table dibentuk object-nya dari class swing JTable dengan default tampilan sebagai berikut:



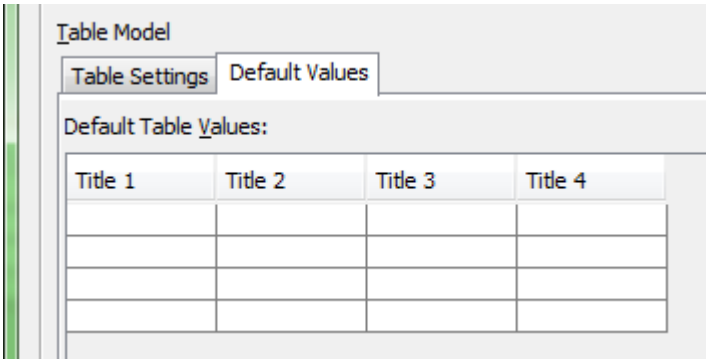
Gambar 11: Layout JTable

Jumlah kolom yang ditampilkan, serta title dapat diubah di bagian model pada properties table.



Gambar 12: Konfigurasi Obyek JTable

Rows merupakan jumlah baris yang ditampilkan (biasanya berisi data). Pengubahan data pada rows dapat dilakukan pada tab “default values”.



Gambar 13: Konfigurasi Konten Obyek JTable

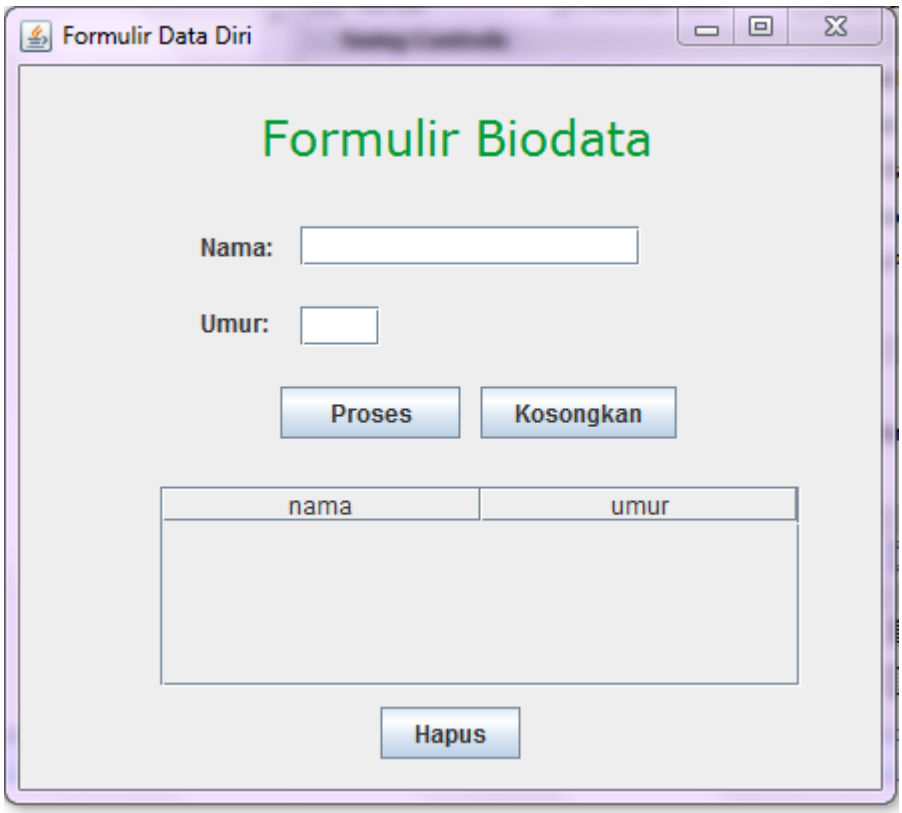
7.3.2 Model

JTable merupakan tempat peletakan dan cara menampilkan data, sedangkan tempat untuk meletakkan data yang ditampilkan terdapat pada model tabel, salah satunya DefaultTableModel. Terdapat beberapa method dari DefaultTableModel:

a) addColumn: method ini berfungsi untuk menambah header dari table. Terdapat beberapa versi dari parameter masukan (overloading), tapi yang paling sering digunakan nantinya adalah addCloumn(Object columnName).
b) addRow: method ini berfungsi untuk menambah isi dari sebuah data pada model. Parameter masukan dapat berupa Vector atau Array of Object. Jenis kedua akan sering digunakan nantinya.
c) setValueAt(x,y,z): mengeset nilai “x” pada baris y dan kolom z.
d) getValueAt(x,y): mengambil nilai dari model pada baris x dan kolom y.
e) setRowCount(x): melakukan set nilai baris menjadi 0 pada model yang memanggil method ini.
f) setColumnCount(x): melakukan set nilai kolom menjadi 0 pada model yang memanggil method ini.

