**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 : 8**

**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  **(Ahmad Faisol, ST, MT)**  **NIP.P :1031000431** | Menyetujui  Dosen Pembimbing  **(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 kehadirat Allah SWT, karena atas berkah rahmat dan karunia-Nya sehingga penulis dapat menyelesaikan Laporan Praktikum Pemrogramman Web, 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 Pemrograman Web.
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. nstruktur 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](#_Toc153625687)

[DAFTAR ISI ii](#_Toc153625688)

[DAFTAR GAMBAR iv](#_Toc153625689)

[DAFTAR TABEL vii](#_Toc153625690)

[BAB I PENDAHULUAN I-1](#_Toc153625691)

[I.1 Latar Belakang I-1](#_Toc153625692)

[I.2 Rumusan Masalah I-2](#_Toc153625693)

[I.3 Tujuan I-2](#_Toc153625694)

[I.4 Manfaat I-2](#_Toc153625695)

[BAB II KONSEP DASAR OOP II-3](#_Toc153625696)

[II.1 Landasan Teori II-3](#_Toc153625697)

[II.2 Langkah – Langkah Praktikum II-11](#_Toc153625698)

[II.3 Tugas Praktikum –1: II-11](#_Toc153625699)

[II.4 Tugas Rumah 1 : II-15](#_Toc153625700)

[II.5 Tugas Rumah 2 : II-16](#_Toc153625701)

[II.6 Kesimpulan II-20](#_Toc153625702)

[BAB III KONSTRUCTOR DAN INHERITANCE III-21](#_Toc153625703)

[III.1 Landasan Teori III-21](#_Toc153625704)

[III.2 Langkah – Langkah Praktikum III-23](#_Toc153625705)

[III.3 Tugas Praktikum 1 : III-23](#_Toc153625706)

[III.4 Tugas Praktikum 2 : III-26](#_Toc153625707)

[III.5 Tugas Rumah 1 : III-32](#_Toc153625708)

[III.6 Tugas Rumah 2 : III-36](#_Toc153625709)

[III.7 Kesimpulan III-43](#_Toc153625710)

[BAB IV ENKAPSULASI , OVERRIDING DAN OVERLOADING IV-1](#_Toc153625711)

[IV.1 Landasan Teori IV-1](#_Toc153625712)

[IV.2 Langkah – Langkah Praktikum IV-3](#_Toc153625713)

[IV.3 Tugas Praktikum 1 : IV-4](#_Toc153625714)

[IV.4 Tugas Praktikum 2 : IV-11](#_Toc153625715)

[IV.5 Tugas Rumah 1 : Implementasi Enkapsulasi di 3 *Class* IV-16](#_Toc153625716)

[IV.6 Tugas Rumah 2: IV-24](#_Toc153625717)

[IV.7 Kesimpulans IV-31](#_Toc153625718)

[BAB V ABSTRACT, POLIMORFISME, DAN EXCEPTION HANDLING V-1](#_Toc153625719)

[V.1 Landasan Teori V-1](#_Toc153625720)

[V.2 Langkah – Langkah Praktikum V-8](#_Toc153625721)

[V.3 Tugas Praktikum 1 : V-8](#_Toc153625722)

[V.4 Tugas Praktikum 2 : V-15](#_Toc153625723)

[V.5 Tugas Praktikum 3 : V-18](#_Toc153625724)

[V.6 Tugas Rumah 1 : V-20](#_Toc153625725)

[V.7 Tugas Rumah 2 : V-23](#_Toc153625726)

[V.8 Tugas Rumah 3 : V-26](#_Toc153625727)

[V.9 Kesimpulan V-29](#_Toc153625728)

[BAB VI INTERFACE DAN PENGENALAN DATABASE VI-30](#_Toc153625729)

[VI.1 Landasan Teori VI-30](#_Toc153625730)

[VI.2 Langkah – Langkah Praktikum VI-33](#_Toc153625731)

[VI.3 Tugas Praktikum 1 : VI-34](#_Toc153625732)

[VI.4 Tugas Praktikum 2 : VI-38](#_Toc153625733)

[VI.5 Tugas Praktikum 3 : VI-46](#_Toc153625734)

[VI.6 Tugas Praktikum 4 : VI-53](#_Toc153625735)

[VI.7 Tugas Rumah 1 : VI-60](#_Toc153625736)

[VI.8 Tugas Rumah 2 : VI-60](#_Toc153625737)

[VI.9 Tugas Rumah 3 : VI-63](#_Toc153625738)

[VI.10 Tugas Rumah 4 : VI-65](#_Toc153625739)

[BAB VII KESIMPULAN VII-70](#_Toc153625740)

[DAFTAR PUSTAKA VII-1](#_Toc153625741)

# DAFTAR GAMBAR

[Gambar 2.1 Contoh Class II-5](#_Toc153625597)

[Gambar 2.2 Desain form mahasiswa II-12](#_Toc153625598)

[Gambar 2.3 Hasil Gui\_mahasiswa II-14](#_Toc153625599)

[Gambar 2.4 Flowchart Sistem Informasi Wisata II-15](#_Toc153625600)

[Gambar 2.5 Desain GUI\_Wisata.java II-17](#_Toc153625601)

[Gambar 2.6 Tampilan Hasil GUI\_Wisata.java II-19](#_Toc153625602)

[Gambar 3.1 Desain Gui\_Matkul .java III-24](#_Toc153625603)

[Gambar 3.2 Hasil Tampilan GUI\_Matkul.java III-26](#_Toc153625604)

[Gambar 3.3 Desain GUI\_Penilaian.java III-27](#_Toc153625605)

[Gambar 3.4 Hasil Tampilan GUI\_Penilaian.java III-31](#_Toc153625606)

[Gambar 3.5 Desain GUI\_Wisata.java III-33](#_Toc153625607)

[Gambar 3.6 Hasil Tampilan GUI\_Wisata.java III-35](#_Toc153625608)

[Gambar 3.7 Desain GUI\_Wisata.java III-37](#_Toc153625609)

[Gambar 3.8 Hasil Tampilan GUI\_Wisata java III-38](#_Toc153625610)

[Gambar 3.9 Desain GUI\_Login.java III-39](#_Toc153625611)

[Gambar 3.10 Hasil Tampilan Login.java III-40](#_Toc153625612)

[Gambar 3.11 Desain GUI\_Reservasi.java III-41](#_Toc153625613)

[Gambar 3.12 Hasil Tampilan GUI\_Reservasi.java III-42](#_Toc153625614)

[Gambar 4.1 Desain GUI\_Nilai.java IV-5](#_Toc153625615)

[Gambar 4.2 Hasil Tampilan GUI\_Nilai.java IV-10](#_Toc153625616)

[Gambar 4.3 Desain GUI\_Mahasiswa.java IV-12](#_Toc153625617)

[Gambar 4.4 Hasil Tampilan GUI\_Mahasiswa.java IV-15](#_Toc153625618)

[Gambar 4.5 Desain GUI\_Wisata.java IV-17](#_Toc153625619)

[Gambar 4.6 Hasil Tampilan GUI\_Wisata.java IV-19](#_Toc153625620)

[Gambar 4.7 Desain GUI\_Login.java IV-20](#_Toc153625621)

[Gambar 4.8 Hasil Tampilan GUI\_Login.java IV-21](#_Toc153625622)

[Gambar 4.9 Desain GUI\_Reservasi.java IV-23](#_Toc153625623)

[Gambar 4.10 Hasil Tampilan GUI\_Reservasi.java IV-24](#_Toc153625624)

[Gambar 4.11 Desain GUI\_Wisata.java IV-26](#_Toc153625625)

[Gambar 4.12 Hasil Tampilan GUI\_Wisata.java IV-27](#_Toc153625626)

[Gambar 4.13 Desain GUI\_Reservasi.java IV-28](#_Toc153625627)

[Gambar 4.14 Hasil Tampilan GUI\_Reservasi.java IV-30](#_Toc153625628)

[Gambar 5.1 Konsep Updasting dan Downcasting V-6](#_Toc153625629)

[Gambar 5.2 *Diagram Class* Penilaian V-9](#_Toc153625630)

