

LAPORAN PRAKTIKUM
OBJECT ORIENTED PROGRAMING
SEMESTER GANJIL TAHUN AKADEMIK 2023/2024



Disusun oleh :

Nama : Tesalonika Dua Nurak
NIM : 2218015
Prodi : Teknik Informatika S-1
Kelompok : 08

PROGRAM STUDI TEKNIK INFORMATIKA S-1
FAKULTAS TEKNOLOGI INDUSTRI
INSTITUT TEKNOLOGI NASIONAL MALANG
2023

LEMBAR PERSETUJUAN
PRAKTIKUM OBJECT ORIENTED PROGRAMING
SEMESTER GENAP TAHUN AKADEMIK 2023/2024



Disusun Oleh

NAMA : Tesalonika Dua Nurak
NIM : 2218015
PRODI : Teknik Informatika S-1

Mengetahui
Ka. Lab. Pemrograman Komputer

Menyetujui
Dosen Pembimbing

(Ahmad Faisol, ST, MT)

NIP.P :1031000431

(Ahmad Faisol, ST, MT.)

NIP.P :1031000431

PROGRAM STUDI TEKNIK INFORMATIKA S-1
FAKULTAS TEKNOLOGI INDUSTRI
INSTITUT TEKNOLOGI NASIONAL MALANG
2023

KATA PENGANTAR

Dengan memanjatkan puji syukur kehadiran Allah SWT, karena atas berkah rahmat dan karunia-Nya sehingga penulis dapat menyelesaikan Laporan Praktikum *Object Oriented Programing*, guna persyaratan dalam menempuh mata kuliah tersebut.

Laporan ini disusun berdasarkan percobaan dan teori dasar yang ada dalam buku panduan praktikum, teori yang diperoleh praktikan dari perkuliahan, dan tidak lupa yaitu Internet sehingga penulis dapat menambah tidak hanya menguasai teori saja namun juga memahami serta mengaplikasikannya.

Terwujudnya laporan ini, tentunya tidak lepas dari bantuan-bantuan yang telah penulis terima. Pada kesempatan ini, penulis menyampaikan terima kasih yang sebesar-besarnya kepada yang terhormat:

1. Bapak Ahmad Faisol, ST, MT selaku dosen pembimbing Praktikum *Object Oriented Programing*.
2. Bapak Yoseph Agus Pranoto, ST, MT. dan Dedy Rudhistiar, S.Kom, M.Cs. Selaku dosen matakuliah Object Oriented Programing.
3. Bapak Ahmad Faisol, ST, MT selaku Ketua Pelaksana Praktikum *Object Oriented Programing* Program Studi Teknik Informatika ITN Malang.
4. instruktur Lab.Pemrograman Komputer Teknik Informatika yang telah memberi petunjuk kepada penulis selama pelaksanaan praktikum.
5. Rekan-rekan yang telah membantu dalam pelaksanaan dan penyelesaian laporan ini.

Harapan penulis, laporan praktikum ini bermanfaat bagi penulis sendiri maupun pembaca sekalian.

Malang, ... Desember 2023

Penulis

DAFTAR ISI

KATA PENGANTAR	i
DAFTAR ISI.....	ii
DAFTAR GAMBAR	iv
DAFTAR TABEL	vii
BAB I PENDAHULUAN.....	I-1
I.1 Latar Belakang	I-1
I.2 Rumusan Masalah	I-2
I.3 Tujuan	I-2
I.4 Manfaat	I-2
BAB II KONSEP DASAR OOP	II-3
II.1 Landasan Teori	II-3
II.2 Langkah – Langkah Praktikum.....	II-11
II.3 Tugas Praktikum –1:.....	II-11
II.4 Tugas Rumah 1 :	II-15
II.5 Tugas Rumah 2 :	II-16
II.6 Kesimpulan	II-20
BAB III KONSTRUCTOR DAN INHERITANCE	III-1
III.1 Landasan Teori	III-1
III.2 Langkah – Langkah Praktikum	III-3
III.3 Tugas Praktikum 1 :	III-3
III.4 Tugas Praktikum 2 :	III-6
III.5 Tugas Rumah 1 :	III-12
III.6 Tugas Rumah 2 :	III-16
III.7 Kesimpulan.....	III-23
BAB IV ENKAPSULASI , OVERRIDING DAN OVERLOADING	IV-1
IV.1 Landasan Teori.....	IV-1
IV.2 Langkah – Langkah Praktikum	IV-3
IV.3 Tugas Praktikum 1 :	IV-4
IV.4 Tugas Praktikum 2 :	IV-11
IV.5 Tugas Rumah 1 : Implementasi Enkapsulasi di 3 <i>Class</i>	IV-16
IV.6 Tugas Rumah 2:	IV-24

IV.7 Kesimpulan	IV-31
BAB V ABSTRACT, POLIMORFISME, DAN INTERFACE	V-1
V.1 Landasan Teori	V-1
V.2 Langkah – Langkah Praktikum.....	V-8
V.3 Tugas Praktikum 1 :	V-8
V.4 Tugas Praktikum 2 :	V-15
V.5 Tugas Praktikum 3 :	V-18
V.6 Tugas Rumah 1 :	V-20
V.7 Tugas Rumah 2 :	V-23
V.8 Tugas Rumah 3 :	V-26
V.9 Kesimpulan.....	V-30
BAB VI EXCEPTION DAN PENGENALAN DATABASE	VI-31
VI.1 Landasan Teori.....	VI-31
VI.2 Langkah – Langkah Praktikum	VI-35
VI.3 Tugas Praktikum 1 :	VI-35
VI.4 Tugas Praktikum 2 :	VI-40
VI.5 Tugas Praktikum 3 :	VI-47
VI.6 Tugas Praktikum 4 :	VI-54
VI.7 Tugas Rumah 1 :	VI-61
VI.8 Tugas Rumah 2 :	VI-63
VI.9 Tugas Rumah 3 :	VI-69
VI.10 Kesimpulan.....	VI-74
BAB VII KESIMPULAN	VII-75
DAFTAR PUSTAKA	VII-1

DAFTAR GAMBAR

Gambar 2.1 Contoh Class	II-5
Gambar 2.2 Desain form mahasiswa	II-12
Gambar 2.3 Hasil Gui_mahasiswa	II-14
Gambar 2.4 Flowchart Sistem Informasi Wisata	II-15
Gambar 2.5 Desain GUI_Wisata.java	II-17
Gambar 2.6 Tampilan Hasil GUI_Wisata.java	II-19
Gambar 3.1 Desain Gui_Matkul .java.....	III-4
Gambar 3.2 Hasil Tampilan GUI_Matkul.java.....	III-6
Gambar 3.3 Desain GUI_Penilaian.java	III-7
Gambar 3.4 Hasil Tampilan GUI_Penilaian.java	III-11
Gambar 3.5 Desain GUI_Wisata.java	III-13
Gambar 3.6 Hasil Tampilan GUI_Wisata.java	III-15
Gambar 3.7 Desain GUI_Wisata.java	III-17
Gambar 3.8 Hasil Tampilan GUI_Wisata java	III-18
Gambar 3.9 Desain GUI_Login.java	III-19
Gambar 3.10 Hasil Tampilan Login.java	III-20
Gambar 3.11 Desain GUI_Reservasi.java	III-21
Gambar 3.12 Hasil Tampilan GUI_Reservasi.java.....	III-22
Gambar 4.1 Desain GUI_Nilai.java	IV-5
Gambar 4.2 Hasil Tampilan GUI_Nilai.java	IV-10
Gambar 4.3 Desain GUI_Mahasiswa.java	IV-12
Gambar 4.4 Hasil Tampilan GUI_Mahasiswa.java	IV-15
Gambar 4.5 Desain GUI_Wisata.java.....	IV-17
Gambar 4.6 Hasil Tampilan GUI_Wisata.java	IV-19
Gambar 4.7 Desain GUI_Login.java	IV-20
Gambar 4.8 Hasil Tampilan GUI_Login.java.....	IV-21
Gambar 4.9 Desain GUI_Reservasi.java	IV-23
Gambar 4.10 Hasil Tampilan GUI_Reservasi.java.....	IV-24
Gambar 4.11 Desain GUI_Wisata.java.....	IV-26
Gambar 4.12 Hasil Tampilan GUI_Wisata.java	IV-27
Gambar 4.13 Desain GUI_Reservasi.java	IV-28
Gambar 4.14 Hasil Tampilan GUI_Reservasi.java.....	IV-30
Gambar 5.1 Konsep Updasting dan Downcasting	V-6
Gambar 5.2 <i>Diagram Class</i> Penilaian.....	V-9
Gambar 5.3 <i>Diagram Class</i> KeaktifanMahasiswa.....	V-10
Gambar 5.4 Desain Gui_Penilaian.java	V-11
Gambar 5.5 Hasil Tampilan GUI_Penilaian.java	V-14
Gambar 5.6 <i>Diagram Class</i> Data_kuliah.....	V-15
Gambar 5.7 Desain Gui_Matkul.java.....	V-15
Gambar 5.8 Hasil Tampilan GUI_Matkul.java.....	V-18

Gambar 5.9 Desain Gui_MenuUtama.java	V-19
Gambar 5.10 Hasil Tampilan GUI_MenuUtama.java	V-20
Gambar 5.11 Hasil Tampilan Diagram Hitung	V-20
Gambar 5.12 Desain Gui_Reservasi.java.....	V-21
Gambar 5.13 Hasil Tampilan GUI_Reservasi.java.....	V-22
Gambar 5.14 Hasil Tampilan Diagram Hitung	V-23
Gambar 5.15 Desain Gui_Reservasi.java.....	V-24
Gambar 5.16 Hasil tampilan gui_reservasi	V-26
Gambar 5.17 Desain Gui_Daftar.java.....	V-27
Gambar 5.18 Hasil Tampilan Login Berhasil.java.....	V-29
Gambar 5.19 Hasil Tampilan Login gagal.java	V-29
Gambar 6.1 Desain Gui_Penilaian.java	VI-36
Gambar 6.2 Hasil Tampilan GUI_Penilaian.java	VI-39
Gambar 6.3 Create database oop_2118112.....	VI-40
Gambar 6.4 Membuat table(tb_mahasiswa)	VI-40
Gambar 6.5 Menambahkan <i>Library Connection</i>	VI-40
Gambar 6.6 Desain Form Gui_Mahasiswa	VI-44
Gambar 6.7 Tampilan Hasil Method tampil() Gui_Mahasiswa.....	VI-46
Gambar 6.8 Tampilan Hasil Method tambah() Gui_Mahasiswa	VI-46
Gambar 6.9 Tampilan Hasil Method ubah() Gui_Mahasiswa	VI-46
Gambar 6.10 Tampilan Hasil Method hapus() Gui_Mahasiswa.....	VI-46
Gambar 6.11 Tampilan Hasil Method batal() Gui_Mahasiswa	VI-47
Gambar 6.12 Membuat table (tb_nilai).....	VI-47
Gambar 6.13 Menambahkan <i>Library Connection</i>	VI-47
Gambar 6.14 Desain Form Gui_Nilai	VI-51
Gambar 6.15 Tampilan Hasil Method tampil() Gui_Mahasiswa.....	VI-53
Gambar 6.16 Tampilan Hasil Method tambah() Gui_Mahasiswa	VI-53
Gambar 6.17 Tampilan Hasil Method ubah() Gui_Mahasiswa	VI-54
Gambar 6.18 Tampilan Hasil Method hapus() Gui_Mahasiswa.....	VI-54
Gambar 6.19 Tampilan Hasil Method batal() Gui_Mahasiswa	VI-54
Gambar 6.20 Membuat table (tb_matkul).....	VI-54
Gambar 6.21 Menambahkan <i>Library Connection</i>	VI-54
Gambar 6.22 Desain Form Gui_Matkul.....	VI-58
Gambar 6.23 Tampilan Hasil Method tampil() Gui_Mahasiswa.....	VI-59
Gambar 6.24 Tampilan Hasil Method tambah() Gui_Mahasiswa	VI-60
Gambar 6.25 Tampilan Hasil Method ubah() Gui_Mahasiswa	VI-60
Gambar 6.26 Tampilan Hasil Method hapus() Gui_Mahasiswa.....	VI-60
Gambar 6.27 Tampilan Hasil Method batal() Gui_Mahasiswa	VI-60
Gambar 6.28 Tampilan Desain Daftar.java.....	VI-61
Gambar 6.29 Tampilan Login Berhasil.java.....	VI-62
Gambar 6.30 Tampilan Login Gagal.java.....	VI-63
Gambar 6.31 Membuat database oop_2218015.....	VI-63

Gambar 6.32 Membuat table(tb_list)	VI-63
Gambar 6.33 Menambahkan <i>Library Connection</i>	VI-65
Gambar 6.34 Desain Form GUI_Wisata	VI-65
Gambar 6.35 Tampilan Hasil Method Tampi() GUI_Wisata	VI-68
Gambar 6.36 Membuat table(tb_Reservasi)	VI-69
Gambar 6.37 Menambahkan <i>Library Connection</i>	VI-70
Gambar 6.38 Desain Form GUI_Reservasi	VI-71
Gambar 6.39 Tampilan Hasil Method Tampi() GUI_Reservasi	VI-73

DAFTAR TABEL

Tabel 2.1 Properti Desain GUI_Mahasiswa.java.....	II-12
Tabel 2.2 Properti Desain GUI_Wisata.java.....	II-17
Tabel 3.1 Properti Desain GUI_Matkul.java	III-4
Tabel 3.2 Properti Desain GUI_Penilaian.java	III-8
Tabel 3.3 Properti Desain GUI_CetakPenjualanKaos.java.....	III-13
Tabel 3.4 Properti Desain GUI_Wisata.java.....	III-17
Tabel 3.5 Properti Desain Login.java	III-19
Tabel 3.6 Properti Desain GUI_Reservasi.java	III-21
Tabel 4.1 Properti Desain GUI_Nilai.java	IV-6
Tabel 4.2 Properti Desain GUI_Mahasiswa.java.....	IV-12
Tabel 4.3 Properti Desain GUI_Wisata.java.....	IV-17
Tabel 4.4 Properti Desain GUI_Login.java	IV-20
Tabel 4.5 Properti Desain GUI_Reservasi.java	IV-23
Tabel 4.6 Properti Desain GUI_Wisata.java.....	IV-26
Tabel 4.7 Properti Desain GUI_Resrvasi.java	IV-29
Tabel 5.1 Properti Desain GUI_Penilaian.java.....	V-11
Tabel 5.2 Properti Desain GUI_Matkul.java	V-15
Tabel 5.3 Properti Desain GUI_MenuUtama.java.....	V-19
Tabel 5.4 Properti Desain GUI_Reservasi.java	V-21
Tabel 5.5 Properti Desain GUI_Login.java	V-25
Tabel 5.6 Properti Desain GUI_Login.java	V-28
Tabel 6.1 Properti Desain GUI_Penilaian.java.....	VI-37
Tabel 6.2 Properti Desain GUI_Mahasiswa (Database)	VI-44
Tabel 6.3 Properti Desain GUI_Nilai(Database)	VI-51
Tabel 6.4 Properti Desain GUI_Matkul (Database).....	VI-58
Tabel 6.5 Properti Desain GUI_Matkul (Database).....	VI-62
Tabel 6.6 Properti Desain GUI_Wisata (Database)	VI-66
Tabel 6.7 Properti Desain GUI_Reservasi(Database).....	VI-71



BAB I

PENDAHULUAN

I.1 Latar Belakang

OOP (*Object Oriented Programming*) adalah suatu metode pemrograman yang berorientasi kepada objek. Tujuan dari OOP diciptakan adalah untuk mempermudah pengembangan program dengan cara mengikuti model yang telah ada di kehidupan sehari-hari. Jadi setiap bagian dari suatu permasalahan adalah objek, nah objek itu sendiri merupakan gabungan dari beberapa objek yang lebih kecil lagi. Salah satu contoh perbedaan yang sangat sederhana antara pemrograman prosedural dengan Pemrograman berorientasi objek adalah pada pendefinisian, konstanta atau fungsi. Di dalam pemrograman prosedural seluruh, konstanta, ataupun fungsi yang dibutuhkan didalam rogram wajib didefinisikan sementara di dalam programan beorientasi objek semua kebutuhan, konstanta atau fungsi tersebut cukup dibuatkan di dalam sebuah objek.

Konsep dari OOP sendiri adalah semua pemecahan masalah dibagi ke dalam objek. Proses perancangan atau desain dalam suatu pemrograman merupakan proses yang tidak terpisah dari proses yang mendahului, yaitu analisis dan proses yang mengikutinya. Pembahasan mengenai orientasi objek tidak akan lepas dari konsep objek seperti *inheritance* atau penurunan, *encapsulation* atau pembungkusan, dan *polymorphism* atau kebanyakrapaan. Konsep – konsep ini merupakan fundamental dalam orientasi objek yang perlu sekali dipahami serta digunakan dengan baik, dan menghindari penggunaanya yang tidak tepat.

Model data berorientasi objek dikatakan dapat memberi fleksibilitas yang lebih, kemudahan mengubah program, dan digunakan luas dalam teknik piranti lunak skala besar. Lebih jauh lagi, pendukung OOP mengklaim bahwa OOP lebih mudah dipelajari bagi pemula dibanding dengan pendekatan sebelumnya, dan pendekatan OOP lebih mudah dikembangkan dan dirawat.

**I.2 Rumusan Masalah**

1. Apa itu OOP?
2. Apa tujuan dari OOP?
3. Bagaimana pengaplikasian OOP?

I.3 Tujuan

1. Mampu memahami dengan konsep OOP
2. Mampu menerapkan dan menguasai berdasarkan konsep OOP
3. Mampu membuat program dari OOP

I.4 Manfaat

1. Dapat mengetahui konsep dasar maupun lanjutan dari OOP
2. Dapat mengaplikasikan OOP
3. Dapat mengerti fungsi-fungsi keyword yang terdapat dalam OOP

Nama Aslab :	TTD :
Firman Frezy Pradana 2118112	
Tanggal : 17 Desember 2023	



BAB II

KONSEP DASAR OOP

Jumlah Pertemuan	:	2 x 60 menit
Tujuan Praktikum	:	<ol style="list-style-type: none">1. Praktikan mampu menerapkan konsep dasar pemrograman berorientasi objek.2. Praktikan mampu membuat sebuah <i>Class</i>, atribut, <i>method</i> dan objek.3. Praktikan mampu mengetahui fungsi <i>method getter</i> dan <i>setter</i>.4. Praktikan mampu membuat program untuk memasukkan data mahasiswa berbasis GUI (<i>Graphical User Interface</i>).
Alat / bahan	:	<ol style="list-style-type: none">1. Seperangkat <i>computer</i>.2. Perangkat lunak: <i>Netbeans</i>.3. Modul Praktikum <i>OOP</i> 2022.

II.1 Landasan Teori

A. Pengenalan *OOP*

OOP (Object Oriented Programming) adalah suatu metode pemrograman yang berbasis kepada objek. Tujuan dari *OOP* adalah untuk mempermudah pengembangan program, dengan mengadopsi model yang mirip dengan objek-objek di kehidupan sehari-hari. Dalam kehidupan sehari-hari, kita berinteraksi dengan objek-objek yang memiliki karakteristik atau atribut tertentu dan dapat melakukan tindakan atau perilaku tertentu. Sebagai contoh mobil, mobil adalah sebuah objek yang terbentuk dari beberapa *atribut* seperti roda, kursi, kemudi, mesin, dan lain-lain. Setiap atribut ini, memiliki peran dan fungsinya sendiri dalam membentuk objek mobil secara keseluruhan. Mobil sebagai objek yang terbentuk dari *attribute* yang saling berhubungan dan berinteraksi. Begitu juga dengan program, sebuah objek dalam program dapat terdiri dari beberapa atribut dan metode yang saling berhubungan dan berinteraksi. Atribut ini mewakili data atau informasi yang terkait dengan objek,



sementara metode (*Method*) digunakan untuk menjalankan tindakan atau operasi pada objek tersebut.

B. Class & Object

1. Pengertian Class

Class merupakan rancangan dari sebuah objek yang mendefinisikan atribut (ciri/variabel) dan method (perilaku) umum dari suatu objek yang dibuat, berdasarkan *class* tersebut. Dalam OOP, sebuah program yang berjalan wajib untuk mendefinisikan *class* terlebih dahulu sebelum dapat membuat objek atau melakukan operasi lainnya, karena *Class* merupakan struktur dasar yang digunakan untuk menciptakan objek-objek dalam OOP.

Aturan-aturan saat pendeklarasian *class* di Java:

- a. Penamaan *class* harus dimulai dengan huruf (a-z atau A-Z), atau karakter garis bawah (_).
- b. Hanya boleh ada satu *class public* dalam satu file .java, *non public class* boleh lebih dari satu di dalam satu file .java
- c. Nama *class public* harus sama dengan nama file .java
- d. Komentar bisa diletakkan di mana saja
- e. Jika *class* berada dalam sebuah package, maka harus ada deklarasi nama package di bagian paling atas dengan format “ package nama_package ;“
- f. Import berada antara deklarasi package dan deklarasi *class*

Jenis-jenis *class* yang umum di Java:

- a. *Base / Parent Class* (Kelas Dasar / Induk)

Kelas yang berisi atribut dan metode dasar yang diwariskan ke kelas turunan dan Bertugas membentuk dasar hierarki objek dalam pewarisan.

- b. *Derived / Child Class* (Kelas Turunan / Anak)

Kelas yang mewarisi atribut dan metode dari kelas dasar/induk (*Base/Parent Class*). Kelas ini dapat memiliki atribut dan metode tambahan serta dapat meng-*override* metode dari kelas dasar/induk (*Base/Parent Class*).



c. *Abstract Class* (Kelas Abstrak)

Kelas yang tidak dapat diinstansiasi secara langsung. Class biasanya digunakan sebagai kerangka untuk kelas-kelas turunan dengan definisi metode abstrak yang harus diimplementasikan oleh kelas turunan.

d. *Interface* (Kelas Antarmuka)

Serupa dengan kelas abstrak, tetapi hanya berisi metode abstrak dan konstanta. Kelas lain dapat mengimplementasikan beberapa antarmuka, mendukung multiple inheritance dari perilaku.

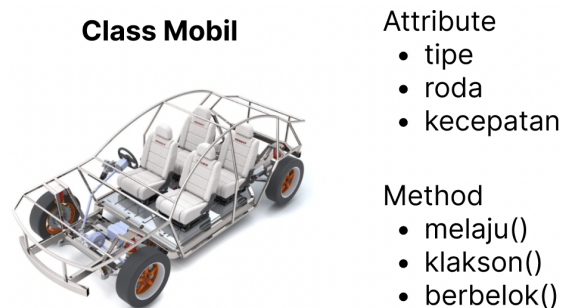
e. *Final Class* (Kelas Final)

Kelas yang tidak dapat diwariskan oleh kelas lain (tidak dapat menjadi kelas dasar). Biasanya digunakan untuk mencegah modifikasi kelas dalam pewarisan.

Berdasarkan peran atau karakteristik, class terbagi menjadi 2 jenis:

- Object Class* : Class yang tidak ada main, biasanya berisi *method* dan *attribute* yang nantinya akan di panggil di *Drive Class*.
- Drive Class* : Class yang menggunakan main atau dapat di running.

Contoh dari class:



Gambar 2.1 Contoh Class

Class dideklarasikan sebagai berikut:

```
class NamaClass{  
    //deklarasi atribut  
    //deklarasi method  
}
```



Keterangan :

- a. `class` adalah kata kunci yang digunakan untuk mendeklarasikan suatu kelas.
- b. `NamaClass` merupakan indentifier.
- c. Atribut dan Method bisa berjumlah 0 atau lebih.

2. Pengertian Object

Object adalah realisasi dari *class*. Dapat di bayangkan bahwa *class* adalah sebuah cetakan/*template* , dan *object* adalah bentuk dari representasi cetakan/*template* dari *class*. Setiap *object* akan mempunyai *state* / keadaan (*instance* variabel/*properties*) yang membedakan satu *object* dengan *object* lain. Kemudian *object* juga mempunyai *behaviour* (*method*) di mana *logic* dari *class* disimpan. Terdapat istilah “*instansiasi*” dalam OOP yaitu proses pembuatan *object real* dari *class*.

Ketika kita membuat sebuah *class* bukan berarti kita membuat sebuah objek. Ciri-ciri pembuatan objek adalah dengan adanya operator “*new*”.

Object dideklarasikan sebagai berikut:

```
NamaClass namaObjek = new NamaClass();
```

3. Pengertian Atribut

Atribut adalah karakteristik unik atau ciri dari sebuah objek. Karakteristik tersebut dapat berupa data/variabel yang akan dimiliki oleh objek dari kelas tersebut. Atribut dapat memiliki hak akses *private*, *public* maupun *protected* (akan dijelaskan di bab 4).

Attribute dideklarasikan sebagai berikut:

```
Hak_akses tipe_data nama_Atribut;
```

Berikut adalah beberapa jenis atribut umum dalam OOP Java:

a. Atribut *Instance* (*Instance Variables*)

Atribut yang berkaitan dengan objek yang dihasilkan dari *class* yang dipakai. Setiap objek memiliki salinan unik dari atribut ini.



Atribut instance mendefinisikan karakteristik atau properti unik dari objek.

b. Atribut *Static* (*Static Variables*)

Atribut yang terkait dengan *class* itu sendiri, bukan dengan objek. Tidak ada salinan unik dari atribut untuk setiap objek. Dideklarasikan dengan kata kunci *static* sebelum tipe data atribut. Biasanya digunakan untuk menyimpan data yang bersifat bersama antara semua objek kelas.

c. Atribut Final (*Final Variables*)

Atribut yang nilai awalnya harus ditetapkan saat deklarasi dan tidak dapat diubah lagi setelahnya. Atribut final digunakan untuk menyimpan konstanta atau nilai tetap yang tidak boleh diubah.

```
public class Mobil {  
  
    // Atribut instance  
    String merk;  
    int tahunProduksi;  
  
    // Atribut static  
    static int jumlahMobil;  
  
    // Atribut final  
    final int maksKecepatan = 200;  
  
    // Atribut final  
    static final String WARNA_MERAH = "Merah";  
}
```

4. Pengertian Method

Method adalah aksi atau tindakan yang dapat dilakukan oleh objek dari suatu kelas. Dalam Pemrograman Berorientasi Objek (OOP)



sebuah method biasanya berupa blok kode yang berisi serangkaian pernyataan atau instruksi yang dapat dieksekusi. *Method* beroperasi pada objek atau kelas dan dapat digunakan untuk melakukan tindakan tertentu seperti mengembalikan nilai, atau mengubah keadaan objek. *Method* biasanya digunakan untuk mengorganisir dan mengelompokkan kode, sehingga tindakan atau fungsi tertentu dapat dijalankan dengan cara yang terstruktur.

Dalam Pemrograman Berorientasi Objek (OOP) menggunakan bahasa Java, terdapat beberapa jenis metode yang dapat didefinisikan dalam kelas. Berikut adalah beberapa jenis metode umum dalam OOP Java:

a. *Instance Methods*

Instance Methods merupakan metode dalam OOP yang terkait langsung dengan objek spesifik yang diciptakan dari suatu kelas. *Method* instance bekerja pada atribut dan perilaku objek tertentu yang diwakili oleh instansiasi objek yang telah dibuat dari kelas tersebut. *Method instance* dapat mengakses dan memanipulasi atribut objek, serta melakukan berbagai tindakan yang relevan terhadap objek.

```
public class Mahasiswa {
    String nama;

    int umur;

    // Metode instance untuk menampilkan informasi mahasiswa

    public void tampilkanInfo() {

        System.out.println("Nama: " + nama);

        System.out.println("Umur: " + umur + " tahun");

    }

    // Metode instance untuk mengubah umur mahasiswa

    public void ubahUmur(int newUmur) {

        umur = newUmur;

    }

}
```



```
    }  
}  
  
public class Main {  
    public static void main(String[] args) {  
        Mahasiswa mhs1 = new Mahasiswa();  
        mhs1.nama = "Firman";  
        mhs1.umur = 23;  
        mhs1.tampilkanInfo(); // Memanggil metode  
instance  
        mhs1.ubahUmur(21);    // Memanggil metode  
instance untuk mengubah umur  
        mhs1.tampilkanInfo(); // Memanggil metode  
instance lagi  
    }  
}
```

b. *Static Methods*

Static methods adalah *method* yang terkait dengan kelas itu sendiri, bukan dengan objek yang dibuat dari kelas. *method static* diakses melalui nama kelas tanpa perlu membuat objek dari kelas tersebut. Karena *method* tidak beroperasi pada objek spesifik, mereka tidak dapat mengakses atribut non-*static* kelas atau variabel anggota non-*static*.

```
public class staticDemo {  
    static int x;  
    static int y;  
    static int hasil;  
  
    static int jumlah(){  
        hasil = x * y ;  
        return hasil;  
    }  
    public static void main(String[] args) {  
        staticDemo.x = 4;  
        staticDemo.y = 5;  
        System.out.println("Hasil      Penjualan      :  
"+staticDemo.jumlah());  
    }  
}
```



c. *Final Methods*

Final methods adalah metode yang dideklarasikan sebagai final dalam kelas dasar atau "parent class". Metode tersebut tidak dapat di-override oleh kelas turunan atau "child class". Final methods digunakan ketika Anda ingin mencegah kelas turunan untuk mengubah perilaku khusus dari metode yang ada dalam kelas dasar. Method final digunakan untuk menjaga konsistensi perilaku metode tersebut di seluruh aplikasi.

```
public class Animal {
    public final void suara() {
        System.out.println("Hewan      mengeluarkan
suara.");
    }
}
public class kucing extends Animal{
    public void suara() {
        System.out.println("Kucing   mengeluarkan
suara: Meow!");
    }
}
public class exampleFinal {
    public static void main(String[] args) {
        kucing kcng = new kucing();
        kcng.makeSound();

    }
}
```

Berdasarkan return typenya *method* terdiri dari 2 jenis, yakni *Method void* dan *Method Non Void*.

a. *Method Void (Procedure)*

Merupakan *method* yang tidak memiliki nilai balik.

Method Void (Procedure) dideklarasikan sebagai berikut:

```
Void cetakHalo() //tipe_method nama_method

    System.out.println("HELLO  WORLD  !"); //badan
method
}
```

b. *Method Non-Void (Fungsi)*

Merupakan *method* yang mempunyai nilai balik. Nilai yang dikembangkan sebagai hasil fungsi harus bertipe sama dengan tipe fungsi. Nilai itu sendiri berupa data, ekspresi maupun *variable*.



Method Non-Void (Fungsi) dideklarasikan sebagai berikut:

```
int    jumlahRoda(int    x)    //tipe_method
nama_method(parameter)
{
    return x; //nilai kembalian
}
```

Keterangan:

Terdapat *script* “return x;” yang berarti mengembalikan nilai variable “x” sesuai dengan nilai yang tersimpan pada variable “x”.