7.4 Latihan

Latihan 1



Keterangan:

a) Masukkan komponen dalam panel; Jangan memasukkan langsung komponen dalam frame
b) Gunakan Absolute Layout
c) Terdapat 2 input
d) Frame muncul di tengah screen
e) Setelah menekan “OK” data input muncul di tabel
f) Tabel dapat ditambah dan dihapus tapi tidak dapat diubah
g) Jika ada yang salah satu textfield yang kosong, maka program menampilkan pesan kesalahan

Modul 8 : Layout

8.1 Tujuan

Setelah mengikuti praktikum ini mahasiswa diharapkan dapat:

1. Mampu membuat aplikasi sederhana menggunakan komponen swing.
2. Mampu menggunakan Matisse Builder sebagai editor GUI pada pemrograman visual Java.
3. Mampu menambahkan aksi pada aplikasi yang dibuat.

8.2 Alat & Bahan

Alat & Bahan Yang digunakan adalah hardware perangkat PC beserta Kelengkapannya berjumlah 40 PC dengan Java SDK dan editor IDE (disarankan Netbeans) telah terinstall di masing-masing unit.

8.3 Dasar Teori

8.3.1 Layouting

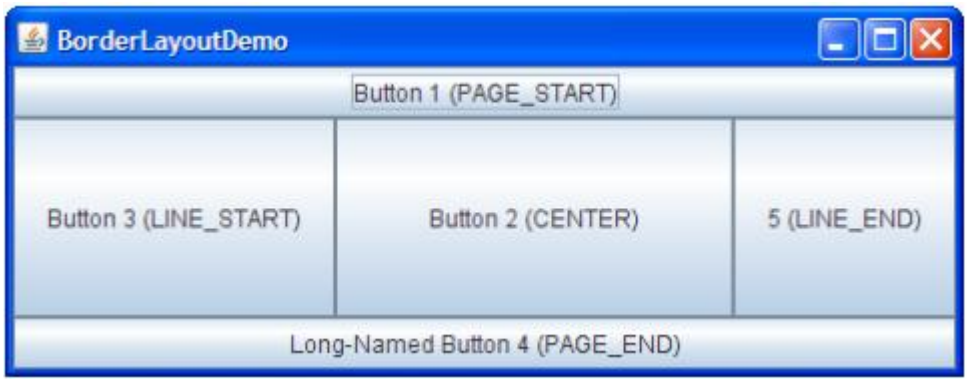
Peletakan komponen pada JFrame mengikuti layout tertentu. Pada drag & drop dari Matisse Builder, layout yang digunakan adalah GroupLayout dengan pengaturan ditekankan pada penambahan gap ataupun container gap. Pada pengkodean awal sebelum terdapatnya builder, pengaturan letak dapat menggunakan beberapa layout seperti BorderLayout, FlowLayout, GridLayout, GridBagLaout, dan lainnya.

8.3.2 BorderLayout

Layout yang memungkinkan komponen hanya dapat diletakkan di 5 area saja:

- PAGE_START
- PAGE_END
- LINE_START
- LINE_END
- CENTER

Ilustrasi:

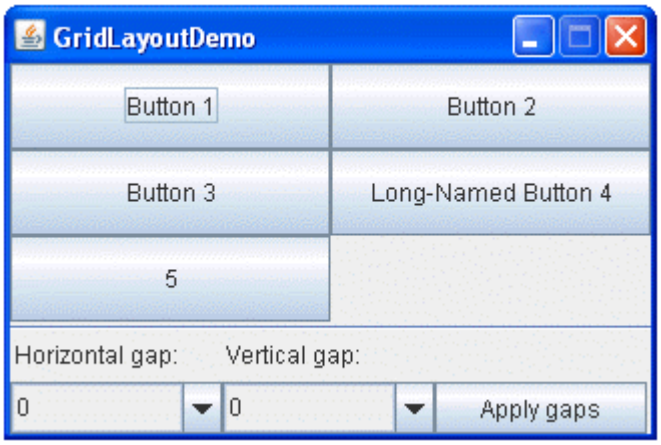


Gambar 14: BorderLayout

8.3.3 GridLayout

Grid layout menempatkan objek komponen berdasarkan grid cell. Dengan setiap cell memiliki ukuran yang sama.

Ilustrasi:



Gambar 15: GridLayout

Terdapat beberapa method yang digunakan pada layout ini:

Method	Fungsi
setRows(3)	Menge-set jumlah maksimal baris dari peletakan komponen.
setColumns(4)	Menge-set jumlah maksimal kolom dari peletakan komponen. Orientasi setRows lebih besar dibandingkan setColumns.
setHgap(30)	Menge-set jarak antar komponen secara horizontal
setVgap(30)	Menge-set jarak antar komponen secara vertikal

Jumlah baris dan kolom maksimum pada layout juga dapat di-set via parameter konstruktor. Parameter pertama adalah nilai baris maksimal, parameter kedua merupakan pengaturan nilai kolom maksimal.

