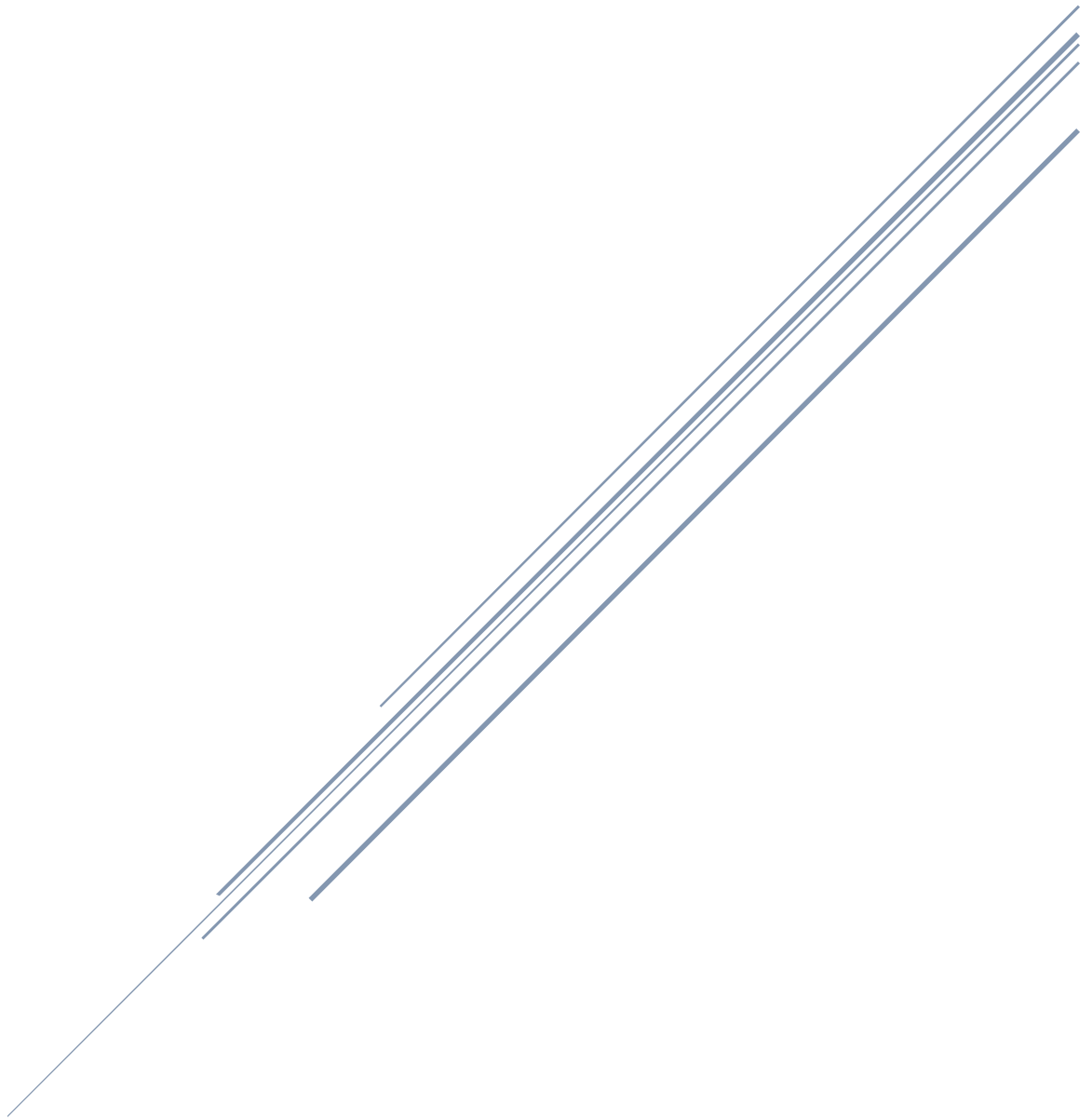


MODUL PEMOGRAMAN BERORIENTASI OBJEK LANJUT

Pemograman JAVA



UNIVERSITAS WIJAYA PUTRA
PROGRAM STUDI TEKNIK INFORMATIKA

Winner vs. Loser

Winner is always a part of solutions

Loser is always a part of problems

Winner sees answer in every problem

Loser sees problem in every answer

Winner always has a program

Loser always has an excuse

Winner always says, "It's difficult, but it's possible."

Loser always says, "It's possible, but it's difficult."



KATA PENGANTAR

Alhamdulillah, segala puji bagi Allah karena hanya dengan rahmat-Nya modul ini dapat terselesaikan dengan baik. Modul ini dibuat untuk melaksanakan praktikum mata kuliah Pemrograman Berorientasi Objek Lanjut bagi mahasiswa semester empat. Pada semester lalu kita telah mempelajari sistem pemrograman berorientasi objek. Kita telah mengenal class, obyek, method, overload dan override, pewarisan dan polimorfisme. Materi tersebut merupakan materi dasar yang harus dipahami oleh programmer Java.

Pada tahap selanjutnya kita akan mempelajari tentang materi OOP lebih lanjut. Pada modul ini kita akan mempelajari bagaimana membuat sebuah aplikasi GUI berbasis OOP menggunakan Netbeans. Materi yang dipelajari meliputi exception, komponen GUI pada Java, GUI event handling, koneksi ke database. Diharapkan mahasiswa dapat membuat sebuah aplikasi GUI sederhana lengkap dengan database setelah melakukan praktikum semua bab.

Penyusun berharap semoga Modul ini dapat bermanfaat bagi semua pihak, terutama bagi mahasiswa yang mengambil mata kuliah pemrograman berorientasi objek di Universitas Wijaya Putra Surabaya. Namun kami menyadari bahwa modul ini belum sempurna. Tinjauan dan saran yang bersifat membangun tetaplah sangat diharapkan demi peningkatan kesempurnaan modul praktikum ini.

Surabaya, April 2017

Penyusun



DAFTAR ISI

DAFTAR ISI.....	3
BAB 1	5
EXCEPTIONS & ASSERTIONS	5
POKOK BAHASAN.....	5
TUJUAN BELAJAR.....	5
Dasar Teori	6
Percobaan.....	10
Latihan	13
Tugas.....	13
BAB 2	14
SWING DAN GUI EVENT HANDLING.....	14
POKOK BAHASAN.....	14
TUJUAN BELAJAR.....	14
Dasar Teori	15
Percobaan.....	21
Latihan	22
Tugas	23
BAB 3	25
JTABLE.....	25
POKOK BAHASAN.....	25
TUJUAN BELAJAR.....	25
Dasar Teori	26
Percobaan.....	27
Latihan	31
BAB 4	32
LAYOUT	32



POKOK BAHASAN.....	32
TUJUAN BELAJAR.....	32
Dasar Teori	33
Percobaan.....	35
Latihan	37
BAB 5	37
JAVA DATABASE CONNECTIVITY (JDBC).....	Error! Bookmark not defined.
POKOK BAHASAN.....	38
TUJUAN BELAJAR.....	38
Dasar Teori	39
Percobaan.....	41
Latihan	45
BAB 6	46
TUGAS AKHIR PRAKTIKUM.....	46
INSTRUKSI	46



BAB 1

EXCEPTIONS & ASSERTIONS

POKOK BAHASAN

- Penanganan exception menggunakan try, catch, finally
- Penggunaan throw dan throws
- Perkenalan beberapa exception class
- Pembuatan exception class manual
- Penggunaan assertions

TUJUAN BELAJAR

Setelah melakukan praktikum dalam bab ini, mahasiswa diharapkan mampu:

- Menangani exception dengan menggunakan try, catch dan finally
- Membedakan penggunaan antara throw dengan throws
- Menggunakan exception class yang berbeda – beda
- Membedakan antara checked exceptions dan unchecked exceptions
- Membuat exception class tersendiri
- Menjelaskan keunggulan penggunaan assertions
- Menggunakan assertions



Dasar Teori

Bugs dan error dalam sebuah program sangat sering muncul meskipun program tersebut dibuat oleh programmer berkemampuan tinggi. Untuk menghindari pemborosan waktu pada proses error-checking, Java menyediakan mekanisme penanganan exception.

Exception adalah singkatan dari Exceptional Events. Kesalahan (errors) yang terjadi saat runtime, menyebabkan gangguan pada alur eksekusi program. Terdapat beberapa tipe error yang dapat muncul. Sebagai contoh adalah error pembagian 0, mengakses elemen di luar jangkauan sebuah array, input yang tidak benar dan membuka file yang tidak ada.

- **Error Class dan Exception Class**

Seluruh exceptions adalah subclasses, baik secara langsung maupun tidak langsung, dari sebuah root class Throwable. Kemudian, dalam class ini terdapat dua kategori umum : Error class dan Exception class.

Exception Class → menunjukkan kondisi yang dapat diterima oleh user program. Umumnya hal tersebut disebabkan oleh beberapa kesalahan pada kode program. Contoh dari exceptions adalah pembagian oleh 0 dan error di luar jangkauan array.

Error Class → digunakan oleh Java run-time untuk menangani error yang muncul pada saat dijalankan. Secara umum hal ini di luar control user karena kemunculannya disebabkan oleh run-time environment. Sebagai contoh adalah out of memory dan harddisk crash.

- **Menangkap Exception**

- **Try-catch**

Seperti yang telah dijelaskan sebelumnya, keyword try, catch dan finally digunakan dalam menangani bermacam tipe exception. 3 Keyword tersebut digunakan bersama, namun finally bersifat opsional. Akan lebih baik jika memfokuskan pada dua keyword pertama, kemudian membahas finally pada bagian akhir.

Syntax try-catch :

```
try {  
    <code to be monitored for exceptions>  
} catch (<ExceptionType1> <ObjName>) {  
    <handler if ExceptionType1 occurs>  
}  
...  
} catch (<ExceptionTypeN> <ObjName>) {  
    <handler if ExceptionTypeN occurs>  
}
```

- **Keyword finally**

Finally bersifat optional pada exception try-catch.

Syntax try-catch-finally :

```
try {  
    <kode monitor exception>  
} catch (<ExceptionType1> <ObjName>) {  
    <penanganan jika ExceptionType1 terjadi>  
} ...
```



```

} finally {
<kode yang akan dieksekusi saat blok try berakhir>
}

```

• Melempar Exception

○ Keyword throw

Disamping menangkap exception, Java juga mengizinkan seorang user untuk melempar sebuah exception. Sintaks pelemparan exception cukup sederhana :

```
throw <exception object>;
```

○ Keyword throws

Jika sebuah method dapat menyebabkan sebuah exception namun tidak menangkapnya, maka digunakan keyword throws. Aturan ini hanya berlaku pada checked exception. Berikut ini penulisan syntax menggunakan keyword throws :

```

<type> <methodName> (<parameterList>) throws <exceptionList> {
<methodBody>
}

```

• Kategori Exception

○ Exception Classes dan Hierarki

Seperti yang disebutkan sebelumnya, root class dari seluruh exception classes adalah Throwable class. Yang disebutkan dibawah ini adalah exception class hierarki. Seluruh exceptions ini terdefinisi pada package java.lang.