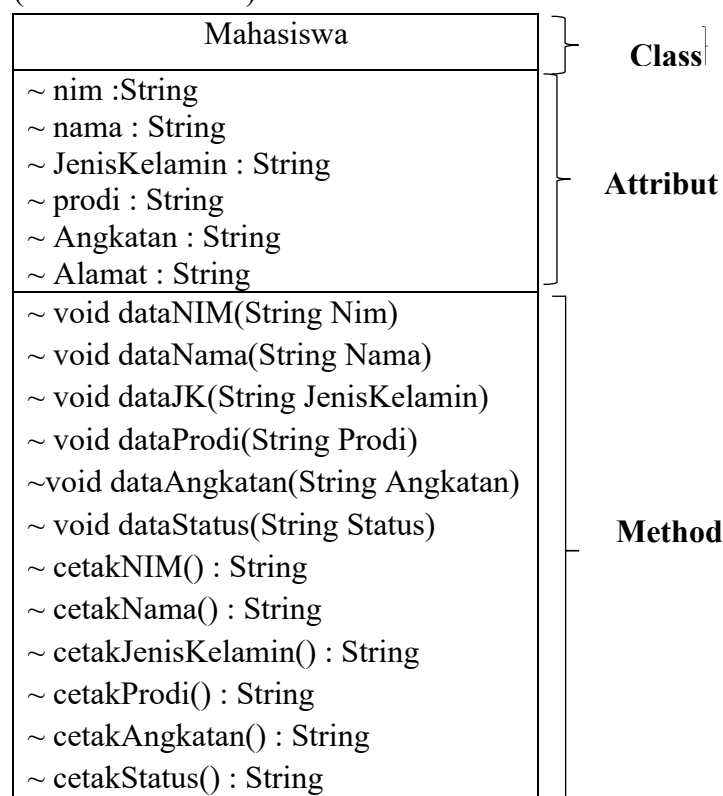
II.2 Langkah – Langkah Praktikum

1. Buka Aplikasi *Netbeans*.
2. Buat *Class*.
3. Memberi *script* pada kelas tersebut.
4. Buat *form*.
5. Memberi *script* pada *form* tersebut.
6. Menjalankan program.

II.3 Tugas Praktikum –1:

Membuat kelas Mahasiswa.java dan GUI_Mahasiswa.java

Diagram Class (Class Mahasiswa):





Source code (Mahasiswa.java) :

```
public class Mahasiswa {
    String nim, nama, prodi, angktn;
    void dataNIM(String Nim){
        this.nim = Nim;
    }
    void dataNama(String Nama){
        this.nama = Nama;
    }
    void dataProdi(String Prodi){
        this.prodi = Prodi;
    }
    void dataAngkatan(String Angktn){
        this.angktn = Angktn;
    }

    String cetakNIM(){
        return nim;
    }
    String cetakNama(){
        return nama;
    }
    String cetakProdi(){
        return prodi;
    }
    String cetakAngkatan(){
        return angktn;
    }
}
```

Desain form (GUI_Mahasiswa.java):

The screenshot shows a Java Swing window titled "Data Mahasiswa". It contains a form with the following elements:

- Labels: Nim, Nama, Jenis Kelamin, Prodi, Angkatan, Alamat.
- Input fields: Text boxes for Nim, Nama, Prodi, Angkatan, and Alamat.
- Radio buttons: Two radio buttons for "Jenis Kelamin", labeled "Laki-laki" and "Perempuan".
- Button: A "Cetak" button at the bottom right.
- Empty area: A large empty rectangular area on the right side of the form.

Gambar 2.2 Desain form mahasiswa

Tabel 2.1 Properti Desain GUI_Mahasiswa.java

No	Komponen	Properti	Nilai
1	jLabel1	Text	DATA MAHASISWA
2	jLabel2	Text	NIM
3	jLabel3	Text	Nama
4	jLabel4	Text	Jenis Kelamin



5	jLabel5	Text	Prodi
6	jLabel6	Text	Angkatan
7	jLabel7	Text	Alamat
8	jTextField1	Name	txtNim
		Text	
9	jTextField2	Name	txtNama
		Text	
10	jTextField3	Name	txtProdi
		Text	
11	jTextField4	Name	txtAngkatan
		Text	
12	jTextField5	Name	txtAlamat
		Text	
13	jRadioButton1	Name	radiobtnLaki
		Text	Laki-laki
14	jRadioButton2	Name	radiobtnPerempuan
		Text	Perempuan
15	jButton1	Name	btnKTM
		Text	Cetak KTM
16	jButton2	Name	btnCLOSE
		Text	Close
17	jTextArea	Name	memoKTM
		Text	

Source code pada button Cetak KTM:

```
private void CetakActionPerformed(java.awt.event.ActionEvent  
evt) {  
    // TODO add your handling code here:  
    // TODO add your handling code here:  
    memoKTM.setText("");  
    Mahasiswa mhs = new Mahasiswa();  
    mhs.dataNIM(txtNim.getText());  
    mhs.dataNama(txtNama.getText());  
    String JenKel="";  
    if (radiobtnLaki.isSelected()) {  
        mhs.dataJenisKelamin(radiobtnLaki.getText());  
    }else{
```



```
mhs.dataJenisKelamin(radiobtnPerempuan.getText());
    }
    mhs.dataProdi(txtProdi.getText());
    mhs.dataAngkatan(txtAngkatan.getText());
    mhs.dataAlamat(txtAlamat.getText());

    memoKTM.append("Kartu Tanda Mahasiswa\n");
    memoKTM.append("-----\n");
    memoKTM.append("NIM                : " + mhs.cetakNIM()
+"\\n");
    memoKTM.append("Nama                : " +
mhs.cetakNama() + "\\n");
    memoKTM.append("Jenis Kelamin      : " +
mhs.cetakJenisKelamin() + "\\n");
    memoKTM.append("Prodi              : " +
mhs.cetakProdi() + "\\n");
    memoKTM.append("Angkatan          : " +
mhs.cetakAngkatan() + "\\n");
    memoKTM.append("Alamat            : " +
mhs.cetakAlamat() + "\\n");
    }
```

Tampilan Hasil :

Data Mahasiswa

Nim: 2118112

Nama: Firman Frezy Pradana

Jenis Kelamin: ☒ Laki-laki ☐ Perempuan

Prodi: Teknik informatika

Angkatan: 2021

Alamat: Trenggalek

Kartu Tanda Mahasiswa

NIM : 2118112
Nama : Firman Frezy Pradana
Jenis Kelamin : Laki-laki
Prodi : Teknik informatika
Angkatan : 2021
Alamat : Trenggalek

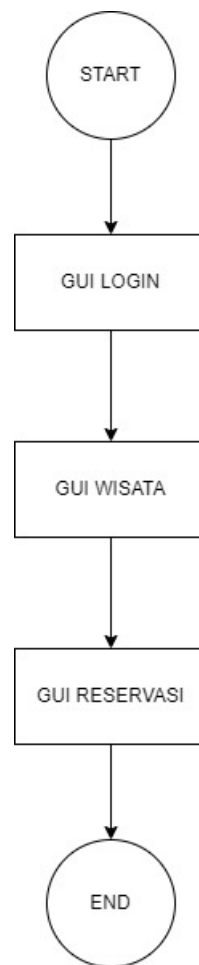
Cetak KTM Close

Gambar 2.3 Hasil Gui_mahasiswa



II.4 Tugas Rumah 1 :

Flowchart Sistem Informasi Wisata



Gambar 2.4 Flowchart Sistem Informasi Wisata

Analisa :

Pertama program akan dimulai dengan menginput nama, Kota, Dan deskripsi wisata. Kemudian program akan menyimpan semua data inputan tadi. Berikutnya program akan menampilkan data input dalam bentuk List wisata yang sesuai.

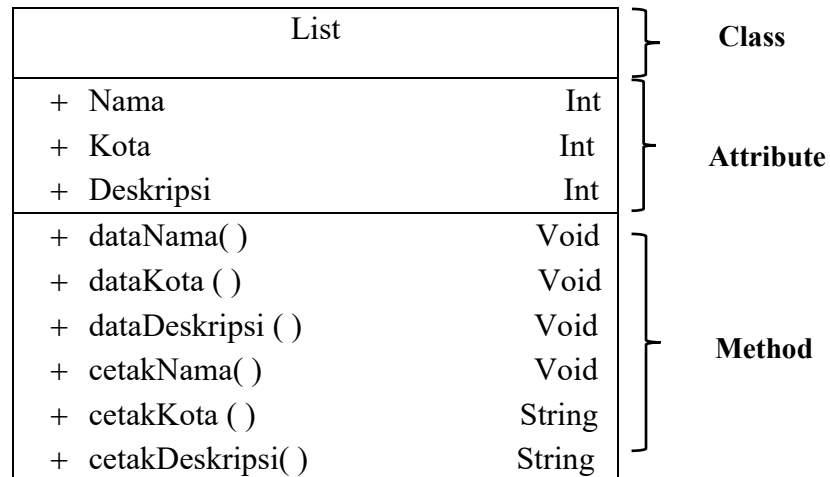


II.5 Tugas Rumah 2 :

Membuat *Class* List.java dan GUI_Wisata.java

Judul : Sistem Informasi Wisata

Diagram Class (List.java):



Source code Object Class/Abstact(Wisata.java)

```
package ProjectPrak;

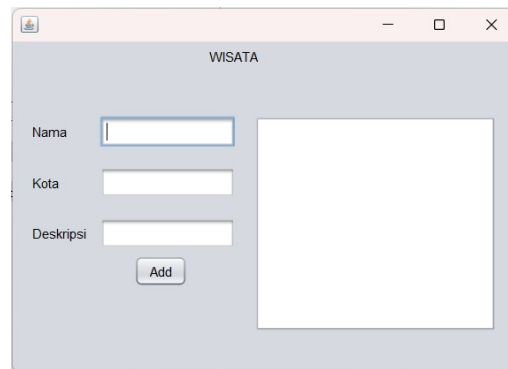
public class List {
    String Nama, Kota, Deskripsi;

    void dataNama(String Nama) {
        this.Nama = Nama;
    }
    void dataKota(String Kota) {
        this.Kota = Kota;
    }
    void dataDeskripsi(String Deskripsi) {
        this.Deskripsi = Deskripsi;
    }

    String cetakNama() { return Nama;
    }
    String cetakKota() { return Kota;
    }
    String cetakDeskripsi() { return Deskripsi;
    }
}
```



Desain *form* (GUI_Wisata.java):



Gambar 2.5 Desain GUI_Wisata.java

Tabel 2.2 Properti Desain GUI_Wisata.java

No	Objek	Properti	Nilai
1	jLabel1	Text	WISATA
2	jLabel2	Text	Nama
3	jLabel3	Text	Kota
4	jLabel4	Text	Deskripsi
5	jTextField1	Name	txtNama
		Text	“ “
6	jTextField2	Name	txtKota
		Text	“ “
7	jTextField3	Name	txtDes
		Text	“ “
8	jButton1	Name	Add
		Text	btnDta
9	jTextArea	Name	-
		Text	memoData
10	jScrollPane1	-	-

*Source code Button/combobox :*

```
private void initComponents() {

    jLabel1 = new javax.swing.JLabel();
    jLabel2 = new javax.swing.JLabel();
    txtNama = new javax.swing.JTextField();
    txtKota = new javax.swing.JTextField();
    jLabel3 = new javax.swing.JLabel();
    jLabel4 = new javax.swing.JLabel();
    txtDes = new javax.swing.JTextField();
    btnData = new javax.swing.JButton();
    jScrollPane1 = new javax.swing.JScrollPane();
    memoData = new javax.swing.JTextArea();

    setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_
    ON_CLOSE);

    jLabel1.setText("WISATA");

    jLabel2.setText("Nama");

    txtNama.setName("txtNama"); // NOI18N

    jLabel3.setText("Kota");

    jLabel4.setText("Deskripsi");

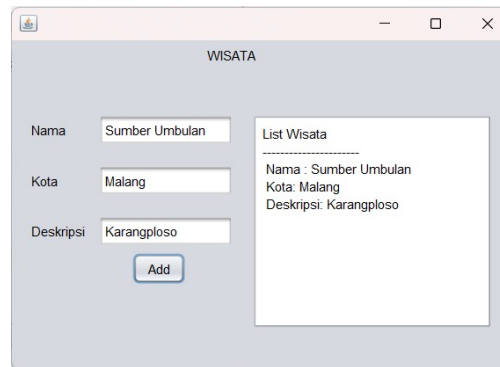
    btnData.setText("Add");
    btnData.addActionListener(new
    java.awt.event.ActionListener() {
        public
        actionPerformed(java.awt.event.ActionEvent evt) {
            btnDataActionPerformed(evt);
        }
    });

    memoData.setText("");
    //      Wisata wst = new Wisata();
    List wst = new List();
    wst.dataNama(txtNama.getText());
    wst.dataKota(txtKota.getText());
    wst.dataDeskripsi(txtDes.getText());

    memoData.append("List Wisata\n");
    memoData.append("-----\n");
    memoData.append(" Nama : " + wst.Nama + "\n");
    memoData.append(" Kota: " + wst.Kota + "\n");
}
```



Hasil Tampilan:



Gambar 2.6 Tampilan Hasil GUI_Wisata.java

Analisa:

Program akan meminta untuk menginputkan nama, lokasi wisata, deskripsi tempat tersebut bertipe string. Kemudian untuk memanggil text area diperlukan variable dari text area berupa memomabe. Karena yang sebelumnya system.out berubah menjadi variable dari text area.



II.6 Kesimpulan

1. OOP (*Object Oriented Programming*) adalah suatu metode pemrograman yang berbasis kepada objek. OOP mengadopsi model yang mirip dengan objek-objek di kehidupan sehari-hari.
2. *Class* merupakan rancangan dari sebuah objek yang mendefinisikan atribut (ciri/variabel) dan *method* (perilaku) umum dari suatu objek yang dibuat. Dalam OOP, sebuah program yang berjalan wajib untuk mendefinisikan *class* terlebih dahulu, sebelum dapat membuat objek atau melakukan operasi lainnya.
3. Atribut adalah karakteristik unik atau ciri dari sebuah objek. Karakteristik tersebut dapat berupa data/variabel yang akan dimiliki oleh objek dari kelas tersebut.

Nama Aslab :	TTD :
Firman Frezy Pradana 2118112	
Tanggal : 17 Desember 2023	



BAB III

KONSTRUCTOR DAN INHERITANCE

Jumlah Pertemuan	:	2 x 60 menit
Tujuan Praktikum	:	<ol style="list-style-type: none">1. Praktikan mampu mengetahui tipe data dan <i>java identifier</i>.2. Praktikan mampu merubah tipe data (<i>Casting</i>).3. Praktikan mampu mengetahui dan membuat <i>method</i> konstruktor.4. Praktikan mampu mengetahui Inheritance
Alat / bahan	:	<ol style="list-style-type: none">1. Seperangkat <i>computer</i>.2. Perangkat lunak: <i>Netbeans</i>.3. Modul Praktikum <i>OOP</i> 2023.

III.1 Landasan Teori

A. Konstruktor

Constructor adalah *method* khusus yang otomatis dieksekusi pada saat menginstansiasi *object* dari *class* tertentu. Secara sintaks, *constructor* mirip seperti *method*, namun *constructor* tidak memiliki *return* nilai pengembalian seperti *method*. *Constructor* digunakan khusus untuk membuat dan menginisialisasi objek baru, sehingga tujuan utamanya untuk memberikan atau mendefinisikan nilai awal pada sebuah *attribute* di dalam *class*. *Constructor* yang tidak memiliki parameter disebut dengan *default constructor*. Setiap *class* pasti memiliki setidaknya satu *constructor*, jika dalam deklarasi *class* tidak ada *constructor* sama sekali, maka *java* secara default akan membuat default constructor.

1. Cara penulisan method Konstruktor

Berikut adalah cara penulisan method konstruktor pada pemrograman java:

Source Code:

```
class NamaClass {  
    Integer a ,b ,c ;  
  
    public NamaClass()
```



```
{  
    a = 1;  
    b = 2;  
    c = 3;  
}  
}
```

Keterangan:

- a. “NamaClass” merupakan nama class dapat di ganti sesuai keinginan
- b. “public NamaClass()” merupakan nama method konstruktor nama harus sesuai dengan nama class yang di naungi.

Nantinya ketika penginisailasian object terjadi maka data dari atribut akan otomatis terisi oleh data yang ada dalam method konstruktor

2. Aturan-aturan Constructor

- a. Mempunyai nama yang sama dengan nama class,
- b. Tidak mempunyai tipe return,
- c. Digunakan untuk menginstansiasi object,
- d. Hanya mempunyai access modifier, tidak ada keyword lain yang diletakkan sebelum nama method pada deklarasi constructor.

B. Inheritance

Inheritance atau Pewarisan/Penurunan adalah konsep pemrograman dimana sebuah class dapat ‘menurunkan’ properti dan *method* yang dimilikinya kepada *class* lain. Konsep *inheritance* ialah membuat sebuah struktur atau ‘*hierarchy*’ *class* dalam kode program, hal tersebut memungkinkan untuk melakukan pewarisan *attribute* atau *method* dari *class* yang umum ke *class* yang lebih spesifik.

Class yang akan ‘diturunkan’ bisa disebut sebagai *class* induk (*parent class*), *super class*, atau *base class*. Sedangkan *class* yang ‘menerima penurunan’ bisa disebut sebagai *class* anak (*child class*), *sub class*, *derived class*. Konsep *inheritance* digunakan untuk memanfaatkan fitur ‘*code reuse*’ untuk menghindari duplikasi kode program. Fungsi dari *inheritance* memperluas fungsi dari *parent class*.



Tidak semua *property* dan *method* dari *class* tidak akan diturunkan. *Property* dan *method* dengan hak akses *private*, tidak akan diturunkan kepada *class* anak. Hanya *property* dan *method* dengan hak akses *public*, *protected* dan *default* saja yang bisa di akses dari *class* anak.

Pada pemrograman berorientasi objek atau OOP, konsep *inheritance* menjadi salah satu topik yang penting. Suatu objek diwariskan dengan menggunakan *keyword extends*.

Istilah penting dalam konsep *inheritance*:

- a. *Super Class* : kelas induk yang mewariskan atribut dan *method* kepada turunannya.
- b. *Sub Class* atau *Child Class* : kelas turunan yang mewarisi atribut dan *method*. *Sub Class* dapat menambah atribut dan *method*-nya sendiri sebagai tambahan dari kelas yang memberi warisan.
- c. *Reusability* : menggunakan kembali atribut dan *method* dari *super class* di *sub class*.

III.2 Langkah – Langkah Praktikum

1. Buka Aplikasi *Netbeans*.
2. Buat *Class*.
3. Memberi *script* pada kelas tersebut.
4. Buat *form*.
5. Memberi *script* pada *form* tersebut.
6. Menjalankan program.

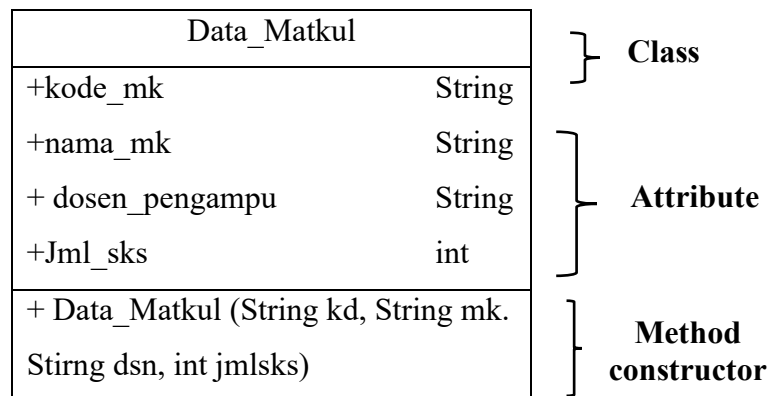
III.3 Tugas Praktikum 1 :

Membuat *Class* *Data_Matkul* dan *GUI_Matkul*

Judul : Konstruktor pada *class* *Data_Matkul*



Diagram Class (Data Matkul):



Source code Object Class:

```
public class Data_Matkul {  
    //atribur  
    int jml_sks;  
    String kode_mk, nama_mk, dosen_pengampu;  
  
    public Data_Matkul() {  
        this.kode_mk = "IF2023";  
        this.nama_mk = "Teknik Informatika";  
        this.dosen_pengampu= "Yosep Agus Pranoto";  
        this.jml_sks = 4;  
    }  
}
```

Desain form (GUI_Matkul.java):

Gambar 3.1 Desain Gui_Matkul .java

Tabel 3.1 Properti Desain GUI_Matkul.java

No	Objek	Properti	Nilai
1	jLabel1	Text	Data Matakuliah
2	jLabel2	Text	Kode Matakuliah
3	jLabel3	Text	Mata Kuliah
4	jLabel4	Text	Dosen Pengajar



5	jLabel5	Text	Jumlah SKS
6	(jTextField1)	Name	txtKode
		Text	
7	(jTextField2)	Name	txtMK
		Text	
8	(jTextField3)	Name	txtDosen
		Text	
9	(jTextField4)	Name	txtSKS
		Text	
10	(jButton1)	Name	btnCDM
		Text	Cetak Dosen Matakuliah
11	jTextArea	Name	memoDosen
		Text	

Source code Button Cetak Dosen Matakuliah pada GUI_Matkul.java:

```
public GUI_Matkul() {
    initComponents();
    Data_Matkul dtM = new Data_Matkul();
    txtKode.setText(dtM.kode_mk);
    txtKode.setEnabled(false);
    txtDosen.setText(dtM.dosen_pengampu);
    txtDosen.setEnabled(false);
    txtMk.setText(dtM.nama_mk);
    txtMk.setEnabled(false);
    txtSKS.setText(Integer.toString(dtM.jml_sks));
    txtSKS.setEnabled(false);
    memoDosen.setEnabled(true);
}

private void
btnCDMActionPerformed(java.awt.event.ActionEvent evt) {
    Data_Matkul mk = new Data_Matkul();
    memoDosen.append("Data Mata Kuliah = "+ "\n");
    memoDosen.append("Kode Mata Kuliah="+mk.kode_mk+"\n");
    memoDosen.append("Nama Mata Kuliah = "+mk.nama_mk+"\n");
    memoDosen.append("Dosen Pengajar
="+mk.dosen_pengampu+"\n");
    memoDosen.append("Jumlah SKS          =
"+mk.jml_sks+"\n");
}
```



Hasil Tampilan:

DATA MATAKULIAH

Kode Matakuliah: IF2023

Mata Kuliah: Teknik Informatika

Dosen Pengajar: Yosep Agus Pranoto

Jumlah SKS: 4

Data Mata Kuliah =
Kode Mata Kuliah = IF2023
Nama Mata Kuliah = Teknik Informatika
Dosen Pengajar = Yosep Agus Pranoto
Jumlah SKS = 4

Cetak Dosen Matakuliah

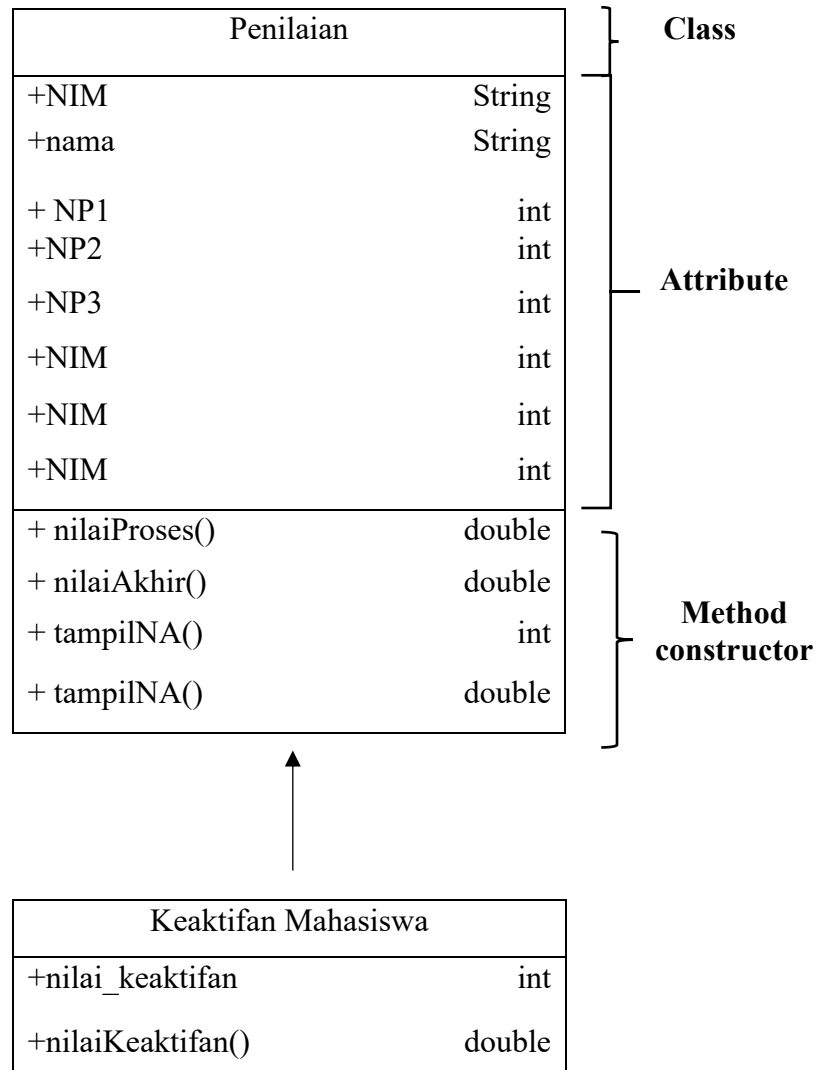
Gambar 3.2 Hasil Tampilan GUI_Matkul.java

III.4 Tugas Praktikum 2 :

Membuat *Class* Penilaian dan GUI_Penilaian

Judul : *Inheritance* pada class Penilaian

Diagram Class (Penilaian):





Source code Object Class:

```
public class Penilaian {
    String NIM, Nama, kode_mk;
    int NP1, NP2, NP3, NilaiPrak, UTS, UAS;

    double nilaiProses(){
        return ((NP1 * 0.1)+(NP2 * 0.2)+(NP3 *
0.3)+(NilaiPrak * 0.4));
    }
    double NilaiAkhir(){
        return (nilaiProses() * 0.6)+(UAS * 0.3);
    }
    double tampilNA(){
        return NilaiAkhir();
    }
    double nilaiKeaktifan(){
        return 0;
    }
}
```

Desain form (GUI_Penilaian.java):

Penilaian Matakuliah

NIM

Nama Matakuliah

Kode Matakuliah

NP 1 UTS

NP 2 Praktikum

NP 3 UAS

Nilai Keaktifan ☐ Tambahkan Nilai Keaktifan

Hasil Nilai Akhir

Gambar 3.3 Desain GUI_Penilaian.java



Tabel 3.2 Properti Desain GUI_Penilaian.java

No	Objek	Properti	Nilai
1	jLabel1	Text	Penilaian Matakuliah
2	jLabel2	Text	NIM
3	jLabel3	Text	Nama Matakuliah
4	jLabel4	Text	Kode Matakuliah
5	jLabel5	Text	NP1
6	jLabel6	Text	NP2
7	jLabel7	Text	NP3
8	jLabel8	Text	Nilai keaktifan
9	jLabel9	Text	UTS
10	Jlabel10	Text	Praktikum
11	jLabel11	Text	UAS
12	jTextField1	Name	txtNIM
		Text	
13	jTextField2	Name	txtMk
		Text	
14	jTextField3	Name	txtKodeMk
		Text	
15	jTextField4	Name	txtNP1
		Text	
16	jTextField5	Name	txtNP2
		Text	
17	jTextField6	Name	txtNP3
		Text	



18	(jTextField7	Name	txtNKeaktif
		Text	
19	(jTextField8	Name	txtUTS
		Text	
20	(jTextField9	Name	txtPrak
		Text	
21	(jTextField9	Name	txtUAS
		Text	
22	jCheckBox1	Name	checkBox
		Text	Tambahkan Nilai Keaktifan
23	jButton	Name	btnHNA
		Text	Hasil Nilai Akhir
24	jTextArea	Name	memoNilai
		Text	

Source code Button (Hasil Nilai Akhir) pada GUI:

```
public GUI_Penilaian() {
    initComponents();
    KeaktifanMahasiswa      nilai      =      new
    KeaktifanMahasiswa();

    txtNKeaktif.setText(Integer.toString(nilai.nilai_keaktifan)
);
    txtNKeaktif.setEnabled(false);
}
private void
checkBoxActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    if (checkBox.isSelected()){
        txtNKeaktif.setEnabled(true);
    }else{
        txtNKeaktif.setEnabled(false);
    }
}

private void
btnHNAAActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    memoNilai.setText(" ");
}
```



```
KeaktifanMahasiswa      nilai      =      new
KeaktifanMahasiswa();
    nilai.NIM = txtNIM.getText();
    nilai>Nama = txtMk.getText();
    nilai.kode_mk = txtKodeMk.getText();
    nilai.NP1 = Integer.parseInt(txtNP1.getText());
    nilai.NP2 = Integer.parseInt(txtNP2.getText());
    nilai.NP3 = Integer.parseInt(txtNP3.getText());
    nilai.NilaiPrak      =
Integer.parseInt(txtPrak.getText());
    nilai.UTS = Integer.parseInt(txtUTS.getText());
    nilai.UAS = Integer.parseInt(txtUAS.getText());

    nilai.nilai_keaktifan      =
Integer.parseInt(txtNKeaktif.getText());
    if(checkBox.isSelected()){
        memoNilai.append("      Nilai Akhir Mata Kuliah
\n");
        memoNilai.append("-----
-----\n");
        memoNilai.append("NIM
:"+nilai.NIM + "\n");
        memoNilai.append("Nama      Matakuliah
:"+nilai>Nama + "\n");
        memoNilai.append("Kode      Matakuliah
:"+nilai.kode_mk + "\n");
        memoNilai.append("Nilai      NP1
:"+nilai.NP1 + "\n");
        memoNilai.append("Nilai      NP2
:"+nilai.NP2 + "\n");
        memoNilai.append("Nilai      NP3
:"+nilai.NP3 + "\n");
        memoNilai.append("Nilai      Praktikum
:"+nilai.NilaiPrak + "\n");
        memoNilai.append("Nilai      UTS
:"+nilai.UTS + "\n");
        memoNilai.append("Nilai      UAS
:"+nilai.UAS + "\n");
        memoNilai.append("-----
-----\n");
        memoNilai.append("      Nilai      Akhir
:"+nilai.nilai_keaktifan + "\n");
    }else{
        memoNilai.append("Nilai Akhir Mata Kuliah \n");
        memoNilai.append("-----
-----\n");
        memoNilai.append("NIM
:"+nilai.NIM + "\n");
        memoNilai.append("Nama      Matakuliah
:"+nilai>Nama + "\n");
        memoNilai.append("Kode      Matakuliah
:"+nilai.kode_mk + "\n");
        memoNilai.append("Nilai      NP1
:"+nilai.NP1 + "\n");
        memoNilai.append("Nilai      NP2
:"+nilai.NP2 + "\n");
        memoNilai.append("Nilai      NP3
:"+nilai.NP3 + "\n");
        memoNilai.append("Nilai      Praktikum
:"+nilai.NilaiPrak + "\n");
    }
```



```
: "+nilai.NilaiPrak + "\n");  
    memoNilai.append("Nilai                                UTS  
: "+nilai.UTS + "\n");  
    memoNilai.append("Nilai                                UAS  
: "+nilai.UAS + "\n");  
    memoNilai.append("-----  
-----\n");  
    memoNilai.append("                                Nilai        Akhir  
: "+nilai.tampilNA() + "\n");  
    }  
}
```

Hasil Tampilan:

The screenshot shows a Java Swing window titled "Penilaian Matakuliah". It contains several input fields for student data: NIM (2218096), Nama Matakuliah (Object Oriented Programming), Kode Matakuliah (IF2023), and a table for scores. The table has columns for NP 1, NP 2, NP 3, UTS, Praktikum, and UAS. Below the table is a checkbox for "Tambahkan Nilai Keaktifan" and a button "Hasil Nilai Akhir". At the bottom is a text area displaying the calculated final score.