8.3.4 GridBagLayout

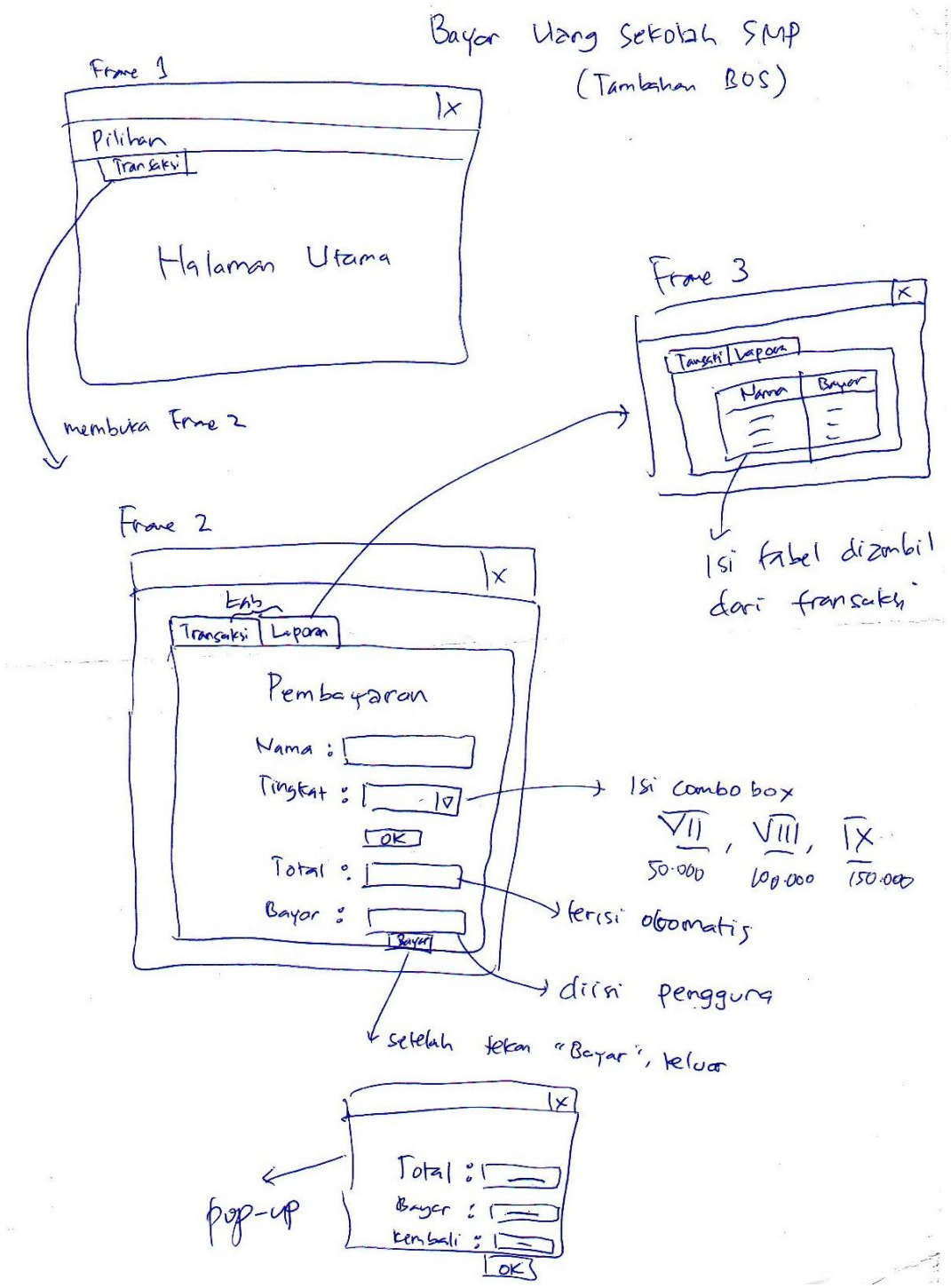
Merupakan layout yang paling sering digunakan programmer java untuk pengaturan peletakan komponen objek swing karena fleksibilitas yang ditawarkan. Sama seperti grid layout, membagi peletakan dalam grid cells. Hanya saja ukuran dari setiap komponen dapat berbeda. Contohnya, saat melakukan peletakan komponen, komponen tersebut dapat memakai 2 grid secara horizontal dan 3 grid vertical.

Ilustrasi:

Gambar 16: GridBagLayout

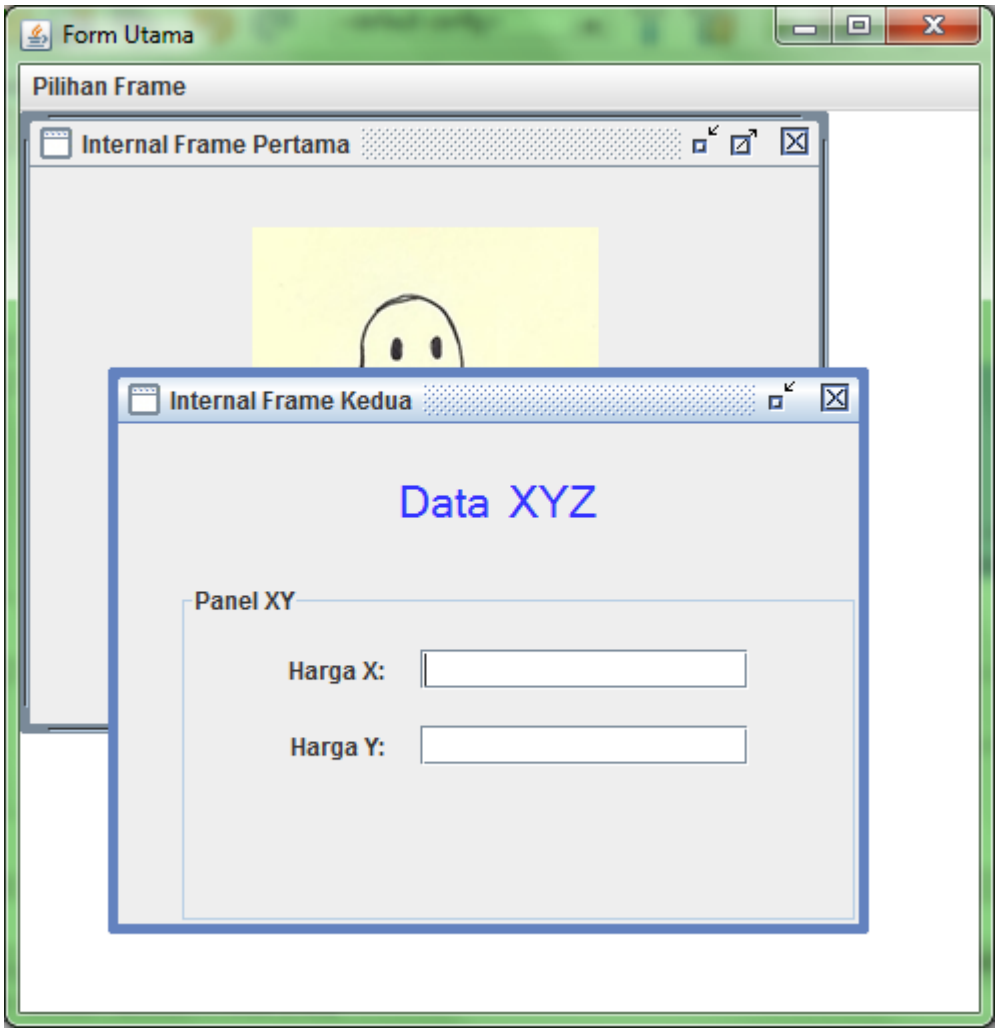
8.4 Latihan

Latihan 1
Buatlah sebuah GUI dengan window jenis tabbed pane seperti mockup tampilan di bawah ini.



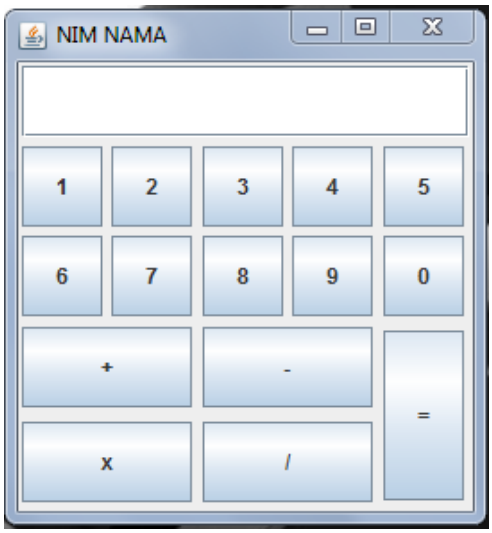
Latihan 2

Buatlah beberapa form menggunakan JInternalFrame dengan mockup seperti yang ditampilkan di bawah ini:



Latihan 3

Buatlah menu kalkulator dengan acuan tampilan sebagai berikut:



Gunakan GridBag Layout dan Border Layout. Tombol tidak harus semuanya diberikan action. Input user masih diperbolehkan jika hanya bisa input dengan skenario:

a) Tekan angka pertama sebagai operan pertama
b) Tekan operator sebagai operasi kedua operan
c) Tekan angka kedua sebagai operan kedua

Modul 9 : Akses Database - 1

9.1 Tujuan

Setelah mengikuti praktikum ini mahasiswa diharapkan dapat:

1. Mampu membuat aplikasi sederhana menggunakan akses database.
2. Mampu menerapkan query DML & SELECT pada basis data Access menggunakan bahasa pemrograman Java.

9.2 Alat & Bahan

Alat & Bahan Yang digunakan adalah hardware perangkat PC beserta Kelengkapannya berjumlah 40 PC dengan Java SDK dan editor IDE (disarankan Netbeans) telah terinstall di masing-masing unit.

9.3 Dasar Teori

9.3.1 JDBC

JDBC dibutuhkan untuk menghubungkan bahasa pemrograman java dengan database. JDBC merupakan singkatan dari Java DataBase Connectivity. JDBC merupakan driver untuk mengakses database. Analoginya seperti driver printer untuk menggunakan sebuah printer melalui computer. Driver JDBC sendiri merupakan koleksi class-class Java yang dikumpulkan dalam satu atau beberapa file .jar. JDBC yang digunakan berbeda-beda utuk setiap database yang digunakan.

9.3.2 ODBC

ODBC merupakan singkatan dari Open Database Connectivity. Merupakan API yang digunakan untuk mengakses database management system (DBMS). Fungsi dari ODBC hampir sama dengan JDBC. Perbedaannya yang mendasar, ODBC merupakan open interface yang dapat digunakan aplikasi untuk berkomunikasi dengan DBMS, sedangkan JDBC dikhususkan hanya dengan bahasa Java.