Tabel 1.1 Hirarki Exception Class

Exception Class Hierarchy		
Throwable	Error	LinkageError, ... VirtualMachineError, ...
	Exception	ClassNotFoundException, CloneNotSupportedException, IllegalAccessException, InstantiationException, InterruptedException, IOException, EOFException, FileNotFoundException, ...



RuntimeException,	ArithmeticException,
	ArrayStoreException,
	ClassCastException,
	IllegalArgumentException,
	(IllegalThreadStateException and NumberFormatException as subclasses)
	IllegalMonitorStateException,
	IndexOutOfBoundsException,
	NegativeArraySizeException,
	NullPointerException,
	SecurityException
...	

Dengan mengetahui hirarki tersebut, maka perlu diingat bahwa untuk program yang memiliki catch lebih dari satu penulisan exception harus berurutan dari subclass ke superclass.

○ **Checked dan Unchecked Exceptions**

Checked exceptions adalah exception yang diperiksa oleh Java compiler. Compiler memeriksa keseluruhan program apakah menangkap atau mendaftarkan exception yang terjadi dalam syntax throws. Apabila checked exception tidak didaftarkan ataupun ditangkap, maka compiler error akan ditampilkan.

Tidak seperti checked exceptions, unchecked exceptions tidak berupa compile-time checking dalam penanganan exceptions. Pondasi dasar dari unchecked exception classes adalah Error, RuntimeException dan subclass-nya.

○ **User Defined Exceptions**

Meskipun beberapa exception classes terdapat pada package java.lang namun tidak mencukupi untuk menampung seluruh kemungkinan tipe exception yang mungkin terjadi. Sehingga sangat mungkin bahwa kita perlu untuk membuat tipe exception tersendiri.

Dalam pembuatan tipe exception manual, kita hanya perlu membuat sebuah extended class terhadap RuntimeException class, maupun Exception class lain. Selanjutnya tergantung pada bagaimana kita memodifikasi class sesuai permasalahan yang akan diselesaikan. Members dan constructors dapat dimasukkan pada exception class tersebut.

• **Assertions**

○ **User Defined Exceptions**

Assertions mengizinkan programmer untuk menentukan asumsi yang dihadapi. Sebagai contoh, sebuah tanggal dengan area bulan tidak berada antara 1 hingga 12 dapat diputuskan bahwa data tersebut tidak valid. Programmer dapat menentukan bulan harus berada diantara area tersebut. Meskipun hal itu dimungkinkan untuk menggunakan constructor lain untuk mensimulasikan fungsi dari assertions, namun sulit untuk dilakukan karena fitur assertion dapat tidak digunakan. Hal yang menarik dari assertions adalah seorang user memiliki pilihan untuk digunakan atau tidak pada saat runtime.



Assertion dapat diartikan sebagai extensi atas komentar yang menginformasikan pembaca kode bahwa sebagian kondisi harus terpenuhi. Dengan menggunakan assertions, maka tidak perlu untuk membaca keseluruhan kode melalui setiap komentar untuk mencari asumsi yang dibuat dalam kode. Namun, menjalankan program tersebut akan memberitahu Anda tentang assertion yang dibuat benar atau salah. Jika assertion tersebut salah, maka `AssertionError` akan terjadi.

o **Mengaktifkan dan menonaktifkan exceptions**

Penggunaan assertions tidak perlu melakukan import package `java.util.assert`. Menggunakan assertions lebih tepat ditujukan untuk memeriksa parameter dari non-public methods jika public methods dapat diakses oleh class lain. Hal itu mungkin terjadi bila penulis dari class lain tidak menyadari bahwa mereka dapat menonaktifkan assertions. Dalam hal ini program tidak dapat bekerja dengan baik. Pada non-public methods, hal tersebut tergunakan secara langsung oleh kode yang ditulis oleh programmer yang memiliki akses terhadap methods tersebut. Sehingga mereka menyadari bahwa saat menjalankannya, assertion harus dalam keadaan aktif.

Untuk mengkompilasi file yang menggunakan assertions, sebuah tambahan parameter perintah diperlukan seperti yang terlihat dibawah ini :

```
javac -source 1.4 MyProgram.java
```

Jika Anda ingin untuk menjalankan program tanpa menggunakan fitur assertions, cukup jalankan program secara normal.

```
java MyProgram
```

Namun, jika Anda ingin mengaktifkan assertions, Anda perlu menggunakan parameter `-enableassertions` atau `-ea`.

```
java -enableassertions MyProgram
```

o **Sintak Assertions**

Penulisan assertions memiliki dua bentuk. Bentuk yang paling sederhana terlihat sebagai berikut :

```
assert <expression1>;
```

dimana `<expression1>` adalah kondisi dimana assertion bernilai true. Bentuk yang lain menggunakan dua ekspresi, berikut ini cara penulisannya :

```
assert <expression1> : <expression2>;
```

dimana `<expression1>` adalah kondisi assertion bernilai true dan `<expression2>` adalah informasi yang membantu pemeriksaan mengapa program mengalami kesalahan.



Percobaan

Percobaan 1 : Program DivByZero menggunakan try-catch

```
try {
    System.out.println(3/0);
    System.out.println("Cetak.");
} catch (ArithmeticException exc) {
    //Reaksi atas kejadian
    System.out.println(exc);
}
System.out.println("Setelah Exception.");
```

Percobaan 2 : Menangani lebih dari satu exception

```
try {
    int den = Integer.parseInt(args[0]); //baris 4
    System.out.println(3/den); //baris 5
} catch (ArithmeticException exc) {
    System.out.println("Nilai Pembagi 0.");
} catch (ArrayIndexOutOfBoundsException exc2) {
    System.out.println("Missing argument.");
}
System.out.println("After exception.");
```

Percobaan 3 : Penanganan try-catch bersarang

```
try {
    int a = Integer.parseInt(args[0]);
    try {
        int b = Integer.parseInt(args[1]);
        System.out.println(a/b);
    } catch (ArithmeticException e) {
        System.out.println("Divide by zero error!");
    }
} catch (ArrayIndexOutOfBoundsException e) {
    System.out.println("2 parameters are required!");
}
```

Percobaan 4 : Penggunaan try bersarang tergabung dalam method

```
class NestedTryDemo2 {
    static void nestedTry(String args[]) {
        try {
            int a = Integer.parseInt(args[0]);
            int b = Integer.parseInt(args[1]);
            System.out.println(a/b);
        } catch (ArithmeticException e) {
            System.out.println("Divide by zero error!");
        }
    }

    public static void main(String args[]){
        try {
            nestedTry(args);
        } catch (ArrayIndexOutOfBoundsException e) {
            System.out.println("2 parameters are required!");
        }
    }
}
```



Percobaan 5 : Penggunaan keyword finally

FinallyDemo.java

```
class FinallyDemo {
    static void myMethod(int n) throws Exception{
        try {
            switch(n) {
                case 1: System.out.println("case pertama");
                return;
                case 3: System.out.println("case ketiga");
                throw new RuntimeException("demo case ketiga");
                case 4: System.out.println("case keempat");
                throw new Exception("demo case keempat");
                case 2: System.out.println("case Kedua");
            }
        } catch (RuntimeException e) {
            System.out.print("RuntimeException terjadi: ");
            System.out.println(e.getMessage());
        } finally {
            System.out.println("try-block entered.");
        }
    }

    public static void main(String args[]){
        for (int i=1; i<=4; i++) {
            try {
                FinallyDemo.myMethod(i);
            } catch (Exception e){
                System.out.print("Exception terjadi: ");
                System.out.println(e.getMessage());
            }
            System.out.println();
        }
    }
}
```

Percobaan 6 : Melempar exception dengan keyword throw

```
/* Melempar exception jika terjadi kesalahan input */
class ThrowDemo {
    public static void main(String args[]){
        String input = "invalid input";
        try {
            if (input.equals("invalid input")) {
                throw new RuntimeException("throw demo");
            } else {
                System.out.println(input);
            }
            System.out.println("After throwing");
        } catch (RuntimeException e) {
            System.out.println("Exception caught here.");
            System.out.println(e);
        }
    }
}
```



Percobaan 7 : Melempar exception dengan keyword throws

Dari hasil percobaan ini, jelaskan apa maksud dari program tersebut?

```
class ThrowingClass {
    static void myMethod() throws ClassNotFoundException {
        throw new ClassNotFoundException ("just a demo");
    }
}

class ThrowsDemo {
    public static void main(String args[]) {
        try {
            ThrowingClass.myMethod();
        } catch (ClassNotFoundException e) {
            System.out.println(e);
        }
    }
}
```

Percobaan 8 : Mengetes hirarki exception

Mengapa kode di bawah ini error ketika di-compile? Coba betulkan agar program dapat di-compile.

```
class MultipleCatchError {
    public static void main(String args[]){
        try {
            int a = Integer.parseInt(args [0]);
            int b = Integer.parseInt(args [1]);
            System.out.println(a/b);
        } catch (Exception e) {
            System.out.println(e);
        } catch (ArrayIndexOutOfBoundsException e2) {
            System.out.println(e2);
        }
        System.out.println("After try-catch-catch.");
    }
}
```

Percobaan 9 : Membuat exception sendiri

```
class HateStringException extends RuntimeException{
    /* Tidak perlu memasukkan member ataupun konstruktor */
}

class TestHateString {
    public static void main(String args[]) {
        String input = "invalid input";
        try {
            if (input.equals("invalid input")) {
                throw new HateStringException();
            }
            System.out.println("String accepted.");
        } catch (HateStringException e) {
            System.out.println("I hate this string: " + input +
                ".");
        }
    }
}
```



Percobaan 10 : Membuat assertion

```
class AgeAssert {
public static void main(String args[]) {
int age = Integer.parseInt(args[0]);
assert(age>0);
/* jika masukan umur benar (misal, age>0) */
if (age >= 18) {
System.out.println("Congrats! You're an adult!");
}
}
}
```

Perhatikan jika argumen diberi angka lebih dari 0, maka program akan berjalan normal. Ganti argumen dengan bilangan negatif. Bagaimana hasilnya?

Exception tersebut termasuk dalam Unchecked Exception yang tidak bisa di deteksi pada saat kompilasi. Tentu saja kita tidak menginginkan program kita berhenti karena exception. Oleh karena itu, Java mulai menyediakan cara untuk menguji input dengan menggunakan assertion. Assertion pada JVM secara bawaan adalah tidak aktif sehingga perlu diaktifkan terlebih dahulu.

Cara mengaktifkan assertion : Klik kanan pada project → pilih Properties → Pilih Run → pada VM Options ketikkan -ea → Klik Ok

Latihan

Latihan 1 : Heksadesimal ke desimal

Tentukan sebuah angka heksadesimal sebagai input. Konversi angka tersebut menjadi bilangan desimal. Tentukan exception class Anda sendiri dan lakukan penanganan jika input dari user bukan berupa bilangan heksadesimal.