NP 1	NP 2	NP 3	UTS	Praktikum	UAS
78	80	82	80	85	80

Nilai Keaktifan: 60 ☐ Tambahkan Nilai Keaktifan

Hasil Nilai Akhir

NIM :2218096
Nama Matakuliah :Object Oriented Programming
Kode Matakuliah :IF2023
Nilai NP1 :78
Nilai NP2 :80
Nilai NP3 :82
Nilai Praktikum :85
Nilai UTS :80
Nilai UAS :80

Nilai Akhir :73.44

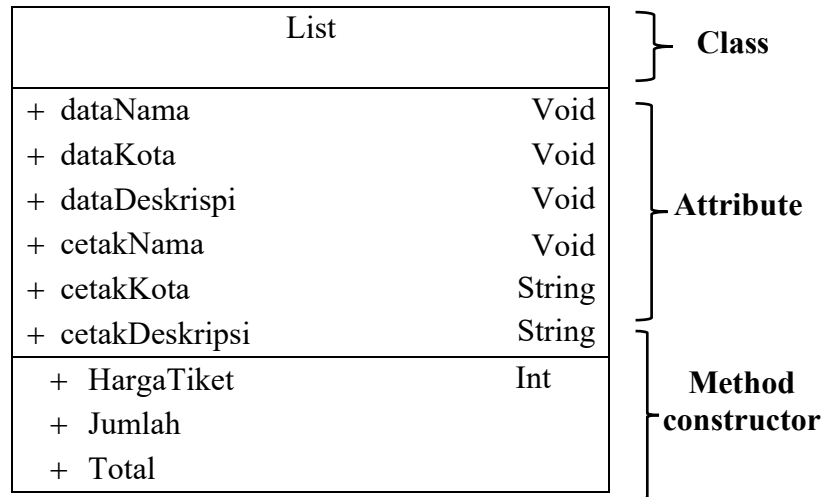
Gambar 3.4 Hasil Tampilan GUI_Penilaian.java



III.5 Tugas Rumah 1 :

Judul : Konstruktork pada *class* List

Diagram Class (List.java):



Source code Object Class List.java:

```
public class List {
    String Nama, Kota, Deskripsi;
    int HargaTiket, Jumlah, Total;

    void dataNama(String Nama) {
        this.Nama = Nama;
    }
    void dataKota(String Kota) {
        this.Kota = Kota;
    }
    void dataDeskripsi(String Deskripsi) {
        this.Deskripsi = Deskripsi;
    }

    String cetakNama() {
        return Nama;
    }
    String cetakKota() { return Kota;
    }
    String cetakDeskripsi() { return Deskripsi;
    }

    int HargaTiket() {
        return HargaTiket;
    }
    int Jumlah() {
        return Jumlah;
    }
    int Total() {
        return Total;
    }
    public int List() {
        Total = (HargaTiket * Jumlah);
    }
}
```



```
return Total;  
}  
  
}
```

Desain *form* (GUI_Wisata.java):

Gambar 3.5 Desain GUI_Wisata.java

Tabel 3.3 Properti Desain GUI_CetakPenjualanKaos.java

No	Nama Komponen	Properti	Value
1	jLabel1	Text	WISATA
2	jLabel2	Text	Nama
3	jLabel3	Text	Kota
4	jLabel4	Text	Deskripsi
5	jTextField1	Name	txtNama
		Text	“ “
6	jTextField3	Name	txtDes
		Text	“ “
7	jButton1	Name	Add
		Text	btnDta
8	jTextArea	Name	-
		Text	memodata
9	ScrollPane1	-	-



10	jLabel5	Text	Harga Tiket
11	jLabel6	Text	Jumlah
12	jTextField4	Name	HargaTiket
		Text	“ “
13	jTextField5	Name	jumlah
		Text	“ “
14	jTextField2	Name	txtKota
		Text	“ “

Source code Button Cetak pada GUI_Wisata.java:

```
private void
btnDataActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    memoData.setText("");
    // Wisata wst = new Wisata();
    List wst = new List();
    wst.dataNama(txtNama.getText());
    wst.dataKota(txtKota.getText());
    wst.dataDeskripsi(txtDes.getText());
    wst.HargaTiket =
Integer.parseInt(HargaTiket.getText());
    wst.Jumlah = Integer.parseInt(jumlah.getText());
    memoData.append("List Wisata\n");
    memoData.append("-----\n");
    memoData.append(" Nama : " + wst.Nama + "\n");
    memoData.append(" Kota: " + wst.Kota + "\n");
    memoData.append(" Deskripsi: " + wst.Deskripsi +
"\n");
    memoData.append("Harga Tiket : " + "\n");
    memoData.append("Total : "+wst.List());
}
```



Hasil Tampilan:

WISATA

Nama: Sumber Umbulan

Kota: Malang

Deskripsi: Karanglo

Harga Tiket: 2000

Jumlah: 4

Add

List Wisata

Nama : Sumber Umbulan
Kota: Malang
Deskripsi: Karanglo
Harga Tiket :
Total : 8000

Gambar 3.6 Hasil Tampilan GUI_Wisata.java

Analisa:

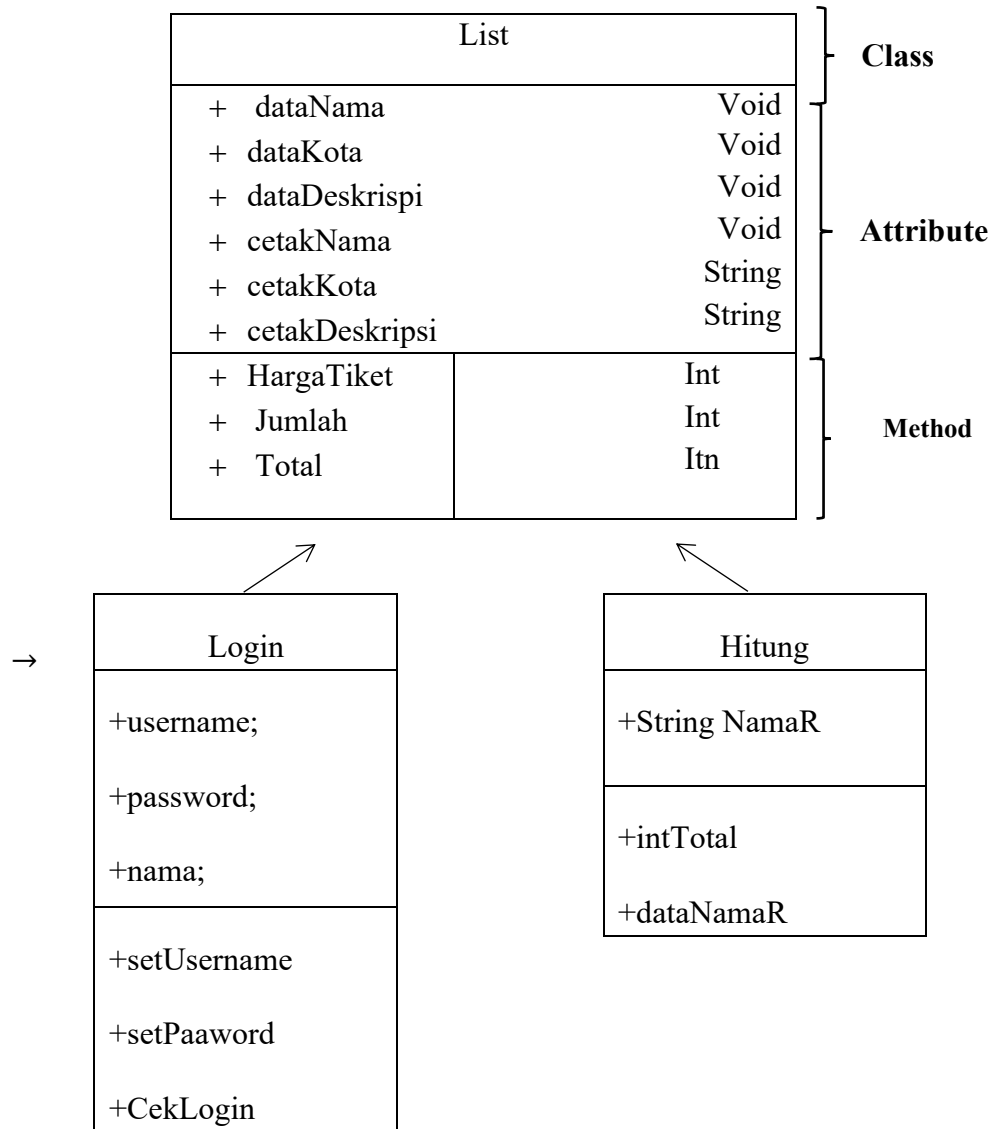
Program akan meminta untuk menginputkan nama, lokasi wisata, deskripsi tempat tersebut bertipe string dan juga menginput harga tiket, jumlah, dan total yang bertipe int. Kemudian untuk memanggil text area diperlukan variable dari text area berupa `memomabe`. Karena yang sebelumnya `system.out` berubah menjadi variable dari text area.



III.6 Tugas Rumah 2 :

Judul : Membuat 2 class tambahan, GUI, dan Menerapkan Inheritance

Diagram Class (List):



Source code Object Class(List) :

```
public class List {
    String Nama, Kota, Deskripsi;
    int Harga, Jumlah, Total;

    void dataNama(String Nama) {
        this.Nama = Nama;
    }
    void dataKota(String Kota) {
        this.Kota = Kota;
    }
    void dataDeskripsi(String Deskripsi) {
        this.Deskripsi = Deskripsi;
    }
}
```



Desain *form* (GUI_Wisata.java):

Gambar 3.7 Desain GUI_Wisata.java

Tabel 3.4 Properti Desain GUI_Wisata.java

No	Objek	Properti	Nilai
1	jLabel1	Text	Wisata
2	jLabel2	Text	Nama
3	jLabel3	Text	Kota
4	jLabel4	Text	Deskripsi
5	jTextField1	Name	txtNama
		Text	“ “
6	jTextField2	Name	txtKota
		Text	“ “
7	jTextField3	Name	txtDes
		Text	“ “
8	jButton1	Name	Add
		Text	btnData
9	jTextArea	Name	MemoDta
		Text	

*Source code Button Add.java:*

```
private void
btnDataActionPerformed(java.awt.event.ActionEvent
evt) {
    // TODO add your handling code here:
    #PraktikumOOP2023
    memoData.setText("");
    // Wisata wst = new Wisata();
    List wst = new List();
    wst.dataNama(txtNama.getText());
    wst.dataKota(txtKota.getText());
    wst.dataDeskripsi(txtDes.getText());
    memoData.append("List Wisata\n");
    memoData.append("-----\n");
    memoData.append(" Nama : " + wst.Nama + "\n");
    memoData.append(" Kota: " + wst.Kota + "\n");
    memoData.append(" Deskripsi: " + wst.Deskripsi + "\n");

    Reservasi r = new Reservasi();
    r.setVisible(true);
}
```

Hasil Tampilan:

Gambar 3.8 Hasil Tampilan GUI_Wisata java

Source Code Object Class/Abstract (Login.java) :

```
private void
btnLoginActionPerformed(java.awt.event.ActionEvent
evt) {
    // TODO add your handling code here:
    Login l = new Login();
    String username = txtNama.getText();
    String password = txtPass.getText();
    boolean Authenticated = l.CekLogin(username, password);
    if (Authenticated)
    {
        JOptionPane.showMessageDialog(rootPane, "LOGIN
        #PraktikumOOP2023
        BERHASIL, "+l.nama + "!");
        Wisata w = new Wisata();
        w.setVisible(true);
    }
}
```



```
this.dispose();
}else
{
    JOptionPane.showMessageDialog(rootPane, "LOGIN
GAGAL. Silahkan periksa kembali username dan password
Anda.");
}
}
```

Desain form (Login.java) :

Gambar 3.9 Desain GUI_Login.java

Tabel 3.5 Properti Desain Login.java

No	Objek	Properti	Nilai
1	jLabel1	Text	Login
2	jLabel2	Text	Nama
3	jLabel3	Text	Password
4	jTextField1	Name	txtnama
		Text	-
5	jTextField2	Name	txtPass
		Text	“ “
6	JButton1	Name	btnLogin
		Text	Login

Source code Button/combobox btnLogin :

```
private void
btnLoginActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    Login l = new Login();
    String username = txtNama.getText();
    #PraktikumOOP2023
    String password = txtPass.getText();
```




```
boolean Authenticated = l.CekLogin(username,
password);
if (Authenticated)
{
JOptionPane.showMessageDialog(rootPane, "LOGIN
BERHASIL, "+l.nama + "!");
Wisata w = new Wisata();
w.setVisible(true);
this.dispose();
}else
{
JOptionPane.showMessageDialog(rootPane, "LOGIN
GAGAL. Silahkan periksa kembali username dan password
Anda.");
}
}
```

Hasil Tampilan :



Gambar 3.10 Hasil Tampilan Login.java

Analisa:

Program diatas merupakan tampilan gui wisata dari class list,pada bagian ini akan tampil nama,kota dan deskripsi.pada tampilan gui kedua tampilan login.pada login ada nama dan password.

Source Code Object Class/Abstract (Hitung.java) :

```
public class Hitung extends List{
String NamaR;

void dataNamaR(String NamaR)
{
this.NamaR = NamaR;
}

public Hitung(){
this.Harga = 25000;
this.Jumlah = Jumlah;
this.Total = Total;
}
public int Total()
{
```



```
#PraktikumOOP2023
Total = (Harga * Jumlah);
return Total;
}

}
```

Desain form (GUI_Reservasi.java) :

Gambar 3.11 Desain GUI_Reservasi.java

Tabel 3.6 Properti Desain GUI_Reservasi.java

No	Objek	Properti	Nilai
1	jLabel1	Text	Reservasi
2	jLabel2	Text	Nama
3	jLabel3	Text	Harga Tiket
4	jLabel4	Text	Jumlah Orang
5	jTextField1	Name	txtR -
		Text	-
6	jTextField2	Name	txtHarga
		Text	-
7	jTextField3	Name	txtJumlah “ “
		Text	-



6	JButton1	Name	btnHitung
		Text	Hitung
7	jScrollPane1	Name	memo
			-

Source code Button/combobox btnHitung :

```
private void
btnDataActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    memoData.setText("");
    // Wisata wst = new Wisata();
    List wst = new List();
    wst.dataNama(txtNama.getText());
    wst.dataKota(txtKota.getText());
    wst.dataDeskripsi(txtDes.getText());
    memoData.append("List Wisata\n");
    memoData.append("-----\n");
    memoData.append(" Nama : " + wst.Nama + "\n");
    memoData.append(" Kota: " + wst.Kota + "\n");
    memoData.append(" Deskripsi: " + wst.Deskripsi +
"\n");

    Reservasi r = new Reservasi();
    r.setVisible(true);
}
```

Hasil Tampilan :

Gambar 3.12 Hasil Tampilan GUI_Reservasi.java



Analisa :

Program diatas merupakan implementasi dari class hitung yang di hubungkan gui reservasi. Tampil yang amuncul nama pemesan, harga tiket dan jumlah orang. Setelah itu akan muncul di memo sesuai apa yang di inputkan.

III.7 Kesimpulan

1. Konstruktor adalah method yang pertama kali dijalankan pada saat sebuah objek pertama kali diciptakan. Syarat penulisan *method* konstruktor harus sama dengan nama *class* yang di naungi.
2. *Inheritance* atau Pewarisan/Penurunan adalah konsep pemrograman dimana sebuah class dapat ‘menurunkan’ properti dan method yang dimilikinya kepada class lain. Konsep inheritance digunakan untuk memanfaatkan fitur ‘code reuse ’ untuk menghindari duplikasi kode program. Fungsi dari inheritance memperluas fungsi dari parent class.
3. Tidak semua property dan *method* dari *class* induk akan diturunkan. Property dan *method* dengan hak akses *private*, tidak akan diturunkan kepada *class* anak. Pada class yang memiliki class turunan disebut dengan parent class atau base class, sedangkan class turunan itu sendiri kerap disebut dengan subclass atau child class yang bisa menurunkan atau mewariskan apa pun dari parent class

Nama Aslab :	TTD :
Firman Frezy Pradana 2118112	
Tanggal : 17 Desember 2023	



BAB IV

ENKAPSULASI , OVERRIDING DAN OVERLOADING

Jumlah Pertemuan	:	2 x 60 menit
Tujuan Praktikum	:	<ol style="list-style-type: none">1. Praktikan mampu memahami tentang hak akses (Enkapsulasi).2. Praktikan mampu mengimplementasikan konsep <i>Encapsulasi</i> kedalam program.3. Praktikan mampu memahami dan membuat <i>method override</i>.
Alat / bahan	:	<ol style="list-style-type: none">1. Seperangkat <i>computer</i>.2. Perangkat lunak: <i>Netbeans</i>.3. Modul Praktikum <i>OOP 2022</i>.

IV.1 Landasan Teori

A. Pengertian *Encapsulasi*

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 akan dibatasi dalam 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 dipublish.

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. Alasan menggunakan Enkapsulasi
 - a. Untuk meningkatkan keamanan data;
 - b. Agar lebih mudah dalam mengontrol atribut dan method;
 - c. Class bisa kita buat menjadi *read-only* dan *write-only*; dan fleksibel: programmer dapat mengganti sebagian dari kode tanpa harus takut berdampak pada kode yang lain.

2. Jenis Enkapsulasi (Hak Akses)

- a. *Public*

Dengan mendeklarasikan data dan *method* dengan tingkat akses *public*, maka data dan *method* dapat diakses semua kelas yang ada di dalam program, baik yang merupakan kelas turunan maupun kelas yang tidak mempunyai hubungan sama sekali.

- b. *Private*

Dengan mendeklarasikan data dan *method* menggunakan tingkat akses *private*, maka data dan *method* tersebut hanya dapat diakses oleh kelas tersebut. Sehingga data dan *method* tersebut tidak dapat diakses oleh kelas lain.

- c. *Protected*

Dengan mendeklarasikan data dan *method* menggunakan tingkat akses *protected*, maka data dan *method* tersebut hanya dapat diakses oleh kelas yang memilikinya dan kelas-kelas yang masih memiliki hubungan turunan.

- d. Default (tidak ada modifier)

Untuk hak akses *default* ini, sebenarnya hanya ditujukan untuk *class* yang ada dalam satu paket, atau istilahnya hak akses yang berlaku untuk satu folder saja (tidak berlaku untuk *class* yang tidak satu folder/package)

- B. Pengertian *Overloading* dan *Overriding*.

1. *Overloading*

Method Overloading adalah sebuah kemampuan yang membolehkan sebuah class mempunyai 2 atau lebih method dengan

nama yang sama, namun parameter yang berbeda. Pada *method overloading* perbedaan parameter mencakup : Jumlah parameter ,serta tipe data dari parameter tersebut. Dengan menggunakan *method overloading* , kita dapat membuat beberapa versi dari sebuah *method* dengan implementasi yang berbeda berdasarkan argumen yang diberikan padanya. jadi ini akan membantu kita menggunakan kembali nama *method* yang sama untuk tujuan yang berbeda. Dengan begitu *method overloading* akan memberikan fleksibilitas saat menggunakan sebuah *method/function*.

2. *Overriding*

Overriding *method* adalah kemampuan dari subclass (*child class*) untuk memodifikasi *method* dari *superclass*-nya, dengan cara mendefinisikan kembali *method* *superclass*-nya. Namun masih dengan nama dan parameter yang sama tetapi isi (*statement*) berbeda. *Method overriding* akan meningkatkan fleksibilitas dalam penggunaan *class inheritance*. Dengan adanya *overriding*, *subclass* dapat menyesuaikan implementasi dari *method* yang diwarisi dari *superclass* sesuai dengan kebutuhan *subclass*.

Aturan *Overriding*:

- a. Parameter yang terdapat pada *method Overriding* di *subclass* harus sama dengan parameter yang terdapat pada *parent class*.
- b. *Method* pada *subclass* *class* yang akan di *override*, sebaiknya tidak menggunakan hak akses *private*, karena *method* pada *subclass* tidak boleh membatasi aksesibilitas yang lebih luas dari pada *method* pada *parent class*.

IV.2 Langkah – Langkah Praktikum

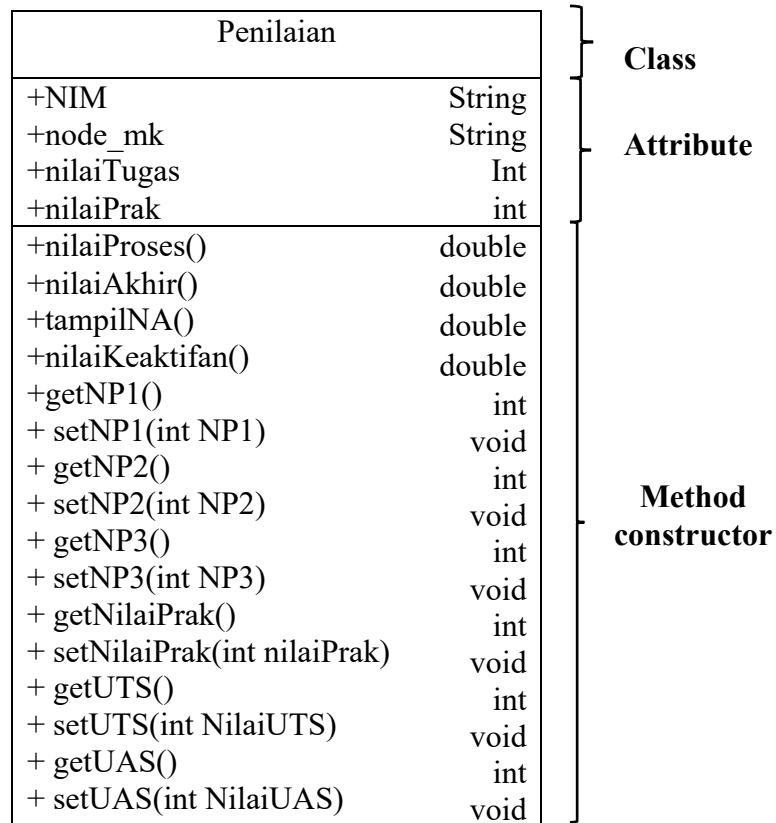
1. Buka Aplikasi *Netbeans*.
2. Buat *Class*.
3. Memberi *script* pada kelas tersebut.
4. Buat *form*.
5. Memberi *script* pada *form* tersebut.
6. Menjalankan program.

IV.3 Tugas Praktikum 1 :

Membuat *Class* Penilaian dan GUI_Nilai

Judul : Implementasi Enkapsulasi *Class* Penilaian

Diagram Class (Penilaian):



Source code Object Class (Penilaian.java):

```
public class Penilaian {
    public String NIM, nama, kode_mk;
    private int NP1, NP2, NP3, NilaiPrak, UTS, UAS;
    public double nilaiProses() {
        return
        ((NP1*0.1)+(NP2*0.2)+(NP3*0.1)+(UTS*0.2)+(NilaiPrak*0.4));
    }
    public double nilaiAkhir(){
        return (nilaiProses()*0.6)+(UAS*0.3);
    }
    public double tampilNA(){
        return nilaiAkhir();
    }
    public double nilaiKeaktifan(){
        return 0;
    }
    public int getNP1(){
        return NP1;
    }
    public void setNP1(int NP1){
        this.NP1 = NP1;
    }
}
```



```

    }
    public int getNP2(){
        return NP2;
    }
    public void setNP2(int NP2){
        this.NP2 = NP2;
    }
    public int getNP3(){
        return NP3;
    }
    public void setNP3(int NP3){
        this.NP3 = NP3;
    }
    public int getNilaiPrak(){
        return NilaiPrak;
    }
    public void setNilaiPrak(int NilaiPrak){
        this.NilaiPrak = NilaiPrak; }
    public int getUTS(){
        return UTS;}
    public void setUTS(int UTS){
        this.UTS = UTS;}
    public int getUAS(){
        return UAS;}
    public void setUAS(int UAS){
        this.UAS = UAS;
    }
}

```

Desain *form* (GUI_Nilai.java):

The screenshot shows a Java Swing window titled "Penilaian Matakuliah". It features a light gray background and standard window controls (minimize, maximize, close). The form includes the following elements:

- Input Fields:**
 - NIM (text box)
 - Nama Matakuliah (text box)
 - Kode Matakuliah (text box)
 - NP 1, NP 2, NP 3 (text boxes)
 - UTS (text box)
 - Praktikum (text box)
 - UAS (text box)
 - Nilai Keaktifan (text box with value 0)
- Checkbox:** "Tambahkan Nilai Keaktifan" (unchecked)
- Button:** "Hasil Nilai Akhir" (disabled)
- Text Area:** A large empty rectangular area at the bottom of the window.

Gambar 4.1 Desain GUI_Nilai.java

Tabel 4.1 Properti Desain GUI_Nilai.java

No	Objek	Properti	Nilai
1	Jlabel1	Text	Penilaian Matakuliah
2	Jlabel2	Text	NIM
3	Jlabel3	Text	Nama Matakuliah
4	Jlabel4	Text	Kode Matakuliah
5	Jlabel5	Text	NP1
6	Jlabel6	Text	NP2
7	Jlabel7	Text	NP3
8	Jlabel8	Text	UTS
9	Jlabel9	Text	UAS
10	Jlabel10	Text	Praktikum
11	Jlabel11	Text	Nilai Keaktifan
12	Jtextfield1	Name	txtNIM
		Text	-
13	Jtextfield2	Name	txtMk
		Text	-
14	Jtextfield3	Name	txtKodeMk
		Text	-
15	Jtextfield4	Name	txtNP1
		Text	-
16	Jtextfield5	Name	txtNP2
		Text	-
17	Jtextfield6	Name	txtNP3
		Text	-

18	Jtextfield7	Name	txtUTS
		Text	-
19	Jtextfield8	Name	txtPrak
		Text	-
20	Jtextfield9	Name	txtUAS
		Text	-
21	Jtextfield10	Name	txtNKeaktif
		Text	-
22	Jbutton1	Name	btnHNA
		Text	Hasil Nilai Akhir
23	Jtextarea	Name	MemoNilai
		Text	-
24	Jcheckbox	Name	checkbox
		Text	Tambahkan Nilai Keaktifan

Source code Button pada GUI_Nilai.java:

```

public GUI_Nilai() {
    initComponents();
    KeaktifanMahasiswa nilai = new KeaktifanMahasiswa();

    txtNKeaktif.setText(Integer.toString(nilai.nilai_keaktifan));
    txtNKeaktif.setEnabled(false);
}

private void btnHNAActionPerformed(java.awt.event.ActionEvent
evt) {
    // TODO add your handling code here:
    memoNilai.setText(" ");
    KeaktifanMahasiswa nilai = new KeaktifanMahasiswa();
    nilai.NIM = txtNIM.getText();
    nilai.nama = txtMk.getText();
    nilai.kode_mk = txtKodeMk.getText();
    nilai.setNP1 (Integer.parseInt(txtNP1.getText()));
    nilai.setNP2 (Integer.parseInt(txtNP2.getText()));
    nilai.setNP3 (Integer.parseInt(txtNP3.getText()));
    nilai.setNilaiPrak
(Integer.parseInt(txtPrak.getText()));
    nilai.setUTS (Integer.parseInt(txtUTS.getText()));
    nilai.setUAS (Integer.parseInt(txtUAS.getText()));
}

```

```

        nilai.nilai_keaktifan =
Integer.parseInt(txtNKeaktif.getText());
        if(checkBox.isSelected()){
            memoNilai.append("          Nilai Akhir Mata Kuliah
\n");
            memoNilai.append("-----\n");
            memoNilai.append("NIM
:"+nilai.NIM + "\n");
            memoNilai.append("Nama          Matakuliah
:"+nilai.nama + "\n");
            memoNilai.append("Kode          Matakuliah
:"+nilai.kode_mk + "\n");
            memoNilai.append("Nilai          NP1
:"+nilai.getNP1() + "\n");
            memoNilai.append("Nilai          NP2
:"+nilai.getNP2() + "\n");
            memoNilai.append("Nilai          NP3
:"+nilai.getNP3() + "\n");
            memoNilai.append("Nilai          Praktikum
:"+nilai.getNilaiPrak() + "\n");
            memoNilai.append("Nilai          UTS
:"+nilai.getUTS() + "\n");
            memoNilai.append("Nilai          UAS
:"+nilai.getUAS() + "\n");
            memoNilai.append("-----\n");
            memoNilai.append("          Nilai          Akhir
:"+nilai.nilai_keaktifan + "\n");
        }else{
            memoNilai.append("Nilai Akhir Mata Kuliah \n");
            memoNilai.append("-----
---\n");
            memoNilai.append("NIM
:"+nilai.NIM + "\n");
            memoNilai.append("Nama          Matakuliah
:"+nilai.nama + "\n");
            memoNilai.append("Kode          Matakuliah
:"+nilai.kode_mk + "\n");
            memoNilai.append("Nilai          NP1
:"+nilai.getNP1() + "\n");
            memoNilai.append("Nilai          NP2
:"+nilai.getNP2() + "\n");
            memoNilai.append("Nilai          NP3
:"+nilai.getNP3() + "\n");
            memoNilai.append("Nilai          Praktikum
:"+nilai.getNilaiPrak() + "\n");
            memoNilai.append("Nilai          UTS
:"+nilai.getUTS() + "\n");
            memoNilai.append("Nilai          UAS
:"+nilai.getUAS() + "\n");
            memoNilai.append("-----
-----\n");
            memoNilai.append("  Nilai Akhir          : " +
nilai.nilai_keaktifan + "\n");
        }
    }
    memoNilai.setText(" ");
    KeaktifanMahasiswa nilai = new KeaktifanMahasiswa();
    nilai.NIM = txtNIM.getText();

```

```

        nilai.nama = txtMk.getText();
        nilai.kode_mk = txtKodeMk.getText();
        nilai.setNP1 (Integer.parseInt(txtNP1.getText()));
        nilai.setNP2 (Integer.parseInt(txtNP2.getText()));
        nilai.setNP3 (Integer.parseInt(txtNP3.getText()));
        nilai.setNilaiPrak
(Integer.parseInt(txtPrak.getText()));
        nilai.setUTS (Integer.parseInt(txtUTS.getText()));
        nilai.setUAS (Integer.parseInt(txtUAS.getText()));

        nilai.nilai_keaktifan =
Integer.parseInt(txtNKeaktif.getText());
        if(checkBox.isSelected()){
            memoNilai.append("          Nilai Akhir Mata Kuliah
\n");
            memoNilai.append("-----\n");
            memoNilai.append("NIM
:"+nilai.NIM + "\n");
            memoNilai.append("Nama          Matakuliah
:"+nilai.nama + "\n");
            memoNilai.append("Kode          Matakuliah
:"+nilai.kode_mk + "\n");
            memoNilai.append("Nilai          NP1
:"+nilai.getNP1()+ "\n");
            memoNilai.append("Nilai          NP2
:"+nilai.getNP2() + "\n");
            memoNilai.append("Nilai          NP3
:"+nilai.getNP3() + "\n");
            memoNilai.append("Nilai          Praktikum
:"+nilai.getNilaiPrak() + "\n");
            memoNilai.append("Nilai          UTS
:"+nilai.getUTS() + "\n");
            memoNilai.append("Nilai          UAS
:"+nilai.getUAS() + "\n");
            memoNilai.append("-----\n");
            memoNilai.append("          Nilai          Akhir
:"+nilai.nilai_keaktifan + "\n");
        }else{
            memoNilai.append("Nilai Akhir Mata Kuliah \n");
            memoNilai.append("-----
---\n");
            memoNilai.append("NIM
:"+nilai.NIM + "\n");
            memoNilai.append("Nama          Matakuliah
:"+nilai.nama + "\n");
            memoNilai.append("Kode          Matakuliah
:"+nilai.kode_mk + "\n");
            memoNilai.append("Nilai          NP1
:"+nilai.getNP1()+ "\n");
            memoNilai.append("Nilai          NP2
:"+nilai.getNP2() + "\n");
            memoNilai.append("Nilai          NP3
:"+nilai.getNP3() + "\n");
            memoNilai.append("Nilai          Praktikum
:"+nilai.getNilaiPrak() + "\n");
            memoNilai.append("Nilai          UTS
:"+nilai.getUTS() + "\n");
            memoNilai.append("Nilai          UAS
:"+nilai.getUAS() + "\n");

```

```

        memoNilai.append("-----\n");
        memoNilai.append("  Nilai Akhir          : " +
nilai.nilai_keaktifan + "\n");
    }

private void
checkBoxActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    if (checkBox.isSelected()){
        txtNKeaktif.setEnabled(true);
    }else{
        txtNKeaktif.setEnabled(false);
    }
}

```

Hasil Tampilan:

The image displays two side-by-side screenshots of a Java Swing application window titled "Penilaian Mata Kuliah". The window contains several text input fields for student information and scores, a checkbox, and a summary table.