[Gambar 5.3 *Diagram Class* KeaktifanMahasiswa V-10](#_Toc153625631)

[Gambar 5.4 Desain Gui\_ Penilaian.java V-11](#_Toc153625632)

[Gambar 5.5 Hasil Tampilan GUI\_Penilaian.java V-14](#_Toc153625633)

[Gambar 5.6 *Diagram Class* Data\_kuliah V-15](#_Toc153625634)

[Gambar 5.7 Desain Gui\_Matkul.java V-15](#_Toc153625635)

[Gambar 5.8 Hasil Tampilan GUI\_Matkul.java V-18](#_Toc153625636)

[Gambar 5.9 Desain Gui\_MenuUtama.java V-19](#_Toc153625637)

[Gambar 5.10 Hasil Tampilan GUI\_MenuUtama.java V-20](#_Toc153625638)

[Gambar 5.11 Hasil Tampilam Diagram Hitung V-20](#_Toc153625639)

[Gambar 5.12 Desain Gui\_Reservasi.java V-21](#_Toc153625640)

[Gambar 5.13 Hasil Tampilan GUI\_Reservasi.java V-22](#_Toc153625641)

[Gambar 5.14 Desain Gui\_Login.java V-24](#_Toc153625642)

[Gambar 5.15 Hasil tampilan gui\_login.java yang berhasil V-25](#_Toc153625643)

[Gambar 5.16 Hasil Tampilan gui\_login.java yang gagal login V-25](#_Toc153625644)

[Gambar 5.17 Desain Gui\_Reservasi.java V-27](#_Toc153625645)

[Gambar 5.18 Hasil Tampilan GUI\_Reservasi.java V-28](#_Toc153625646)

[Gambar 6.1 Desain Gui\_ Penilaian.java VI-35](#_Toc153625647)

[Gambar 6.2 Hasil Tampilan GUI\_Penilaian.java VI-38](#_Toc153625648)

[Gambar 6.3 Membuat table(tb\_mahasiswa) VI-39](#_Toc153625649)

[Gambar 6.4 Menambahkan *Library Connection* VI-39](#_Toc153625650)

[Gambar 6.5 Desain Form Gui\_Mahasiswa VI-42](#_Toc153625651)

[Gambar 6.6 Tampilan Hasil Method tampil() Gui\_Mahasiswa VI-45](#_Toc153625652)

[Gambar 6.7 Tampilan Hasil Method tambah() Gui\_Mahasiswa VI-45](#_Toc153625653)

[Gambar 6.8 Tampilan Hasil Method ubah() Gui\_Mahasiswa VI-45](#_Toc153625654)

[Gambar 6.9 Tampilan Hasil Method hapus() Gui\_Mahasiswa VI-46](#_Toc153625655)

[Gambar 6.10 Tampilan Hasil Method batal() Gui\_Mahasiswa VI-46](#_Toc153625656)

[Gambar 6.11 Membuat table (tb\_nilai) VI-46](#_Toc153625657)

[Gambar 6.12 Menambhakan *Library Connection* VI-46](#_Toc153625658)

[Gambar 6.13 Desain Form Gui\_Nilai VI-50](#_Toc153625659)

[Gambar 6.14 Tampilan Hasil Method tampil() Gui\_Mahasiswa VI-52](#_Toc153625660)

[Gambar 6.15 Tampilan Hasil Method tambah() Gui\_Mahasiswa VI-53](#_Toc153625661)

[Gambar 6.16 Tampilan Hasil Method ubah() Gui\_Mahasiswa VI-53](#_Toc153625662)

[Gambar 6.17 Tampilan Hasil Method hapus() Gui\_Mahasiswa VI-53](#_Toc153625663)

[Gambar 6.18 Tampilan Hasil Method batal() Gui\_Mahasiswa VI-53](#_Toc153625664)

[Gambar 6.19 Membuat table (tb\_matkul) VI-53](#_Toc153625665)

[Gambar 6.20 Menambhakan *Library Connection* VI-54](#_Toc153625666)

[Gambar 6.21 Desain Form Gui\_Matkul VI-57](#_Toc153625667)

[Gambar 6.22 Tampilan Hasil Method tampil() Gui\_Mahasiswa VI-58](#_Toc153625668)

[Gambar 6.23 Tampilan Hasil Method tambah() Gui\_Mahasiswa VI-59](#_Toc153625669)

[Gambar 6.24 Tampilan Hasil Method ubah() Gui\_Mahasiswa VI-59](#_Toc153625670)

[Gambar 6.25 Tampilan Hasil Method hapus() Gui\_Mahasiswa VI-59](#_Toc153625671)

[Gambar 6.26 Tampilan Hasil Method batal() Gui\_Mahasiswa VI-59](#_Toc153625672)

[Gambar 6.27 1Tampilan Hasil Method tampil() GUI………. VI-60](#_Toc153625673)

[Gambar 6.28 Membuat database oop\_judul VI-60](#_Toc153625674)

[Gambar 6.29 Membuat table(tb\_...........) VI-60](#_Toc153625675)

[Gambar 6.30 Menambahkan *Library Connection* VI-61](#_Toc153625676)

[Gambar 6.31 Desain Form GUI\_……. VI-62](#_Toc153625677)

[Gambar 6.32 Tampilan Hasil Method Tampi() GUI\_....... VI-63](#_Toc153625678)

[Gambar 6.33 Membuat table(tb\_...........) VI-63](#_Toc153625679)

[Gambar 6.34 Menambahkan *Library Connection* VI-64](#_Toc153625680)

[Gambar 6.35 Desain Form GUI\_……. VI-64](#_Toc153625681)

[Gambar 6.36 Tampilan Hasil Method Tampi() GUI\_....... VI-65](#_Toc153625682)

[Gambar 6.37 Membuat table(tb\_...........) VI-65](#_Toc153625683)

[Gambar 6.38 Menambahkan *Library Connection* VI-66](#_Toc153625684)

[Gambar 6.39 Desain Form GUI\_……. VI-67](#_Toc153625685)

[Gambar 6.40 Tampilan Hasil Method Tampi() GUI\_....... VI-68](#_Toc153625686)

# DAFTAR TABEL

[Tabel 2.1 Properti Desain GUI\_Mahasiswa.java II-12](#_Toc153625479)

[Tabel 2.2 Properti Desain GUI\_Wisata.java II-17](#_Toc153625480)

[Tabel 3.1 Properti Desain GUI\_Matkul.java III-24](#_Toc153625481)

[Tabel 3.2 Properti Desain GUI\_Penilaian.java III-28](#_Toc153625482)

[Tabel 3.3 Properti Desain GUI\_CetakPenjualanKaos.java III-33](#_Toc153625483)

[Tabel 3.4 Properti Desain GUI\_Wisata.java III-37](#_Toc153625484)

[Tabel 3.5 Properti Desain Login,java III-39](#_Toc153625485)

[Tabel 3.6 Properti Desain GUI\_Reservasi.java III-41](#_Toc153625486)

[Tabel 4.1 Properti Desain GUI\_Nilai.java IV-6](#_Toc153625487)

[Tabel 4.2 Properti Desain GUI\_Mahasiswa.java IV-12](#_Toc153625488)

[Tabel 4.3 Properti Desain GUI\_Wisata.java IV-17](#_Toc153625489)

[Tabel 4.4 Properti Desain GUI\_Login.java IV-20](#_Toc153625490)

[Tabel 4.5 Properti Desain GUI\_Reservasi.java IV-23](#_Toc153625491)

[Tabel 4.6 Properti Desain GUI\_Wisata.java IV-26](#_Toc153625492)

[Tabel 4.7 Properti Desain GUI\_Resrvasi.java IV-29](#_Toc153625493)

[Tabel 5.1 Properti Desain GUI\_Penilaian.java V-11](#_Toc153625494)

[Tabel 5.2 Properti Desain GUI\_Matkul.java V-15](#_Toc153625495)

[Tabel 5.3 Properti Desain GUI\_MenuUtama.java V-19](#_Toc153625496)

[Tabel 5.4 Properti Desain GUI\_Reservasi.java V-21](#_Toc153625497)

[Tabel 5.5 Properti Desain GUI\_Login.java V-24](#_Toc153625498)

[Tabel 5.6 Properti Desain GUI\_Reservasi.java V-27](#_Toc153625499)

[Tabel 6.1 Properti Desain GUI\_Penilaia.java VI-35](#_Toc153625500)

[Tabel 6.2 Properti Desain GUI\_Mahasiswa (Database) VI-43](#_Toc153625501)

[Tabel 6.3 Properti Desain GUI\_ Nilai(Database) VI-50](#_Toc153625502)

[Tabel 6.4 Properti Desain GUI\_ Matkul (Database) VI-57](#_Toc153625503)

[Tabel 6.5 Properti Desain GUI\_.................... (Database) VI-62](#_Toc153625504)

[Tabel 6.6 Properti Desain GUI\_.................... (Database) VI-64](#_Toc153625505)

[Tabel 6.7 Properti Desain GUI\_.................... (Database) VI-67](#_Toc153625506)

# PENDAHULUAN

## 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 kebanyakrupaan. 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.