Tugas

Tugas 1 : Menampilkan sebuah berlian

Tentukan nilai integer positif sebagai input. Tampilkan sebuah berlian menggunakan karakter asterisk (*) sesuai angka yang diinput oleh user. Jika user memasukkan bilangan integer negatif, gunakan assertions untuk menanganinya. Sebagai contoh, jika user memasukkan integer bernilai 3, program Anda harus menampilkan sebuah berlian sesuai bentuk berikut :

```
  *
 ***
*****
 ***
  *
```



BAB 2

SWING DAN GUI EVENT HANDLING

POKOK BAHASAN

- Desain GUI dengan SWING
- Delegation Event Model
- Class-class event
- Event Listeners

TUJUAN BELAJAR

Dengan praktikum ini mahasiswa diharapkan dapat:

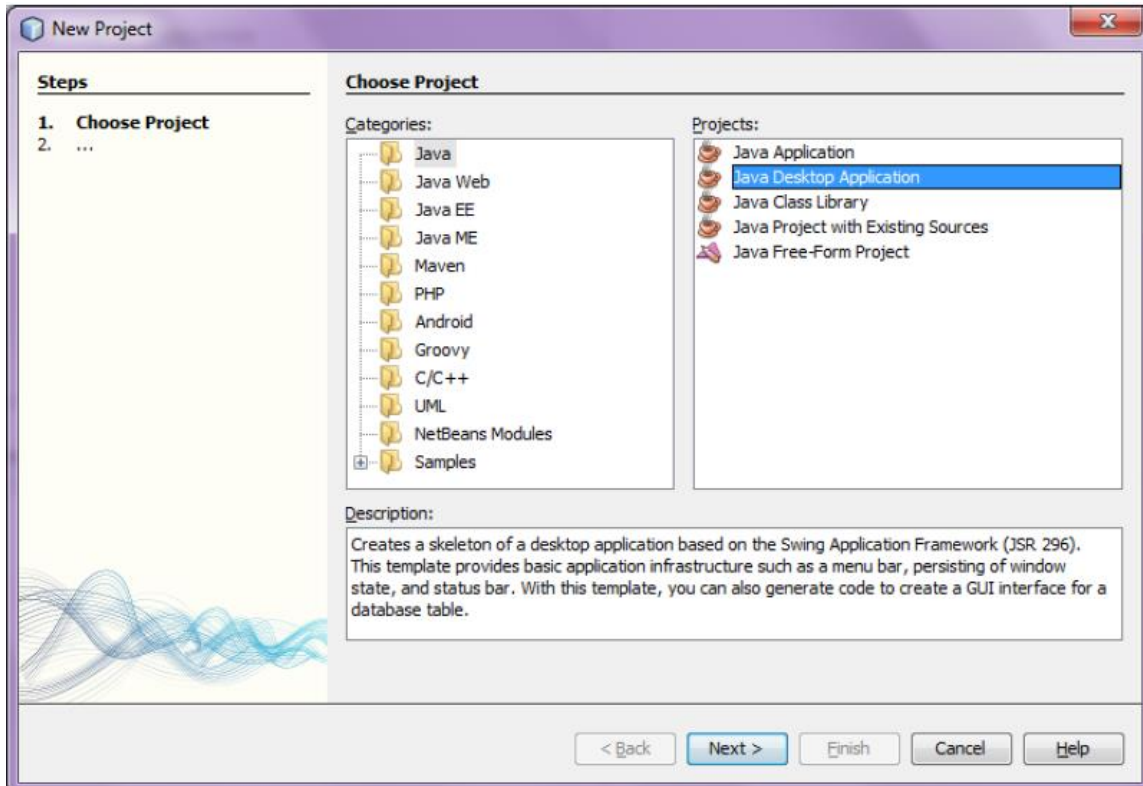
- Mampu membuat aplikasi sederhana menggunakan komponen swing.
- Mampu menggunakan Matisse Builder sebagai editor GUI pada pemrograman visual Java.
- Mampu menambahkan aksi pada aplikasi yang dibuat



Dasar Teori

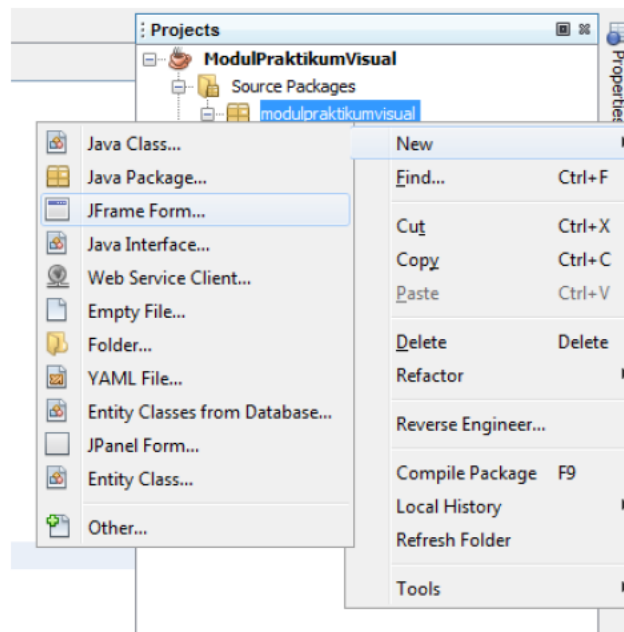
Netbeans memiliki sebuah project bernama Project Matisse untuk membuat sebuah builder GUI dengan basis swing menggunakan bahasa pemrograman java. Swing GUI builder ini membantu para programmer untuk membangun sebuah aplikasi desktop karena dapat membangun GUI secara visual dan bukan hanya sekedar text-based 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.



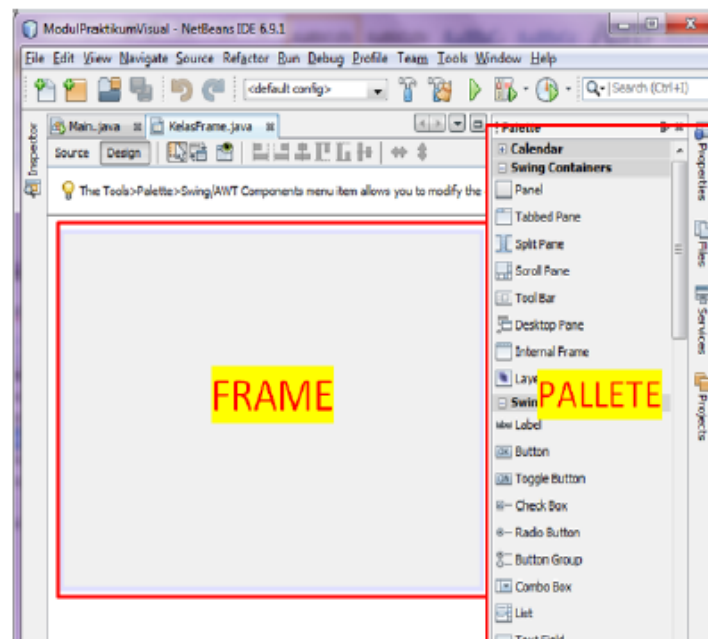
Gambar 2.1 Java Desktop Application

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



Gambar 2.2 Java Application

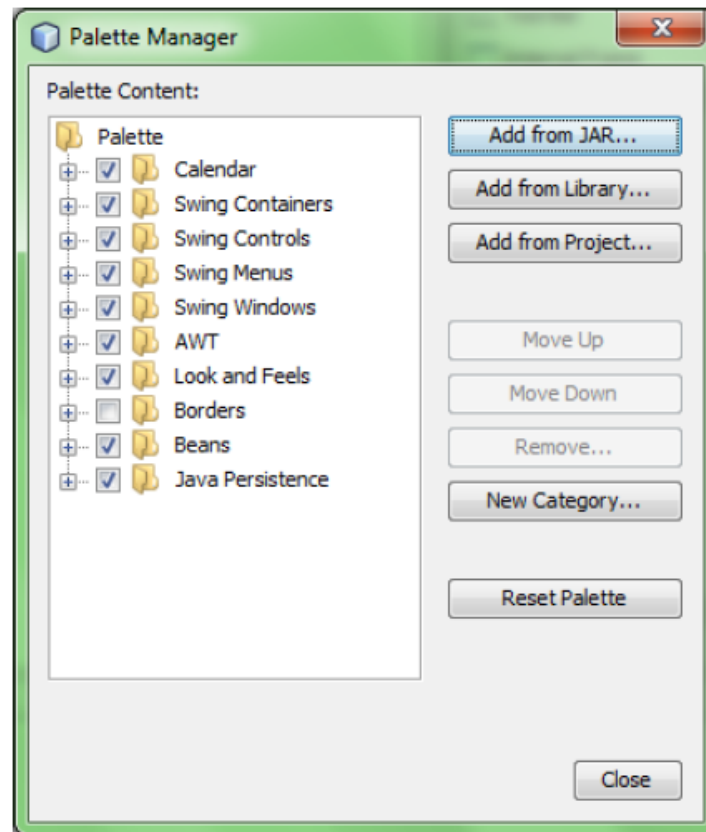
Hal ini akan menciptakan satu kelas java yang meng-extends JFrame yang merupakan container tertinggi dari sebuah aplikasi java. Matisse Builder/GUI editor ini terdiri dari beberapa bagian, 2 diantaranya yaitu bagian frame dan palette.



Gambar 2.3 Frame dan Pallette 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).

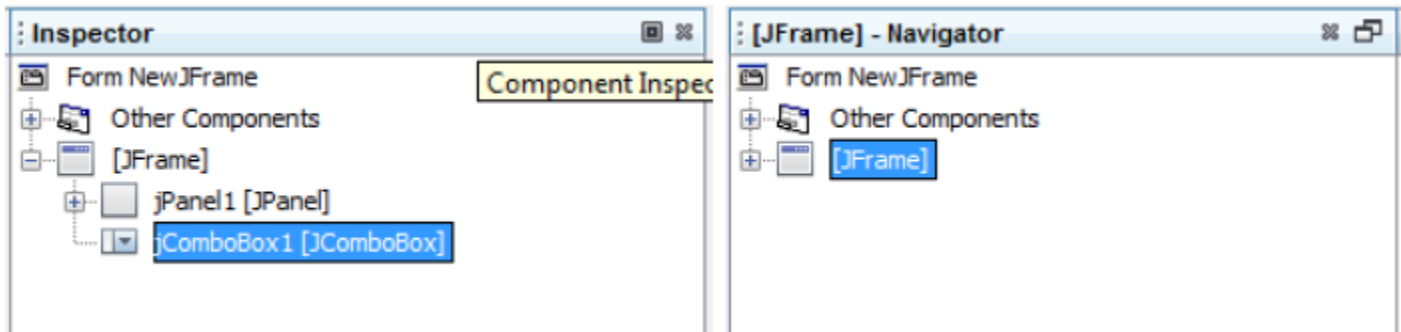


Gambar 2.4 Palette Manager

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

- Swing container: merupakan container pada pemrograman gui menggunakan bahasa java. Biasa digunakan untuk windowing (cara/bentuk menampilkan aplikasi ke user)
- 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
- 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 18ias diubah terdapat di window properties. Untuk melihat susunan hirarki gui, 18ias dilihat di window inspector.