Left Screenshot (Initial State):

- NIM: 2218096
- Nama Mata Kuliah: Object Oriented Programming
- Kode Mata Kuliah: IF2023
- NP 1: 75, UTS: 80
- NP 2: 78, Praktikum: 90
- NP 3: 80, UAS: 80
- Nilai Keaktifan: 0
- ☐ Tambahkan Nilai Keaktifan
- Hasil Nilai Akhir button
- Summary Table:

Nilai Akhir Mata Kuliah	
NIM	2218096
Nama Mata Kuliah	Object Oriented Programming
Kode Mata Kuliah	IF2023
Nilai NP1	75
Nilai NP2	78
Nilai NP3	80
Nilai Praktikum	90
Nilai UTS	80
Nilai UAS	80
Nilai Akhir	80.5

Right Screenshot (After Action):

- NIM: 2218096
- Nama Mata Kuliah: Object Oriented Programming
- Kode Mata Kuliah: IF2023
- NP 1: 75, UTS: 80
- NP 2: 78, Praktikum: 90
- NP 3: 80, UAS: 80
- Nilai Keaktifan: 70
- ☒ Tambahkan Nilai Keaktifan
- Hasil Nilai Akhir button
- Summary Table:

Nilai Akhir Mata Kuliah	
NIM	2218096
Nama Mata Kuliah	Object Oriented Programming
Kode Mata Kuliah	IF2023
Nilai NP1	75
Nilai NP2	78
Nilai NP3	80
Nilai Praktikum	90
Nilai UTS	80
Nilai UAS	80
Nilai Akhir	70

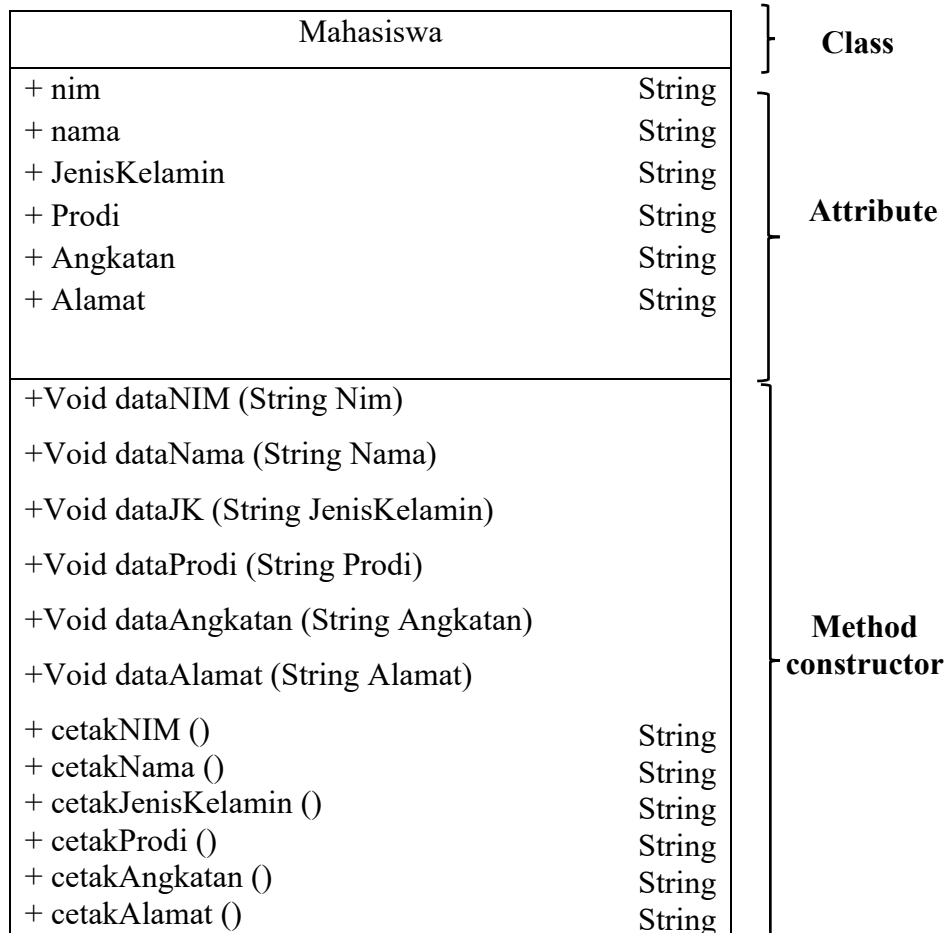
Gambar 4.2 Hasil Tampilan GUI_Nilai.java

IV.4 Tugas Praktikum 2 :

Membuat *Class* Mahasiswa dan GUI_Mahasiswa

Judul : Implementasi Overloading dan Overriding pada *class* Mahasiswa

Diagram Class (Mahasiswa):



Source code Object Class:

```
public class Mahasiswa {
    String nim, nama, JenisKelamin, prodi, angkatan, alamat;
    void dataNIM(String Nim) {
        this.nim = Nim;
    }
    void dataNama(String Nama) {
        this.nama = Nama;
    }
    void dataJenisKelamin(String JenisKelamin) {
        this.JenisKelamin = JenisKelamin;
    }
    void dataProdi(String Prodi) {
        this.prodi = Prodi;
    }
    void dataAngkatan(String angkatan) {
        this.angkatan = angkatan;
    }
    void dataAlamat(String alamat) {
```

```

        this.alamat = alamat;
    }
    String cetakNIM(){
        return nim;
    }
    String cetakNama(){
        return nama;
    }
    String cetakJenisKelamin(){
        return JenisKelamin;
    }
    String cetakProdi(){
        return prodi;
    }
    String cetakAngkatan(){
        return angkatan;
    }
    String cetakAlamat(){
        return alamat;
    }
}

```

Desain *form* (GUI_Mahasiswa.java):

Gambar 4.3 Desain GUI_Mahasiswa.java

Tabel 4.2 Properti Desain GUI_Mahasiswa.java

No	Komponen	Properti	Nilai
1	jLabel1	Text	DATA MAHASISWA
2	jLabel2	Text	NIM
3	jLabel3	Text	Nama
4	jLabel4	Text	Jenis Kelamin
5	jLabel5	Text	Prodi
6	jLabel6	Text	Angkatan
7	jLabel7	Text	Alamat

8	jLabel8	Text	Search
9	jTextField1	Name	txtNIM
		Text	
10	jTextField2	Name	txtNama
		Text	
11	jTextField3	Name	txtProdi
		Text	
12	jTextField4	Name	txtAngkatan
		Text	
13	jTextField5	Name	txtAlamat
		Text	
14	jRadioButton1	Name	radiobtnLaki
		Text	Laki-laki
15	jRadioButton2	Name	radiobtnPerempuan
		Text	Perempuan
16	jButton1	Name	btnSimpan
		Text	Simpan
17	jButton2	Name	btnClose
		Text	Close
18	jButton3	Name	btnHapus
		Text	Hapus
19	jButton4	Name	btnBatal
		Text	Batal
20	jButton5	Name	btnPenilaian
		Text	Form Penilaian
21	jTable	Name	table_data_mahasiswa
		Text	

Source code Button pada GUI_Mahasiswa.java:

```
import java.util.ArrayList;
import java.util.List;
import javax.swing.JOptionPane;
import javax.swing.table.DefaultTableColumnModel;
import javax.swing.table.DefaultTableModel;

public GUI_Mahasiswa() {
    initComponents();
    //mengambil model data dari tabel dan menyimpannya
    dalam objek DefaultTableModel dataModel
    DefaultTableModel dataModel = (DefaultTableModel)
    table_data_mahasiswa.getModel();
    //mendapatkan jumlah baris yang ad dalam model
    data saat ini
    int rowCount = dataModel.getRowCount();
    while (rowCount > 0 ){
        //menghapus baris terakhir dari model data
        dataModel.removeRow(rowCount - 1);
        //memperbaharui nilai rowCount setelah
    penghapusan baris terkahir
        rowCount = dataModel.getRowCount();
    }
}

public void clear (){
    txtNIM.setText(" ");
    txtNama.setText(" ");
    txtProdi.setText(" ");
    txtAngkatan.setText(" ");
    txtAlamat.setText(" ");
    btnGroupJk.clearSelection();
}

private void
btnSimpanActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    JOptionPane.showMessageDialog(null, "Data
    ditambahkan di tabel");
    DefaultTableModel dataModel = (DefaultTableModel)
    table_data_mahasiswa.getModel();
    List list = new ArrayList<>();

    table_data_mahasiswa.setAutoCreateColumnsFromModel(true);
    Mahasiswa mhs = new Mahasiswa();
    mhs.dataNIM(txtNIM.getText());
    mhs.dataNama(txtNama.getText());
    String JenKel = " ";
    if(radiobtnLaki.isSelected()){
        mhs.dataJenisKelamin(radiobtnLaki.getText());
    }else{
        mhs.dataJenisKelamin(radiobtnPerempuan.getText());
    }
    mhs.dataProdi(txtProdi.getText());
    mhs.dataAngkatan(txtAngkatan.getText());
    mhs.dataAlamat(txtAlamat.getText());
}
```

```

        list.add(mhs.cetakNIM());
        list.add(mhs.cetakNama());
        list.add(mhs.cetakJenisKelamin());
        list.add(mhs.cetakProdi());
        list.add(mhs.cetakAngkatan());
        list.add(mhs.cetakAlamat());

        dataModel.addRow(list.toArray());
        clear();
    }

    private void
    btnCloseActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        dispose();
    }

    private void
    btnBatalActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        clear();
    }

    private void
    btnHapusActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        DefaultTableModel dataModel = (DefaultTableModel)
        table_data_mahasiswa.getModel();
        int rowCount = dataModel.getRowCount();
        while (rowCount > 0){
            dataModel.removeRow(rowCount - 1);
            rowCount = dataModel.getRowCount();
        }
    }
}

```

Hasil Tampilan:

NIM	Nama	JK	Prodi	Angkatan	Alamat
2218096	Maria Avr...	Perempu...	Infomati...	2022	NTT
2012049	Yohanes...	Laki-laki	Elektro	2020	NTT

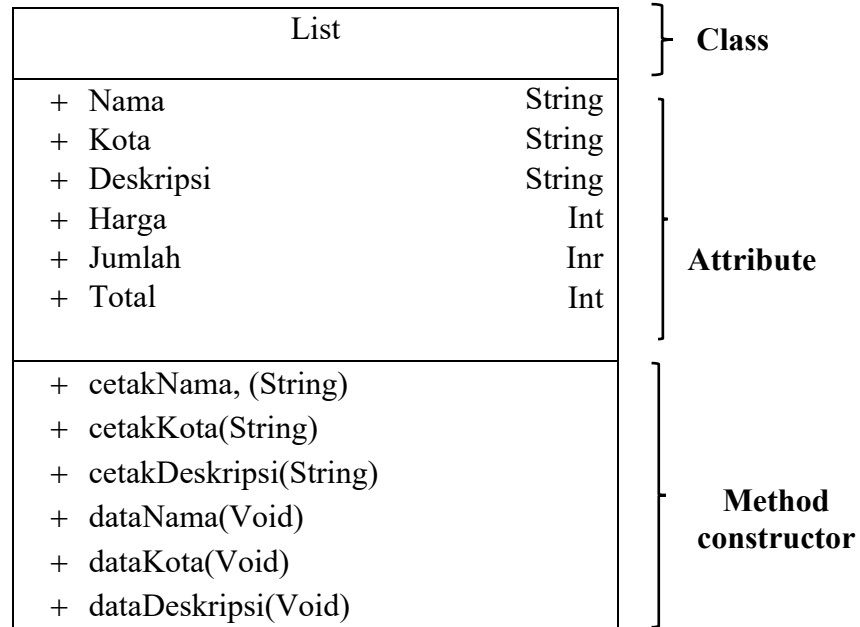
Gambar 4.4 Hasil Tampilan GUI_Mahasiswa.java

IV.5 Tugas Rumah 1 : Implementasi Enkapsulasi di 3 Class

Judul Judul : Informasi Wisata

Tema : Sistem Informasi

Diagram Class (List.java):



Source code Object Class/Abstact(List.java):

```
public class List {
    String Nama, Deskripsi;
    private String Kota;
    int Jumlah, Total;

    public String getkota()
    {
        return Kota;
    }
    public void setKota(String Kota){
        this.Kota = Kota;
    }

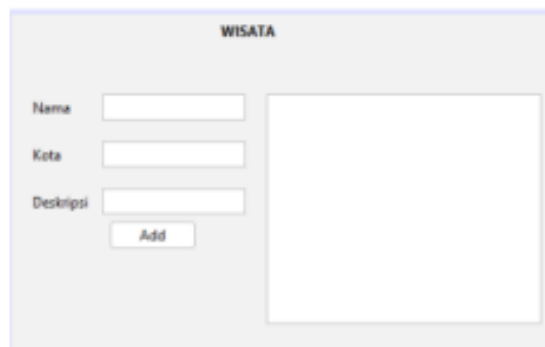
    void dataNama(String Nama){
        this.Nama = Nama;
    }
    void dataDeskripsi(String Deskripsi){
        this.Deskripsi = Deskripsi;
    }
}
List
□ Nama,
□ Kota,
□ Deskripsi
□ Harga,
□ Jumlah,
```

```

□ Total
□ void dataNama
□ void dataKota
□ void dataDeskripsi
□ String cetakNama
□ String cetakKota
□ String cetakDeskripsi
#PraktikumOOP2023
Desain form (Wisata.java):
Gambar 4.1 Desain GUI_Wisata .java
Tabel 4.1 Properti GUI_Wisata.java
No Nama Komponen Properti Value
1 jLabel1 Text Wisata
2 jLabel2 Text Nama
3 jLabel3 Text Kota
4 jLabel4 Text Deskripsi
5 jTextField1
Name txtNama
Text " "
void dataNama(String Nama){
    this.Nama = Nama;
}
void dataDeskripsi(String Deskripsi){
    this.Deskripsi = Deskripsi;
}
}

```

Desain *form* (GUI_Wisata.java):



Gambar 4.5 Desain GUI_Wisata.java

Tabel 4.3 Properti Desain GUI_Wisata.java

No	Nama Komponen	Properti	Value
1	jLabel1	Text	Wisata
2	jLabel2	Text	Nama
3	jLabel3	Text	Kota
4	jLabel4	Text	Deskripsi
5	jTextField1	Name	txtNama

		Text	“”
6	jTextField2	Name	txtKota
		Text	txtDes
7	jTextField3	Name	“ “
		Text	
8	jButton1	Name	Add
		Text	btnData
9	jScrollPane1	Name	memoDta
		Text	“ “

Source code Button /combobox GUI_Wisata.java:

```
private void
btnDataActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    memoData.setText("");
    // Wisata wst = new Wisata();
    List wst = new List();
    wst.dataNama(txtNama.getText());
    wst.setKota(txtKota.getText());
    wst.dataDeskripsi(txtDes.getText());
    memoData.append("List Wisata\n");
    memoData.append("-----\n");
    memoData.append(" Nama : " + wst.Nama + "\n");
    memoData.append(" Kota: " + wst.getkota() +
"\n");
    memoData.append(" Deskripsi: " + wst.Deskripsi +
"\n");

    Reservasi r = new Reservasi();
    r.setVisible(true);
}
```

Hasil Tampilan:



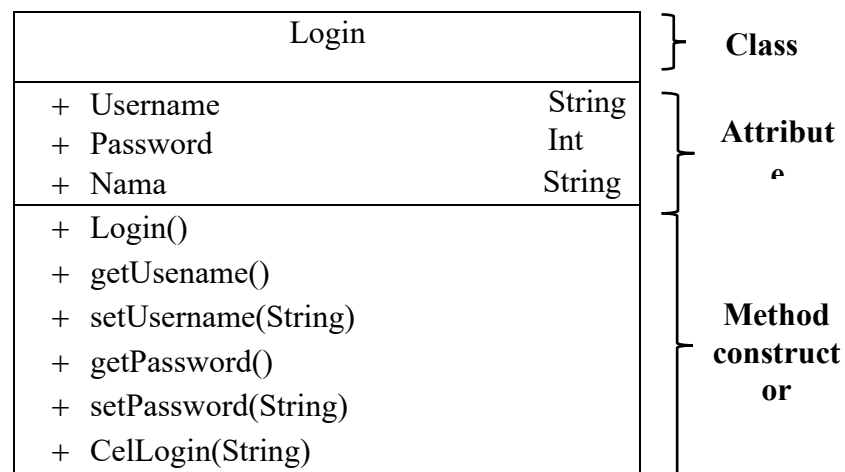
Gambar 4.6 Hasil Tampilan GUI_Wisata.java

Analisa:

Pada program diatas kita membuat sebuah *class* list yang tersambung dengan gui wisata. Atribut yang dipakai yaitu nama, kota, dan jumlah serta metode yang dipakai yaitu `getkota`, `setkota`, `datanama`, dan `datadeskripsi`. Setelah di input nama, kota dan deskripsi kita klik add setelah itu akan muncul inputan yang di sebelah kanan.

Implementasi Enkapsulasi *Class* Login

Diagram Class (Login):



Source code Object Class /Abstact (Login.java):

```
public class Login {
    private String username, password;
    public String nama;
    public Login()
    {
        nama = "Tesa";
        username = "tesalonika";
        password = "12345";
    }
}
```

```

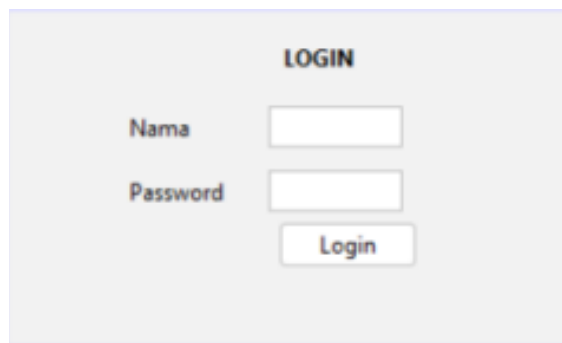
    }

    public String getUsername()
    {
        return username;
    }
    public void setUsername(String username)
    {
        this.username = username;
    }
    public String getPassword()
    {
        return password;
    }
    public void setPassword(String password)
    {
        this.password = password;
    }

    boolean CekLogin(String username, String
password)
    {
        if (username.equals(getUsername()) &&
password.equals(getPassword()))
        {
            return true;
        }
        return false;
    }
}

```

Desain *form* (GUI_Login.java):



Gambar 4.7 Desain GUI_Login.java

Tabel 4.4 Properti Desain GUI_Login.java

No	Nama Komponen	Properti	Value
1	jLabel1	Text	Login
2	jLabel2	Text	Nama
3	jLabel3	Text	Password

4	jTextField1	Name	txtnama
		Text	“ “
5	jTextField2	Name	txtPass
		Text	“ “
6	jButton1	Name	txtPass
		Text	Login

Source code Button/combobox (.btnLogin):

```
private void
btnLoginActionPerformed(java.awt.event.ActionEvent
evt) {
    // TODO add your handling code here:
    Login l = new Login();
    String username = txtNama.getText();
    String password = txtPass.getText();
    boolean Authenticated =
l.CekLogin(username, password);
    if (Authenticated)
    {
        JOptionPane.showMessageDialog(rootPane,
"LOGIN BERHASIL, "+l.nama + "!");
        Wisata w = new Wisata();
        w.setVisible(true);
        this.dispose();
    }else
    {
        JOptionPane.showMessageDialog(rootPane,
"LOGIN GAGAL. Silahkan periksa kembali username dan
password Anda.");
    }
}
```

Hasil Tampilan:



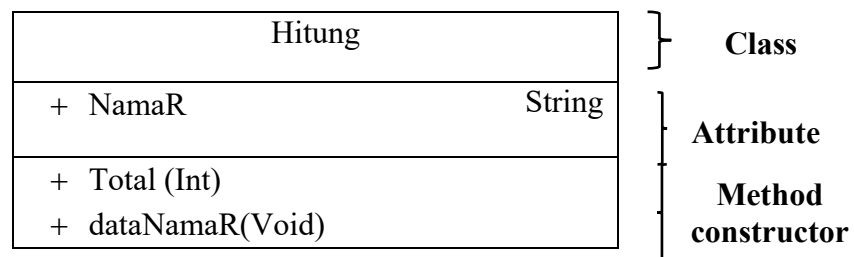
Gambar 4.8 Hasil Tampilan GUI_Login.java

Analisa:

Program diatas merupakan tampilan gui wisata dari class list,pada bagian ini akan tampil nama,kota dan deskripsi.pada tampilan gui kedua tampilan login.pada login ada nama dan password.Pada bagian class list yang menjadi private yaitu kota dan pada bagian login private String username, password.

Implementasi Enkapsulasi *Class* Hitung

Diagram Class (Hitung.java):



Source code Object Class/Abstract (Hitung):

```
public class Hitung extends List{
    String NamaR;
    private int Harga;

    void dataNamaR(String NamaR)
    {
        this.NamaR = NamaR;
    }
    public int getHarga()
    {
        return Harga;
    }
    public void setHarga(int Harga)
    {
        this.Harga = Harga;
    }

    public Hitung(){
        this.Harga = 25000;
        this.Jumlah = Jumlah;
        this.Total = Total;
    }
    public int Total()
    {
        Total = (Harga * Jumlah);
        return Total;
    }
}
```

Desain *form* (GUI_Reservasi.java):

Gambar 4.9 Desain GUI_Reservasi.java

Tabel 4.5 Properti Desain GUI_Reservasi.java

No	Nama Komponen	Properti	Value
1	jLabel1	Text	Deskripsi Hoodie
2	jLabel2	Text	Merk
3	jLabel3	Text	Warna
4	jLabel4	Text	Bahan
5	jTextField1	Name	txtR
		Text	“ “
6	jTextField2	Name	txtHarga
		Text	“ “
7	jTextField3	Name	txtHarga
		Text	“ “
8	jButton1	Name	btnHitung
		Text	Hitung
9	jScrollPane1	Name	memo
		Text	—

Source code Button/combobox (GUI_Reservasi.java):

```
private void
btnHitungActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    memo.setText("");
    Hitung h = new Hitung();
    h.dataNamaR(txtR.getText());
    h.Jumlah = Integer.parseInt(txtJumlah.getText());

    memo.append(" Reservasi\n");
    memo.append("Nama : "+h.NamaR+"\n");
    memo.append("Jumlah Orang : "+h.Jumlah+"\n");
    memo.append("Total : "+Integer.toString(h.Total()));
}
```

Hasil Tampilan:



Gambar 4.10 Hasil Tampilan GUI_Reservasi.java

Analisa:

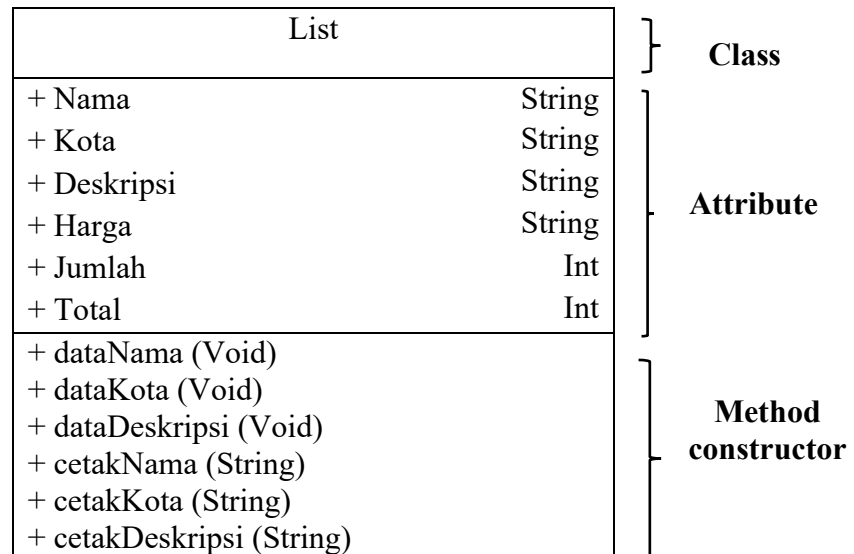
Program diatas merupakan implementasi dari class hitung yang di hubungkan gui reservasi.Tampil yang amuncul nama pemesan,harga tiket dan jumlah orang.Setelah itu akan muncul di memo sesuai apa yang di inputkan.Pada class ini yang di jadikan private yaitu Harga.

IV.6 Tugas Rumah 2:

Tema :Sistem Informasi

Judul : Implementasi *Overrloading* dan *overriding*

Diagram Class (List.java):



Source code Object Class Listyang ditambahkan Overrloading:

```
public class List {
    String Nama, Deskripsi;
    private String Kota;
    int Jumlah, Total;

    public String getkota()
    {
        return Kota;
    }
    public void setKota(String Kota){
        this.Kota = Kota;
    }
    void dataNama(String Nama){
        this.Nama = Nama;
    }
    void dataDeskripsi(String Deskripsi){
        this.Deskripsi = Deskripsi;
    }
}
```

Desain *form* (GUI_Wisata.java):

Gambar 4.11 Desain GUI_Wisata.java

Tabel 4.6 Properti Desain GUI_Wisata.java

No	Nama Komponen	Properti	Value
1	jLabel1	Text	Wisata
2	jLabel2	Text	Nama
3	jLabel3	Text	Kota
4	jLabel4	Text	Deskripsi
5	jTextField1	Name	txtNama
		Text	“ “
6	jTextField2	Name	txtKota
		Text	“ “
8	jTextField3	Name	txtDes
		Text	“ “
9	JLabel8	Name	memoDta
		Text	“ “

Source code Button Add pada GUI_Wisata.java:

```
private void
btnDataActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    memoData.setText("");
    // Wisata wst = new Wisata();
    List wst = new List();
```

```

wst.dataNama(txtNama.getText());
wst.setKota(txtKota.getText());
wst.dataDeskripsi(txtDes.getText());
memoData.append("List Wisata\n");
memoData.append("-----\n");
memoData.append(" Nama : " + wst.Nama + "\n");
memoData.append(" Kota: " + wst.getkota() +
"\n");
memoData.append(" Deskripsi: " + wst.Deskripsi +
"\n");

Reservasi r = new Reservasi();
r.setVisible(true);
}

```

Hasil Tampilan:

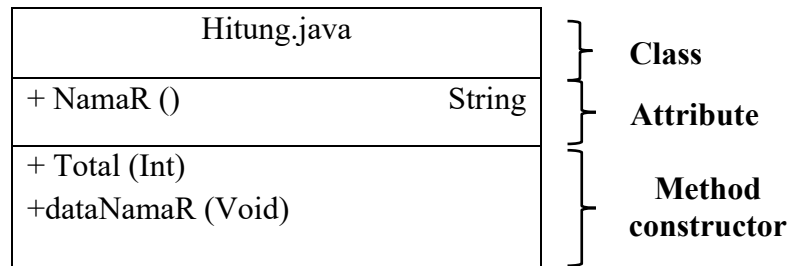


Gambar 4.12 Hasil Tampilan GUI_Wisata.java

Analisa:

Pada program diatas kita sebuah class listkita menambahkan metode overriding. Pada gui terdapat label (jLabel1, jLabel2, jLabel3, jLabel4) untuk menandai Wisata, Nama, Kota, dan Deskripsi. Pada JTextField (jTextField1, jTextField2, jTextField3) untuk memasukkan data Nama, Kota, dan Deskripsi. Class List yang melibatkan overriding (overriding adalah suatu konsep dalam pemrograman berorientasi objek di mana suatu metode yang didefinisikan dalam kelas induk (superclass) dapat diimplementasikan ulang dalam kelas anak (subclass)).

Diagram Class Overring (Class Hitung):



Source code Object Class (Hitung.java):

```
public class Hitung extends List{
    String NamaR;
    private int Harga;

    void dataNama(String NamaR)
    {
        this.NamaR = NamaR;
    }
    public int getHarga()
    {
        return Harga;
    }
    public void setHarga(int Harga)
    {
        this.Harga = Harga;
    }

    public Hitung(){
        this.Harga = 25000;
        this.Jumlah = Jumlah;
        this.Total = Total;
    }
    public int Total()
    {
        Total = (Harga * Jumlah);
        return Total;
    }
}
```

Desain form (GUI_Reservasi.java):

The image shows a Java Swing window titled "RESERVASI". Inside the window, there are three text input fields arranged vertically, each with a label to its left: "Nama", "Harga Tiket", and "Jumlah Orang". Below these input fields is a button labeled "Hitung". At the bottom of the window is a large, empty rectangular area, likely intended for displaying the result of the calculation.

Gambar 4.13 Desain GUI_Reservasi.java

Tabel 4.7 Properti Desain GUI_Resrvasi.java

No	Nama Komponen	Properti	Value
1	jLabel1	Text	Reservasi
2	jLabel2	Text	Nama
3	jLabel3	Text	Harga Tiket
4	jLabel4	Text	Jumlah Orang
5	jTextField1	Name	txtR
		Text	“ “
6	jTextField2	Name	txtHarga
		Text	“ “
7	jTextField3	Name	txtJumlah
		Text	“ “
8	jButton1	Name	btnHitung
		Text	Hitung
9	jScrollPane1	Name	memo
		Text	-

Source code Button Hitung pada GUI_Reservasi.java:

```
private void
btnHitungActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    memo.setText("");
    Hitung h = new Hitung();
    h.dataNamaR(txtR.getText());
    h.Jumlah = Integer.parseInt(txtJumlah.getText());

    memo.append(" Reservasi\n");
    memo.append("Nama : "+h.NamaR+"\n");
    memo.append("Jumlah Orang : "+h.Jumlah+"\n");
    memo.append("Total : "+Integer.toString(h.Total()));
}
```

Hasil Tampilan:



The screenshot shows a Java Swing window titled "RESERVASI". Inside the window, there are three text input fields: "Nama" with the value "Tesa", "Harga Tiket" with the value "25000", and "Jumlah Orang" with the value "5". Below these fields is a button labeled "Hitung". At the bottom of the window, there is a text area displaying the following information: "Reservasi", "Nama : Tesa", "Jumlah Orang : 5", and "Total :125000".