## Rumusan Masalah

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

## Tujuan

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

## Manfaat

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

# KONSEP DASAR OOP

|  |  |  |
| --- | --- | --- |
| **Jumlah Pertemuan** | **:** | **2 x 60 menit** |
| **Tujuan Praktikum** | **:** | 1. Praktikan mampu menerapkan konsep dasar pemrograman berorientasi objek. 2. Praktikan mampu membuat sebuah *Class*, atribut, *method* dan objek. 3. Praktikan mampu mengetahui fungsi *method* *getter* dan *setter*. 4. Praktikan mampu membuat program untuk memasukkan data mahasiswa berbasis GUI (*Graphical User Interface* ). |
| **Alat / bahan** | **:** | 1. Seperangkat *computer*. 2. Perangkat lunak: *Netbeans*. 3. Modul Praktikum *OOP* 2022. |

## Landasan Teori

1. 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.

1. Class & Object
   * + 1. Pengertian Class

*Class* merupakan rancangan dari sebuah objek yang mendefenisikan 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:

* 1. Penamaan *class* harus dimulai dengan huruf (a-z atau A-Z), atau karakter garis bawa ( **\_** ).
  2. Hanya boleh ada satu class public dalam satu file .java, *non* *public* *class* boleh lebih dari satu di dalam satu file .java
  3. Nama *class* *public* harus sama dengan nama file .java
  4. Komentar bisa diletakkan di mana saja
  5. Jika class berada dalam sebuah package, maka harus ada deklarasi nama package di bagian paling atas dengan format “ package nama\_package ;“
  6. Import berada antara deklarasi package dan deklarasi class

Jenis-jenis class yang umum di Java:

1. *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.

1. *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*).

1. *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.

1. *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.

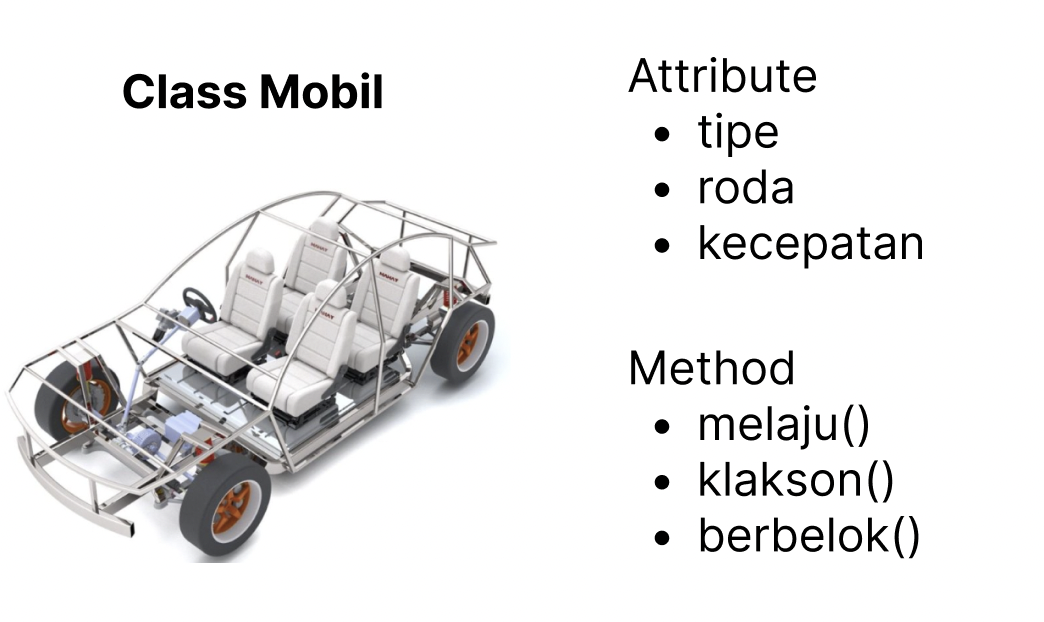
1. *Final* *Class* ( Kelas Final )

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:

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

Contoh dari class:



### Contoh Class

*Class* dideklarasikan sebagai berikut:

class NamaClass{

//deklarasi atribut

//deklarasi method

}

Keterangan :

* + 1. class adalah kata kunci yang digunakan untuk mendeklarasikan suatu kelas.
    2. NamaClass merupakan indentifier.
    3. Atribut dan Method bisa berjumlah 0 atau lebih.
       1. Pengertian Object

*Object* adalah realisasi dari *class*. Dapat di bayangkan bahwa class adalah sebuah cetakan/*tamplate* , dan *object* adalah bentuk dari representasi cetakan/*tamplate* 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();

* + - 1. 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:

1. Atribut *Instance* (*Instance Variables*)

Atribut yang berkaitan dengan objek yang dihasilkan dari *class* yang dipakai. Setiap objek memiliki salinan unik dari atribut ini. Atribut insance mendefinisikan karakteristik atau properti unik dari objek.

1. 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.

1. 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";  } |

* + - 1. 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:

1. *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  }  } |

1. *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());  }  } |

1. *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*.

1. *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**

}

1. *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”.

## 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.

## Tugas Praktikum –1:

Membuat kelas Mahasiswa.java dan GUI\_Mahasiswa.java

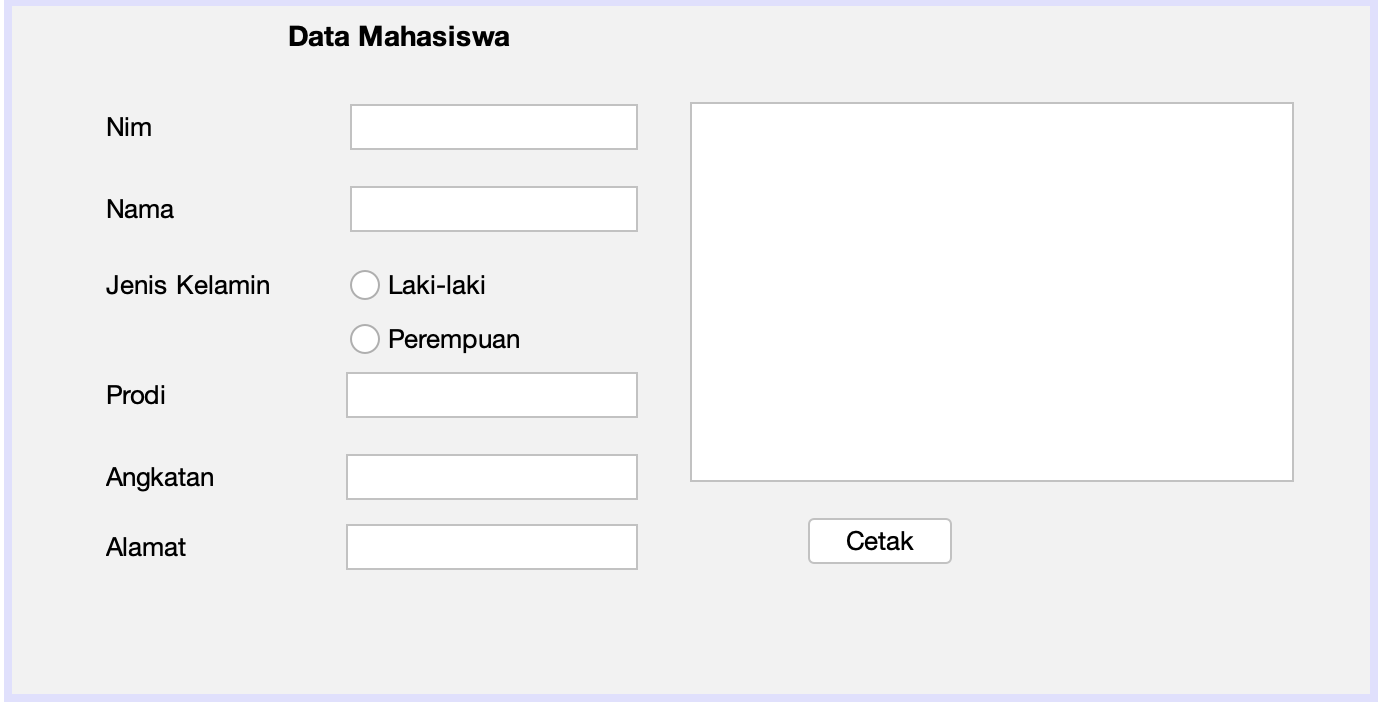
Diagram Class (Class Mahasiswa):