Gambar 2.5 Inspector dan Navigator

GUI EVENT HANDLING

- **Delegation Event Model (Model Pendelegasian Event)**

Delegasi event model menjelaskan bagaimana program Anda merespon interaksi dari user. Tiga komponen model :

- Event Source** : Event source mengacu pada komponen GUI yang meng-generate event. Sebagai contoh, jika user menekan tombol, event source dalam hal ini adalah tombol.
- Event Listener/Handler** : Event listener menerima berita dari event-event dan proses-proses interaksi user. Ketika tombol ditekan, listener akan mengendalikan dengan menampilkan sebuah informasi yang berguna untuk user.
- Event Object** : Ketika sebuah event terjadi (misal, ketika user berinteraksi dengan komponen GUI), sebuah object event diciptakan. Object berisi semua informasi yang perlu tentang event yang telah terjadi. Informasi meliputi tipe dari event yang telah terjadi, seperti ketika mouse telah di-klik. Ada beberapa class event untuk kategori yang berbeda dari user action. Sebuah event object mempunyai tipe data mengenai salah satu dari class ini.

- **Registrasi Listeners**

Event source mendaftarkan sebuah listener melalui method `add<Type>Listener`.

```
void add<Type>Listener(<Type>Listener listenerObj)
```

<Type> tergantung pada tipe dari event source, dapat berupa Key, Mouse, Focus, Component, Action dan lainnya. Beberapa listeners dapat diregistrasi dengan satu event source untuk menerima pemberitahuan event. Listener yang telah teregistrasi dapat juga tidak diregistrasikan lagi menggunakan method `remove<Type>Listener`.

```
void remove<Type>Listener(<Type>Listener listenerObj)
```



- **Class-class event**

Sebuah event object mempunyai sebuah class event sebagai tipe data acuannya. Akar dari hirarki class event adalah class EventObject, yang dapat ditemukan pada paket java.util. Immediate subclass dari class EventObject adalah class AWTEvent. Class AWTEvent didefinisikan pada paket java.awt. Itu merupakan akar dari semua AWT-based events. Berikut ini beberapa dari class-class AWT event.

Class Event	Deskripsi
ComponentEvent	Extends AWTEvent. Dijalankan ketika sebuah komponen dipindahkan, di-resize, dibuat visible atau hidden.
InputEvent	Extends ComponentEvent. Abstrak root class event untuk semua komponen-level input class-class event.
ActionEvent	Extends AWTEvent. Dijalankan ketika sebuah tombol ditekan, melakukan double-klik daftar item, atau memilih sebuah menu.
ItemEvent	Extends AWTEvent. Dijalankan ketika sebuah item dipilih atau di-deselect oleh user, seperti sebuah list atau checkbox.
KeyEvent	Extends InputEvent. Dijalankan ketika sebuah key ditekan, dilepas atau diketikkan.
MouseEvent	Extends InputEvent. Dijalankan ketika sebuah tombol mouse ditekan, dilepas, atau di-klik (tekan dan lepas), atau ketika sebuah kursor mouse masuk atau keluar dari bagian visible dari komponen.
TextEvent	Extends AWTEvent. Dijalankan ketika nilai dari text field atau text area dirubah.
WindowEvent	Extends ComponentEvent. Dijalankan sebuah object Window dibuka, ditutup, diaktifkan, nonaktifkan, iconified, deiconified, atau ketika focus ditransfer kedalam atau keluar window.

Catatan, bahwa semua subclass-subclass AWTEvent mengikuti konvensi nama berikut ini:

```
<Type>Event
```

- **Event Listeners**

Event listeners adalah class yang mengimplementasikan interfaces <Type>Listener. Berikut ini adalah beberapa listener interfaces yang biasanya digunakan.

➤ ActionListener → Bereaksi atas perubahan mouse atau keyboard, hanya memiliki 1 method :

Method ActionListener
public void actionPerformed(ActionEvent e)
Mengendalikan ActionEvent e yang terjadi.

• MouseListener → Bereaksi atas pergerakan mouse. Beberapa method yang digunakan dalam class :

Method MouseListener



<code>public void mouseClicked(MouseEvent e)</code>
Dipanggil pada saat tombol mouse di click (seperti tekan dan lepas).
<code>public void mouseEntered(MouseEvent e)</code>
Dipanggil pada saat kursor mouse memasuki area komponen.
<code>public void mouseExited(MouseEvent e)</code>
Dipanggil pada saat kursor mouse meninggalkan area komponen.
<code>public void mousePressed(MouseEvent e)</code>
Dipanggil pada saat tombol mouse ditekan di atas komponen
<code>public void mouseReleased(MouseEvent e)</code>
Dipanggil pada saat tombol mouse dilepas di atas komponen

- `MouseListener` → Interface `MouseListener` mendukung `MouseListener`. Menyediakan method-method yang akan memantau pergerakan mouse, seperti drag dan pemindahan mouse. Memiliki dua method :

Method <code>MouseListener</code>
<code>public void mouseDragged(MouseEvent e)</code>
Digunakan untuk memantau pergerakan mouse yang melintasi object pada saat tombol mouse ditekan. Tindakan ini persis sama dengan tindakan pada saat memindahkan sebuah window.
<code>public void mouseMoved(MouseEvent e)</code>
Digunakan untuk memantau pergerakan mouse pada saat mouse melintasi area suatu object. Pada saat ini tidak ada mouse yang ditekan, hanya memindahkan pointer mouse melalui object.

- `WindowListener` → Bereaksi atas perubahan window. Beberapa method dari interface `WindowListener` :

Method <code>WindowListener</code>
<code>public void windowOpened(WindowEvent e)</code>
Dipanggil pada saat object window dibuka (pertama kali window dibuat tampil).
<code>public void windowClosing(WindowEvent e)</code>
Dipanggil pada saat user mencoba untuk menutup object Window dari menu sistem object. Il pada saat kursor mouse memasuki area komponen.
<code>public void windowClosed(WindowEvent e)</code>
Dipanggil pada saat object Window ditutup setelah memanggil penempatan (misal, release dari resource-resource yang digunakan oleh source) pada object.
<code>public void windowActivated(WindowEvent e)</code>
Dilibatkan ketika object Window adalah window yang aktif (window masih dipakai).



<code>public void windowDeactivated(WindowEvent e)</code>
Dilibatkan ketika object Window tidak lagi merupakan window yang aktif.
<code>public void windowIconified(WindowEvent e)</code>
Dipanggil ketika object Window di-minimize.
<code>public void windowDeiconified(WindowEvent e)</code>
Dipanggil ketika object Window kembali setelah di-minimize ke keadaan normal.

• Petunjuk untuk Menciptakan Aplikasi Handling GUI Events

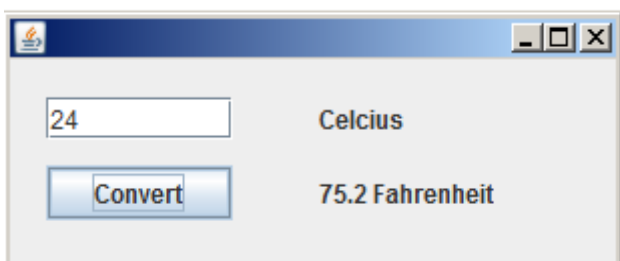
Berikut ini langkah-langkah yang Anda butuhkan untuk mengingat ketika ingin membuat aplikasi GUI dengan event handling.

1. Buatlah sebuah class yang menguraikan dan membuat suatu tampilan dari aplikasi GUI Anda.
2. Buatlah sebuah class yang menerapkan interface listener yang sesuai. Class ini boleh mengacu pada class yang sama seperti pada langkah awal.
3. Dalam menerapkan class, gunakan semua method-method dengan interface listener yang sesuai. Uraikan masing-masing method bagaimana Anda ingin mengendalikan event-event. Anda dapat memberikan implementasi kosong untuk method yang tidak ingin Anda gunakan.
4. Daftarkan object listener, instansiate dari class listener pada langkah 2, dengan source component menggunakan method `add<Type>Listener`.

Percobaan

Percobaan 1 : Aplikasi Konversi Suhu

Buatlah sebuah form seperti di bawah ini :



GUI Component: TextField, Label, Button
Events: actionPerformed, mouseClicked

Langkah pembuatan aplikasi GUI :

1. Membuat project baru : File → New Project → Java Application
2. Membri nama project : Nama : KonversiSuhu, Unchecked pada bagian “Create Main Class”
3. Menambahkan JFrameForm pada project : Nama Frame : CelciusToFahrenheit, Package : GUI



4. Menambahkan komponen pada frame : 1 panel (Sebagai dasar), 1 text field, 1 button, 2 label
5. Mengubah Text dari GUI Component : (sesuaikan dengan yang ada pada gambar)
6. Mengubah nama variabel dari setiap GUI Component : JTextField1: celciusTextField, JLabel1: celciusLabel, JLabel2: fahrenheitLabel, JButton1: convertButton
7. Rapikan tampilan Frame : potong bagian yang tidak perlu
8. Buat event untuk convert Button : Klik kanan pada Convert Button Pilih : Event → Action → ActionPerformed atau Event → Mouse → MouseClick
9. Buat code untuk event handling di convert button :

```
private void convertButtonActionPerformed(java.awt.event.ActionEvent evt) {  
    double fahrenheit = Double.parseDouble(celciusTextField.getText()) * 1.8 + 32;  
    fahrenheitLabel.setText(fahrenheit + " Fahrenheit");  
}
```

10. Build Project (F11) dan Jalankan (Run, F6).

Latihan

Latihan 1 : Membuat Kalkulator Sederhana

Buatlah sebuah GUI dengan mockup tampilan sebagai berikut:



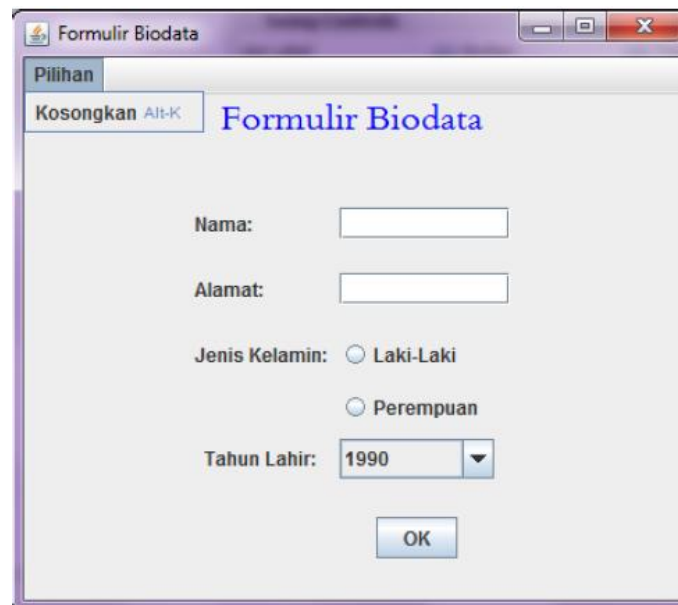
Keterangan:

- a) Masukkan komponen dalam panel; Jangan memasukkan langsung komponen dalam frame
- b) Gunakan Absolute Layout
- c) Field hasil tidak dapat diubah dan berwarna 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 : Membuat Formulir Biodata

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

Tugas

Tugas 1 : Tic-Tac-Toe

Buatlah tampilan GUI untuk program tic-tac-toe. Papannya terdiri dari sembilan kotak. Tambahkan event handlers ke dalam coding untuk membuat program berfungsi penuh. Permainan Tic-Tac-Toe dimainkan dengan dua pemain. Permainan dilakukan secara bergantian. Pemain 1 mendapat giliran pertama, setiap kali pemain 1 meng-klik papan akan bertanda X, setelah itu pemain 2 mendapat giliran bermain. Setiap kali pemain 2 meng-klik papan akan bertanda O. Pemain yang sukses menaklukkan 3 kotak membentuk garis horisontal, vertikal, atau diagonal, memenangkan permainan. Permainan akan berakhir ketika salah satu pemain menang atau ketika semua kotak telah terisi.