Gambar 4.14 Hasil Tampilan GUI_Reservasi.java

Analisa:

Program diatas merupakan implementasi dari class hitung yang di hubungkan gui reservasi.Tampil yang amuncul nama pemesan,harga tiket dan jumlah orang.Setelah itu akan muncul di memo sesuai apa yang di inputkan.Pada class ini yang di jadikan private yaitu Harga.Pada class overloading terdapat pada class hitung yang berfungsi untuk berfungsi kemampuan dari subclass untuk memodifikasi methoddarisuperclass.



IV.7 Kesimpulan

1. *Method Overloading* adalah sebuah kemampuan yang membolehkan sebuah class mempunyai 2 atau lebih *method* dengan nama yang sama.
2. Enkapsulasi merupakan proses pemaketan objek beserta methodnya untuk menyembunyikan rincian implementasi dari pemakai/objek lainnya. Dengan menggunakan *method overloading* , kita dapat membuat beberapa versi dari sebuah *method* dengan statemen yang berbeda berdasarkan parameter yang diberikan padanya
3. Ini akan membantu menggunakan kembali nama *method* yang sama untuk tujuan yang berbeda, dengan begitu *method overloading* akan memberikan fleksibilitas saat menggunakan sebuah *method*. Overriding method adalah kemampuan dari subclass(child class) untuk memodifikasi method dari superclass, dengan cara mendefinisikan kembali method superclass-nya.

Nama Aslab :	TTD :
Firman Frezy Pradana 2118112	
Tanggal : 17 Desember 2023	



BAB V

ABSTRACT, POLIMORFISME, DAN INTERFACE

Jumlah Pertemuan	:	2 x 60 menit
Tujuan Praktikum	:	<ol style="list-style-type: none">1. Mampu menerapkan konsep abstract, polimorfisme, interface2. Mampu mengetahui fungsi konsep abstract, polimorfisme, interface3. Mampu membuat penerapan abstract, polimorfisme, interface
Alat / bahan	:	<ol style="list-style-type: none">1. Seperangkat <i>computer</i>.2. Perangkat lunak: <i>Netbeans</i>.3. Modul Praktikum <i>OOP 2022</i>.

V.1 Landasan Teori

A. Abstract

Abstract Class adalah sebuah *class* yang tidak bisa di-*instansiasi* (tidak bisa dibuat menjadi objek) dan berperan sebagai ‘kerangka dasar’ bagi *class* turunannya. Di dalam sebuah *abstract class* setidaknya memiliki satu atau lebih *method* abstrak. Fungsi dari *class abstract* ini adalah untuk mempertahankan *hirarky* dari *parent class* ke kelas turunan dari induknya. *Abstract class* digunakan ketika ingin mendefinisikan sebuah *class* yang tidak dapat diinstansiasi langsung, tetapi dapat digunakan sebagai dasar untuk membuat subclass lain yang dapat diinstansiasi. Hal tersebut memungkinkan untuk menyediakan struktur dasar dan peraturan yang harus dipatuhi oleh subclass-subclass yang akan di buat.

Berikut bentuk deklarasi *Class Abstract* pada Java:

```
Akses_modifier abstract class namaClassAbstrak
{
    .....// isi
}
```

Contoh penerapan:

```

14 public abstract class bentuk {
15     //isi class abstract
16 }

```

Abstract Method adalah sebuah ‘method dasar’ yang harus direpresentasikan ulang di dalam class anak (*child class*). Abstract method ditulis tanpa isi dari method, melainkan hanya ‘signature’-nya saja (ciri khas). Signature dari sebuah method adalah bagian method yang terdiri dari nama method dan parameternya (jika ada).

Berikut bentuk deklarasi Class Abstract pada Java:

```

abstract tipe_data namaMethodAbstract(parameter); //non-void
abstract void namaMethodAbstract(parameter); // void

```

Contoh penerapan:

```

22 abstract double luas();
    abstract double keliling();

```

Abstract class digunakan di dalam inheritance (pewarisan class) untuk ‘memaksakan’ implementasi method yang sama bagi seluruh class yang diturunkan dari abstract class. Abstract class digunakan untuk membuat struktur logika penurunan di dalam pemrograman objek.

Java memiliki aturan-aturan dalam penggunaan method abstrak dan class abstrak sebagai berikut:

- 1 Class yang di dalamnya terdapat abstract method harus dideklarasikan sebagai abstract class.
- 2 Abstract class tidak dapat diinstansi, tetapi harus di turunkan.
- 3 Abstract class tidak dapat diinstansi (menjadi objek dari class abstract), tetapi kita dapat mendeklarasikan suatu variable yang bertipe abstract class dan membuat instansi dari variabel tersebut yang bertipe class turunan dari abstract class tersebut (*teknik polymorphism*).
- 4 Sebuah class dapat dideklarasikan sebagai abstract class meskipun class tersebut tidak memiliki abstract method.

- 5 Abstract method tidak boleh mempunyai body method dan demikian juga sebaliknya bahwa method yang tidak ditulis body methodnya maka harus dideklarasikan sebagai abstract method.

B. Polimorfisme

1. Pengetian Polimorfisme

Poly artinya banyak, *morfisme* artinya bentuk. Dalam konteks OOP, polimorfisme memungkinkan suatu entitas (seperti method atau function) untuk berperilaku dengan cara yang berbeda tergantung pada konteks di mana entitas tersebut digunakan “Bentuk” di sini dapat kita artikan: isinya berbeda (overriding), parameternya berbeda dan tipe datanya berbeda (overloading). Polymorphism digunakan untuk mengimplementasi suatu fungsi dari sebuah induk class maupun Interface, baik fungsi yang abstract maupun sudah terdefinisi, untuk diimplementasikan sesuai dengan relevansi suatu class. Polimorfisme mengizinkan kelas induk untuk mendefinisikan sebuah method general (bersifat umum) untuk semua kelas turunannya.

Polymorphisme pada *java* terdiri dari 2 jenis, yaitu *Static Polymorphism* dan *Dynamic Polymorphism*. *Static Polymorphism* adalah *Polymorphism* yang dilakukan pada waktu *compile (compile time)*, sedangkan *Dynamic Polymorphism* adalah *Polymorphism* yang dilakukan pada waktu berjalannya program (*run time*).

- a. *Compile-time*: Tipe *compile-time* atau *static* terjadi ketika metode dijalankan pada waktu kompilasi (*compile-time*), yaitu saat kode sumber diubah menjadi kode yang dapat dieksekusi. Tipe ini terjadi saat menjalankan metode *overloading*.
- b. *Run time*: Tipe *runtime* atau *dynamic* merujuk pada metode dijalankan tepat saat kode yang dapat dieksekusi mulai dijalankan. *Run-time* berlangsung saat metode *overriding*.

Berikut bentuk deklarasi *polimorfisme*:

a. *Static Polimorfisme*

Pada static polimorfisme menggunakan konsep overloading, overloading sendiri memiliki ciri ciri bahwa pada suatu class

terdapat method dengan nama yang sama , lalu tipe data dan parameter berbeda, inilah yang dinamakan method overloading.

Berikut salah satu bentuk deklarasi *Polimorfisme static*

SuperClass

```
Public nama_class() {
    double nama_method(double d) {
        //isi
    }
    double nama_method(double d, double f) {
        //isi
    }
}
```

Main class

```
Public nama_class {
    // Main driver method
    public static void main(String[] args)
    {
        // Calling method by passing
        // input as in arguments
        System.out.println(nama_method(2,4));
        System.out.println(nama_method(5.5, 6.3));
    }
}
```

b. *Dynamic Polymorphism*

Polimorfisme dinamik menggunakan konsep method overridding dalam penerapannya dan dilakukan disaat run time(selesai menjalankan program). Pada polimorfisme dinamis biasanya terjadi saat kita menggunakan pewarisan (inheritance) dan implementasi interface.

Berikut salah satu bentuk deklarasi *polimorfisme dynamic*

Super Class

```
Public Super_Class() {
    Void nama_method() {
        isi
    }
}
```

Sub Class 1

```
Public Sub_Class1 extend Super_Class() {
@Override
    Void nama_method() {Isi2
    }
}
```

Sub Class 2

```
Public Sub_Class2 extend
Super_Class() { @override
    Void nama_method() {Isi3
    }
    }}
}
```

Main Class

```
public class main {
    public static void main(String[] args) {
        Super_Class induk; // instansiasi
                           super class

        induk = new Sub_Class1();
        // memanggil method dari class Sub_Class1
        induk.nama_method();

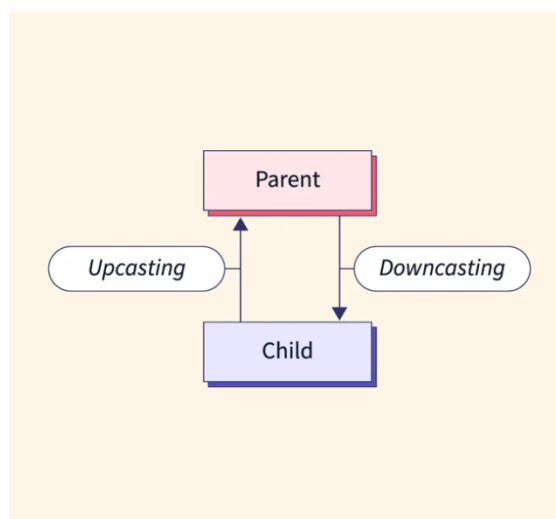
        induk = new Sub_Class2();
        // memanggil method dari class Sub_Class2
        induk.nama_method();
    }
}
```

Keterangan:

- 1) Instansiasi: pembuatan objek dari suatu class (objek “induk” dari Super Class).
- 2) Induk merupakan objek polimorfisme dari class Super_Class, kemudian diisi dengan objek dari Sub_Class1
- 3) Method nama_method() adalah overridden method dari kelas Super_Class, yang di override sub class.
- 4) Ketika dipanggil induk.nama_method(), yang dimaksud adalah nama_method() dari Super_Class, namun ketika di running, yang ditampilkan adalah method nama_method() dari Sub_Class1 (karena objek induk diinstansiasi sebagai objek dari Sub_Class). Begitu juga pada Sub_Class2.

2. Casting

Casting adalah proses mengubah tipe data dari satu tipe ke tipe (attribut maupun method) lainnya. Casting diperlukan untuk mengubah atau menyesuaikan tipe data antara tipe data yang berbeda. Terdapat 2 jenis casting yaitu Upcasting dan downcasting adalah dua konsep yang terkait dengan pewarisan (inheritance) pada pemrograman berorientasi objek yang memanfaatkan hubungan antara kelas-kelas dalam hierarki pewarisan. Kedua konsep ini akan memberikan fleksibilitas dan kemampuan untuk mengelola hubungan antara kelas-kelas dalam hierarki pewarisan.



Gambar 5.1 Konsep Updasting dan Downcasting

a. Upcasting

Upcasting adalah Proses mengkonversi objek, dari tipe sub class (class anak/turunan) ke tipe super class (class parent), ke arah atas pada hirarki pewarisan (inheritance). Dalam upcasting, objek dari kelas turunan dapat dianggap sebagai objek dari kelas induknya. Hal ini berarti object dari kelas anak dapat mengakses anggota kelas induk menggunakan objek kelas turunan setelah dilakukan upcasting.

Bentuk Penerapan

```
SuperClass obj = new subclass()
```

c. Downcasting

Downcasting adalah proses mengubah objek dari tipe induk ke tipe turunannya dalam hierarki pewarisan. Dalam downcasting, objek anak diubah kembali dari objek kelas induk menjadi objek kelas turunan agar dapat mengakses metode atau atribut khusus dari kelas turunan tersebut.

Bentuk Penerapan

```
SuperClass obj1 = new subclass();

subClass1 obj2 = (subClass2) obj1;
```

C. Interface

1. Pengertian Interface

Interface merupakan sebuah antarmuka ,yang secara umum interface berfungsi sebagai penghubung antar sesuatu yang '*Abstract*' dengan sesuatu yang nyata. Terdapat istilah '*able*' atau 'mampu' yaitu istilah yang sering digunakan untuk menunjukkan bahwa , objek yang mengimplementasikan interface tersebut memiliki sesuatu atau memiliki kemampuan tertentu. Dalam OOP, sebuah interface dapat dianggap sebagai prototipe atau templat untuk sebuah kelas. Analoginya, jika sebuah class abstrak adalah kerangka dasar untuk kelas-kelas lain, maka interface adalah templat yang memberikan struktur khusus untuk kelas tertentu.

Ketika kita mendefinisikan metode di dalam sebuah interface, kita hanya memberikan gambaran atau "prototipe" dari metode tersebut tanpa memberikan implementasi konkret (abstract method). Kelas-kelas yang mengimplementasikan interface tersebut wajib meng-Override method untuk mendefinisikan implementasi metode – metode ini. Jadi Interface ini ,digunakan sebagai protokol untuk kelas yang mengimplementasi interface tersebut memiliki method yang ada di interface.

2. Deklarasi Interface

Interface secara struktur mirip dengan *class*, tetapi dengan perbedaan penting. Isi dari *interface* terdiri dari deklarasi metode yang

bersifat *abstrak*, yaitu metode yang hanya didefinisikan tanpa isi (badan metode). Deklarasi metode dalam *interface* mirip dengan deklarasi metode pada kelas *abstrak*.

variabel yang dideklarasikan dalam *interface* otomatis dianggap sebagai *static* dan *final*. Sedangkan Metode dalam *interface* secara otomatis dianggap sebagai *public* dan *abstract*. Selain itu, metode dalam *interface* juga dapat memiliki implementasi *default* menggunakan kata kunci *default*. Implementasi default ini dapat digunakan oleh kelas yang mengimplementasikan *interface* tanpa harus mengubahnya.

Sintak untuk menciptakan *interface* serupa dengan cara menciptakan sebuah *class* tetapi terdapat beberapa pengecualian, yaitu:

- a. Seluruh *method* yang di deklarasikan pada *Interface* pasti bersifat *public* dan *abstract*.
- b. *Variable* selalu bersifat *public*, *final* dan *static*.

V.2 Langkah – Langkah Praktikum

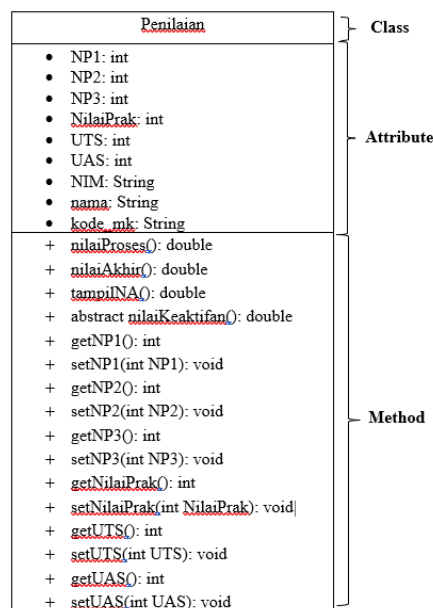
1. Buka Aplikasi *Netbeans*.
2. Buat *Class*.
3. Memberi *script* pada kelas tersebut.
4. Buat *form*.
5. Memberi *script* pada *form* tersebut.
6. Menjalankan program.

V.3 Tugas Praktikum 1 :

Membuat *Class* Penilaian dan Keaktifan Mahasiswa dan GUI_Penilaian

Judul : Mendesain ulang GUI_Penilaian

Diagram Class (Penilaian):



Gambar 5.2 Diagram Class Penilaian

Source code Object Class:

```

public abstract class Penilaian {
    public String NIM,nama,kode_mk;
    private int NP1,NP2,NP3,NilaiPrak,UTS,UAS;
    public double nilaiProses() {
        return
        ((NP1*0.1)+(NP2*0.2)+(NP3*0.1)+(UTS*0.2)+(NilaiPrak*0.4));
    }

    public double nilaiAkhir(){
        return  (nilaiProses()*0.6) +(UAS*0.3);
    }

    public double tampilNA(){
        return nilaiAkhir();
    }

    abstract double nilaiKeaktifan();

    public int getNP1() {
        return NP1;
    }

    public void setNP1(int NP1) {
        this.NP1 = NP1;
    }

    public int getNP2() {
        return NP2;
    }

    public void setNP2(int NP2) {
        this.NP2 = NP2;
    }

    public int getNP3() {

```

```

        return NP3;
    }

    public void setNP3(int NP3) {
        this.NP3 = NP3;
    }

    public int getNilaiPrak() {
        return NilaiPrak;
    }

    public void setNilaiPrak(int NilaiPrak) {
        this.NilaiPrak = NilaiPrak;
    }

    public int getUTS() {
        return UTS;
    }

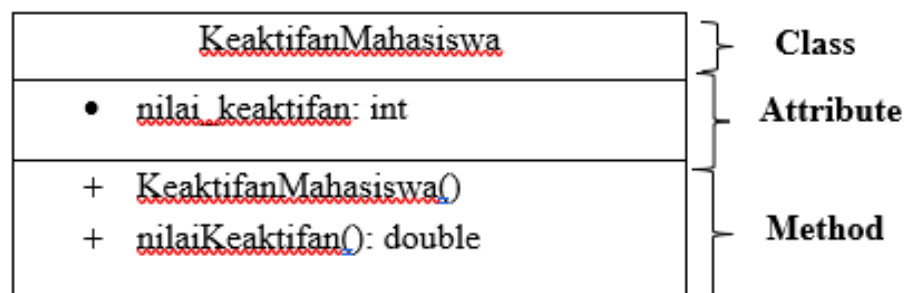
    public void setUTS(int UTS) {
        this.UTS = UTS;
    }

    public int getUAS() {
        return UAS;
    }

    public void setUAS(int UAS) {
        this.UAS = UAS;
    }
}

```

Diagram Class (KeaktifanMahasiswa):



Gambar 5.3 Diagram Class KeaktifanMahasiswa

Source code Object Class:

```

public class KeaktifanMahasiswa extends Penilaian{
    int nilai_keaktifan;
    public KeaktifanMahasiswa(){
        this.nilai_keaktifan = 0;
    }
    @Override
    public double nilaiKeaktifan(){
        return ((nilai_keaktifan* 0.1) + nilaiAakhir());
    }
}

```

Desain *form* (GUI_Penilaian.java):

Gambar 5.4 Desain Gui_ Penilaian.java

Tabel 5.1 Properti Desain GUI_Penilaian.java

No	Objek	Properti	Nilai
1	jLabel1	Text	PROGRAM PENILAIAN
2	jLabel2	Text	NIM
3	jLabel3	Text	Nama
4	jLabel4	Text	Kode Matakuliah
5	jLabel5	Text	NP1
6	jLabel6	Text	NP2
7	jLabel7	Text	NP3
8	jLabel8	Text	UTS
9	jLabel9	Text	UAS
10	jLabel10	Text	Praktikum
11	jLabel11	Text	Nilai Keaktifan
12	jTextField1	Name	txtNIM
		Text	“ “
13	jTextField2	Name	txtNama
		Text	“ “
14	jTextField3	Name	txtKodeMK
		Text	“ “

15	jTextField4	Name	txtNP1
		Text	“ “
16	jTextField5	Name	txtNP2
		Text	“ “
17	jTextField6	Name	txtNP3
		Text	“ “
18	jTextField7	Name	txtUts
		Text	“ “
19	jTextField8	Name	txtPraktikum
		Text	“ “
20	jTextField9	Name	txtUas
		Text	“ “
21	jTextField10	Name	txtKeaktifan
		Text	“ “
22	jButton1	Name	btnNA
		Text	Hasil Nilai Akhir
23	jTable	Name	table_penilaian_matakuliah
		Text	“ “
24	JCheckBox	Name	checkBox
		Text	Tambahkan Nilai Keaktifan

Source code Button Cetak pada GUI:

```
import java.util.ArrayList; import java.util.List;
import javax.swing.JOptionPane;
import javax.swing.table.DefaultTableModel
```

Source code Constructor GUI:

```
public Gui_Penilaian() {
    initComponents();
    // Mengambil model data dari tabel dan menyimpannya
    dalam objek DefaultTableModel dataModel
    DefaultTableModel dataModel = (DefaultTableModel)
    table_penilaian_matakuliah.getModel();
    // Mendapatkan jumlah baris yang ada dalam model
    data saat ini
    int rowCount = dataModel.getRowCount();
    while (rowCount > 0) {
        // Menghapus baris terakhir dari model data
        dataModel.removeRow(rowCount - 1);
        // Memperbarui nilai rowCount setelah
        penghapusan baris terakhir
        rowCount = dataModel.getRowCount(); // Update
        rowCount after removal
    }

    KeaktifanMahasiswa nilai = new KeaktifanMahasiswa();
    txtKeaktifan.setText(Integer.toString(nilai.nilai_keaktifan)
    );
    txtKeaktifan.setEnabled(false);

}
```

Source code Clear pada GUI:

```
public void clear() {
    txtNim.setText("");
    txtNama.setText("");
    txtKodeMK.setText("");
    txtNP1.setText("");
    txtNP2.setText("");
    txtNP3.setText("");
    txtPraktikum.setText("");
    txtUts.setText("");
    txtUas.setText("");
    txtKeaktifan.setText("0");
}
```

Source code Button Hasil Nilai Akhir pada GUI:

```
private void btnNAActionPerformed(java.awt.event.ActionEvent
    evt) {
    double keaktifan;
    JOptionPane.showMessageDialog(null, "Data anda
    Ditambahkan Ke tabel");
    // Mengambil model data dari tabel
    DefaultTableModel dataModel = (DefaultTableModel)
    table_penilaian_matakuliah.getModel();
    // Inisialisasi sebuah ArrayList bernama 'list'
```



```

List list = new ArrayList<>();
KeaktifanMahasiswa nilai = new KeaktifanMahasiswa();
nilai.NIM = txtNim.getText();
nilai.nama = txtNama.getText();
nilai.kode_mk = txtKodeMK.getText();
nilai.setNP1(Integer.parseInt(txtNP1.getText()));
nilai.setNP2(Integer.parseInt(txtNP2.getText()));
nilai.setNP3(Integer.parseInt(txtNP3.getText()));

nilai.setNilaiPrak(Integer.parseInt(txtPraktikum.getText()));
nilai.setUTS(Integer.parseInt(txtUts.getText()));
nilai.setUAS(Integer.parseInt(txtUas.getText()));
nilai.nilai_keaktifan =
Integer.parseInt(txtKeaktifan.getText());

if (checkBox.isSelected()) {
    keaktifan = nilai.nilaiKeaktifan();
} else {
    keaktifan = nilai.tampilNA();
}
// Mengatur tabel untuk membuat kolom dari model
secara otomatis

table_penilaian_matakuliah.setAutoCreateColumnsFromModel(true
);

list.add(nilai.NIM);
list.add(nilai.kode_mk);
list.add(nilai.getNP1());
list.add(nilai.getNP2());
list.add(nilai.getNP3());
list.add(nilai.getNilaiPrak());
list.add(nilai.getUTS());
list.add(nilai.getUAS());
list.add(keaktifan);
// Menambahkan baris baru ke model tabel menggunakan
data dari ArrayList 'list'
dataModel.addRow(list.toArray());
// Memanggil fungsi 'clear' untuk membersihkan nilai
dari komponen
clear();

}

```

Hasil Tampilan:

Nim	KD MK	NP 1	NP 2	NP 3	UTS	PRAK	UAS	NA
2218	IF2001	95	95	95	95	95	95	93.0

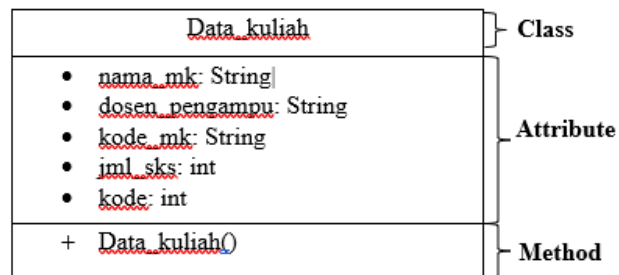
Gambar 5.5 Hasil Tampilan GUI_Penilaian.java

V.4 Tugas Praktikum 2 :

Membuat *Class* Data_kuliah dan GUI_Matkul

Judul : GUI_Matkul

Diagram Class (Data_Kuliah):



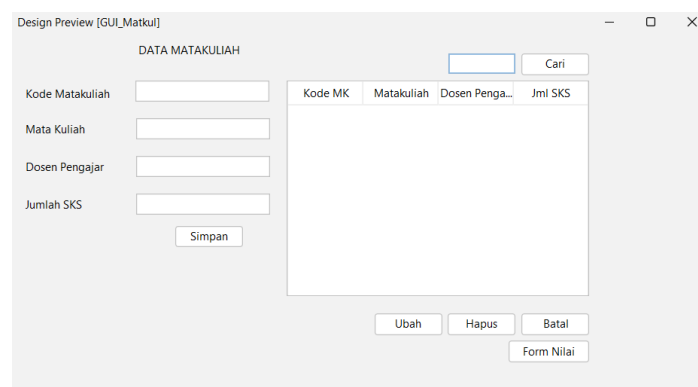
Gambar 5.6 *Diagram Class* Data_kuliah

Source code Object Class:

```

public class Data_kuliah {
    String nama_mk , dosen_pengampu ,kode_mk;
    int jml_sks,kode;
    public Data_kuliah(){
        this.kode_mk = "IF2112";
        this.nama_mk = "OOP";
        this.dosen_pengampu = "YOSEP AGUS PRANOTO,
ST.MT.";
        this.jml_sks = 4;
    }
}
  
```

Desain form (GUI_Matkul.java):



Gambar 5.7 Desain Gui_Matkul.java

Tabel 5.2 Properti Desain GUI_Matkul.java

No	Objek	Properti	Nilai
1	jLabel1	Text	DATA MATA KULIAH
2	jLabel2	Text	Kode Mata Kuliah

3	jLabel3	Text	Mata Kuliah
4	jLabel4	Text	Dosen Pengajar
5	jLabel5	Text	Jumlah SKS
6	jTextField1	Name	txtKdMatakuliah
		Text	-
7	jTextField2	Name	txtMatakuliah
		Text	-
8	jTextField3	Name	txtDosenPengajar
		Text	-
9	jTextField4	Name	txtJmlSks
		Text	-
10	jTextField5	Name	txtCari
		Text	--
11	jButton1	Name	btnSimpan
		Text	Simpan
12	jButton2	Name	btnUbah
		Text	Ubah
13	jButton3	Name	btnHapus
		Text	Hapus
14	jButton4	Name	btnBatal
		Text	Batal
15	jButton5	Name	btnCari
		Text	Cari
16	jButton6	Name	btnNilai
		Text	Form Nilai

	JTable	Name	
--	--------	------	--

Source code Library pada GUI:

```
import java.util.ArrayList; import java.util.List;
import javax.swing.JOptionPane;
import javax.swing.table.DefaultTableModel;
```

Source code Constructor pada GUI:

```
public GUI_Matkul() {
    initComponents();
    DefaultTableModel dataModel = (DefaultTableModel)
jTable1.getModel();
// Mendapatkan jumlah baris yang ada dalam model data saat
ini
int rowCount = dataModel.getRowCount();
while (rowCount > 0) {
// Menghapus baris terakhir dari model data
dataModel.removeRow(rowCount - 1);
// Memperbarui nilai rowCount setelah penghapusan baris
terakhir
rowCount = dataModel.getRowCount(); // Update rowCount
after removal
}
Data_Matkul dtMatkul = new Data_Matkul();
jTextField1.setText(dtMatkul.kode_mk);
jTextField2.setText(dtMatkul.nama_mk);
jTextField3.setText(dtMatkul.dosen_pengampu);
jTextField4.setText(dtMatkul.jml_sks);
}
```

Source code Clear pada GUI:

```
public void clear(){
jTextField1.setText("");
jTextField2.setText("");
jTextField3.setText("");
jTextField4.setText("");
}
```

Source code button Simpan pada GUI:

```
JOptionPane.showMessageDialog(null, "Data anda Ditambahkan
Ke tabel");
// Mengambil model data dari tabel
DefaultTableModel dataModel = (DefaultTableModel)
jTable1.getModel();
// Inisialisasi sebuah ArrayList bernama 'list'
List list = new ArrayList<>();
// Mengatur tabel untuk membuat kolom dari model secara
otomatis
jTable1.setAutoCreateColumnsFromModel(true);
String kode_mk = jTextField1.getText();
String nama_mk = jTextField2.getText();
String dosen_pengampu = jTextField3.getText();
String jml_sks = jTextField4.getText();
```

```

Data_Matkul dtMatkul = new
Data_Matkul(jml_sks,dosen_pengampu,nama_mk,kode_mk);
list.add(dtMatkul.dosen_pengampu);
list.add(dtMatkul.nama_mk);
list.add(dtMatkul.kode_mk);
list.add(dtMatkul.jml_sks);
// Menambahkan baris baru ke model tabel menggunakan data
dari ArrayList 'list'
dataModel.addRow(list.toArray());
// Memanggil fungsi 'clear' untuk membersihkan nilai dari
komponen
clear();

```

Source code button Hapus pada GUI:

```

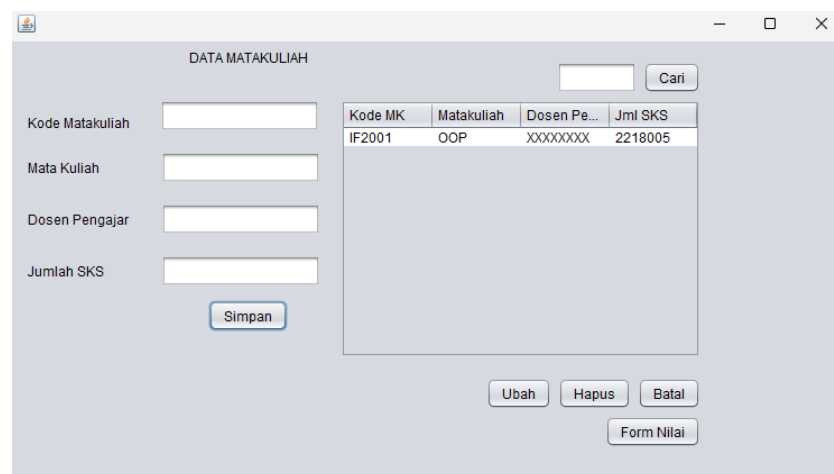
DefaultTableModel dataModel = (DefaultTableModel)
jTable1.getModel();
int rowCount = dataModel.getRowCount();
while (rowCount > 0) {
dataModel.removeRow(rowCount - 1);
rowCount = dataModel.getRowCount(); // Update
}

```

Source code button Batal pada GUI:

```
clear();
```

Hasil Tampilan:



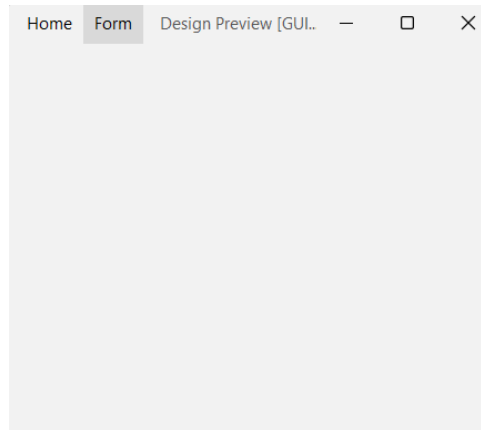
Gambar 5.8 Hasil Tampilan GUI_Matkul.java

V.5 Tugas Praktikum 3 :

Membuat GUI_MenuUtama

Judul : Membuat GUI Menu Utama

Desain *form* (GUI_MenuUtama.java):



Gambar 5.9 Desain Gui_MenuUtama.java

Tabel 5.3 Properti Desain GUI_MenuUtama.java

No	Objek	Properti	Nilai
1	jMenuBar1	jMenu	[1] = Home
			[2] = Form
		jMenuItem	[1] = GUI Mahasiswa
			[2] = GUI Mata Kuliah
			[3] = GUI Nilai

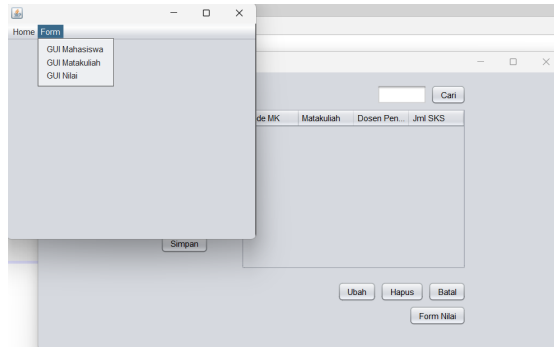
Source code Menu Form pada GUI:

```
private void
 jMenuItem1ActionPerformed(java.awt.event.ActionEvent evt) {
    GUI_Mahasiswa mhs = new GUI_Mahasiswa();
    mhs.show();
}

private void
 jMenuItem2ActionPerformed(java.awt.event.ActionEvent evt) {
    GUI_Matkul mk = new GUI_Matkul();
    mk.show();
}

private void
 jMenuItem3ActionPerformed(java.awt.event.ActionEvent evt) {
    GUI_Penilaian nilai = new GUI_Penilaian();
    nilai.show();
}
```

Hasil Tampilan:



Gambar 5.10 Hasil Tampilan GUI_MenuUtama.java

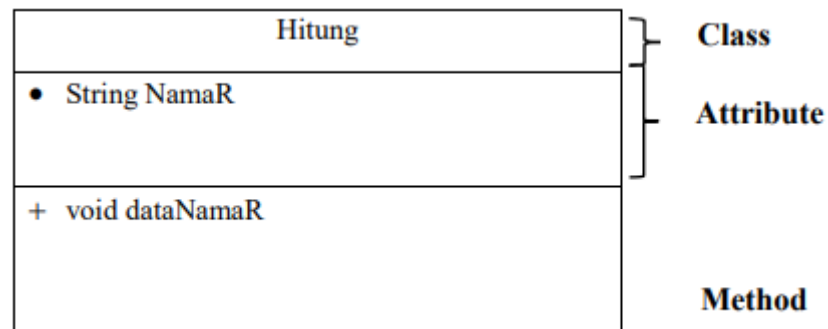
V.6 Tugas Rumah 1 :

Implementasi Class Abstract

Judul : Informasi Wisata

Tema :Informasi Wisata

Diagram Class (Hitung.java):



Gambar 5.11 Hasil Tampilam Diagram Hitung

Source code Object Class(Hitung.java):

```
public class Hitung extends Data{
    String NamaR;
    private int Harga;
    int Jumlah, Total;

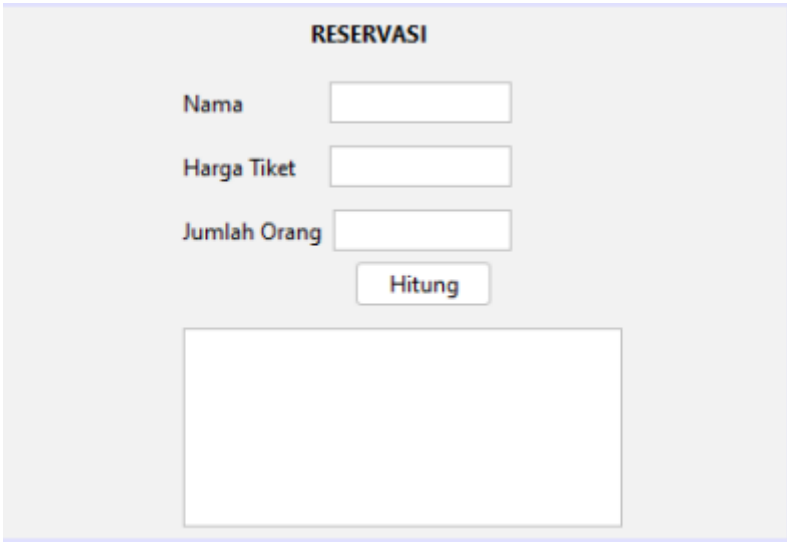
    void dataNama(String NamaR)
    {
        this.NamaR = NamaR;
    }
    public int getHarga()
    {
        return Harga;
    }
    public void setHarga(int Harga)
    {
        this.Harga = Harga;
    }
}
```

```
public Hitung(){
    this.Harga = 25000;
    this.Jumlah = Jumlah;
    this.Total = Total;
}
public int Total()
{
    Total = (Harga * Jumlah);
    return Total;
}
}
```

Source code Object Class(Data.java):

```
public abstract class Data {
    abstract int Total();
}
```

Desain form (GUI_Reservasi.java):



Gambar 5.12 Desain Gui_Reservasi.java

Tabel 5.4 Properti Desain GUI_Reservasi.java

No	Objek	Properti	Nilai
1	jLabel1	Text	Reservasi
2	jLabel2	Text	Nama
3	jLabel3	Text	Harga Tiket
4	jLabel4	Text	Jumlah Orang
5	jTextField1	Name	txtR

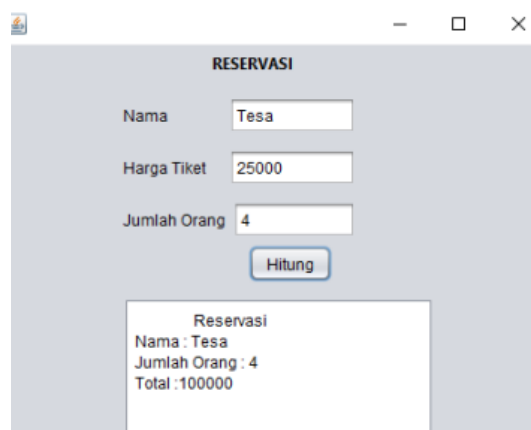
		Text	“ “
6	jTextField2	Name	txtHarga
		Text	“ “
7	jTextField3	Name	txtHarga
		Text	“ “
8	jButton1	Name	btnHitung
		Text	Hitung
9	jScrollPane1	Name	memo
		Text	-

Source code Button Hitung pada GUI:

```
private void
btnHitungActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    memo.setText("");
    Hitung h = new Hitung();
    h.dataNama(txtR.getText());
    h.Jumlah = Integer.parseInt(txtJumlah.getText());

    memo.append(" Reservasi\n");
    memo.append("Nama : "+h.NamaR+"\n");
    memo.append("Jumlah Orang : "+h.Jumlah+"\n");
    memo.append("Total :"+Integer.toString(h.Total()));
}
}
```

Hasil Tampilan:



Gambar 5.13 Hasil Tampilan GUI_Reservasi.java

Analisa:

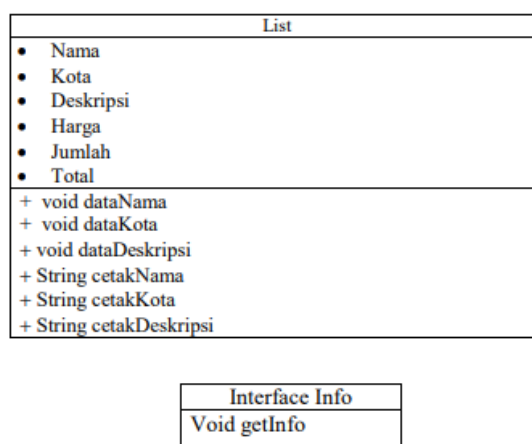
Program diatas merupakan implementasi dari class hitung yang di hubungkan gui reservasi. Tampil yang muncul nama pemesan, harga tiket dan jumlah orang. Setelah itu akan muncul di memo sesuai apa yang di inputkan. Pada class ini yang di jadikan private yaitu Harga. Pada bagian ini juga kita buat class baru yaitu kelas data yang berisi Total dan Total di buat abstract. Abstract class adalah sebuah konsep dalam pemrograman berorientasi objek (OOP) yang digunakan untuk membuat kelas dasar yang tidak dapat diinstansiasi sendiri. Sebagai contoh class Hitung dan class Data diatas.

V.7 Tugas Rumah 2 :

Tema : Sistem Informasi

Judul : Menerapkan Konsep Interface

Diagram Class (Hitung.java):



Gambar 5.14 Hasil Tampilam Diagram Hitung

Source code Object Class (Hitung.java):

```

public class Hitung extends Data{
    String NamaR;
    private int Harga;
    int Jumlah, Total;

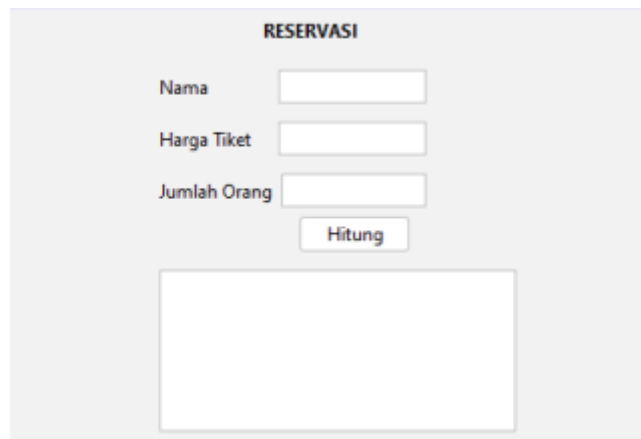
    void dataNama(String NamaR)
    {
        this.NamaR = NamaR;
    }
    public int getHarga()
    {
        return Harga;
    }
    public void setHarga(int Harga)
  
```

```
{  
    this.Harga = Harga;  
}  
  
    public Hitung(){  
        this.Harga = 25000;  
        this.Jumlah = Jumlah;  
        this.Total = Total;  
    }  
    public int Total()  
    {  
        Total = (Harga * Jumlah);  
        return Total;  
    }  
}
```

Source code Object Class (Info.java):

```
public interface Info {  
    void getInfo();  
}
```

Desain *form* (GUI_Reservasi.java):



The image shows a Java Swing window titled "RESERVASI". Inside the window, there are three text labels: "Nama", "Harga Tiket", and "Jumlah Orang", each followed by a text input field. Below these input fields is a button labeled "Hitung". At the bottom of the window is a large, empty rectangular area, likely intended for displaying the result of the calculation.

Gambar 5.15 Desain Gui_Reservasi.java

Tabel 5.5 Properti Desain GUI_Login.java

No	Objek	Properti	Nilai
1	jLabel1	Text	Reservasi
2	jLabel2	Text	Nama
	jLabel3	Text	Harga Tiket
	jLabel4	Text	Jumlah Orang
	jTextField1	Name	xtR
		Text	“ “
	jTextField2	Name	txtHarga
		Text	“ “
	jTextField3	Name	txtHarga
		Tex	“ “
	jButton1	Name	btnHitung
		Text	Hitung
	jScrollPane1	Name	memo
		Text	-

Source code Button Hitung:

```
private void
btnHitungActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    memo.setText("");
    Hitung h = new Hitung();
    h.dataNama(txtR.getText());
    h.Jumlah = Integer.parseInt(txtJumlah.getText());

    memo.append(" Reservasi\n");
    memo.append("Nama : "+h.NamaR+"\n");
    memo.append("Jumlah Orang : "+h.Jumlah+"\n");
    memo.append("Total : "+Integer.toString(h.Total()));

    #PraktikumOOP2023
}
```

Hasil Tampilan:

Gambar 5.16 Hasil tampilan gui_reservasi

Analisa:

Polimorfisme adalah salah satu konsep dalam pemrograman komputer yang memungkinkan suatu entitas, seperti fungsi, operator, atau objek, untuk dapat memiliki beberapa bentuk atau perilaku yang berbeda. Dengan kata lain, polimorfisme memungkinkan suatu entitas untuk bersifat banyak bentuk. Sebagai contoh pada Program diatas merupakan implementasi dari class hitung yang di hubungkan gui reservasi. Tampil yang muncul nama pemesan, harga tiket dan jumlah orang. Setelah itu akan muncul di memo sesuai apa yang di inputkan. Pada class ini yang di jadikan private yaitu Harga. Pada bagian ini juga kita buat class baru yaitu kelas data yang berisi Info dan Info di buat interface . Pada gui wisata terjadi masalah yang membuat yang membuat memo tidak keluar.

V.8 Tugas Rumah 3 :

Tema : Sistem Informasi

Judul : Menerapkan Konsep Exception Handling

Source code Object Class(Login):

```
public class Login {
    private String username, password;
    public String nama;
    public Login()
    {
        nama = "Tesa";
        username = "tesalonika";
        password = "12345";
    }
}
```

```
}

public String getUsername()
{
    return username;
}

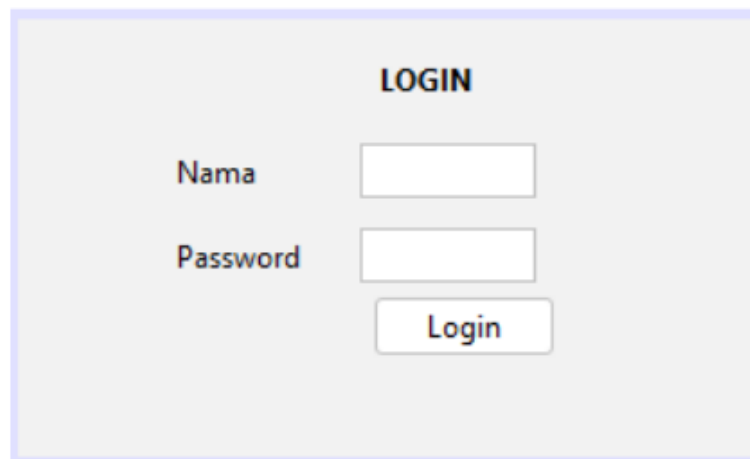
public void setUsername(String username)
{
    this.username = username;
}

public String getPassword()
{
    return password;
}

public void setPassword(String password)
{
    this.password = password;
}

boolean CekLogin(String username, String password)
{
    if (username.equals(getUsername()) &&
        password.equals(getPassword()))
    {
        return true;
    }
    return false;
}
}
```

Desain *form* (GUI_Daftar.java):

The image shows a simple login form titled "LOGIN" in bold black text at the top center. Below the title, there are two labels: "Nama" and "Password", both in bold black text. To the right of each label is a white rectangular input field with a thin black border. Below the "Password" input field, there is a button labeled "Login" in bold black text, also with a thin black border. The entire form is set against a light gray background.

Gambar 5.17 Desain Gui_Daftar.java

Tabel 5.6 Properti Desain GUI_Login.java

No	Objek	Properti	Nilai
1	jLabel1	Text	Login
2	jLabel2	Text	Nama
3	jLabel3	Text	Password
4	jTextField1	Name	txtnama
		Text	“ “
5	jTextField2	Name	txtPass
		Text	“ “
6	jButton1	Name	txtPass g
		Text	Login

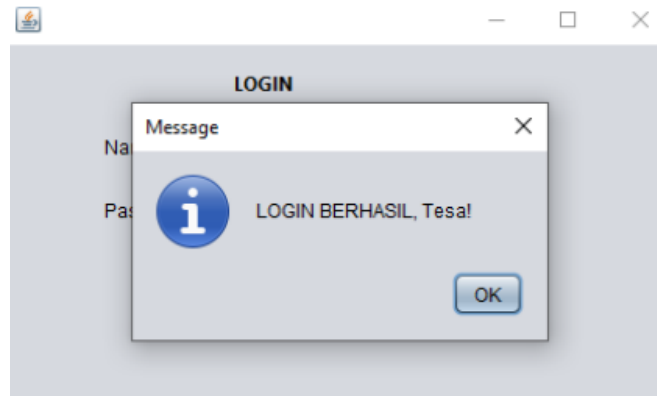
Source code Button Login

```

private void
btnLoginActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    Login l = new Login();
    String username = txtNama.getText();
    String password = txtPass.getText();
    boolean Authenticated = l.CekLogin(username, password);
    if (Authenticated)
    {
        JOptionPane.showMessageDialog(rootPane, "LOGIN BERHASIL,
        "+l.nama + "!");
        Wisata w = new Wisata();
        w.setVisible(true);
        this.dispose();
    }else
    {
        JOptionPane.showMessageDialog(rootPane, "LOGIN GAGAL.
        Silahkan periksa kembali username dan password Anda.");
    }
}

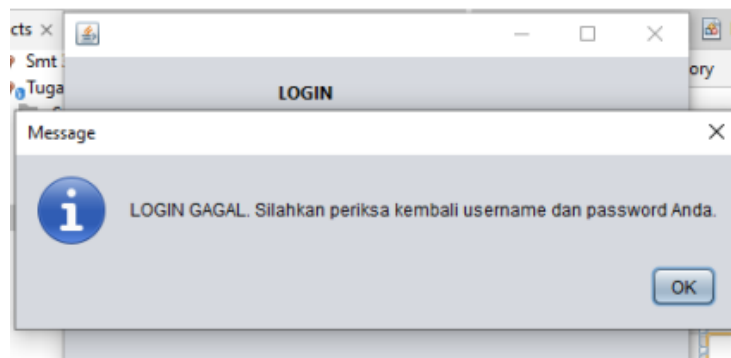
```

Hasil Tampilan:



Gambar 5.18 Hasil Tampilan Login Berhasil.java

Hasil Tampilan:



Gambar 5.19 Hasil Tampilan Login gagal.java

Analisa:

Polimorfisme adalah salah satu konsep dalam pemrograman komputer yang memungkinkan suatu entitas, seperti fungsi, operator, atau objek, untuk dapat memiliki beberapa bentuk atau perilaku yang berbeda. Dengan kata lain, polimorfisme memungkinkan suatu entitas untuk bersifat banyak bentuk. Sebagai contoh pada Program diatas merupakan implementasi dari class hitung yang di hubungkan gui reservasi. Tampil yang muncul nama pemesan, harga tiket dan jumlah orang. Setelah itu akan muncul di memo sesuai apa yang di inputkan. Pada class ini yang di jadikan private yaitu Harga. Pada bagian ini juga kita buat class baru yaitu kelas data yang berisi Total dan Total di buat abstract. Serta ada class baru yaitu data yang menggunakan abstract. Pada gui wisata terjadi masalah yang membuat yang membuat memo tidak keluar.



V.9 Kesimpulan

1. *Abstract Class* adalah sebuah *class* yang tidak bisa di-*instansiasi* (tidak bisa dibuat menjadi objek) dan berperan sebagai ‘kerangka dasar’ bagi class turunannya.
2. Fungsi dari *class abstract* ini adalah untuk mempertahankan *hirarky* dari *parent class* ke kelas turunan dari induknya
3. Polimorfisme memungkinkan suatu entitas (seperti method atau function) untuk berperilaku dengan cara yang berbeda tergantung pada konteks di mana entitas tersebut digunakan “Bentuk” di sini dapat kita artikan: isinya berbeda (overriding), parameternya berbeda dan tipe datanya berbeda (overloading).
4. Polymorphism digunakan untuk mengimplementasi suatu fungsi dari sebuah induk class maupun Interface, baik fungsi yang abstract maupun sudah terdefinisi, untuk diimplementasikan sesuai dengan relevansi suatu class.
5. Interface berfungsi sebagai penghubung antar sesuatu yang ‘*Abstract*’ dengan sesuatu yang nyata.
6. Dalam OOP, sebuah interface dapat dianggap sebagai prototipe atau templat untuk sebuah kelas. Analoginya, jika sebuah class abstrak adalah kerangka dasar untuk kelas-kelas lain, maka interface adalah templat yang memberikan struktur khusus untuk kelas tertentu.

Nama Aslab :	TTD :
Firman Frezy Pradana 2118112	
Tanggal : 17 Desember 2023	



BAB VI

EXCEPTION DAN PENGENALAN DATABASE

Jumlah Pertemuan	:	2 x 60 menit
Tujuan Praktikum	:	<ol style="list-style-type: none">1. Praktikan mampu memahami tentang <i>exception</i> dan menerapkan <i>exception</i> ke dalam program.2. Praktikan mampu membuat <i>database</i> dengan menggunakan MySQL.3. Praktikan mampu mengkoneksikan <i>database</i> dengan program yang sudah dibuat sebelumnya.4. Praktikan mampu memberi akses <i>insert</i>, <i>update</i>, <i>delete</i>, dan <i>searching</i> (memasukkan, merubah, menghapus, dan pencarian).
Alat / bahan	:	<ol style="list-style-type: none">1. Seperangkat <i>computer</i>.2. Perangkat lunak: <i>Netbeans</i>.3. Modul Praktikum <i>OOP</i> 2022.

VI.1 Landasan Teori

A. Interface

1. Pengertian Exception

Exception merupakan kondisi yang tidak diinginkan yang dapat terjadi selama eksekusi program. Exception muncul ketika program menghadapi situasi yang tidak dapat diatasi atau tidak terduga, seperti kesalahan pembagian nol, akses data yang tidak valid, ataupun masalah jaringan. Exception dapat menyebabkan program keluar dari jalur normalnya, menghentikan proses eksekusi, dan memerlukan penanganan khusus agar program dapat berjalan dengan baik. Oleh karenanya diperlukan mekanisme yang membantu menangani *error* tersebut atau kesalahan yang terjadi, baik saat pembuatan maupun implementasi program, mekanisme ini dikenal dengan *Exception Handling*.

Exception Handling merupakan mekanisme yang diperlukan dalam menangani *error* yang terjadi pada saat *runtime* (program

berjalan) atau yang lebih dikenal dengan sebutan *runtime error*. Secara umum, adanya kesalahan / *error* yang terjadi pada program pada saat *runtime* dapat menyebabkan program berhenti atau *hang*. Untuk itulah diperlukan mekanisme untuk memastikan bahwa program tetap dapat berjalan meskipun terdapat kesalahan yang terjadi.

Keuntungan penggunaan *Exception Handling* :

- a) *Exception* bisa membawa sangat banyak informasi tentang kesalahan yang terjadi (deskripsi, stack trace, lokasi baris kode dll)
- b) *Exception* sudah didukung oleh banyak IDE modern yang ada di pasar
- c) Mekanisme *Exception* terintegrasi secara baik dengan OOP

2. Hierarki Kelas Exception pada Java

Exception adalah subkelas dari kelas *java.lang.Throwable*. Karena *Exception* merupakan sebuah kelas, saat program berjalan dan muncul bug atau kesalahan, bug tersebut dapat dianggap sebagai sebuah objek. Selain *Exception*, *java.lang.Throwable* juga memiliki subclass lain yaitu class *Error* dan *Exception*. Tetapi, penting untuk diketahui bahwa *Error* dan *Exception* memiliki perbedaan mendasar. *Error* merujuk pada masalah serius yang umumnya di luar kendali developer dan sebaiknya tidak ditangkap. Di sisi lain, *Exception* adalah representasi dari kesalahan atau pengecualian yang lebih terkendali, sering kali diantisipasi oleh developer untuk diperlakukan atau ditangani sesuai kebutuhan.

Pada dasarnya ada 3 jenis exception menurut Oracle:

- a) *Checked Exception*, adalah exception yang terjadi saat *compile time*. *Compile time* error terjadi apabila *exceptions* ini tidak ditangani menggunakan *block try-catch* atau menggunakan *keyword throws*
- b) *Unchecked Exception*, adalah *exception* yang terjadi saat *execution time*. *Error* ini terjadi dalam lingkup internal dari aplikasi, biasanya terjadi karena salah penulisan kode. Contoh: *NullPointerException*.

- c) *Error*, adalah *exception* yang diluar kendali *user* atau *programmer*. *Error* ini terjadi di lingkup eksternal dari aplikasi. *Ketika exception* ini terjadi, maka tidak ada yang bisa dilakukan untuk mengatasinya. Contoh: ketika perangkat kerasnya rusak saat ingin membaca data.

Dalam mengatasi berbagai macam *Exception*, terdapat 3 *block* kode yaitu *try*, *catch* dan *finally*.

a) *Try-Catch*

Kode yang rawan dengan *exception* kita masukkan ke dalam *block try-catch*. Kode yang dimasukkan dalam *block try-catch* biasa disebut sebagai *protected code*.

Berikut bentuk penulisan *Try-Catch*:

```
try {
    //Protected code
} catch (ExceptionType nama_variabel) {
    //Catch Block
}
```

b) *Multiple catch*

Dengan menggunakan *multiple catch* dapat menangani lebih dari 1 *exception*. Berikut bentuk penulisan *Multiple catch*:

```
try {
    //Protected code
} catch (ExceptionType nama_variabel) {
    //Catch Block
} catch (ExceptionType nama_variabel) {
    //Catch Block }
```

c) *Finally*

Block finally adalah *block* yang di tambahkan di akhir *block try-catch*. *Finally* akan selalu dijalankan setelah *try-catch* baik terjadi *exception* atau tidak. Berikut bentuk penulisan *finally*:

```
try {
    //Protected code
} catch (ExceptionType nama_variabel){
    //Catch Block }
finally {
    //finally Block }
```

B. Pengenalan Database

1. Basis Data (*Database*)

Basis data (*database*) adalah kumpulan data yang disimpan secara sistematis di dalam komputer yang dapat diolah atau dimanipulasi menggunakan perangkat lunak (program aplikasi) untuk

menghasilkan informasi. Pendefinisian basis data meliputi spesifikasi berupa tipe data, struktur data dan juga batasan-batasan pada data yang akan disimpan.

2. XAMPP

Xampp itu kepanjangan dari Apache, PHP, MySQL dan phpMyAdmin. XAMPP merupakan tool yang menyekdiakan paket perangkat lunak ke dalam satu buah paket. XAMPP merupakan perangkat lunak bebas, yang mendukung banyak sistem operasi. Fungsinya adalah sebagai server yang berdiri sendiri (localhost), yang terdiri atas program Apache HTTP Server, MySQL database, dan penerjemah bahasa yang ditulis dengan bahasa pemrograman PHP dan Perl. Sifatnya hampir mirip dengan Web Server yang ada di internet, hanya bedanya Xampp tidak perlu terkoneksi ke internet alias dapat berdiri sendiri atau localhost.

3. Koneksi java ke mysql

MySQL adalah salah satu DBMS (Database Management System) yang menggunakan bahasa SQL (Structured Query Language) untuk mengelola informasi dalam basis data. MySQL memungkinkan kita untuk membuat dan mengelola basis data serta kontennya, termasuk melakukan operasi seperti menambahkan, mengubah, dan menghapus data dalam basis data tersebut.

Dalam pengembangan aplikasi menggunakan bahasa pemrograman Java, kita dapat menghubungkan aplikasi tersebut dengan basis data MySQL. Untuk melakukan koneksi antara Java dan basis data MySQL, kita memerlukan JDBC (Java Database Connectivity) driver. JDBC API (Java Database Connectivity Application Programming Interface) adalah antarmuka pemrograman dalam bahasa Java yang memungkinkan kita berinteraksi dengan informasi dalam basis data.

VI.2 Langkah – Langkah Praktikum

1. Buka Aplikasi *Netbeans*.
2. Buat *Class*.
3. Memberi *script* pada kelas tersebut.
4. Buat *form*.
5. Memberi *script* pada *form* tersebut.
6. Menjalankan program.