|  |
| --- |
| Mahasiswa  **Class** |
| ~ nim :String  ~ nama : String  ~ JenisKelamin : String  **Attribute**  ~ prodi : String  ~ Angkatan : String  ~ Alamat : String |
| ~ void dataNIM(String Nim)  ~ void dataNama(String Nama)  ~ void dataJK(String JenisKelamin)  ~ void dataProdi(String Prodi)  ~void dataAngkatan(String Angkatan)  **Method**  ~ void dataStatus(String Status)  ~ cetakNIM() : String  ~ cetakNama() : String  ~ cetakJenisKelamin() : String  ~ cetakProdi() : String  ~ cetakAngkatan() : String  ~ cetakStatus() : String |

*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):



### Desain form mahasiswa

#### 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");  } |

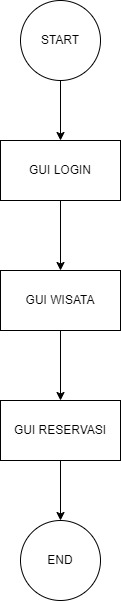
Tampilan Hasil :



### Hasil Gui\_mahasiswa

## Tugas Rumah 1 :

Flowchart Sistem Informasi Wisata



### 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.

## Tugas Rumah 2 :

Membuat *Class* List.java dan GUI\_Wisata.java

Judul : Sistem Informasi Wisata

*Diagram Class* (List.java):

**Method**

**Attribute**

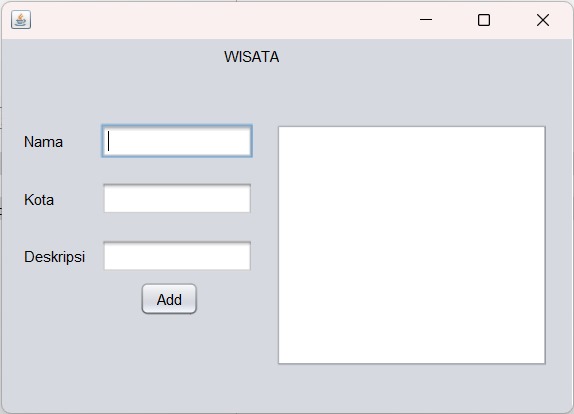
**Class**

|  |
| --- |
| List |
| * Nama Int * Kota Int * Deskripsi Int |
| * dataNama( ) Void * dataKota ( ) Void * dataDeskripsi ( ) Void * cetakNama( ) Void * cetakKota ( ) String * cetakDeskripsi( ) String |

*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):



### Desain GUI\_Wisata.java

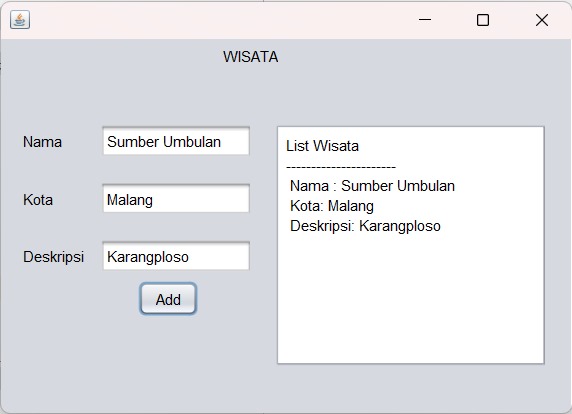
#### Properti Desain GUI\_Wisata.java

|  |  |  |  |
| --- | --- | --- | --- |
| **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 void 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:



### 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.

## 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 mendefenisikan 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.

# KONSTRUCTOR DAN INHERITANCE

|  |  |  |
| --- | --- | --- |
| **Jumlah Pertemuan** | **:** | **2 x 60 menit** |
| **Tujuan Praktikum** | **:** | 1. Praktikan mampu mengetahui tipe data dan *java* *identifier*. 2. Praktikan mampu merubah tipe data (*Casting*). 3. Praktikan mampu mengetahui dan membuat *method* konstruktor. 4. Praktikan mampu mengetahui Inheritance |
| **Alat / bahan** | **:** | 1. Seperangkat *computer*. 2. Perangkat lunak: *Netbeans*. 3. Modul Praktikum *OOP* 2023. |

## Landasan Teori

1. 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:

1. “NamaClass” merupakan nama class dapat di ganti sesuai keinginan
2. “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 otmatis terisi oleh data yang ada dalam method konstruktor

* + - 1. Aturan-aturan Constructor
  1. Mempunyai nama yang sama dengan nama class,
  2. Tidak mempunyai tipe return,
  3. Digunakan untuk menginstansiasi object,
  4. Hanya mempunyai access modifer, tidak ada keyword lain yang diletakkan sebelum nama method pada deklarasi constructor.

1. 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*:

1. *Super Class* : kelas induk yang mewariskan atribut dan *method* kepada turunannya.
2. *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.
3. Reusability : menggunakan kembali atribut dan *method* dari *super class* di *sub class*.

## 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.

## Tugas Praktikum 1 :

Membuat *Class* Data\_Matkul dan GUI\_Matkul

Judul : Konstruktorpada *class* Data\_Matkul

*Diagram Class* ( Data Matkul ):

**Class**

**Attribute**

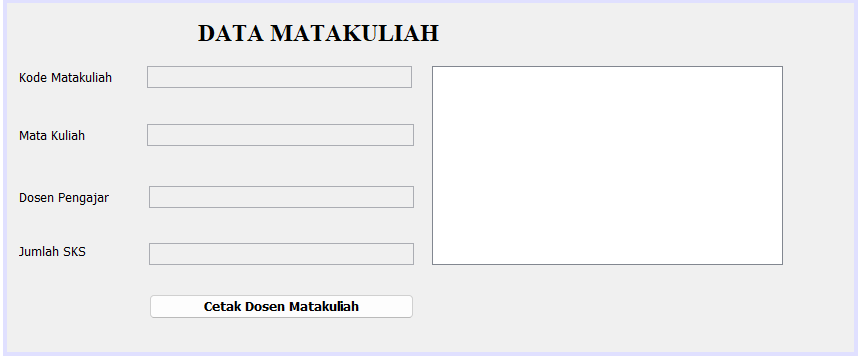
**Method constructor**

|  |  |
| --- | --- |
| Data\_Matkul | |
| +kode\_mk | String |
| +nama\_mk | String |
| + dosen\_pengampu | String |
| +Jml\_sks | int |
| + Data\_Matkul (String kd, String mk. Stirng dsn, int jmlsks) | |

*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):