BAB 3

JTABLE

POKOK BAHASAN

- Jtable
- Model

TUJUAN BELAJAR

Dengan praktikum ini mahasiswa diharapkan dapat:

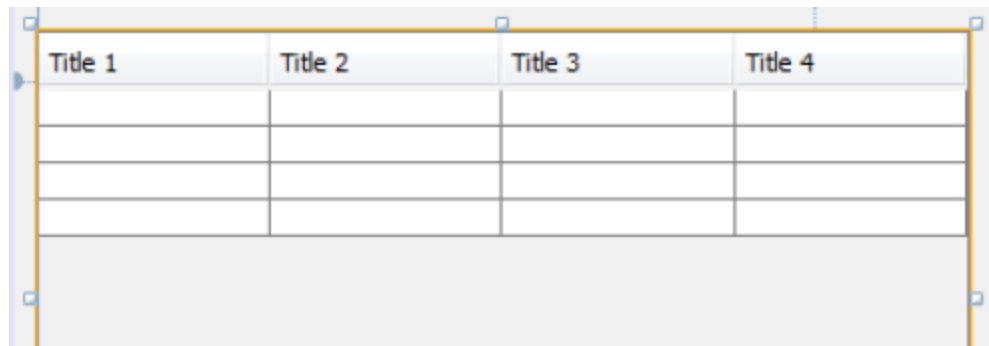
- Mampu membuat aplikasi sederhana menggunakan komponen swing JTable
- Mampu melakukan operasi pada Jtable
- Mampu menambahkan aksi pada aplikasi yang dibuat.



Dasar Teori

- **JTable**

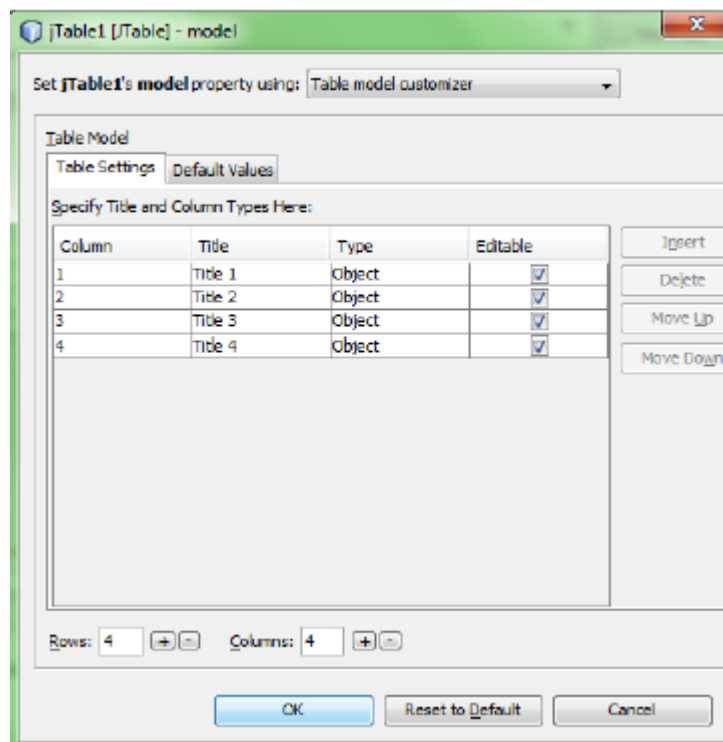
Sama seperti komponen swing lain, table dibentuk object-nya dari class swing JTable dengan default tampilan sebagai berikut:



Title 1	Title 2	Title 3	Title 4

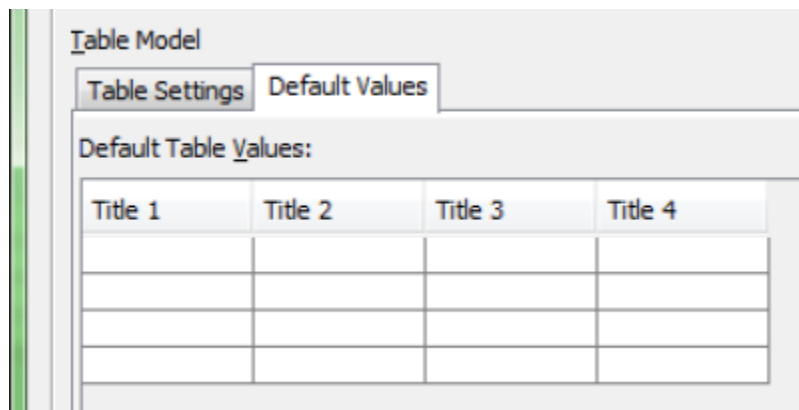
Gambar 3.1 Layout Jtable

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



Gambar 3.2 Konfigurasi Objek JTable

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



Gambar 3.3 Konfigurasi Konten Objek Jtable

- **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:

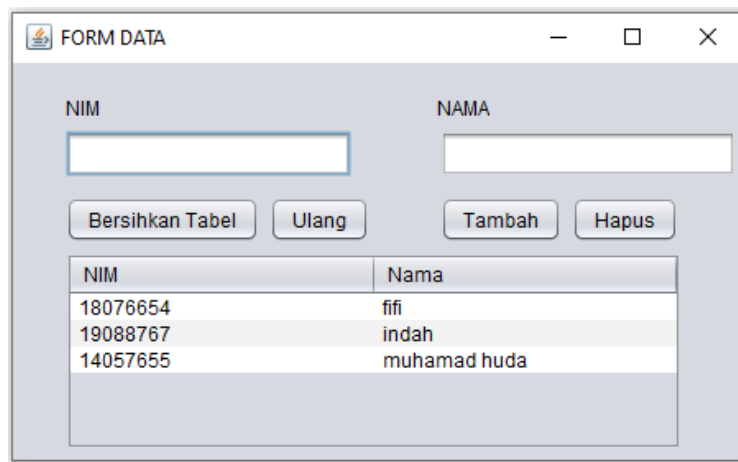
- a) `addColumn`: method ini berfungsi untuk menambah header dari table. Terdapat beberapa versi dari parameter masukan (overloading), tapi yang paling sering digunakan nantinya adalah `addColumn(Object columnName)`.
- b) `addRow`: method ini berfungsi untuk menambah isi dari sebuah data pada model. 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.

Percobaan

Percobaan 1 : Aplikasi Demo Jtable

Contoh aplikasi berikut akan menunjukkan cara kerja JTable. Ada 2 buah JTextField untuk menerima inputan NIM dan Nama. Selanjutnya, jika ditekan tombol **Tambah**, maka data NIM dan Nama tersebut akan berpindah ke JTable. Tombol **Ulang**, Jika ditekan maka data pada isian NIM dan Nama pada JTextField akan terhapus dan memfokuskan cursor pada NIM (JTextField). Di bawah terdapat tombol **Bersihkan JTable**. Jika tombol itu ditekan, maka JTable akan bersih kembali (tidak ada data).

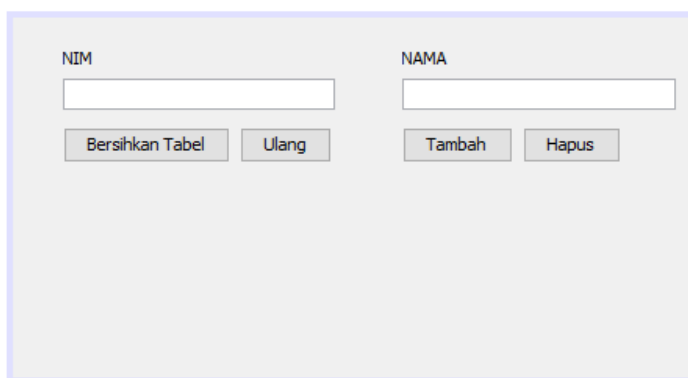




NIM	Nama
18076654	fifi
19088767	indah
14057655	muhamad huda

Langkah-langkah desain form dan pengkodean :

- 1) Letakkanlah komponen JLabel, JTextField, JButton seperti pada Gambar. Sediakan ruang kosong di bawah untuk peletakan JTable.



Atur Variable Name untuk 2 JTextField

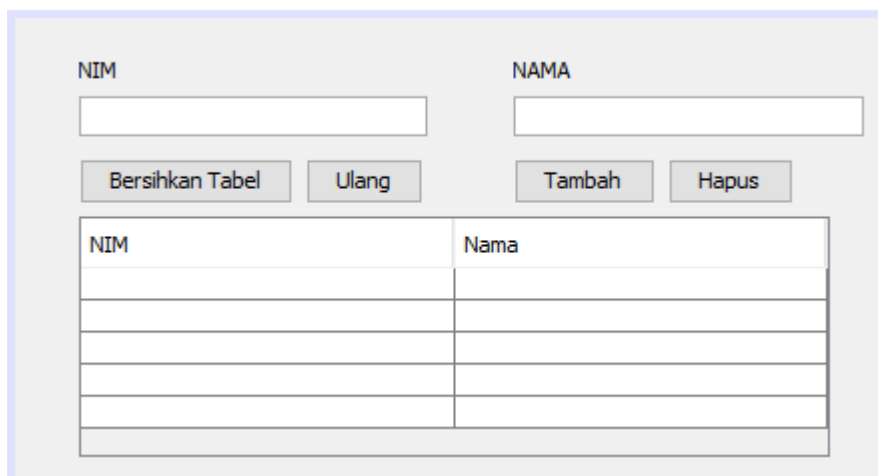
adalah : txt_nim dan txt_nama.

4 JButton adalah tb_ulang (**Ulang**),

tb_tambah (**Tambah**), tb_Hapus (Hapus) dan

tb_bersih (**Bersihkan Table**).