9.3.3 UCanAccess

UCanAccess merupakan open-source Java JDBC driver yang memungkinkan Java developers and JDBC client programs (seperti DBeaver, NetBeans, SQLeo, OpenOffice Base, LibreOffice Base, Squirrel SQL) untuk membaca dan memanipulasi data dari Microsoft Access databases. Keterangan lebih lanjut dapat dilihat pada homepage-nya: http://ucanaccess.sourceforge.net/site.html

9.3.4 Eksekusi Query

Pengaksesan database pasti tidak lepas dari query yang akan dikirimkan agar dieksekusi. Hal ini membutuhkan sebuah objek dengan tipe Statement untuk membangun dan men-submit SQL statement ke database. Terdapat 3 interface yang biasa digunakan:

a) Statement: merupakan general-purpose pengaksesan database. Digunakan untuk mengeksekusi query yang static seperti “select * from nama_tabel”. Interface ini tidak menerima parameter.
b) PreparedStatement: Digunakan untuk mengeksekusi query dinamis dan memiliki input parameter. Jika query static harus dieksekusi berulang kali, penggunaan preparedStatement akan lebih efektif dibandingkan Statement.
c) CallableStatement: berfungsi untuk mengakses stored procedure dari database (procedure, function dll)

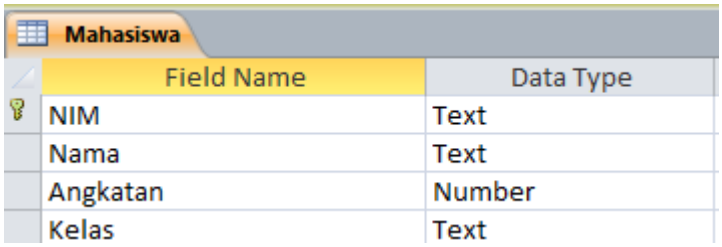
Untuk mengeksekusi query, terdapat 3 method yang digunakan (terdapat pada objek bertipe Statement dan **PreparedStatement**):

a) boolean execute(String SQL) : mengembalikan nilai true jika objek ResultSet dapat diambil dan mengembalikan nilai false jika kondisi tersebut tidak terpenuhi; Method ini digunakan untuk mengeksekusi DDL statements atau ketika mengeksekusi truly dynamic SQL.
b) int executeUpdate(String SQL) : Mengembalikan jumlah baris yang terpengaruh oleh query yang dieksekusi. Penggunaannya ketika mengeksekusi query INSERT, UPDATE, atau DELETE statement.
c) ResultSet executeQuery(String SQL) : Mengembalikan objek ResultSet. Penggunaan method ini ketika diharapkan ada data yang diambil dari database menggunakan query select. Perbedaan dengan yang pertama, method tersebut hanya mengembalikan nilai true/false dan tidak mengembalikan datanya. Sedangkan method ini memiliki return value berupa data dari database.

9.3.5 Koneksi Java – Ms. Access

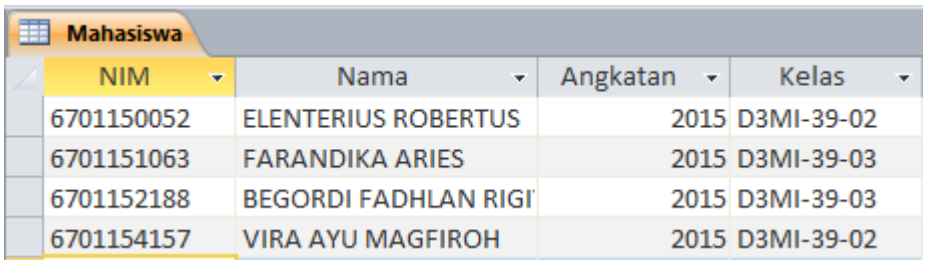
Pada modul ini, digunakan UCanAccess sebagai JDBC driver, karena modul ini menggunakan Java 8. Mulai Java 8, ODBC-JDBC bridge telah dihilangkan. Sebagai penggantinya, untuk akses koneksi menggunakan UCanAccess.

Buatlah sebuah tabel pada database (database1.accdb) menggunakan Ms. Access sebagai berikut:


Mahasiswa

Field Name	Data Type
NIM	Text
Nama	Text
Angkatan	Number
Kelas	Text

Isilah tabel Mahasiswa dengan contoh isian sebagai berikut:


Mahasiswa

NIM	Nama	Angkatan	Kelas
6701150052	ELENTERIUS ROBERTUS	2015	D3MI-39-02
6701151063	FARANDIKA ARIES	2015	D3MI-39-03
6701152188	BEGORDI FADHLAN RIGI	2015	D3MI-39-03
6701154157	VIRA AYU MAGFIROH	2015	D3MI-39-02

Langkah berikutnya, pada editor (IDE) yang digunakan, tambah library UCanAccess. Akan tetapi, driver ini membutuhkan dependensi berupa hsqldb, jackcess, commons-lang, dan commons-logging. Dependensi ini sudah terdapat pada paket archive dari UCanAccess.