****

### Desain Gui\_Matkul .java

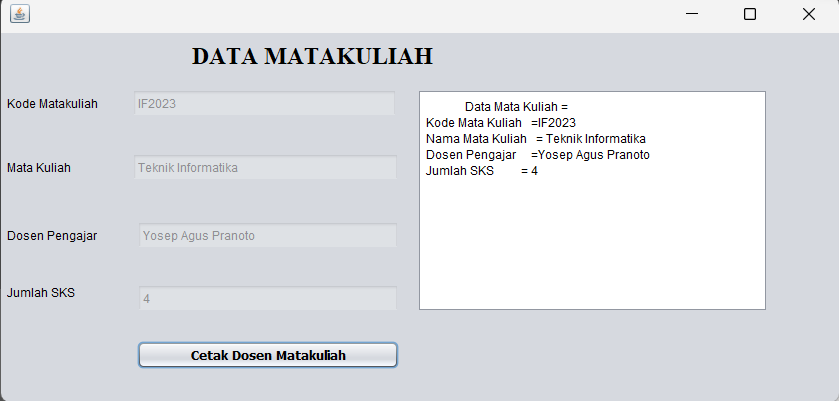
#### Properti Desain GUI\_Matkul.java

|  |  |  |  |
| --- | --- | --- | --- |
| **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:

****

### Hasil Tampilan GUI\_Matkul.java

## Tugas Praktikum 2 :

Membuat *Class* Penilaian dan GUI\_Penilaian

Judul : *Inheritance* pada class Penilaian

*Diagram Class* (Penilaian):

**Class**

**Attribute**

**Method constructor**

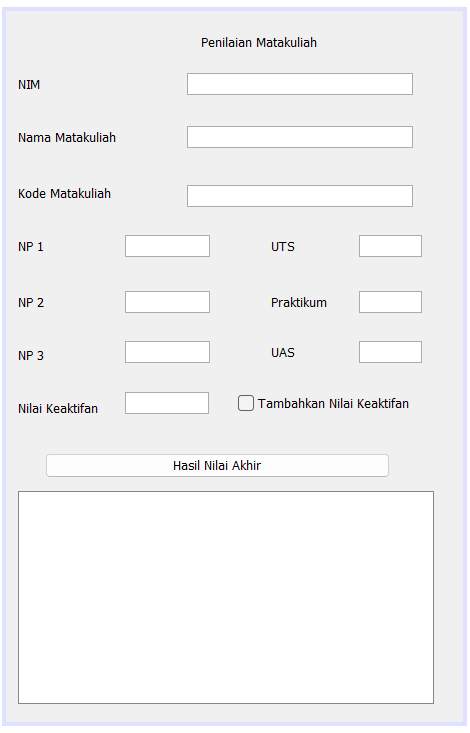
|  |  |
| --- | --- |
| Penilaian | |
| +NIM | String |
| +nama | String |
| + NP1 | int |
| +NP2 | int |
| +NP3 | int |
| +NIM | int |
| +NIM | int |
| +NIM | int |
| + nilaiProses() | double |
| + nilaiAkhir() | double |
| + tampilNA() | int |
| + tampilNA() | double |

|  |  |  |
| --- | --- | --- |
| Keaktifan Mahasiswa | | |
| +nilai\_keaktifan |  | int |
| +nilaiKeaktifan() |  | double |

*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):

****

### Desain GUI\_Penilaian.java

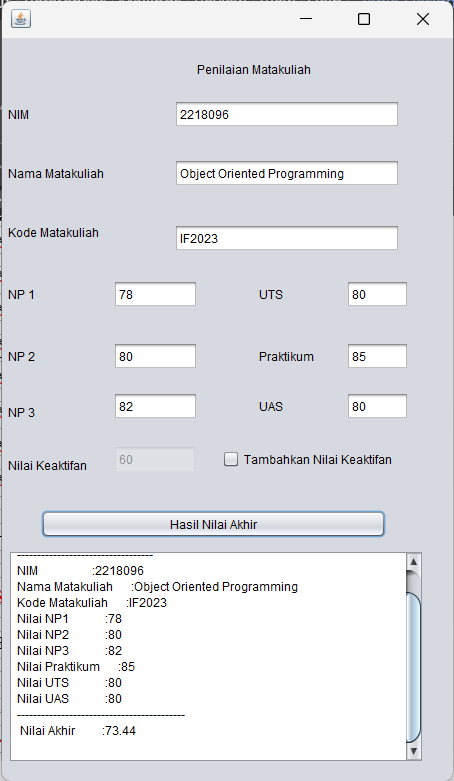
#### Properti Desain GUI\_Penilaian.java

|  |  |  |  |
| --- | --- | --- | --- |
| **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 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.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");  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:

****

### Hasil Tampilan GUI\_Penilaian.java

## Tugas Rumah 1 :

Judul : Konstruktor pada *class* List

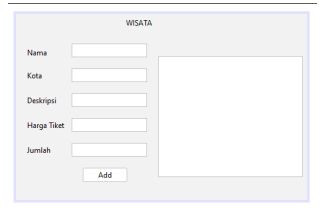
*Diagram Class* (List.java):

|  |  |
| --- | --- |
| List  **Class**  **Attribute**  **Method constructor** | |
| * dataNama | Void |
| * dataKota | Void |
| * dataDeskrispi | Void |
| * cetakNama * cetakKota * cetakDeskripsi | Void  String  String |
| * HargaTiket * Jumlah * Total | Int |

*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):

****

### Desain GUI\_Wisata.java

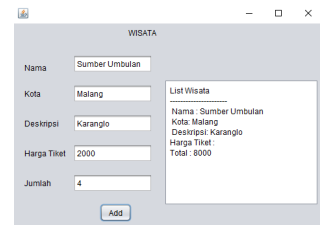
#### Properti Desain GUI\_CetakPenjualanKaos.java

|  |  |  |  |
| --- | --- | --- | --- |
| **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:

****

### 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.

## Tugas Rumah 2 :

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

*Diagram Class* (List):

**Class**

**Attribute**

**Method**

|  |  |  |
| --- | --- | --- |
| List | | |
| * dataNama * dataKota * dataDeskrispi * cetakNama * cetakKota * cetakDeskripsi | | Void  Void  Void  Void  String  String |
| * HargaTiket * Jumlah * Total |  | Int  Int  Itn |

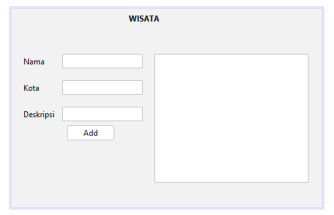
|  |
| --- |
| Login |
| +username;  +password;  +nama; |
| +setUsername  +setPaaword  +CekLogin |

|  |
| --- |
| Hitung |
| +String NamaR |
| +intTotal  +dataNamaR |

*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):

****

### Desain GUI\_Wisata.java

#### 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:

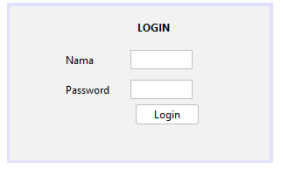
****

### 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) :



### Desain GUI\_Login.java

#### 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 bttnLogin :

|  |
| --- |
| 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 :



### 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) :



### Desain GUI\_Reservasi.java

#### 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 bttnHitung :

|  |
| --- |
| 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 :



### 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.

## Kesimpulan

* + - 1. Konsktruktor 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

# ENKAPSULASI , OVERRIDING DAN OVERLOADING

|  |  |  |
| --- | --- | --- |
| **Jumlah Pertemuan** | **:** | **2 x 60 menit** |
| **Tujuan Praktikum** | **:** | 1. Praktikan mampu memahami tentang hak akses (Enkapsulasi). 2. Praktikan mampu mengimplementasikan konsep *Encapsulasi* kedalam program. 3. Praktikan mampu memahami dan membuat *method override*. |
| **Alat / bahan** | **:** | 1. Seperangkat *computer*. 2. Perangkat lunak: *Netbeans*. 3. Modul Praktikum *OOP* 2022. |

## Landasan Teori

1. 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

1. Untuk meningkatkan keamanan data;
2. Agar lebih mudah dalam mengontrol atribut dan method;
3. 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.
   * + 1. Jenis Enkapsulasi (Hak Akses)
4. *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.

1. *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.

1. *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.

1. 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*)

1. Pengertian *Overloading* dan *Overriding*.
2. *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.*

1. *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*:

* 1. Parameter yang terdapat pada *method* *Overriding* di *subclass* harus sama dengan parameter yang terdapat pada *parent class*.
  2. 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.