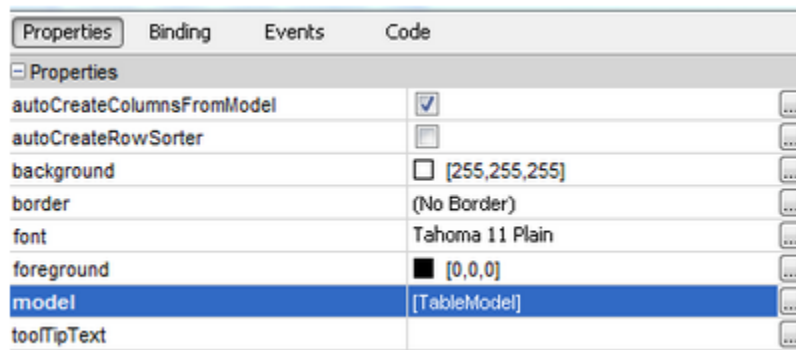
- 2) Letakkan komponen **JTable** di bagian bawah



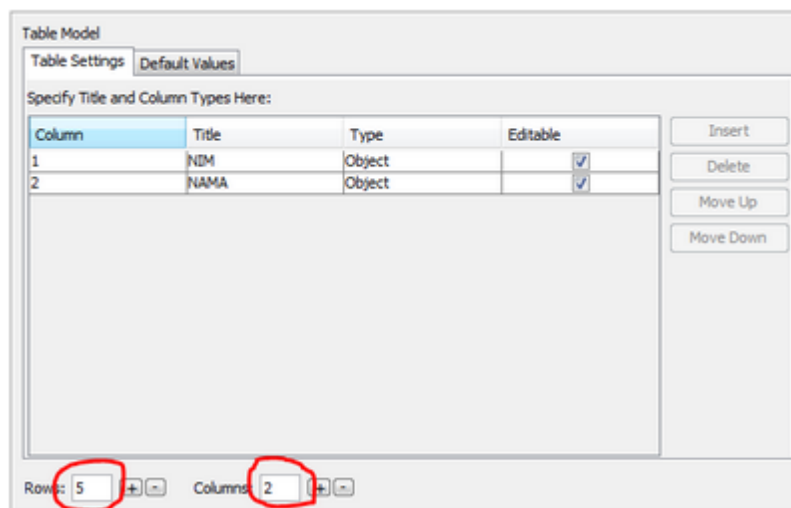
NIM	Nama

Keterangan:

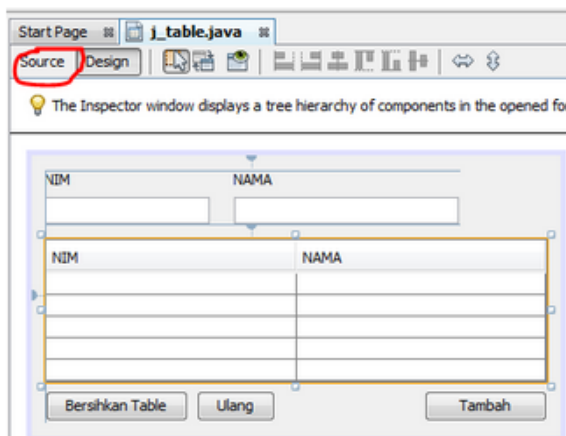
Cara mengatur **JTable** agar nampak seperti Gambar diatas. Klik kanan pada komponen JTable > **Properties** > **model** > (...)



Selanjutnya akan masuk ke bagian ini., sesuaikan saja dengan gambar di bawah (kolom 1 ganti title dengan NIM, kolom 2 ganti title dengan Nama, hapus kolom yang lain. Rows isi 5 dan Column isi 2). Kalau sudah klik Ok dan Close.



3) Klik tab **Source**



Ketikkan kode deklarasi model `JTable` di bagian bawah (sebelum kurung kurawal terakhir).

```
int baris = 0 ;
static Object kolom[] = {" NIM ", " Nama " } ;
DefaultTableModel mdl = new DefaultTableModel(kolom,baris) ;
```

```
185 private javax.swing.JLabel jLabel2;
186 private javax.swing.JScrollPane jScrollPane1;
187 private javax.swing.JTable jTable1;
188 private javax.swing.JTextField txt_nama;
189 private javax.swing.JTextField txt_nim;
190 // End of variables declaration
191 int baris = 0 ;
192 static Object kolom[] = {" NIM ", " Nama " } ;
193 DefaultTableModel mdl = new DefaultTableModel(kolom,baris) ;
194 }
```

- 4) Pada Gambar di atas., Jika terlihat ada tanda lampu di samping kode `DefaultTableModel`...Klik tanda lampu tersebut dan pilih Add import for javax.swing.table.DefaultTableModel
- 5) Akan tercipta kode : import javax.swing.table.DefaultTableModel; terletak di bagian paling atas.

```
11 package javaapplication2;
12
13 import javax.swing.table.DefaultTableModel;
14
15 /**
16  *
17  * @author iLuphi
18  */
```

- 6) Pastikan Frame dalam posisi aktif (terpilih). Klik kanan pada **JFrame (JFrame Form) > Events > Component > componentShown**. Ketikkan kode berikut di dalamnya :

```
jTable1.setModel(mdl);
```

```
218 private void formComponentShown(java.awt.event.ComponentEvent
219     // TODO add your handling code here:
220     jTable1.setModel(mdl);
221 }
```

- 7) Buat event pada tombol Tambah (klik kanan **tombol Tambah > Events > Action > actionPerformed**). Isikan kode berikut :

```
private void tmbTambahActionPerformed(java.awt.event.ActionEvent evt) {
    mdl.addRow(new Object []
    {txt_nim.getText(),txt_nama.getText() } ) ;
    jTable1.setModel(mdl) ;
}
```

- 8) Buat event pada tombol Hapus (klik kanan **tombol Hapus > Events > Action > actionPerformed**). Isikan kode berikut:

```
private void hapusButtonActionPerformed(java.awt.event.ActionEvent evt) {
    int row = TabelData.getSelectedRow();
    if(row>=0){
        int ok=JOptionPane.showConfirmDialog(null, "Yakin Mau Hapus?", "Konfirmasi",
        JOptionPane.YES_NO_OPTION);
        if(ok==0){
```



```

        mdl.removeRow(row); }
    }
    jTable1.setModel(mdl);
}

```

- 9) Buat event pada tombol Bersihkan JTable (klik kanan **tombol Bersihkan JTable > Events > Action > actionPerformed**). Isikan kode berikut :

```

private void bt_bersihActionPerformed(java.awt.event.ActionEvent evt) {
    mdl.getDataVector().removeAllElements();
    mdl.fireTableDataChanged();
    jTable1.setModel(mdl);
    txt_nim.requestFocus();
}

```

- 10) Buat event pada tombol Ulang (klik kanan **tombol Ulang > Events > Action > actionPerformed**). Isikan kode berikut :

```

private void bt_ulangActionPerformed(java.awt.event.ActionEvent evt) {
    txt_nama.setText("");
    txt_nim.setText("");
    txt_nim.requestFocus();
}

```

Latihan

Latihan 1 : Aplikasi Biodata Mahasiswa

Buatlah aplikasi biodata mahasiswa dengan tampilan berikut :

Nama	NIM	Jurusan	Alamat	Phone
Romi Satria W...	12387	Teknik Inform...	Jakarta	08158231

Keterangan :

- Aplikasi dengan frame lengkap beserta menubar dan menuitem
- Data diisi kemudian diklik Tambahkan maka data akan masuk ke table
- Bersihkan: membersihkan isian di Textfield
- Hapus: salah satu record di table di klik kemudian klik hapus maka akan menghapus record di table
- Simpan: menyimpan record di file (simpan dalam file .txt)
- Keluar: keluar aplikasi



BAB 4

LAYOUT

POKOK BAHASAN

Percabangan

- BorderLayout
- GridLayout
- GridBagLayout

TUJUAN BELAJAR

Dengan praktikum ini mahasiswa diharapkan dapat:

- Menjelaskan tentang flow layout, border layout, dan grid layout dalam komponen GUI
- Membuat tampilan yang kompleks dalam mendesain aplikasi GUI.



Dasar Teori

- **Layouting**

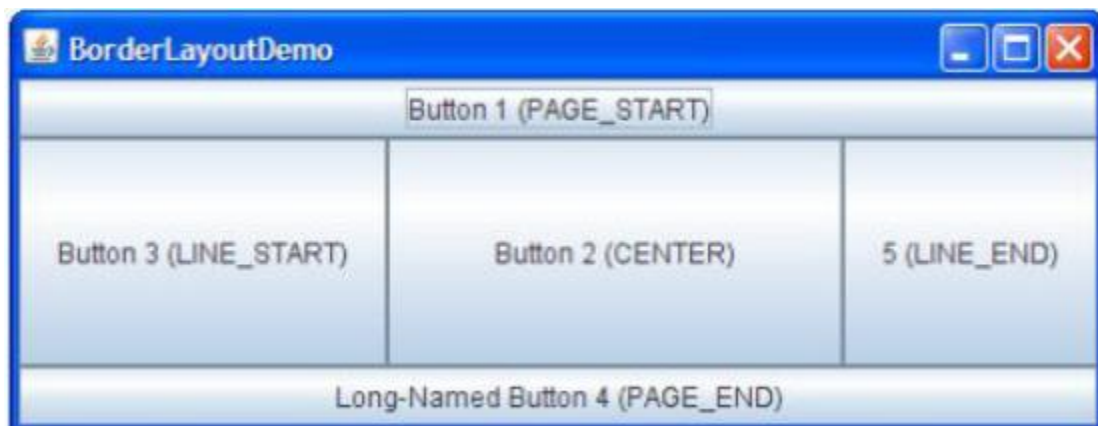
Peletakan komponen pada JFrame mengikuti layout tertentu. Pada drag & drop dari Matisse Builder, layout yang digunakan adalah GroupLayout dengan pengaturan ditekankan pada penambahan gap ataupun container gap. Pada pengkodean awal sebelum terdapatnya builder, pengaturan letak dapat menggunakan beberapa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

- **Border Layout**

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

- PAGE_START
- PAGE_END
- LINE_START
- LINE_END
- CENTER

Ilustrasi:.



Gambar 4.1 Border Layout

- **Grid Layout**

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

Ilustrasi :



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.

- **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:

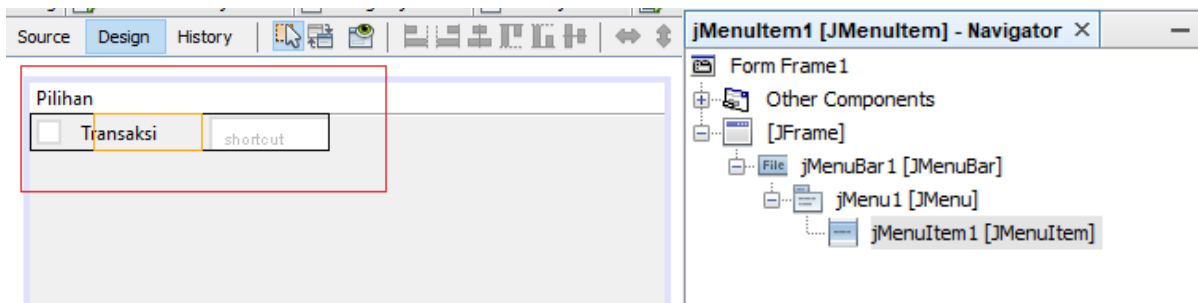


Percobaan

Percobaan 1 : Membuat Sebuah GUI dengan tabbed pane

Berikut ini adalah langkah-langkah membuat GUI dengan multiframe.