Untuk melakukan SELECT, kodekan hal berikut pada Java class:

Listing Program 18: Akses DB - SELECT

```
import java.sql.*;

public class MsAcessAkses {

    public static void main(String[] args) {
        try {
            // membuat objek Connection & PreparedStatement
            String alamat = "jdbc:ucanaccess://E:/database/Database1.accdb";
            Connection con = DriverManager.getConnection(alamat);
            String kueri = "Select nim,nama,angkatan,kelas from Mahasiswa";
            PreparedStatement ps = con.prepareStatement(kueri);

            //eksekusi query kueri
            ResultSet rs = ps.executeQuery();

            //ekstrak data result set
            //sesuaikan angka dengan urutan data pada query
            while(rs.next()){
                System.out.println("NIM: "+rs.getString(1));
                System.out.println("Nama: "+rs.getString(2));
                System.out.println("Angkatan: "+rs.getString(3));
                System.out.println("Kelas: "+rs.getString(4));
                System.out.println();
            }
            ps.close();
            con.close();
        } catch (Exception ex) {
            System.err.print("Exception: ");
            System.err.println(ex);
        }
    }
}
```

Hasil:

```
Output - Modul PBO Gasal1617 (run)
run:
NIM: 6701150052
Nama: ELENTERIUS ROBERTUS
Angkatan: 2015
Kelas: D3MI-39-02

NIM: 6701151063
Nama: FARANDIKA ARIES
Angkatan: 2015
Kelas: D3MI-39-03

NIM: 6701152188
Nama: BEGORDI FADHLAN RIGIYAN
Angkatan: 2015
Kelas: D3MI-39-03

NIM: 6701154157
Nama: VIRA AYU MAGFIROH
Angkatan: 2015
Kelas: D3MI-39-02
```

Modul 10 : Akses Database - 2

10.1 Tujuan

Setelah mengikuti praktikum ini mahasiswa diharapkan dapat:

1. Mampu membuat aplikasi sederhana menggunakan akses database.
2. Mampu menerapkan query DML & SELECT pada basis data Access menggunakan bahasa pemrograman Java.

10.2 Alat & Bahan

Alat & Bahan Yang digunakan adalah hardware perangkat PC beserta Kelengkapannya berjumlah 40 PC dengan Java SDK dan editor IDE (disarankan Netbeans) telah terinstall di masing-masing unit.

10.3 Dasar Teori

10.3.1 DML Ms. Access – Java

DML merupakan singkatan dari Data Manipulation Language. Query tipe DML akan memanipulasi konten data dari database yang digunakan. Contoh query DML: INSERT, UPDATE, dan DELETE. Untuk mengeksekusi query DML, biasanya digunakan method executeUpdate dari interface Statement atau PreparedStatement (lihat modul 9).

10.3.2 Penambahan Data

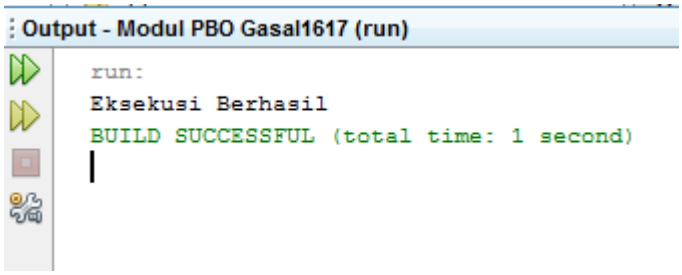
Contoh kode:

Listing Program 19: Akses DB - INSERT

```
import java.sql.*;

public class MsAccessTambah {
    public static void main(String[] args) {
        try {
            // membuat objek Connection & PreparedStatement
            String alamat = "jdbc:ucanaccess://E:/database/Database2.accdb";
            Connection con = DriverManager.getConnection(alamat);
            String kueri = "INSERT INTO Mahasiswa(NIM,Nama,Angkatan,Kelas)"
                           + "VALUES (?,?,?,?)";
            PreparedStatement ps = con.prepareStatement(kueri);

            //mengisi ?
            ps.setString(1, "6701150053");
            ps.setString(2, "Rayizan Alfi");
            ps.setInt(3, 2016);
            ps.setString(4, "D3MI-40-01");
            //eksekusi query kueri
            int jumRow = ps.executeUpdate();

            if(jumRow > 0){
                System.out.println("Eksekusi Berhasil");
            }else{
                System.out.println("Eksekusi Gagal");
            }

            ps.close();
            con.close();
        } catch (Exception ex) {
            System.err.print("Exception: ");
            System.err.println(ex);
        }
    }
```

```
}
```

Hasil:



Mahasiswa

NIM	Nama	Angkatan	Kelas
6701150052	ELENTERIUS ROBERTUS	2015	D3MI-39-02
6701150053	Rayizan Alfi	2016	D3MI-40-01
6701151063	FARANDIKA ARIES	2015	D3MI-39-03
6701152188	BEGORDI FADHLAN RIGI	2015	D3MI-39-03
6701154157	VIRA AYU MAGFIROH	2015	D3MI-39-02