## 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.

## Tugas Praktikum 1 :

Membuat *Class* Penilaian dan GUI\_Nilai

Judul : Implementasi Enkapsulasi *Class* Penilaian

*Diagram Class* (Penilaian):

**Class**

**Attribute**

**Method constructor**

|  |  |
| --- | --- |
| Penilaian | |
| +NIM  +node\_mk  +nilaiTugas  +nilaiPrak | String  String  Int  int |
| +nilaiProses()  +nilaiAkhir()  +tampilNA()  +nilaiKeaktifan()  +getNP1()  + setNP1(int NP1)  + getNP2()  + setNP2(int NP2)  + getNP3() + setNP3(int NP3)  + getNilaiPrak() + setNilaiPrak(int nilaiPrak)  + getUTS() + setUTS(int NilaiUTS)  + getUAS() + setUAS(int NilaiUAS) | double  double  double  double  int  void  int  void  int  void  int  void  int  void  int  void |

*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):

****

### Desain GUI\_Nilai.java

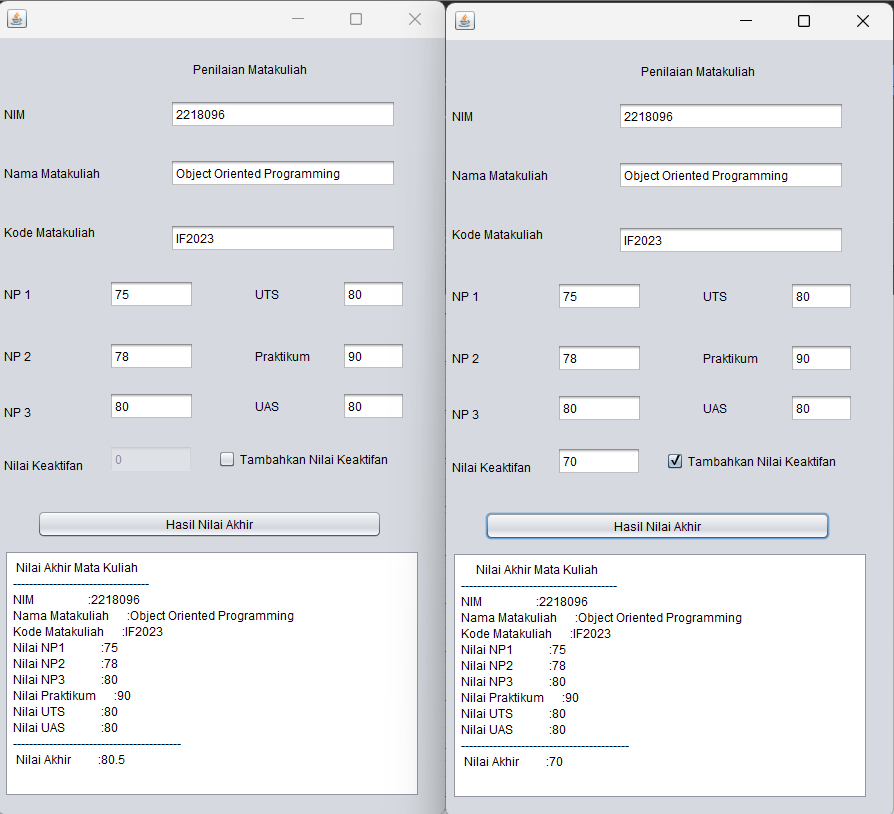
#### Properti Desain GUI\_Nilai.java

|  |  |  |  |
| --- | --- | --- | --- |
| **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:

****

### Hasil Tampilan GUI\_Nilai.java

## Tugas Praktikum 2 :

Membuat *Class* Mahasiswa dan GUI\_Mahasiswa

Judul : Implementasi Overloading dan Overriding pada *class* Mahasiswa

*Diagram Class* (Mahasiswa):

**Class**

**Attribute**

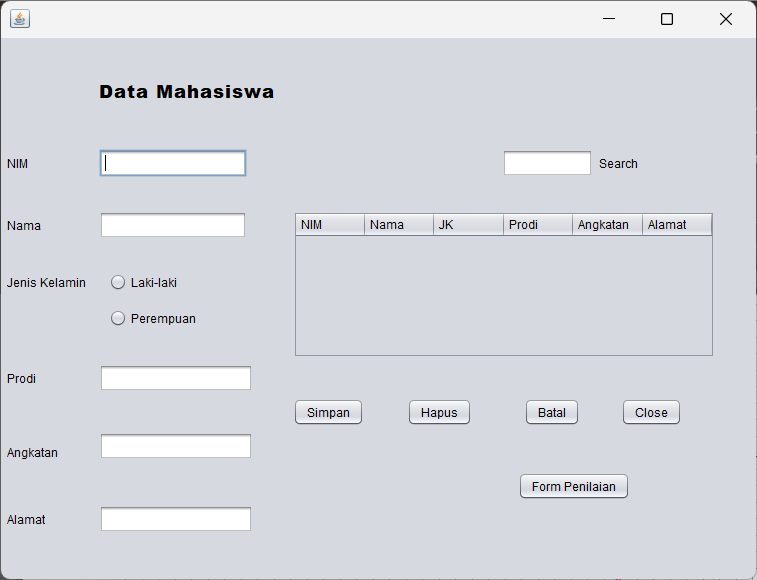
**Method constructor**

|  |  |
| --- | --- |
| Mahasiswa | |
| + nim  + nama  + JenisKelamin  + Prodi  + Angkatan  + Alamat | String  String  String  String  String  String |
| +Void dataNIM (String Nim)  +Void dataNama (String Nama)  +Void dataJK (String JenisKelamin)  +Void dataProdi (String Prodi)  +Void dataAngkatan (String Angkatan)  +Void dataAlamat (String Alamat)  + cetakNIM ()  + cetakNama ()  + cetakJenisKelamin ()  + cetakProdi ()  + cetakAngkatan ()  + cetakAlamat () | String  String  String  String  String  String |

*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):

****

### Desain GUI\_Mahasiswa.java

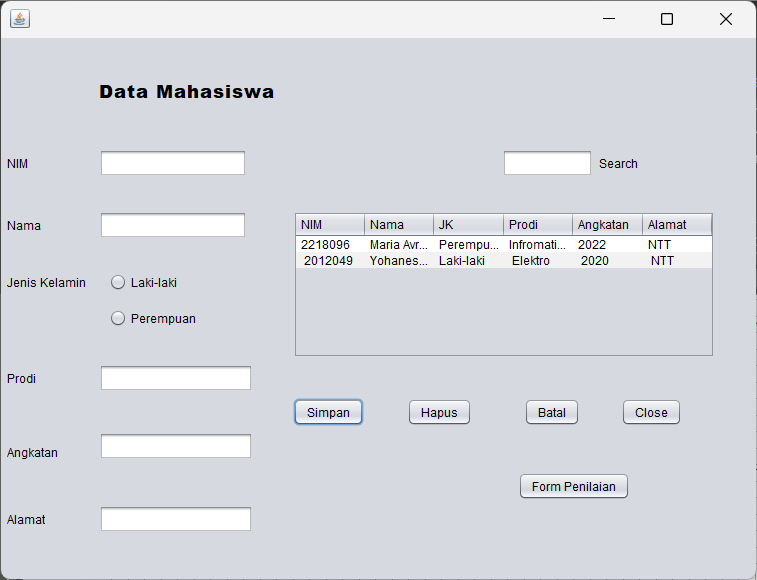
#### Properti Desain GUI\_Mahasiswa.java

|  |  |  |  |
| --- | --- | --- | --- |
| **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 ditmbahkan 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:

****

### Hasil Tampilan GUI\_Mahasiswa.java

## Tugas Rumah 1 : Implementasi Enkapsulasi di 3 *Class*

Judul Judul : Informasi Wisata

Tema : Sistem Informasi

*Diagram Class* (List.java):

**Class**

**Attribute**

**Method constructor**

|  |  |
| --- | --- |
| List | |
| * Nama * Kota * Deskripsi * Harga * Jumlah * Total | String  String  String  Int  Inr  Int |
| * cetakNama, (String) * cetakKota(String) * cetakDeskripsi(String) * dataNama(Void) * dataKota(Void) * dataDeskripsi(Void) | |

*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):