1. Buat project baru New → Java Application → beri nama Project → Uncheck Create Main Class
2. Buat JFrame Form baru → beri nama Frame1 → Finish
3. Tambahkan 1 menu bar pada frame, letakkan pada bagian atas. Edit nama File menjadi Pilihan. Tambahkan 1 menu item dengan cara klik kanan pada Pilihan tab Navigator → Add From Palette → Menu Item. Edit Menu Item menjadi Transaksi. (Lihat Gambar)



Tugas :

Buat Frame1 menjadi Main Class Program.

Beri coding pada menu item Transaksi agar ketika di klik maka akan membuka Frame2.

4. Buat JFrame baru → beri nama Frame2 → Finish. Tambahkan komponen-komponen agar Frame2 terlihat seperti gambar di bawah.



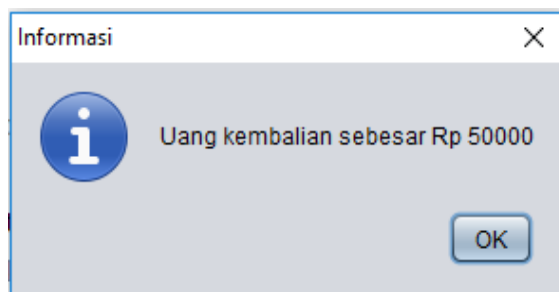
Komponen :

- 1 menu bar
- 2 menu item (di dalam menu bar)
- 1 panel sebagai dasar semua peletakkan komponen
- 5 label
- 3 text field
- 1 combo box
- 2 button

Tugas :

Buatlah coding untuk Frame2 dengan spesifikasi sebagai berikut :

- Isi combo box dengan pilihan tingkat : VII , VIII, IX
- Beri variabel baru dengan nama **total**, kemudian tambahkan kondisi pada coding sebagai berikut :
 - Jika user memilih tingkat VII maka total = 50.000
 - Jika user memilih tingkat VIII maka total = 100.000
 - Jika user memilih tingkat IX maka total = 150.000
- Ketika user sudah memilih tingkat, maka tekan Tombol Ok, kemudian jumlah total akan muncul di Textfield Total (terisi otomatis setelah user memilih tingkat dan klik OK)
- Text field Bayar diisi oleh user. Kemudian klik Bayar.
- Ketika Bayar di klik maka akan muncul pop-up sebagai berikut :



Menampilkan pop-up informasi uang kembalian yang diterima user. Uang kembalian didapatkan dari Bayar dikurangi Total. Tombol Ok untk kembali ke Frame2.

- Beri coding pada menu item **Laporan** agar ketika di dklik maka akan menampilkan Frame3.
5. Buat JFrame baru → beri nama Frame3 → Finish. Tambahkan komponen-komponen agar Frame3 terlihat seperti gambar di bawah.



Transaksi Laporan

Daftar Pembayaran Siswa

Nama	Bayar

Isi tabel pembayaran diambil dari transaksi.

Latihan

Latihan 1 : Membuat Kalkulator

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:

- Tekan angka pertama sebagai operan pertama
- Tekan operator sebagai operasi kedua operan
- Tekan angka kedua sebagai operan kedua.



BAB 5

JDBC

POKOK BAHASAN

Percabangan

- Koneksi Basis Data
- JDBC
- MySQL

TUJUAN BELAJAR

Dengan praktikum ini mahasiswa diharapkan dapat:

- Memahami Koneksi dan Pemrosesan Basis Data di Java
- Memahami JDBC
- Menggunakan MySQL pada program Java



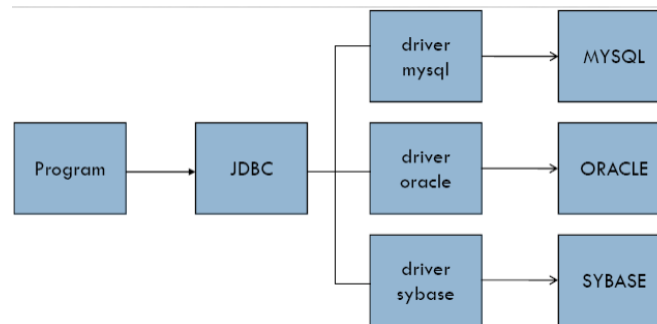
Dasar Teori

- **Persiapan Database**

Install Xampp dari <https://www.apachefriends.org/download.html>. Jika sudah, jalankan MySQL server dan Apache server melalui Xampp Control Panel. Masuk ke localhost/phpmyadmin melalui browser. Jika belum ada database, create new database.

- **JDBC Library**

JDBC adalah suatu fitur di Java yang memungkinkan kita untuk melakukan koneksi ke hampir semua sistem RDBMS yang ada saat ini, eksekusi perintah SQL, dan memproses hasil perintah SQL melalui program Java. Library JDBC terdiri atas class-class yang berguna untuk setiap tasks di dalam pemrosesan database, misalnya class untuk: (1) membuat koneksi ke database, (2) membuat statement menggunakan SQL, (3) mengeksekusi query SQL (statement) di dalam database, (4) menampilkan records yang dihasilkan dan (5) semua class-class JDBC adalah bagian dari **java.sql package**.



Gambar 4.1 Alur akses database melalui JDBC

- **JDBC Driver**

Sebelum Anda dapat menulis program Java yang mengakses database melalui JDBC, pertama-tama Anda harus meng-install sebuah Driver yang menghubungkan class-class di dalam Java's database API dengan database yang akan digunakan. Untuk setiap database yang ada di pasaran saat ini, hampir semuanya memiliki driver JDBC-nya. Biasanya driver ini dapat didownload secara gratis melalui website perusahaan database masing-masing. Driver ini seringkali disebut juga sebagai "connector". Untuk NetBeans IDE, letakkan connector ini di folder Libraries.

- **Enam langkah koneksi dan akses database**

- (1) **Memanggil driver JDBC**

Sebelum kita dapat menggunakan JDBC untuk mengakses database SQL, kita harus melakukan koneksi terlebih dahulu. Langkah pertama dalam melakukan koneksi adalah registrasi driver. Caranya adalah dengan menggunakan method "forName" dari kelas "Class". Misalnya untuk meregistrasi connector MySQL, gunakan perintah berikut:

```
Class.forName("com.mysql.jdbc.Driver");
```



Perlu diingat bahwa method `forName` memiliki exception, sehingga kita harus memasukkan statement ini dalam blok `try-catch`, seperti berikut :

```
try {
    Class.forName("com.mysql.jdbc.Driver");
}
catch (ClassNotFoundException) {
    // error handling
}
```

(2) Mendefinisikan URL untuk koneksi Database

Buatlah sebuah method yang akan me-return sebuah objek `Connection`. Method ini akan memanggil static method class `DriverManager` yaitu `getConnection`. Static method `getConnection` (class `DriverManager`) memiliki 3 parameter yaitu url database, user name, dan password berikut :

```
String url = "jdbc:mysql://localhost/firstjdbc";
String user = "root";
String pw = "";
con = DriverManager.getConnection(url, user, pw);
```

Statement diatas akan melemparkan sebuah `SQLException` jika terjadi kesalahan. Dengan demikian, buat sebuah blok `try-catch` untuk exception ini. Lihat pada percobaan 1 yang merupakan sebuah method yang me-return objek `Connection` yang berfungsi sebagai penghubung ke database MySQL.

(3) Membuat objek statement

Kita memerlukan objek `Statement` untuk melakukan query dan objek ini dapat dibuat dari objek `Connection`.

```
Statement st = con.createStatement();
```

`Statement` adalah interface yang memiliki method penting untuk mengirimkan perintah ke database dan mengembalikan hasil querynya. Gunakan method `executeQuery` method untuk mengeksekusi perintah database dan gunakan method `executeUpdate` untuk melakukan perintah insert, update, atau delete.

(4) Melakukan query atau update

Setelah kita memiliki objek `Statement`, kita dapat menggunakannya untuk mengirimkan query dan mengeksekusinya dengan method `executeQuery` yang menghasilkan objek bertipe `ResultSet`.

ResultSet : interface `ResultSet` merepresentasikan baris yang dihasilkan dari queri. `ResultSet` menyediakan method untuk berpindah dari baris ke baris database.

```
String query = "SELECT * FROM books";
ResultSet rs = st.executeQuery(query);
```

Note: Apabila ingin melakukan insert / update / delete, gunakan **`st.executeUpdate(query);`**



(5) Memproses hasil query

Dalam memproses hasil, kita menggunakan objek resultSet karena hasil query disimpan dalam objek ini. Method utama yang sering digunakan: next dan getString. Contoh pemrosesan hasil query:

```
while (rs.next()){  
System.out.println(rs.getString(1) + " " + rs.getString(2));  
}
```

Kode di atas akan menampilkan semua baris hasil query yang masing-masing menampilkan data kolom pertama dan kedua.

(6) Proses insert/update/delete

Setelah mengerti bagaimana menampilkan data, maka kita perlu mengerti bagaimana menambah / menghapus / mengupdate data ke tabel. Untuk melakukan hal tersebut, kita menggunakan method :

```
executeUpdate("perintah sql untuk insert / update / delete");
```

Method tersebut akan menghasilkan nilai integer yang merupakan jumlah baris yang dipengaruhi oleh proses update tersebut.

```
int i = st.executeUpdate("delete from movie where id = '2' ");
```

Untuk proses updating tabel, kita bisa juga menggunakan objek PreparedStatement. Untuk mengeksekusi PreparedStatement, kita gunakan method executeUpdate().