VI.3 Tugas Praktikum 1 :

Membuat *Class* KeaktifanMahasiswa, Penilaian dan GUI_Penilaian

Judul : Mendesain ulang GUI_Penilaian + Method Batal

Source code Object Class:

```
abstract public class Penilaian {
    String NIM, nama, kode_mk;
    int NP1, NP2, NP3, NilaiPrak, UTS, UAS;

    public double nilaiProses(){
        return
        ((NP1*0.1)+(NP2*0.2)+(NP3*0.1)+(UTS*0.2)+(NilaiPrak*0.4));
    }

    public double nilaiAkhir(){
        return (nilaiProses()*0.6)+(UAS*0.3);
    }

    public double tampilNA(){
        return nilaiAkhir();
    }

    public int getNP1(){
        return NP1;
    }

    public void setNP1(int NP1){
        this.NP1 = NP1;
    }

    public int getNP2(){
        return NP2;
    }

    public void setNP2(int NP2){
        this.NP2 = NP2;
    }

    public int getNP3(){
        return NP3;
    }

    public void setNP3(int NP3){
        this.NP3 = NP3;
    }

    public int getNilaiPrak(){
        return NilaiPrak;
    }

    public void setNilaiPrak(int NilaiPrak){
        this.NilaiPrak = NilaiPrak;
    }
}
```

```

public int getUTS(){
    return UTS;}

public void setUTS(int UTS){
    this.UTS = UTS;}

public int getUAS(){
    return UAS;}

public void setUAS(int UAS){
    this.UAS = UAS;}

int nilai_keaktifan;

public Penilaian(){
    this.nilai_keaktifan = 0;
}

abstract double nilaiKeaktifan();
}

```

Source code Object Class:

```

public class KeaktifanMahasiswa extends Penilaian {
    int nilai_keaktifan;
    public KeaktifanMahasiswa(){
        this.nilai_keaktifan = 0;
    }
    @Override
    public double nilaiKeaktifan(){
        return ((nilai_keaktifan* 0.1) + nilaiAakhir());
    }
}

```

Desain form (GUI_Penilaian.java):

Gambar 6.1 Desain Gui_Penilaian.java

Tabel 6.1 Properti Desain GUI_Penilaian.java

No	Objek	Properti	Nilai
1	jLabel1	Text	PROGRAM PENILAIAN
2	JLabel2	Text	NP1
3	JLabel3	Text	NP2
4	JLabel4	Text	NP3
5	JLabel5	Text	UTS
6	JLabel6	Text	PRAKTIKUM
7	JLabel7	Text	UAS
8	JLabel8	Text	Nilai Keaktifan
9	JLabel9	Text	Nilai Akhir
10	ComboBox	Name	cmbNim
		Text	“ “
11	ComboBox	Name	cmbKodeMk
		Text	“ “
12	jTextField1	Name	txtNP1
		Text	“ “
13	jTextField2	Name	txtNP2
		Text	“ “
14	jTextField3	Name	txtNP3
		Text	“ “
15	jTextField4	Name	txtUTS
		Text	“ “
16	jTextField5	Name	txtPraktikum
		Text	“ “

17	jTextField5	Name	txtUAS
		Text	“ “
18	jTextField5	Name	txtKeaktifan
		Text	“ “
19	jTextField6	Name	txtNA
		Text	“ “
20	jTextField5	Name	txtCari
		Text	“ “
21	jCheckBox	Name	cmKeaktifan
		Text	“ “
22	jButton1	Name	btnProses
		Text	Proses
23	JButton2	Name	btnSimpan
		Text	Simpan
24	JButton3	Name	btnUbah
		Text	Update
25	JButton4	Name	btnHapus
		Text	Delete
26	JButton5	Name	btnBatal
		Text	Batal
27	JButton6	Name	btnCari

		Text	Cari
28	JButton7	Name	btnNim
		Text	Nim
29	JButton8	Name	btnKdMk
		Text	KD MK
30	JTable	Name	table_data

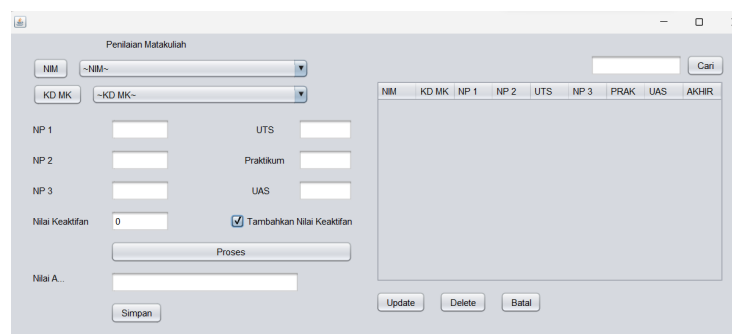
Source code dibawah Method Konstruktork pada GUI:

```
public void batal() {
    txtNP1.setText("");
    txtNP2.setText("");
    txtNP3.setText("");
    txtPraktikum.setText("");
    txtUas.setText("");
    txtUts.setText("");
    txtKeaktifan.setText("");
}
```

Source code Button Batal pada GUI:

```
private void
btnBatalActionPerformed(java.awt.event.ActionEvent
evt) {
    // TODO add your handling code here:
    batal();
}
```

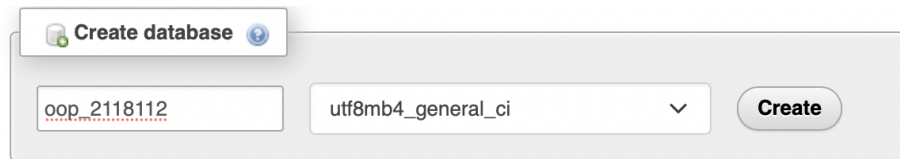
Hasil Tampilan:



Gambar 6.2 Hasil Tampilan GUI_Penilaian.java

VI.4 Tugas Praktikum 2 :

Mengkoneksikan GUI_Mahasiswa Kedalam Database



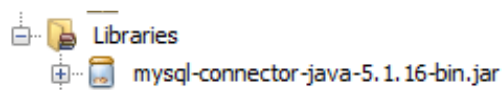
Gambar 6.3 Create database oop_2118112

Structure tb_mahasiswa pada database oop_NIM

#	Nama	Jenis	Penyortiran	Atribut	Tak Ternilai
<input type="checkbox"/> 1	nim	varchar(10)	utf8mb4_general_ci		Tidak
<input type="checkbox"/> 2	nama	varchar(50)	utf8mb4_general_ci		Tidak
<input type="checkbox"/> 3	prodi	varchar(50)	utf8mb4_general_ci		Tidak
<input type="checkbox"/> 4	angkatan	varchar(5)	utf8mb4_general_ci		Tidak

Gambar 6.4 Membuat table(tb_mahasiswa)

Tambahkan library mysql.connector.jar pada library dengan cara klik kanan pada library, kemudian pilih **Add JAR/Folder...**



Gambar 6.5 Menambahkan Library Connection

Source code pada GUI (import Library):

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.swing.JOptionPane;
import javax.swing.table.DefaultTableModel;
```

Source code di bawah Konstruktor GUI_Mahasiswa:

```
public Connection conn;
```

Source code method koneksi :

```
public void koneksi() throws SQLException {
    try {
        conn = null;
        Class.forName("com.mysql.cj.jdbc.Driver");
        conn =
        DriverManager.getConnection("jdbc:mysql://localhost/oop_221
        8005?user=root&password=");
    } catch (ClassNotFoundException ex) {
```

```

Logger.getLogger(GUI_Mahasiswa.class.getName()).log(Level.S
EVERE, null, ex);
    } catch (SQLException e) {

Logger.getLogger(GUI_Mahasiswa.class.getName()).log(Level.S
EVERE, null, e);
    } catch (Exception es) {

Logger.getLogger(GUI_Mahasiswa.class.getName()).log(Level.S
EVERE, null, es);
    }
}

```

Source code di method tampil:

```

public void tampil() {
    DefaultTableModel tabelhead = new
DefaultTableModel();
    tabelhead.addColumn("NIM");
    tabelhead.addColumn("NAMA");
    tabelhead.addColumn("JENIS KELAMIN");
    tabelhead.addColumn("PRODI");
    tabelhead.addColumn("ANGKATAN");
    tabelhead.addColumn("ALAMAT");
    try {
        koneksi();
        String sql = "SELECT * FROM tb_mahasiswa";
        Statement stat = conn.createStatement();
        ResultSet res = stat.executeQuery(sql);
        while (res.next()) {
            tabelhead.addRow(new
Object[]{res.getString(2), res.getString(3),
res.getString(4), res.getString(5), res.getString(6),
res.getString(7),});
        }
        table_data_mahasiswa.setModel(tabelhead);
    } catch (Exception e) {
        JOptionPane.showMessageDialog(null, "BELUM
TERKONEKSI");
    }
}

```

Source code method refresh :

```

public void refresh() {
    new GUI_Mahasiswa().setVisible(true);
    this.setVisible(false);
}

```

Source code method Insert :

```

public void insert() {
    String Nim = txtNim.getText();
    String Nama = txtNama.getText();
    String jk;
    if (radiobtnLaki.isSelected()) {
        jk = "L";
    } else {
        jk = "P";
    }
}

```

```

String Prodi = txtProdi.getText();
String Ang = txtAngkatan.getText();
String alamat = txtAlamat.getText();
try {
    koneksi();
    Statement statement = conn.createStatement();
    statement.executeUpdate("INSERT INTO
tb_mahasiswa (nim, nama,jk, prodi, th_angkatan,alamat)"
        + "VALUES('" + Nim + "','" + Nama +
        "','" + jk + "','" + Prodi + "','" + Ang + "','" + alamat +
        "')");
    statement.close();
    JOptionPane.showMessageDialog(null, "Berhasil
Memasukan Data Mahasiswa!" + "\n" + alamat);
} catch (Exception e) {
    JOptionPane.showMessageDialog(null, "Terjadi
Kesalahan Input!");
}
refresh();
}

```

Source code method delete :

```

public void delete() {
    int ok = JOptionPane.showConfirmDialog(null,
"Apakah Anda yakin akan menghapus data ?", "Konfirmasi",
JOptionPane.YES_NO_OPTION);
    if (ok == 0) {
        try {
            String sql = "DELETE FROM tb_mahasiswa
WHERE nim='" + txtNim.getText() + "'";
            java.sql.PreparedStatement stmt =
conn.prepareStatement(sql);
            stmt.executeUpdate();
            JOptionPane.showMessageDialog(null, "Data
Berhasil di hapus");
            batal();
        } catch (Exception e) {
            JOptionPane.showMessageDialog(null, "Data
gagal di hapus");
        }
    }
    refresh();
}

```

Source code method cari :

```

public void cari() {
    try {
        try ( Statement statement =
conn.createStatement()) {
            String sql = "SELECT * FROM tb_mahasiswa
WHERE `nim` LIKE '%" + txtSearch.getText() + "%'";
            ResultSet rs = statement.executeQuery(sql);
            //menampilkan data dari sql query
            if (rs.next()) // .next() = scanner method
            {
                txtNim.setText(rs.getString(2));
                txtNama.setText(rs.getString(3));
                String jk = rs.getString(4);
                if (jk.equalsIgnoreCase("L")) {

```

```

        radiobtnLaki.setSelected(true);
    } else {
radiobtnPerempuan.setSelected(true);
    }
    txtProdi.setText(rs.getString(4));
    txtAngkatan.setText(rs.getString(5));
    txtAlamat.setText(rs.getString(6));
    } else {
        JOptionPane.showMessageDialog(null,
        "Data yang Anda cari tidak ada");
    }
    }
    } catch (Exception ex) {
        System.out.println("Error." + ex);
    }
}

```

Source code method update :

```

public void update() {
    String Nim = txtNim.getText();
    String Nama = txtNama.getText();
    String jk;
    if (radiobtnLaki.isSelected()) {
        jk = "L";
    } else {
        jk = "P";
    }
    String Prodi = txtProdi.getText();
    String Ang = txtAngkatan.getText();
    String alamat = txtAlamat.getText();
    String Nimlama = nim1;
    try {
        Statement statement = conn.createStatement();
        statement.executeUpdate("UPDATE tb_mahasiswa
SET nim='" + Nim + "'," + "nama='" + Nama + "',"
+ "jk='" + jk + "'" + ",prodi='" +
Prodi + "',alamat='" + alamat + "',th_angkatan='"
+ Ang + "'" WHERE nim = '" + Nimlama +
"'");
        statement.close();
        conn.close();
        JOptionPane.showMessageDialog(null, "Update
Data Mahasiswa Berhasil!");
    } catch (Exception e) {
        System.out.println("Error : " + e);
    }
    refresh();
}

```

Desain form GUI Mahasiswa:

The image shows a Java Swing window titled "Data Mahasiswa". On the left, there are labels and input fields for "Nim", "Nama", "Jenis Kelamin" (with radio buttons for "Laki-laki" and "Perempuan"), "Prodi", "Angkatan", and "Alamat". On the right, there is a search bar with a magnifying glass icon. In the center, there is a table with headers "NIM", "Nama", "JK", "Prodi", "Angkatan", and "Alamat". At the bottom, there are buttons for "Simpan", "Hapus", "Batal", "Close", "Update", and a "Form Penilaian" button.

Gambar 6.6 Desain Form Gui_Mahasiswa

Tabel 6.2 Properti Desain GUI_Mahasiswa (Database)

No	Komponen	Properti	Nilai
1	jLabel1	Text	DATA MAHASISWA
2	jLabel2	Text	NIM
3	jLabel3	Text	Nama
4	jLabel4	Text	Jenis Kelamin
5	jLabel5	Text	Prodi
6	jLabel6	Text	Angkatan
7	jLabel7	Text	Alamat
8	jLabel8	Text	Search
9	jTextField1	Name	txtNim
		Text	
10	jTextField2	Name	txtNama
		Text	
11	jTextField3	Name	txtProdi
		Text	
12	jTextField4	Name	radiobtnLaki
		Text	Laki-laki
13	jTextField5	Name	radiobtnPerempuan
		Text	Perempuan
14	jTextField6	Name	txtAngkatan
		Text	
15	jTextField7	Name	txtAlamat
		Text	

16	jTextField8	Name	txtSearch
		Text	
17	jButton1	Name	btnSimpan
		Text	Simpan
18	jButton2	Name	btnHapus
		Text	Hapus
19	jButton3	Name	btnClose
		Text	Close
20	jButton4	Name	btnBatal
		Text	Batal
21	jButton5	Name	btnPenilaian
		Text	Form Penilaian
22	jTable	Name	table_data_mahasiswa
		Text	

Source code pada tombol tambah:

```
private void
btnSimpanActionPerformed(java.awt.event.ActionEvent evt) {
    insert();
}
```

Source code pada tombol Ubah:

```
private void
btnUbahActionPerformed(java.awt.event.ActionEvent evt) {
    update();
}
```

Source code pada tombol hapus:

```
private void
btnHapusActionPerformed(java.awt.event.ActionEvent evt) {
    delete();
}
```

Source code pada tombol cari:

```
private void
jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
    cari();
}
```

Source code pada tombol batal:

```
private void
```



```

btnBatalActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    clear();
}

```

Tampilan Hasil:

The screenshot shows the 'Data Mahasiswa' window with the following fields and values:

- Nim: S
- Nama: (empty)
- Jenis Kelamin: ☐ Laki-laki, ☐ Perempuan
- Prodi: (empty)
- Angkatan: (empty)
- Alamat: (empty)

The table is empty. Buttons at the bottom include Simpan, Hapus, Batal, Close, Update, and Form Penilaian.

Gambar 6.7 Tampilan Hasil Method tampil() Gui_Mahasiswa

The screenshot shows the 'Data Mahasiswa' window with the following fields and values:

- Nim: (empty)
- Nama: (empty)
- Jenis Kelamin: ☐ Laki-laki, ☐ Perempuan
- Prodi: (empty)
- Angkatan: (empty)
- Alamat: (empty)

The table contains one row of data:

NIM	NAMA	JENIS KEL...	PRODI	ANGKATAN	ALAMAT
2218015	tesalonika	P	Informatika	2022	ntt

Buttons at the bottom include Simpan, Hapus, Batal, Close, Update, and Form Penilaian.

Gambar 6.8 Tampilan Hasil Method tambah() Gui_Mahasiswa

The screenshot shows the 'Data Mahasiswa' window with the following fields and values:

- Nim: 2218088
- Nama: tesalonika
- Jenis Kelamin: ☒ Perempuan
- Prodi: Informatika
- Angkatan: 2022
- Alamat: ntt

The table contains one row of data:

NIM	NAMA	JENIS KEL...	PRODI	ANGKATAN	ALAMAT
2218088	tesalonika	P	informatika	2022	ntt

Buttons at the bottom include Simpan, Hapus, Batal, Close, Update, and Form Penilaian.

Gambar 6.9 Tampilan Hasil Method ubah() Gui_Mahasiswa

The screenshot shows the 'Data Mahasiswa' window with the following fields and values:

- Nim: (empty)
- Nama: (empty)
- Jenis Kelamin: ☐ Laki-laki, ☐ Perempuan
- Prodi: (empty)
- Angkatan: (empty)
- Alamat: (empty)

The table contains one row of data:

NIM	NAMA	JENIS KEL...	PRODI	ANGKATAN	ALAMAT
2277383	Nasya	P	hukum	2023	ntt

Buttons at the bottom include Simpan, Hapus, Batal, Close, Update, and Form Penilaian.

Gambar 6.10 Tampilan Hasil Method hapus() Gui_Mahasiswa

Gambar 6.11 Tampilan Hasil Method batal() Gui_Mahasiswa

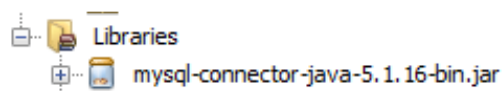
VI.5 Tugas Praktikum 3 :

Mengkoneksikan GUI_Nilai Kedalam Database

#	Nama	Jenis	Penyortiran	Atribut	Tak Ternilai	Bawaan	Komentar	Ekstra	Tindakan
<input type="checkbox"/> 1	id	int(11)			Tidak	Tidak ada		AUTO_INCREMENT	Ubah Hapus Lainnya
<input type="checkbox"/> 2	Nim	varchar(7)	utf8mb4_general_ci		Tidak	Tidak ada			Ubah Hapus Lainnya
<input type="checkbox"/> 3	kd_mk	varchar(6)	utf8mb4_general_ci		Tidak	Tidak ada			Ubah Hapus Lainnya
<input type="checkbox"/> 4	NP1	int(3)			Tidak	Tidak ada			Ubah Hapus Lainnya
<input type="checkbox"/> 5	NP2	int(3)			Tidak	Tidak ada			Ubah Hapus Lainnya
<input type="checkbox"/> 6	UTS	int(3)			Tidak	Tidak ada			Ubah Hapus Lainnya
<input type="checkbox"/> 7	NP3	int(3)			Tidak	Tidak ada			Ubah Hapus Lainnya
<input type="checkbox"/> 8	prak	int(3)			Tidak	Tidak ada			Ubah Hapus Lainnya
<input type="checkbox"/> 9	UAS	int(3)			Tidak	Tidak ada			Ubah Hapus Lainnya
<input type="checkbox"/> 10	NA	int(3)			Tidak	Tidak ada			Ubah Hapus Lainnya

Gambar 6.12 Membuat table (tb_nilai)

Tambahkan library mysql.connector.jar pada library dengan cara klik kanan pada library, kemudian pilih **Add JAR/Folder...**



Gambar 6.13 Menambahkan Library Connection

Source code pada GUI (import Library):

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.swing.JOptionPane;
import javax.swing.table.DefaultTableModel;
```

Source code di bawah Konstruktor GUI_Nilai:

```
public Connection conn;
```

Source code method koneksi :

```
public void koneksi() throws SQLException {
    try {
        conn = null;
        Class.forName("com.mysql.cj.jdbc.Driver");
        conn =
        DriverManager.getConnection("jdbc:mysql://localhost/oop_221
```

```

8005?user=root&password=");
        } catch (ClassNotFoundException ex) {

Logger.getLogger(GUI_Penilaian.class.getName()).log(Level.S
EVERE, null, ex);
        } catch (SQLException e) {

Logger.getLogger(GUI_Penilaian.class.getName()).log(Level.S
EVERE, null, e);
        } catch (Exception es) {

Logger.getLogger(GUI_Penilaian.class.getName()).log(Level.S
EVERE, null, es);
        }
    }
}

```

Source code di method tampil:

```

public void tampil() {
    DefaultTableModel      tabelhead      =      new
DefaultTableModel();
    tabelhead.addColumn("NIM");
    tabelhead.addColumn("Kode MK");
    tabelhead.addColumn("NP1");
    tabelhead.addColumn("NP2");
    tabelhead.addColumn("UTS");
    tabelhead.addColumn("NP3");
    tabelhead.addColumn("PRAK");
    tabelhead.addColumn("UAS");
    tabelhead.addColumn("NA");
    try {
        koneksi();
        String sql = "SELECT * FROM tb_nilai";
        Statement stat = conn.createStatement();
        ResultSet res = stat.executeQuery(sql);
        while (res.next()) {
            tabelhead.addRow(new
Object[]{res.getString(2),      res.getString(3),
res.getString(4),      res.getString(5),      res.getString(6),
res.getString(7),      res.getString(8),      res.getString(9),
res.getString(10),});
        }
        jTable3.setModel(tabelhead);
    } catch (Exception e) {
        JOptionPane.showMessageDialog(null,      "BELUM
TERKONEKSI");
    }
}

```

Source code method refresh :

```

public void refresh() {
    new GUI_Penilaian().setVisible(true);
    this.setVisible(false);
}

```

Source code method insert :

```

public void insert() {
    String Nim = (String) jComboBox1.getSelectedItem();
    String      KodeMK      =      (String)
jComboBox2.getSelectedItem();
    String NP1 = jTextField4.getText();
}

```

```

String NP2 = jTextField5.getText();
String UTS = jTextField7.getText();
String NP3 = jTextField6.getText();
String PRAK = jTextField8.getText();
String UAS = jTextField9.getText();
String NA = jTextField11.getText();
try {
    koneksi();
    Statement statement = conn.createStatement();
    statement.executeUpdate("INSERT INTO
tb_nilai(Nim, kd_mk, NP1, NP2, UTS, NP3, prak, UAS, NA) "
+ "VALUES('" + Nim + "','" + KodeMK +
"','" + NP1 + "','" + NP2 + "','" + UTS + "','" + NP3 +
"','"
+ "'" + PRAK + "','" + UAS + "','" + NA
+ "')");
    statement.close();
    JOptionPane.showMessageDialog(null, "Berhasil
Memasukan Data Nilai!");
} catch (Exception e) {
    JOptionPane.showMessageDialog(null, "Terjadi
Kesalahan Input!");
}
refresh();
}

```

Source code method update :

```

public void update() {
    String Nim = (String) jComboBox1.getSelectedItem();
    String KodeMK = (String)
jComboBox2.getSelectedItem();
    String NP1 = jTextField4.getText();
    String NP2 = jTextField5.getText();
    String UTS = jTextField7.getText();
    String NP3 = jTextField6.getText();
    String PRAK = jTextField8.getText();
    String UAS = jTextField9.getText();
    String NA = jTextField11.getText();

    String nim_lama = nim1;
    String kode_lama = kd_mk1;

    try {
        Statement statement = conn.createStatement();
        statement.executeUpdate("UPDATE tb_nilai SET
Nim='" + Nim + "','" + "kd_mk='" + KodeMK + "'"
+ ",NP1='" + NP1 + "','" + NP2='" + NP2 +
"','UTS='" + UTS + "','" + NP3='" + NP3 + "','" + prak='" + PRAK +
"','UAS='" + UAS + "','" + NA='" + NA + "'" WHERE Nim ='" +
nim_lama + "'" AND kd_mk='" + kode_lama + "'");
        statement.close();
        conn.close();
        JOptionPane.showMessageDialog(null, "Update
Data Nilai!");
    } catch (Exception e) {
        System.out.println("Error : " + e);
    }
    refresh();
}

```

Source code method delete :

```

public void delete() {
    int ok = JOptionPane.showConfirmDialog(null,
    "Apakah Anda yakin akan menghapus data ?", "Konfirmasi",
    JOptionPane.YES_NO_OPTION);
    if (ok == 0) {
        try {
            String sql = "DELETE FROM tb_nilai WHERE
Nim='" + jComboBox1.getSelectedItem() + "' AND kd_mk='" +
jComboBox2.getSelectedItem() + "'";
            PreparedStatement stmt =
conn.prepareStatement(sql);
            stmt.executeUpdate();
            JOptionPane.showMessageDialog(null, "Data
Berhasil di hapus");
            batal();
        } catch (Exception e) {
            JOptionPane.showMessageDialog(null, "Data
gagal di hapus");
        }
    }
    refresh();
}

```

Source code method cari :

```

public void cari() {
    try {
        try (Statement statement =
conn.createStatement()) {
            String sql = "SELECT * FROM tb_nilai WHERE
`Nim` LIKE '%" + jTextField12.getText() + "%'";
            ResultSet rs = statement.executeQuery(sql);
            //menampilkan data dari sql query
            if (rs.next()) // .next() = scanner method
            {

jComboBox1.setSelectedItem(rs.getString(2));

jComboBox2.setSelectedItem(rs.getString(3));
                jTextField4.setText(rs.getString(4));
                jTextField5.setText(rs.getString(5));
                jTextField7.setText(rs.getString(6));
                jTextField6.setText(rs.getString(7));
                jTextField8.setText(rs.getString(8));
                jTextField9.setText(rs.getString(9));
                jTextField11.setText(rs.getString(10));

            } else {
                JOptionPane.showMessageDialog(null,
                "Data yang Anda cari tidak ada");
            }
        }
    } catch (Exception ex) {
        System.out.println("Error." + ex);
    }
}

```

Desain *form* GUI Nilai:

Gambar 6.14 Desain Form Gui_Nilai

Tabel 6.3 Properti Desain GUI_Nilai(Database)

No	Objek	Properti	Nilai
1	jLabel1	Text	Penilaian Matakuliah
2	jLabel2	Text	Nilai Akhir
3	jLabel3	Text	NP1
4	jLabel4	Text	NP2
5	jLabel5	Text	NP3
6	jLabel6	Text	UTS
7	jLabel7	Text	Praktikum
8	jLabel8	Text	UAS
9	jLabel9	Text	Nilai Keaktifan
10	jTextField4	Name	jTextField4
		Text	“ “
11	jTextField5	Name	jTextField5
		Text	“ “
12	jTextField6	Name	jTextField6
		Text	“ “
13	jTextField7	Name	jTextField7
		Text	“ “
14	jTextField8	Name	jTextField8
		Text	“ “
15	jTextField9	Name	jTextField9
		Text	“ “
16	jTextField10	Name	jTextField10

		Text	“ “
17	jTextField11	Name	jTextField11
		Text	“ “
18	jButton1	Name	jButton1
		Text	Proses
19	jButton4	Name	jButton4
		Text	“ “
20	jButton5	Name	jButton5
		Text	Update
21	jButton6	Name	jButton6
		Text	Delete
22	jButton7	Name	jButton7
		Text	Batal
23	jButton9	Name	jButton9
		Text	Ubah
24	jButton8	Name	Ubah
		Text	jButton8
25	jTextField12	Name	jTextField12
		Text	“ “
26	jTable3	Name	jTable3
		Text	“ “
27	jComboBox1	Name	jComboBox1
		Text	~NIM~
28	jComboBox2	Name	jComboBox2
		Text	~KD MK~
29	jButton2	Name	jButton2
		Text	NIM
30	jButton3	Name	jButton3
		Text	KD MK
31	JCheckBox	Text	Tambahkan Nilai Keaktifan

Source code pada tombol tambah:

```
private void
jButton4ActionPerformed(java.awt.event.ActionEvent evt) {
    insert();
}
```

Source code pada tombol Ubah:

```
private void
jButton9ActionPerformed(java.awt.event.ActionEvent evt) {
    update();
}
```

Source code pada tombol hapus:

```
private void
jButton6ActionPerformed(java.awt.event.ActionEvent evt) {
    delete();
}
```

Source code pada tombol cari:

```
private void
jButton8ActionPerformed(java.awt.event.ActionEvent evt) {
    cari();
}
```

Source code pada tombol batal:

```
private void
jButton7ActionPerformed(java.awt.event.ActionEvent evt) {
    batal();
}
```

Tampilan Hasil:

NIM	Kode	NP1	NP2	UTS	NP3	PRAK	UAS	NA
22180...	IF2222	100	100	100	100	1000	100	100

Gambar 6.15 Tampilan Hasil Method tampil() Gui_Mahasiswa

NIM	Kode	NP1	NP2	UTS	NP3	PRAK	UAS	NA
22180...	IF2222	100	100	100	100	1000	100	100
22180...	IF2222	99	99	99	99	99	99	108

Gambar 6.16 Tampilan Hasil Method tambah() Gui_Mahasiswa

NIM	Kode	NP1	NP2	UTS	NP3	PRAK	UAS	NA
22180...	IF2222	100	100	100	100	1000	100	100
22180...	IF2222	99	99	99	99	99	99	103

Gambar 6.17 Tampilan Hasil Method ubah() Gui_Mahasiswa

NIM	Kode	NP1	NP2	UTS	NP3	PRAK	UAS	NA
22180...	IF2222	100	100	100	100	1000	100	100

Gambar 6.18 Tampilan Hasil Method hapus() Gui_Mahasiswa

NIM	Kode	NP1	NP2	UTS	NP3	PRAK	UAS	NA
-----	------	-----	-----	-----	-----	------	-----	----

Gambar 6.19 Tampilan Hasil Method batal() Gui_Mahasiswa

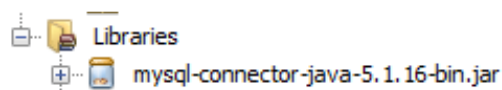
VI.6 Tugas Praktikum 4 :

Mengkoneksikan GUI_Matkul Kedalam Database

#	Nama	Jenis	Penyortiran	Atribut	Tak Ternilai	Bawaan	Komentar	Ekstra	Tindakan
1	id	int(11)			Tidak	Tidak ada		AUTO_INCREMENT	Ubah Hapus Lainnya
2	kode_mk	varchar(10)	utf8mb4_general_ci		Tidak	Tidak ada			Ubah Hapus Lainnya
3	matakuliah	varchar(100)	utf8mb4_general_ci		Tidak	Tidak ada			Ubah Hapus Lainnya
4	dosenpengajar	varchar(100)	utf8mb4_general_ci		Tidak	Tidak ada			Ubah Hapus Lainnya
5	jlmsks	int(1)			Tidak	Tidak ada			Ubah Hapus Lainnya

Gambar 6.20 Membuat table (tb_matkul)

Tambahkan library mysql.connector.jar pada library dengan cara klik kanan pada library, kemudian pilih **Add JAR/Folder...**



Gambar 6.21 Menambahkan Library Connection

Source code pada GUI (import Library):

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.swing.JOptionPane;
import javax.swing.table.DefaultTableModel;
```

Source code di bawah Konstruktor GUI_Matkul:

```
public Connection conn;
```

Source code method koneksi :

```
public void koneksi() throws SQLException {
    try {
        conn = null;
        Class.forName("com.mysql.cj.jdbc.Driver");
        conn =
        DriverManager.getConnection("jdbc:mysql://localhost/oop_221
        8005?user=root&password=");
    } catch (ClassNotFoundException ex) {

        Logger.getLogger(GUI_Matkul.class.getName()).log(Level.SEVERE,
        null, ex);
    } catch (SQLException e) {

        Logger.getLogger(GUI_Matkul.class.getName()).log(Level.SEVERE,
        null, e);
    } catch (Exception es) {

        Logger.getLogger(GUI_Matkul.class.getName()).log(Level.SEVERE,
        null, es);
    }
}
```

Source code di method tampil:

```
public void tampil() {
    DefaultTableModel tabelhead = new
    DefaultTableModel();
    tabelhead.addColumn("KODE MK");
    tabelhead.addColumn("NAMA MK");
    tabelhead.addColumn("DOSEN");
    tabelhead.addColumn("JML SKS");
    try {
        koneksi();
        String sql = "SELECT * FROM tb_matkul";
        Statement stat = conn.createStatement();
        ResultSet res = stat.executeQuery(sql);
        while (res.next()) {
            tabelhead.addRow(new
            Object[]{res.getString(2), res.getString(3),
            res.getString(4), res.getString(5),});
        }
        jTable1.setModel(tabelhead);
    } catch (Exception e) {
        JOptionPane.showMessageDialog(null, "BELUM
        TERKONEKSI");
    }
}
```

```

    }
}

```

Source code method refresh :

```

public void refresh() {
    new GUI_Matkul().setVisible(true);
    this.setVisible(false);
}

```

Source code method insert :

```

public void insert() {
    String Kode = jTextField1.getText();
    String MK = jTextField2.getText();
    String Dosen = jTextField3.getText();
    String jmlsks = jTextField4.getText();
    try {
        koneksi();
        Statement statement = conn.createStatement();
        statement.executeUpdate("INSERT INTO
tb_matkul(kode_mk, matakuliah, dosenpengajar,jmlsks)"
+ "VALUES('" + Kode + "','" + MK +
"','" + Dosen + "','" + jmlsks + "')");
        statement.close();
        JOptionPane.showMessageDialog(null, "Berhasil
Memasukan Data Matakuliah!");
    } catch (Exception e) {
        JOptionPane.showMessageDialog(null, "Terjadi
Kesalahan Input!");
    }
    refresh();
}

```

Source code method update:

```

public void update() {
    String Kode = jTextField1.getText();
    String MK = jTextField2.getText();
    String Dosen = jTextField3.getText();
    String jmlsks = jTextField4.getText();
    String KdMkLama = kode1;
    try {
        Statement statement = conn.createStatement();
        statement.executeUpdate("UPDATE tb_matkul SET
kode_mk='" + Kode + "','" + "matakuliah='" + MK + "','"
+ "dosenpengajar='" + Dosen + "'" +
",jmlsks='" + jmlsks + "'WHERE kode_mk = '" + KdMkLama +
"'");

        statement.close();
        conn.close();
        JOptionPane.showMessageDialog(null, "Update
Data MataKuliah!");
    } catch (Exception e) {
        System.out.println("Error : " + e);
    }
    refresh();
}

```

Source code method hapus :