****

### Desain GUI\_Wisata.java

#### 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:

****

### 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 yaiu 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):

|  |  |
| --- | --- |
| Login | |
| * Username * Password * Nama | String  Int  String |
| * Login() * getUsename() * setUsername(String) * getPassword() * setPassword(String) * CelLogin(String) | |

**Class**

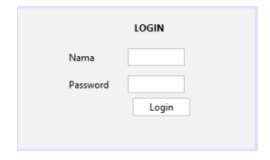
**Attribute**

**Method constructor**

*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):

****

### Desain GUI\_Login.java

#### 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:

****

### 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

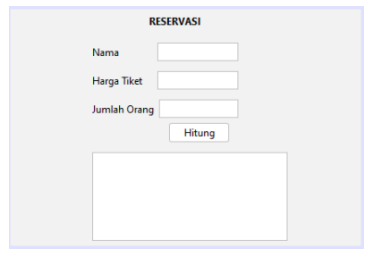
*Diagram Class* (Hitung.java):

|  |  |
| --- | --- |
| Hitung  **Class**  **Attribute**  **Method constructor** | |
| * NamaR | String |
| * Total (Int) * dataNamaR(Void) | |

*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):

****

### Desain GUI\_Reservasi.java

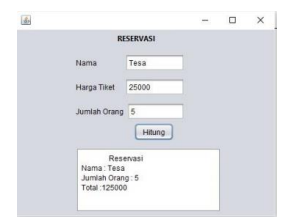
#### Properti Desain GUI\_Reservasi.java

|  |  |  |  |
| --- | --- | --- | --- |
| **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:

****

### 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.

## Tugas Rumah 2:

Tema :Sistem Informasi

Judul : Implementasi *Overrloading* dan *overriding*

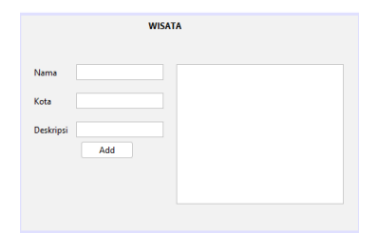
*Diagram Class* (List.java):

|  |  |
| --- | --- |
| List  **Class**  **Attribute**  **Method constructor** | |
| + Nama  + Kota  + Deskripsi  + Harga  + Jumlah  + Total | String  String  String  String  Int  Int |
| + dataNama (Void)  + dataKota (Void)  + dataDeskripsi (Void)  + cetakNama (String)  + cetakKota (String)  + cetakDeskripsi (String) | |

*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):

****

### Desain GUI\_Wisata.java

#### 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:

****

### 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)).

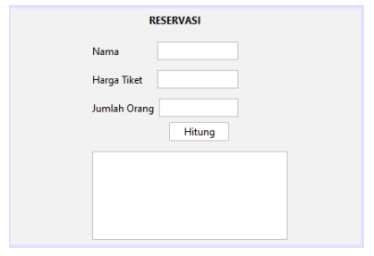
*Diagram Class* *Overring* (Class Hitung):

|  |  |
| --- | --- |
| Hitung.java  **Class**  **Attribute**  **Method constructor** | |
| + NamaR () | String |
| + Total (Int)  +dataNamaR (Void) | |

*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):

****

### Desain GUI\_Reservasi.java

#### 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:

****

### 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.

.

## Kesimpulans

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.

# ABSTRACT, POLIMORFISME, DAN INTERFACE

|  |  |  |
| --- | --- | --- |
| **Jumlah Pertemuan** | **:** | **2 x 60 menit** |
| **Tujuan Praktikum** | **:** | 1. Mampu menerapkan konsep dasar pemrograman web. 2. Mampu mengetahui fungsi tag-tag pada html. 3. Mampu membuat halaman web menggunakan html. |
| **Alat / bahan** | **:** | 1. Seperangkat *computer*. 2. Perangkat lunak: VS Code, XAMPP. 3. Modul Praktikum WEB 2023. |

## Landasan Teori

1. 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:

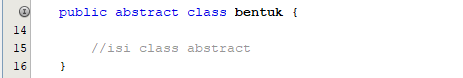
Akses\_modifier **abstract class** namaClassAbstrak

{

.........// isi

}

Contoh penerapan:



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:



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.
6. 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*).

* 1. *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*.
  2. *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*:

* + - * 1. *Static* *Polimorfisme*

Pada static polimorfisme mengunakan 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));

}

}

* + - * 1. *Dynamic Polymorphism*

Polimorfisme dinamik menggunakan konsep method overrdiding 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();

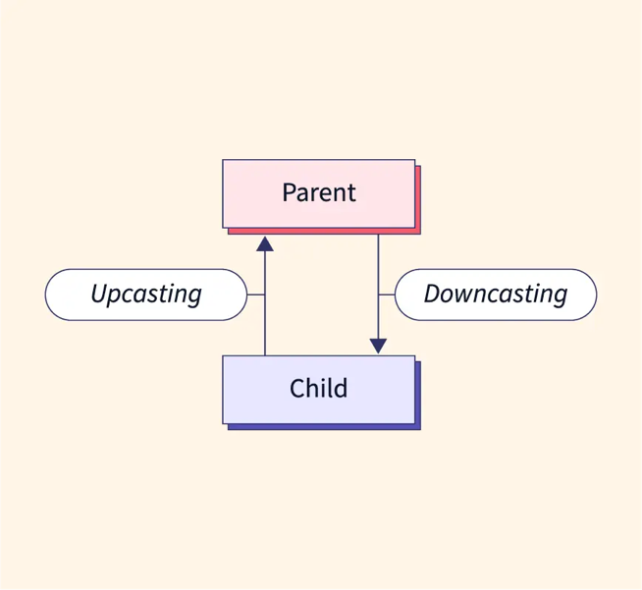
}

}

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 overriden 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 diinstasiasi sebagai objek dari Sub\_Class). Begitu juga pada Sub\_Class2.
   * + 1. 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 jens 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.



### Konsep Updasting dan Downcasting

1. 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() |

1. 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; |

1. 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-Overide 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.

* + - 1. 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:

* + - * 1. Seluruh *method* yang di deklarasikan pada *Interface* pasti bersifat *public* dan *abstract.*
        2. *Variable* selalu bersifat *public*, *final* dan *static.*

## Langkah – Langkah Praktikum

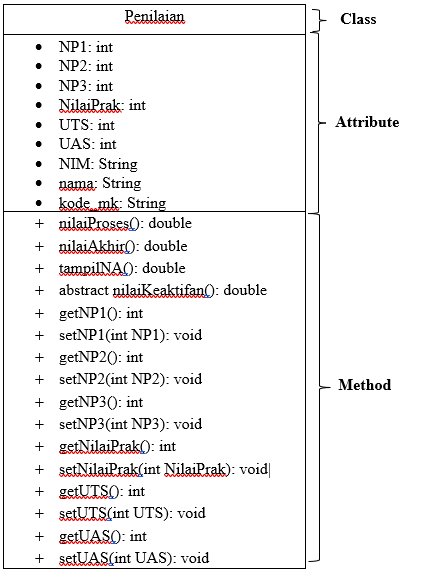
1. Buka Aplikasi *Netbeans*.
2. Buat *Class*.
3. Memberi *script* pada kelas tersebut.
4. Buat *form*.
5. Memberi *script* pada *form* tersebut.
6. Menjalankan program.