Gambar 17: Hasil Penambahan Data

10.3.3 Hapus Data

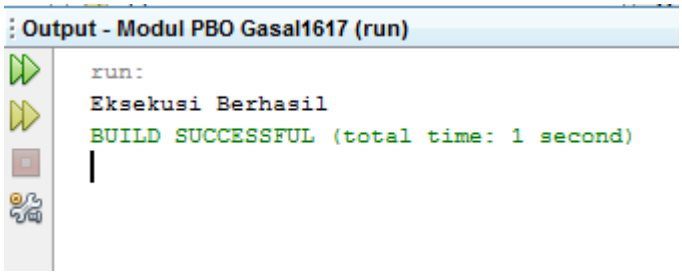
Contoh Kode:

Listing Program 20: Akses DB - DELETE

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;

public class MsAccessHapus {
    public static void main(String[] args) {
        try {
            // membuat objek Connection & PreparedStatement
            String alamat = "jdbc:ucanaccess://E:/ database/Database2.accdb";
            Connection con = DriverManager.getConnection(alamat);
            String kueri = "DELETE FROM Mahasiswa WHERE nim=?";
            PreparedStatement ps = con.prepareStatement(kueri);

            //mengisi ?
            ps.setString(1, "6701150053");

            //eksekusi query kueri
            int jumRow = ps.executeUpdate();

            if(jumRow > 0){
                System.out.println("Eksekusi Berhasil");
            }else{
                System.out.println("Eksekusi Gagal");
            }

            ps.close();
            con.close();
        } catch (Exception ex) {
            System.err.print("Exception: ");
            System.err.println(ex);
        }
    }
```

```
}
```

Hasil:



Mahasiswa

NIM	Nama	Angkatan	Kelas
6701150052	ELENTERIUS ROBERTUS	2015	D3MI-39-02
6701151063	FARANDIKA ARIES	2015	D3MI-39-03
6701152188	BEGORDI FADHLAN RIGI	2015	D3MI-39-03
6701154157	VIRA AYU MAGFIROH	2015	D3MI-39-02

Gambar 18: Hasil Penghapusan Data

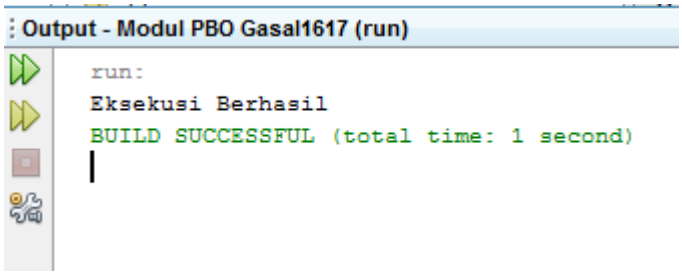
10.3.4 Pengubahan Data

Contoh kode:

Listing Program 21: Akses DB - UPDATE

```
public class MsAccessUbah {
    public static void main(String[] args) {
        try {
            // membuat objek Connection & PreparedStatement
            String alamat = "jdbc:ucanaccess://E/database/Database2.accdb";
            Connection con = DriverManager.getConnection(alamat);
            String kueri = "UPDATE Mahasiswa SET nama=? WHERE nim=?";
            PreparedStatement ps = con.prepareStatement(kueri);

            //mengisi ?
            ps.setString(1, "Name-X");
            ps.setString(2, "6701150052");

            //eksekusi query kueri
            int jumRow = ps.executeUpdate();

            if(jumRow > 0){
                System.out.println("Eksekusi Berhasil");
            }else{
                System.out.println("Eksekusi Gagal");
            }

            ps.close();
            con.close();
        } catch (Exception ex) {
            System.err.print("Exception: ");
            System.err.println(ex);
        }
    }
}
```

Hasil:



Mahasiswa

NIM	Nama	Angkatan	Kelas
6701150052	Name-X	2015	D3MI-39-02
6701151063	FARANDIKA ARIES	2015	D3MI-39-03
6701152188	BEGORDI FADHLAN RIGI	2015	D3MI-39-03
6701154157	VIRA AYU MAGFIROH	2015	D3MI-39-02

Gambar 19: Hasil Pengubahan Data

NOTE: Jika data tidak terlihat berubah di Ms. Access, tutup Ms. Access lalu buka kembali datanya. Refresh data dikhawatirkan tidak menampilkan data perubahan yang dimodifikasi dari bahasa pemrograman.

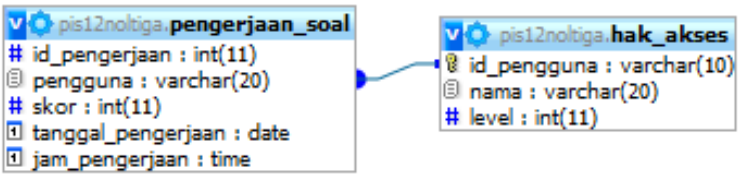
10.4 Latihan

Conan, Ayumi, Genta dan Mitsuhiko merupakan anak SD kelas 1 yang tergabung dalam kelompok detective cilik. Kelompok ini senang memecahkan kode untuk menemukan harta karun. Kode terakhir yang mereka pecahkan berkaitan dengan kriptografi. Sayangnya, metode ini membutuhkan kemampuan matematika yang handal. Conan sebagai yang terpintar di anggotanya, dibantu Mitsuhiko, ingin membuat 1 pelatihan matematika dasar untuk teman-temannya, terutama Genta.

Conan memiliki kemampuan membuat aplikasi sederhana menggunakan bahasa java. Dengan kemampuannya, ia membuat aplikasi pelatihan matematika sederhana yang meng-generate 2 bilangan random (1-100), dan 1 operator (+, *, /, -). Setiap ada yang ingin menggunakan program tersebut, kode melakukan generate soal sebanyak 10 kali. Setiap menjawab benar, point-nya adalah 10. Dan nantinya setelah menjawab soal2 tersebut, hasil akan langsung disimpan ke dalam database.

Terdapat 2 hak akses pengguna, level 1 dan level 2. Level 1 dapat melihat semua hasil pembelajaran teman-teman terurut berdasarkan pengguna dan waktu pembelajaran. Bisa melakukan fungsi “hapus” dan “ubah” untuk setiap hasil pembelajaran ini. Level 2 hanya bisa melihat hasil pembelajaran mereka dan mengerjakan soal. Conan dan Mitsuhiko tergabung pada hak akses level 1, sedangkan Ayumi dan Genta tergabung pada hak akses level 2. Tidak ada menu untuk memodifikasi tabel dari hak akses pengguna.

Data yang digunakan adalah sebagai berikut:



Jalannya Program kira-kira sebagai berikut:

```
Pengguna: <input user>Ayumi
1. Kerjakan Soal
2. Lihat Pengerjaan Soal

Pilihan: <input user>2
Nama: Ayumi
Jumlah Tes: 3
1. 90 || 12 September 2013 || 05.00
2. 100 || 13 September 2013 || 07.00
3. 70 || 13 September 2013 || 10.00
```

Pengguna: <input user> Conan
1. Kerjakan Soal
2. Lihat Progress Teman-Teman
3. Lihat Hasil Sendiri

Pilihan: <input user>1
2 + 58 = <input user> 60
3 * 77= <input user> 12
<muncul soal sampai 10x>

Conan keren mau ngerjain soal matematika. Skor Conan 90!
Skor-nya disimpan di basis data yah…

Pilihan:<input user>2
1. Ayumi || 90 || 12 September 2013 || 05.00
2. Ayumi || 100 || 13 September 2013 || 07.00
3. Ayumi || 70 || 13 September 2013 || 10.00
4. Genta || 60 || 27 Juni 2013 || 18.00
5. Genta || 80 || 21 September 2013 || 04.37

Modul 11 : Responsi Tugas Besar

11.1 Tujuan

Setelah mengikuti praktikum ini mahasiswa diharapkan dapat:

1. Mampu membuat aplikasi sederhana menggunakan akses database.
2. Mampu menerapkan query DML & SELECT pada basis data Access menggunakan bahasa pemrograman Java.
3. Mampu membuat perancanganan aplikasi sederhana dengan konsep yang jelas
4. Mampu menerapkan perancangan yang telah ditetapkan sebelumnya menggunakan bahasa pemrograman Java.

11.2 Alat & Bahan

Alat & Bahan Yang digunakan adalah hardware perangkat PC beserta Kelengkapannya berjumlah 40 PC dengan Java SDK dan editor IDE (disarankan Netbeans) telah terinstall di masing-masing unit.

11.3 Cek Kemajuan

Tabel 2: Penilaian Kemajuan Praktikan

Nomor	Komponen	Deskripsi Kemajuan Praktikan	Penilaian
1	Mahasiswa telah menentukan topik sebagai gambaran bagi tugas besar.		
2	Mahasiswa telah memberikan gambaran cara menerima data masukan pengguna, cara menampilkan data, dan cara mengeluarkan keluaran pada pengguna. Penilaian dilihat dari mockup yang diperlihatkan.		
3	Mahasiswa telah membuat rancangan yang benar pada aplikasi yang akan dibuat. Penilaian dilihat dari UML (class diagram dan atau use case) yang diberikan.		

Modul 12 : Presentasi Tugas Besar

12.1 Tujuan

Setelah mengikuti praktikum ini mahasiswa diharapkan dapat:

1. Mampu membuat aplikasi sederhana menggunakan akses database.
2. Mampu menerapkan query DML & SELECT pada basis data Access menggunakan bahasa pemrograman Java.
3. Mampu membuat perancanganan aplikasi sederhana dengan konsep yang jelas
4. Mampu menerapkan perancangan yang telah ditetapkan sebelumnya menggunakan bahasa pemrograman Java.

12.2 Alat & Bahan

Alat & Bahan Yang digunakan adalah hardware perangkat PC beserta Kelengkapannya berjumlah 40 PC dengan Java SDK dan editor IDE (disarankan Netbeans) telah terinstall di masing-masing unit.

12.3 Penilaian

Tabel 3: Penilaian Tugas Besar Praktikan

Nomor	Komponen	Deskripsi Penilaian Praktikan	Penilaian
1	Mahasiswa telah membuat storage penyimpanan data dengan baik dan benar.		
2	Mahasiswa telah membuat aplikasi dengan tampilan yang baik dan rapih.		
3	Mahasiswa telah membuat aplikasi desktop dengan akses data yang terhubung.		
4	Mahasiswa mampu mengkomunikasikan produk yang telah dibuat dengan baik.		