```

public void delete() {
    int ok = JOptionPane.showConfirmDialog(null,
    "Apakah Anda yakin akan menghapus data ?", "Konfirmasi",
    JOptionPane.YES_NO_OPTION);
    if (ok == 0) {
        try {
            String sql = "DELETE FROM tb_matkul WHERE
kode_mk='" + jTextField1.getText() + "'";
            java.sql.PreparedStatement stmt =
conn.prepareStatement(sql);
            stmt.executeUpdate();
            JOptionPane.showMessageDialog(null, "Data
Berhasil di hapus");
            batal();
        } catch (Exception e) {
            JOptionPane.showMessageDialog(null, "Data
gagal di hapus");
        }
        refresh();
    }
}

```

Source code method cari :

```

public void cari() {
    try {
        try (Statement statement =
conn.createStatement()) {
            String sql = "SELECT * FROM tb_matkul WHERE
`kode_mk` LIKE '%" + jTextField1.getText() + "%'";
            ResultSet rs = statement.executeQuery(sql);
            //menampilkan data dari sql query
            if (rs.next()) // .next() = scanner method
            {
                jTextField1.setText(rs.getString(2));
                jTextField2.setText(rs.getString(3));
                jTextField3.setText(rs.getString(4));
                jTextField4.setText(rs.getString(5));
            } else {
                JOptionPane.showMessageDialog(null,
                "Data yang Anda cari tidak ada");
            }
        }
    } catch (Exception ex) {
        System.out.println("Error." + ex);
    }
}

```

Desain *form* GUI Matkul:

Gambar 6.22 Desain Form Gui_Matkul

Tabel 6.4 Properti Desain GUI_Matkul (Database)

No	Objek	Properti	Nilai
1	jLabel1	Text	DATA MATAKULIAH
2	jLabel2	Text	Kode Matakuliah
3	jLabel3	Text	Matakuliah
4	jLabel4	Text	Dosen Pengajar
5	jLabel5	Text	Nomor Anggota
6	jTextField1	Name	jTextField1
		Text	“ “
7	jTextField2	Name	jTextField2
		Text	“ “
8	jTextField3	Name	jTextField3
		Text	“ “
9	jTextField4	Name	jTextField4
		Text	“ “
10	jButton1	Name	jButton1
		Text	“ “
11	jTable1	Name	jTable1
		Text	“ “
12	jButton1	Name	jButton1
		Text	Simpan
13	jButton2	Name	jButton2
		Text	Hapus
14	jButton3	Name	jButton3
		Text	Batal

	jButton4	Name	jButton4
		Text	Ubah
	jButton5	Name	jButton5
		Text	Form Nilai
	jTextField5	Name	jTextField5
		Text	“ “
	jButton6	Name	jButton6
		Text	Cari

Source code pada tombol tambah:

```
private void
jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
    insert();
}
```

Source code pada tombol Ubah:

```
private void
jButton4ActionPerformed(java.awt.event.ActionEvent evt) {
    update();
}
```

Source code pada tombol hapus:

```
private void
jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
    delete();
}
```

Source code pada tombol cari:

```
private void
jButton6ActionPerformed(java.awt.event.ActionEvent evt) {
    cari();
}
```

Source code pada tombol batal:

```
private void
jButton3ActionPerformed(java.awt.event.ActionEvent evt) {
    batal();
}
```

Tampilan Hasil:

The screenshot shows a Java Swing application window titled "DATA MATAKULIAH". The window has a light gray background. At the top, there is a search bar with a "Cari" button. Below the search bar, there are four input fields labeled "Kode Matakuliah", "Mata Kuliah", "Dosen Pengajar", and "Jumlah SKS", each with a "Simpan" button. To the right of these input fields is a table with the following data:

KODE MIK	NAMA MIK	DOSEN	JML SKS
IF2222	OOP	Pa Yoa	4

At the bottom of the window, there are three buttons: "Ubah", "Hapus", and "Batal". Below these buttons is a "Form Nilai" button.

Gambar 6.23 Tampilan Hasil Method tampil() Gui_Mahasiswa

DATA MATAKULIAH

Kode Matakuliah:

Mata Kuliah:

Dosen Pengajar:

Jumlah SKS:

Simpan

KODE MK	NAMA MK	DOSEN	JML SKS
IF2222	OOP	Pa Yoa	4
IF2133	PCD	xxxxxxx	2

Ubah Hapus Batal

Form Nilai

Gambar 6.24 Tampilan Hasil Method tambah() Gui_Mahasiswa

DATA MATAKULIAH

Kode Matakuliah:

Mata Kuliah:

Dosen Pengajar:

Jumlah SKS:

Simpan

KODE MK	NAMA MK	DOSEN	JML SKS
IF2222	OOP	Pa Yoa	4
IF2133	PCD	tester	2

Ubah Hapus Batal

Form Nilai

Gambar 6.25 Tampilan Hasil Method ubah() Gui_Mahasiswa

DATA MATAKULIAH

Kode Matakuliah:

Mata Kuliah:

Dosen Pengajar:

Jumlah SKS:

Simpan

KODE MK	NAMA MK	DOSEN	JML SKS
IF2222	OOP	Pa Yoa	4

Ubah Hapus Batal

Form Nilai

Gambar 6.26 Tampilan Hasil Method hapus() Gui_Mahasiswa

DATA MATAKULIAH

Kode Matakuliah:

Mata Kuliah:

Dosen Pengajar:

Jumlah SKS:

Simpan

KODE MK	NAMA MK	DOSEN	JML SKS
IF2222	OOP	Pa Yoa	4

Ubah Hapus Batal

Form Nilai

Gambar 6.27 Tampilan Hasil Method batal() Gui_Mahasiswa

VI.7 Tugas Rumah 1 :

Menerapkan *Exception* pada class Komputer

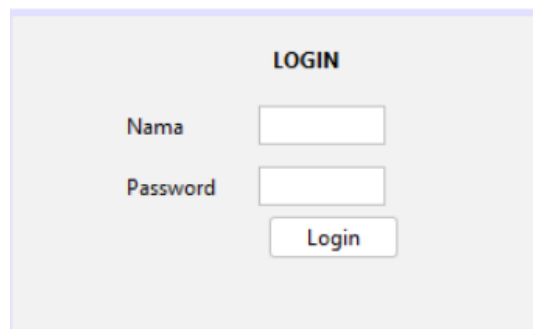
Source code pada *Super Class* Login.Java:

```
public class Login {
    private String username, password;
    public String nama;
    public Login()
    {
        nama = "Tesa";
        username = "tesalonika";
        password = "12345";
    }

    public String getUsername()
    {
        return username;
    }
    public void setUsername(String username)
    {
        this.username = username;
    }
    public String getPassword()
    {
        return password;
    }
    public void setPassword(String password)
    {
        this.password = password;
    }

    boolean CekLogin(String username, String password)
    {
        if (username.equals(getUsername()) &&
            password.equals(getPassword()))
        {
            return true;
        }
        return false;
    }
}
```

Desain form (Daftar.javva):



The image shows a simple login form with a light gray background. At the top center is the title 'LOGIN'. Below it, there are two labels: 'Nama' and 'Password'. Each label is followed by a white rectangular input field. Below the 'Password' field is a white rectangular button with the text 'Login' in the center.

Gambar 6.28 Tampilan Desain Daftar.java

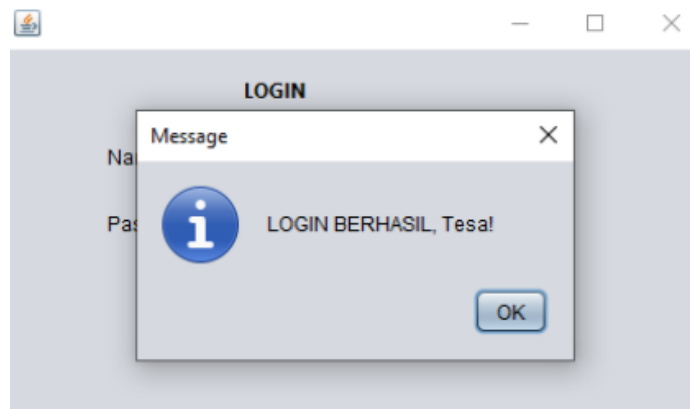
Tabel 6.5 Properti Desain GUI_ Matkul (Database)

No	Objek	Properti	Nilai
1	jLabel1	Text	Login
2	jLabel2	Text	Nama
3	jLabel3	Text	Password
4	jTextField1	Name	txtnama
		Text	“ “
5	jTextField2	Name	txtPass
		Text	“ “
6	jButton1	Name	txtPass
		Text	Login

Source code Button/combobox btnLogin :

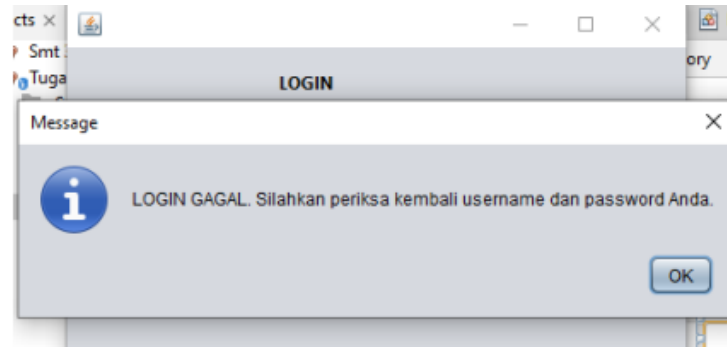
```
private void
btnLoginActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    Login l = new Login();
    String username = txtNama.getText();
    String password = txtPass.getText();
    boolean Authenticated = l.CekLogin(username, password);
    if (Authenticated)
    {
        JOptionPane.showMessageDialog(rootPane, "LOGIN BERHASIL,
        "+l.nama + "!");
        Wisata w = new Wisata();
        w.setVisible(true);
        this.dispose();
    }else
    {
        JOptionPane.showMessageDialog(rootPane, "LOGIN GAGAL.
        Silahkan periksa kembali username dan password Anda.");
    }
}
```

Tampilan Hasil:



Gambar 6.29 Tampilan Login Berhasil.java

Tampilan Hasil:



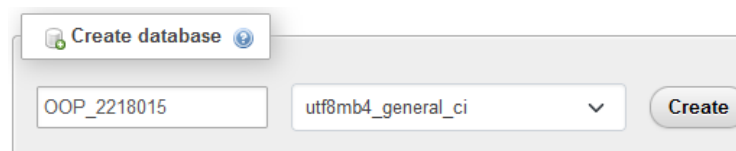
Gambar 6.30 Tampilan Login Gagal.java

Analisa:

Program diatas merupakan program yang mengimplentasikan sebuah exception handling, yang dimana implementasi pada class Login. Jika tidak menginputkan name dan password dengan benar/tidak, maka program akan menampilkan "LOGIN BERHASIL". Dan jika mengisi name dan password dengan benar maka program akan menampilkan "LOGIN GAGAL" sama seperti gambar diatas.

VI.8 Tugas Rumah 2 :

Menerapkan CRUD Pada GUI_Wisata.java



Gambar 6.31 Membuat database oop_2218015

Structure *tb_list* pada database oop_2218015

Table structure										
#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action	
1	Nama	varchar(255)	utf8mb4_general_ci		No	None			Change	Drop More
2	Kota	varchar(255)	utf8mb4_general_ci		No	None			Change	Drop More
3	Deskripsi	varchar(255)	utf8mb4_general_ci		No	None			Change	Drop More

Gambar 6.32 Membuat table(tb_list)

Source code pada GUI:

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.logging.Level;
```

```
import java.util.logging.Logger;
import javax.swing.JOptionPane;
import javax.swing.table.DefaultTableModel;
```

Source code di bawah Konstruktor GUI_Wisata:

```
private Connection connection;
```

Source code method koneksi :

```
private void connectToDatabase() {
    try {
        // Load the JDBC driver
        Class.forName("com.mysql.cj.jdbc.Driver");

        // Establish the connection
        connection =
        DriverManager.getConnection(JDBC_URL, JDBC_USER,
        JDBC_PASSWORD);
    } catch (ClassNotFoundException | SQLException e)
    {
        e.printStackTrace();
        JOptionPane.showMessageDialog(this, "Error
        connecting to the database: " + e.getMessage(), "Error",
        JOptionPane.ERROR_MESSAGE);
        System.exit(1);
    }
}
```

Source code di method tampil:

```
private void loadTableData() {
    try {
        String sql = "SELECT * FROM list";
        PreparedStatement preparedStatement =
        connection.prepareStatement(sql);
        ResultSet resultSet =
        preparedStatement.executeQuery();

        DefaultTableModel model = (DefaultTableModel)
        tabel.getModel();
        model.setRowCount(0);

        while (resultSet.next()) {
            String nama = resultSet.getString("nama");
            String kota = resultSet.getString("kota");
            String deskripsi =
            resultSet.getString("deskripsi");

            model.addRow(new Object[]{nama, kota,
            deskripsi});
        }

        resultSet.close();
        preparedStatement.close();
    } catch (SQLException e) {
        e.printStackTrace();
    }
}
```

```

        JOptionPane.showMessageDialog(this, "Error
loading data from the database: " + e.getMessage(),
"Error", JOptionPane.ERROR_MESSAGE);
    }
}

```

Source code method tambah :

```

preparedStatement.setString(1, txtNama.getText());
preparedStatement.setString(2, txtKota.getText());
preparedStatement.setString(3, txtDes.getText());

```

Source code method ubah :

```

String nama = txtNama.getText();
String kota = txtKota.getText();
String deskripsi = txtDes.getText();

```

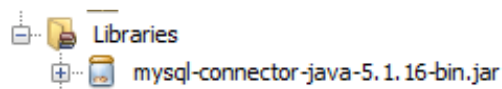
Source code method hapus :

```

String sql = "DELETE FROM list WHERE nama=?";
PreparedStatement preparedStatement =
connection.prepareStatement(sql);

```

Tambahkan library mysql.connector.jar pada library dengan cara klik kanan pada library, kemudian pilih **Add JAR/Folder...**



Gambar 6.33 Menambahkan *Library Connection*

Desain *form* GUI_Wisata:

The image shows a Java Swing window titled 'WISATA'. On the left, there are three text input fields labeled 'Nama', 'Kota', and 'Deskripsi'. To the right of these fields is a table with three columns: 'Nama', 'Kota', and 'Deskripsi'. Below the table are four buttons: 'Insert', 'Update', 'Delete', and 'Close'.

Gambar 6.34 Desain Form GUI_Wisata

Tabel 6.6 Properti Desain GUI_ .Wisata (Database)

No	Objek	Properti	Nilai
1	jLabel	Text	Wisata
2	jLabel2	Text	Nama
3	jLabel3	Text	Kota
4	jLabel4	Text	Deskripsi
5	jTextField1	Name	txtName
		Text	“ “
6	jTextField2	Name	txtKota
		Text	“ “
7	jTextField3	Name	txtDes
		Text	“ “
8	jButton1	Name	btnInsert
		Text	“ “
	jButton2	Name	btnupdate
		Text	“ “
	jButton3	Name	btndelete
		Text	“ “
	jButton4	Name	btnclose
		Text	“ “
	jScrollPane2	Name	jScrollPane2
		Text	“ “

Source code pada tombol Tambah:

```
private void
btninsertActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    try {
        String sql = "INSERT INTO list (nama, kota,
deskripsi) VALUES (?, ?, ?)";
        PreparedStatement preparedStatement =
connection.prepareStatement(sql);

        preparedStatement.setString(1, txtNama.getText());
        preparedStatement.setString(2, txtKota.getText());
        preparedStatement.setString(3, txtDes.getText());

        preparedStatement.executeUpdate();
        preparedStatement.close();

        JOptionPane.showMessageDialog(this, "Data inserted
successfully!");
        loadTableData();
    } catch (SQLException e) {
        e.printStackTrace();
    }
}
```

```

        JOptionPane.showMessageDialog(this, "Error
inserting data into the database: " + e.getMessage(),
"Error", JOptionPane.ERROR_MESSAGE);
    }
}

```

Source code pada tombol Ubah:

```

private void
btnupdateActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    int selectedRow = tabel.getSelectedRow();
    if (selectedRow == -1) {
        JOptionPane.showMessageDialog(this, "Please select
a row to update.", "Warning", JOptionPane.WARNING_MESSAGE);
        return;
    }

    try {
        String nama = txtNama.getText();
        String kota = txtKota.getText();
        String deskripsi = txtDes.getText();

        String sql = "UPDATE list SET nama=?, kota=?,
deskripsi=? WHERE nama=?";
        PreparedStatement preparedStatement =
connection.prepareStatement(sql);

        preparedStatement.setString(1, nama);
        preparedStatement.setString(2, kota);
        preparedStatement.setString(3, deskripsi);
        preparedStatement.setString(4,
tabel.getValueAt(selectedRow, 0).toString()); // Assuming
the first column is 'nama'

        preparedStatement.executeUpdate();
        preparedStatement.close();

        JOptionPane.showMessageDialog(this, "Data updated
successfully!");
        loadTableData();
    } catch (SQLException e) {
        e.printStackTrace();
        JOptionPane.showMessageDialog(this, "Error updating
data in the database: " + e.getMessage(), "Error",
JOptionPane.ERROR_MESSAGE);
    }
}

```

Source code pada tombol Hapus:

```

private void
btndeleteActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    int selectedRow = tabel.getSelectedRow();
    if (selectedRow == -1) {
        JOptionPane.showMessageDialog(this, "Please select
a row to delete.", "Warning", JOptionPane.WARNING_MESSAGE);
    }
}

```

```

        return;
    }

    try {
        String nama = tabel.getValueAt(selectedRow,
0).toString(); // Assuming the first column is 'nama'

        String sql = "DELETE FROM list WHERE nama=?";
        PreparedStatement preparedStatement =
connection.prepareStatement(sql);

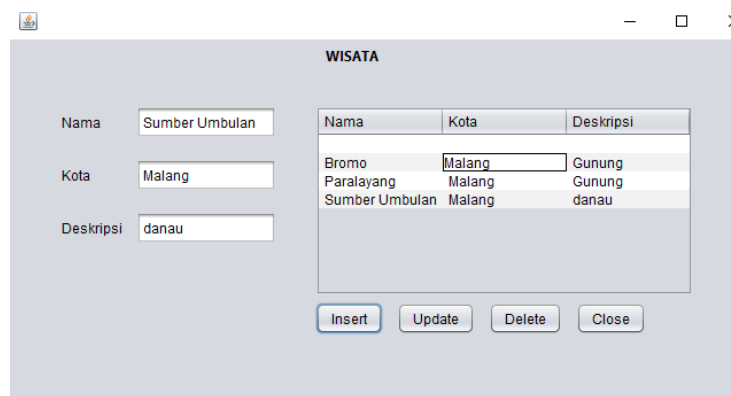
        preparedStatement.setString(1, nama);

        preparedStatement.executeUpdate();
        preparedStatement.close();

        JOptionPane.showMessageDialog(this, "Data deleted
successfully!");
        loadTableData();
    } catch (SQLException e) {
        e.printStackTrace();
        JOptionPane.showMessageDialog(this, "Error deleting
data from the database: " + e.getMessage(), "Error",
JOptionPane.ERROR_MESSAGE);
    }
}

```

Tampilan Hasil:



Gambar 6.35 Tampilan Hasil Method Tampi() GUI_Wisata

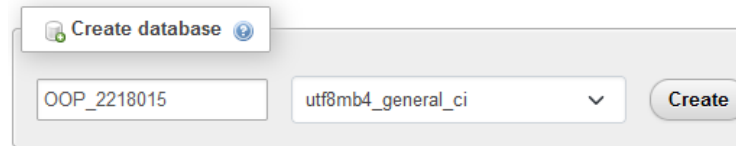
Analisa:

Pada gui wisata kita masukkan nama tempat wisata,kota dan deskripsi tempa wisata.Setelah itu tekan inset untuk menampilkan data yang sudah di input tadi.Setelah itu kita bisa update nama,kota atau deskripsi tempat wisata tersebut.Sedangkan dalete di gunakan untuk menghapus data yang sudah di input tadi.Terakhir kita bisa tekan close untuk keluar dari tampilan wisata tersebut.

VI.9 Tugas Rumah 3 :

Menerapkan CRUD Pada GUI_Reservasi.java

Membuat Database oop_2218015



Structure tb_Reservasi pada database oop_2218015

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	Nama	varchar(255)	utf8mb4_general_ci		No	None			Change Drop More
2	Harga	int(11)			No	0			Change Drop More
3	Jumlah	int(11)			No	None			Change Drop More
4	Total	int(11)			No	None			Change Drop More

Gambar 6.36 Membuat table(tb_Reservasi)

Source code pada GUI:

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.swing.JOptionPane;
import javax.swing.table.DefaultTableModel;
```

Source code di bawah Konstruktor GUI_.....:

```
private Connection conn;
```

Source code method koneksi :

```
private void connectToDatabase() {
    try {
        conn = DriverManager.getConnection(DB_URL,
        USER, PASS);
    } catch (SQLException e) {
        e.printStackTrace();
    }
}
```

Source code di method tampil:

```
private void loadDataIntoTable() {
    try {
        String query = "SELECT Nama, Harga, Jumlah,
        Total FROM reservasi";
        PreparedStatement pst =
        conn.prepareStatement(query);
```



```

        ResultSet rs = pst.executeQuery();

        DefaultTableModel model = (DefaultTableModel)
jTable1.getModel();
        model.setRowCount(0);

        while (rs.next()) {
            String nama = rs.getString("Nama");
            int harga = HARGA; // Use the fixed Harga
value
            int jumlah = rs.getInt("Jumlah");
            int total = rs.getInt("Total");

            model.addRow(new Object[]{nama, harga,
jumlah, total});
        }

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

```

Source code method tambah :

```

String query = "INSERT INTO reservasi (Nama, Jumlah,
Total) VALUES (?, ?, ?)";
PreparedStatement pst =
conn.prepareStatement(query);
pst.setString(1, nama);
pst.setInt(2, jumlah);
pst.setInt(3, total);

```

Source code method ubah :

```

String query = "UPDATE reservasi SET Jumlah=?, Total=?
WHERE Nama=?";
PreparedStatement pst =
conn.prepareStatement(query);
pst.setInt(1, jumlah);
pst.setInt(2, total);
pst.setString(3, nama);

```

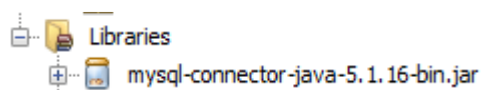
Source code method hapus :

```

String query = "DELETE FROM reservasi WHERE Nama=?";
PreparedStatement pst =
conn.prepareStatement(query);
pst.setString(1, nama);

```

Tambahkan library mysql.connector.jar pada library dengan cara klik kanan pada library, kemudian pilih **Add JAR/Folder...**



Gambar 6.37 Menambahkan *Library Connection*

Desain *form* GUI_Reservasi.java

Gambar 6.38 Desain Form GUI_Reservasi

Tabel 6.7 Properti Desain GUI_Reservasi(Database)

No	Objek	Properti	Nilai
1	jLabel1	Text	Reservasi
	jLabel2	Text	Nama
	jLabel3	Text	Harga Tiket
	jLabel4	Text	Jumlah Orang
	JTextField1	Name	txtR
		Text	“ “
	JTextField2	Name	txtHarga
		Text	“ “
	JTextField3	Name	txtJumlah
		Text	“ “
	jScrollPane2	Name	jScrollPane2
		Text	“ “
	JButton1	Name	btninsert
		Text	“ “
	JButton2	Name	btnupdate
		Text	“ “
	JButton3	Name	btndelete
		Text	“ “

Source code pada tombol Tambah:

```
private void
btninsertActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    try {
        String nama = txtR.getText();
        int harga = Integer.parseInt(txtHarga.getText());
        int jumlah = Integer.parseInt(txtJumlah.getText());
        int total = harga * jumlah;

        String query = "INSERT INTO reservasi (Nama,
        Jumlah, Total) VALUES (?, ?, ?)";
    }
}
```

```

        PreparedStatement          pst          =
conn.prepareStatement(query);
        pst.setString(1, nama);
        pst.setInt(2, jumlah);
        pst.setInt(3, total);

        pst.executeUpdate();

        // Refresh jTable1
        loadDataIntoTable();

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

```

Source code pada tombol Ubah:

```

private void
btnupdateActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    try {
        String nama = txtR.getText();
        int harga = Integer.parseInt(txtHarga.getText());
        int jumlah = Integer.parseInt(txtJumlah.getText());
        int total = harga * jumlah;

        String query = "UPDATE reservasi SET Jumlah=?,
Total=? WHERE Nama=?";
        PreparedStatement          pst          =
conn.prepareStatement(query);
        pst.setInt(1, jumlah);
        pst.setInt(2, total);
        pst.setString(3, nama);

        pst.executeUpdate();

        // Refresh jTable1
        loadDataIntoTable();

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

```

Source code pada tombol Hapus:

```

private void
btndeleteActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    try {
        String nama = txtR.getText();

        String query = "DELETE FROM reservasi WHERE
Nama=?";
        PreparedStatement          pst          =
conn.prepareStatement(query);
        pst.setString(1, nama);
    }
}

```

```

        pst.executeUpdate();

        // Refresh jTable1
        loadDataIntoTable();

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

```

Tampilan Hasil:

Nama	Harga	Jumlah	Total
Elsa	25000	15	375000
Hafid	25000	10	250000
tesa	25000	2	50000

Gambar 6.39 Tampilan Hasil Method Tampi() GUI_Reservasi

Analisa:

Pada tampilan gui reservasi di atas terdapat nama, harga tiket, dan jumlah orang. Tombol insert di gunakan untuk memasukkan inputan nilai nama, harga dan jumlah orang yang telah dimasukkan. Tombol Update di pakai untuk mengupdate data yang telah dimasukkan. Sedangkan tombol delete untuk keluar dari tampilan gui tersebut.



VI.10 Kesimpulan

1. Exception muncul ketika program menghadapi situasi yang tidak dapat diatasi atau tidak terduga, seperti kesalahan pembagian nol, akses data yang tidak valid, ataupun masalah jaringan.
2. *Exception Handling* merupakan mekanisme yang diperlukan dalam menangani *error* yang terjadi pada saat *runtime* (program berjalan) atau yang lebih dikenal dengan sebutan *runtime error*.
4. JDBC API (Java Database Connectivity Application Programming Interface) adalah antarmuka pemrograman dalam bahasa Java yang memungkinkan kita berinteraksi dengan informasi dalam basis data.

Nama Aslab :	TTD :
Firman Frezy Pradana 2118112	
Tanggal : 17 Desember 2023	



BAB VII

KESIMPULAN

1. Class merupakan rancangan dari sebuah objek yang mendefinisikan atribut(ciri/variabel) dan method
2. Object adalah realisasi dari class.
3. Atribut adalah karakteristik unik atau ciri dari sebuah objek.
4. Constructor adalah method khusus yang otomatis dieksekusi pada saat menginstansiasi object dari class tertentu.
5. Inheritance atau Pewarisan/Penurunan adalah konsep pemrograman dimana sebuah class dapat ‘menurunkan” properti dan method yang dimilikinya kepada class lain.
6. Enkapsulasi merupakan proses pemaketan objek beserta methodnya untuk menyembunyikan rincian implementasi dari pemakai/objek lainnya.
7. Overloading adalah sebuah kemampuan yang membolehkan sebuah class mempunyai 2 atau lebih method dengan nama yang sama.
8. Overriding method adalah kemampuan dari subclass(child class) untuk memodifikasi method dari superclass, dengan cara mendefinisikan kembali method superclass-nya.
9. *Abstract Class* adalah sebuah *class* yang tidak bisa di-*instansiasi* (tidak bisa dibuat menjadi objek) dan berperan sebagai ‘kerangka dasar’ bagi class turunannya.
10. Polimorfisme memungkinkan suatu entitas (seperti method atau function) untuk berperilaku dengan cara yang berbeda tergantung pada konteks di mana entitas tersebut digunakan “Bentuk” di sini dapat kita artikan: isinya berbeda (overriding), parameternya berbeda dan tipe datanya berbeda (overloading).
11. Interface berfungsi sebagai penghubung antar sesuatu yang ‘*Abstract*’ dengan sesuatu yang nyata.
12. Exception merupakan kondisi yang tidak diinginkan yang dapat terjadi selama eksekusi program.
13. *Exception Handling* merupakan mekanisme yang diperlukan dalam menangani error yang terjadi pada saat *runtime* (program berjalan) atau yang lebih dikenal dengan sebutan *runtime error*



DAFTAR PUSTAKA

- Laboratorium Pemrograman Komputer. 2023. Modul Praktikum Object Oriented Programming. Laboratorium Pemrograman Komputer Teknik Informatika Institut Teknologi Nasional Malang.
- Rentsch, Tim. "Object oriented programming." *ACM Sigplan Notices* 17.9 (1982): 51-57.
- Stefik, Mark, and Daniel G. Bobrow. "Object-oriented programming: Themes and variations." *AI magazine* 6.4 (1985): 40-40.
- Wegner, Peter. "Concepts and paradigms of object-oriented programming." *ACM Sigplan Oops Messenger* 1.1 (1990): 7-87.
- Stroustrup, Bjarne. "What is object-oriented programming?." *IEEE software* 5.3 (1988): 10-20.
- Cox, Brad J. *Object oriented programming: an evolutionary approach*. Addison-Wesley Longman Publishing Co., Inc., 1986.
- Agha, Gul. "Concurrent object-oriented programming." *Communications of the ACM* 33.9 (1990): 125-141.
- Jusuf, Heni, Nurdin Ibrahim, and Atwi Suparman. "Developing a hybrid learning strategy for students' engagement in object-oriented programming course." *Universal Journal of Educational Research* 7.9 (2019).
- Sari, Ambar Wulan, Resty Wahyuni, and Alfitriani Siregar. "The Effect Of Object-Oriented Programming (Adobe-Flash) Based Multimedia Learning Methods On English For Tourism Courses." *EduTech: Jurnal Ilmu Pendidikan dan Ilmu Sosial* 7.2 (2021): 377845.
- Nahlah, Ms, et al. "The Implementation of OOP (Object Oriented Programming) in Building an E-Commerce Website." *1st International Conference on Advanced Multidisciplinary Research (ICAMR 2018)*. Atlantis Press, 2019.



LABORATORIUM PEMROGRAMAN KOMPUTER
INSTITUT TEKNOLOGI NASIONAL
Kampus II : Jl. Raya Karanglo Km. 2 Malang

LEMBAR ASISTENSI PRAKTIKUM OBJECT ORIENTED PROGRAMING
SEMESTER GENAP TAHUN AKADEMIK 2023/2024

Nama : Tesalonika Dua Nurak
NIM : 2218015
Kelompok : 08

Foto latar
merah
Almamater

No.	Tanggal	Asistensi			Paraf	
		Konsep	Program	Hasil Akhir		
1		Instruktur	<div><input type="checkbox"/> Konsep Dasar OOP</div> <div><input type="checkbox"/> <i>Konstruktor, Inheritance</i></div> <div><input type="checkbox"/> <i>Enkapsulasi</i></div> <div><input type="checkbox"/> <i>Overloading, Overriding</i></div> <div><input type="checkbox"/> <i>Abstract , Polimorfisme</i></div> <div><input type="checkbox"/> <i>Interface</i></div> <div><input type="checkbox"/> Exception, CRUD Java</div>			
2		Dosen				
Batas Akhir:						

Malang, Desember 2023

Asisten,

Dosen Pembimbing,

(Fiman Frezy Pradana)
NIM : 2118112

(Ahmad Faisol, ST, MT.)
NIP.P :1031000431