(7) Menutup koneksi

Sebelum menutup koneksi basis data, kita juga perlu melepaskan objek ResultSet yang ada dengan kode berikut:

```
st.close();
```

Untuk menutup koneksi ke basis data, kita tuliskan sebagai berikut :

```
conn.close();
```

Percobaan

Percobaan 1 : Membuat BookApplication

Langkah 1. Buat project baru dengan nama BookApplication

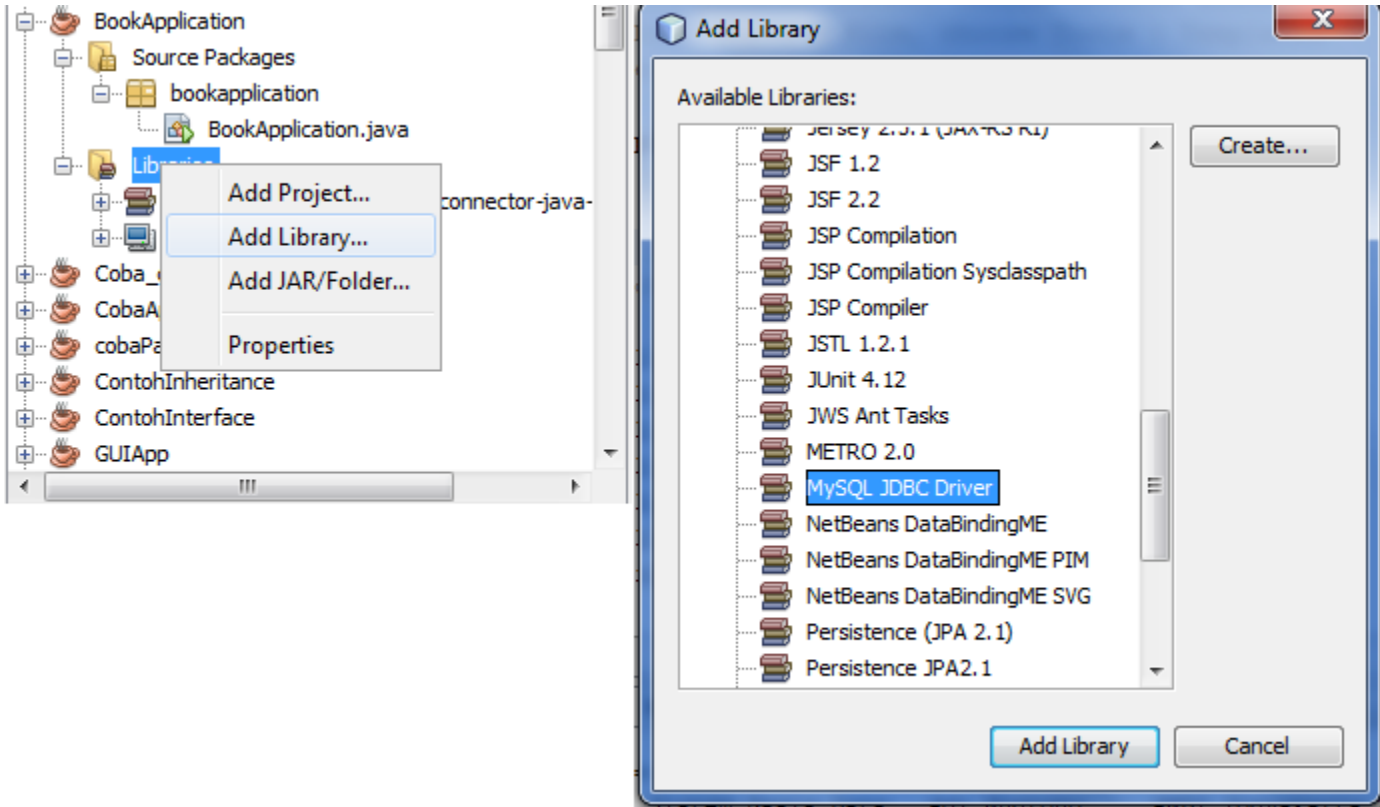
Langkah 2. Instal xampp dan jalankan apache dan mysql server. Jika sudah create database booklibrary, buat 1 tabel “books” dengan query berikut :

```
CREATE TABLE `booklibrary`.`books` ( `idBooks` INT NOT NULL AUTO_INCREMENT , `Nama`  
VARCHAR(50) NOT NULL , `Pengarang` VARCHAR(50) NOT NULL , `Penerbit` VARCHAR(30)  
NOT NULL , `TahunTerbit` INT NOT NULL , PRIMARY KEY (`idBooks`)) ENGINE = InnoDB;
```

Langkah 3. Tambahkan Mysql Connector pada Libraries



Klik kanan pada Libraries → Add Library → Pilih Mysql JDBC Driver → Klik Add Library



Langkah 4. Ketikkan kode berikut pada BookApplication.java

```
package bookapplication;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.Scanner;

public class BookApplication {
    static Scanner sc = new Scanner(System.in);

    public static void main(String[] args) {

        System.out.println("Aplikasi Perpustakaan");
        System.out.println("1. Tampilkan Data");
        System.out.println("2. Tambah Data");
        System.out.println("3. Delete Data");
        System.out.println("Pilihan Anda : ");
        switch (sc.nextInt()) {
            case 1 :
            {
                showSomeData();
                break;
            }
        }
    }
}
```



```

        }
        case 2 :
        {
            insertSomeData();
            break;
        }
        case 3 :
        {
            deleteSomeData();
            break;
        }
        default :
        {
            System.out.println("Pilihan Salah\nSystem Exit");
        }
    }
}

private static void showSomeData() {
    try {
        Connection con = getConnection();
        Statement st = con.createStatement();
        ResultSet rs = st.executeQuery("select * from books");
        System.out.println ("Daftar Buku : ");
        System.out.println ("=====");

        while (rs.next()) {
            int id = rs.getInt("idBooks");
            String nama = rs.getString("Nama");
            String pengarang = rs.getString("Pengarang");
            String penerbit = rs.getString("Penerbit");
            int thnterbit = rs.getInt("TahunTerbit");
            System.out.println(id + "\t" + nama + "\t" + pengarang + "\t" + penerbit
+ "\t" + thnterbit);
        }
        st.close();
        rs.close();
    }
    catch (SQLException e) {
        System.out.println(e.toString());
    }
}

private static void insertSomeData() {
    try {
        Connection con = getConnection();
        PreparedStatement ps = con.prepareStatement("insert into books " +
            "(Nama, Pengarang, Penerbit, TahunTerbit) " +
            "values (?, ?, ?, ?)");
        System.out.println ("Nama Buku : ");
        String nama = sc.next();

        System.out.println ("Pengarang Buku : ");
        String pengarang = sc.next();

        System.out.println ("Penerbit Buku : ");
        String penerbit = sc.next();
    }
}

```



```

        System.out.println ("Tahun terbit Buku : ");
        int tahunterbit = sc.nextInt();

        ps.setString(1, nama);
        ps.setString(2, pengarang);
        ps.setString(3, penerbit);
        ps.setInt(4, tahunterbit);

        int i = ps.executeUpdate();
        System.out.println(i + "rows added");

        System.out.println("Tampilkan data ? (y/n)");
        if ("y".equalsIgnoreCase(sc.next())) {
            showSomeData();
        }
    }

    catch (SQLException e) {
        System.out.println(e.toString());
    }
}

private static void deleteSomeData() {
    try {
        Connection con = getConnection();
        PreparedStatement ps = con.prepareStatement("delete from books " +
            "where idBooks = ?");
        System.out.println("ID buku = ");
        int id = sc.nextInt();
        ps.setInt(1, id);
        int i = ps.executeUpdate();
        System.out.println(i + "rows deleted");

        System.out.println("Tampilkan data ? (y/n)");
        if ("y".equalsIgnoreCase(sc.next())) {
            showSomeData();
        }
        ps.close();
    }
    catch (SQLException e) {
        System.out.println(e.toString());
    }
}

private static Connection getConnection(){
    Connection con = null;
    try
    {
        Class.forName("com.mysql.jdbc.Driver");
        String url="jdbc:mysql://localhost/book_library";
        String user = "root";
        String pw = "";

        con = DriverManager.getConnection(url, user, pw);
    }
    catch ( ClassNotFoundException e){
        System.out.println (e.getMessage());
    }
}

```



```

catch (SQLException e){
System.out.println (e.getMessage());
}
return con;
}
}

```

Latihan

Latihan 1 : Membuat Aplikasi Telepon

Buatlah sebuah aplikasi telepon dengan tampilan GUI. Buat database telepon. Buat satu table bukutelepon, yang berisi field dengan id sebagai primary key (PK):

- | | |
|--------------|--------------------------|
| 1. id | integer (auto increment) |
| 2. nama | varchar(20) |
| 3. alamat | varchar(50) |
| 4. telepon | varchar(20) |
| 5. handphone | varchar(20) |

Buat koneksi ke database, dan lakukan insert, update dan delete data telepon dari aplikasi tersebut.

Tugas

Membuat aplikasi biodata organisasi.

Buatlah form aplikasi biodata organisasi dengsn spesifikasi berikut:

1. Nama dan Alamat ditulis
2. Pekerjaan (PNS, TNI, Karyawan, Pengusaha) dipilih
3. Jenis Kelamin dipilih
4. Pilih salah satu: Cetak tebal atau Cetak Miring
5. Ketika klik Tampilkan, maka data akan tampil di TextArea
6. Tombol Hapus menghapus semua isian di form
7. Tombol Simpan untuk menyimpan di database (buat id sebagai primary key Auto_Increment)

Hint:

```

Font tebal = new Font("Arial", Font.BOLD, 12)
Font tipis = new Font("Arial", Font.PLAIN, 12)
if(yaTebal.getText().equals("Ya")){
    hasilTextArea.setFont(tebal); }
else{
    hasilTextArea.setFont(tipis); }

```



BAB 6

TUGAS AKHIR PRAKTIKUM

INSTRUKSI

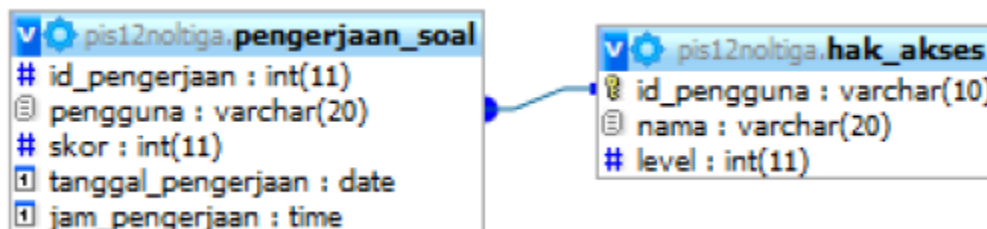
Implementasikan soal kasus di bawah ini dalam kode program!

Conan, Ayumi, Genta dan Mitsuhiro 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 Mitsuhiro, 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 (1-99), 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 Mitsuhiro 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:

- a. Proses pertama saat program dijalankan maka disuruh untuk input nama pengguna
- b. Jika sudah, maka terdapat pilihan sesuai dengan hak akses.
Hak akses level 1 punya 3 pilihan : (1) Kerjakan soal (2) Lihat Hasil (3) Lihat Progres Teman-teman
Hak akses level 2 punya 2 pilihan : (1) Kerjakan soal (2) Lihat Hasil
- c. Jika memilih opsi (1) Kerjakan soal maka : user mengerjakan 10 soal yang di-generate dari program secara otomatis. Setelah selesai mengerjakan soal maka skor ditampilkan dan hasilnya disimpan di database.

Contoh output opsi (1) :

```
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!
```

Catatan : Skor harus disimpan di basis data yah...

- d. Jika memilih opsi (2) Lihat Hasil maka : user akan mengetahui hasil histori pengerjaan soal yang pernah dikerjakannya

Contoh Output opsi (2) :

```
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
```

- e. Jika memilih opsi (3) Lihat Progres Teman-teman bagi hak akses level 1 maka : menampilkan semua hasil pengerjaan yang pernah dikerjakan teman-temannya

Contoh output opsi(3) hak akses level 1:

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