## Tugas Praktikum 1 :

Membuat *Class* Penilaian dan KeaktifanMahasiswa dan GUI\_Penilaian

Judul : Mendesain ulang GUI\_Penilaian

*Diagram Class* (Penilaian):

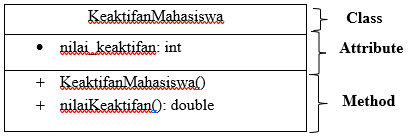


### *Diagram Class* Penilaian

*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):

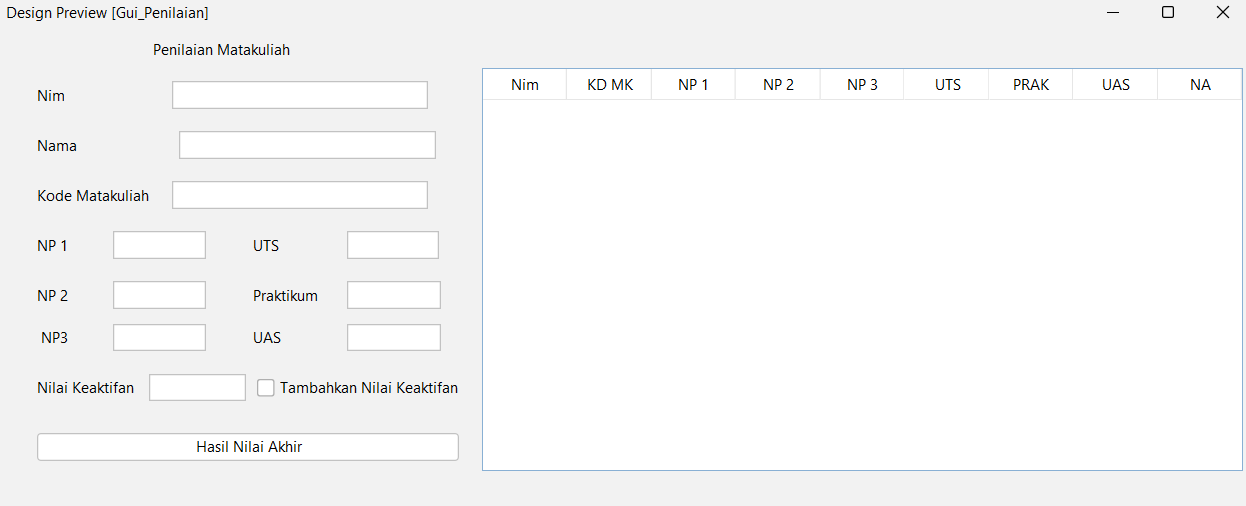


### *Diagram Class* KeaktifanMahasiswa

*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) + nilaiAkhir());  }  } |

Desain *form* (GUI\_Penilaian.java):

* + - * 1. ****

### Desain Gui\_ Penilaian.java

#### 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);  } |

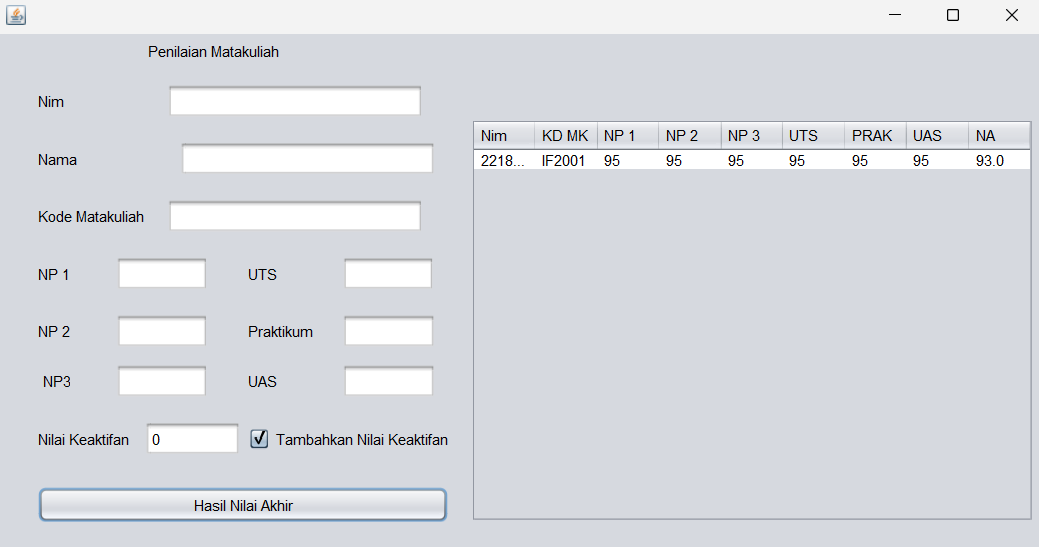
*Source code Clear* pada GUI:

|  |
| --- |
| 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:

****

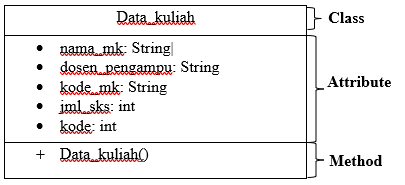
### Hasil Tampilan GUI\_Penilaian.java

## Tugas Praktikum 2 :

Membuat *Class* Data\_kuliah dan GUI\_Matkul

Judul : GUI\_Matkul

*Diagram Class* (Data\_Kuliah):

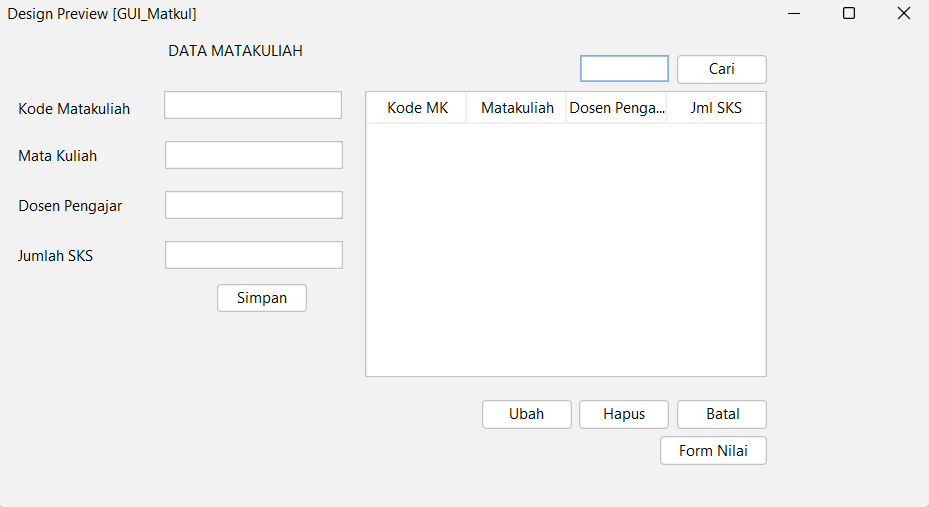


### *Diagram Class* Data\_kuliah

*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):

****

### Desain Gui\_Matkul.java

#### 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(); |

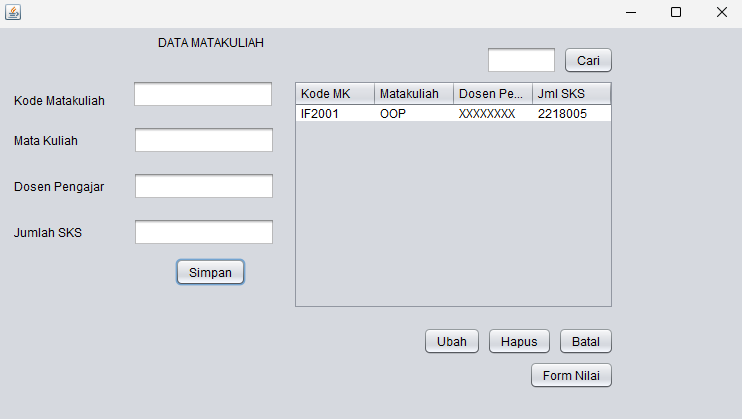
*Source code button* Hapus pada GUI:

|  |
| --- |
| 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:

****

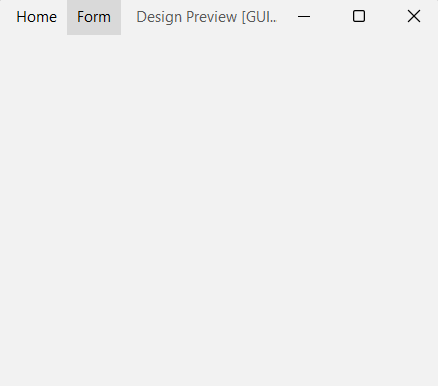
### Hasil Tampilan GUI\_Matkul.java

## Tugas Praktikum 3 :

Membuat GUI\_MenuUtama

Judul : Membuat GUI Menu Utama

Desain *form* (GUI\_MenuUtama.java):

****

### Desain Gui\_MenuUtama.java

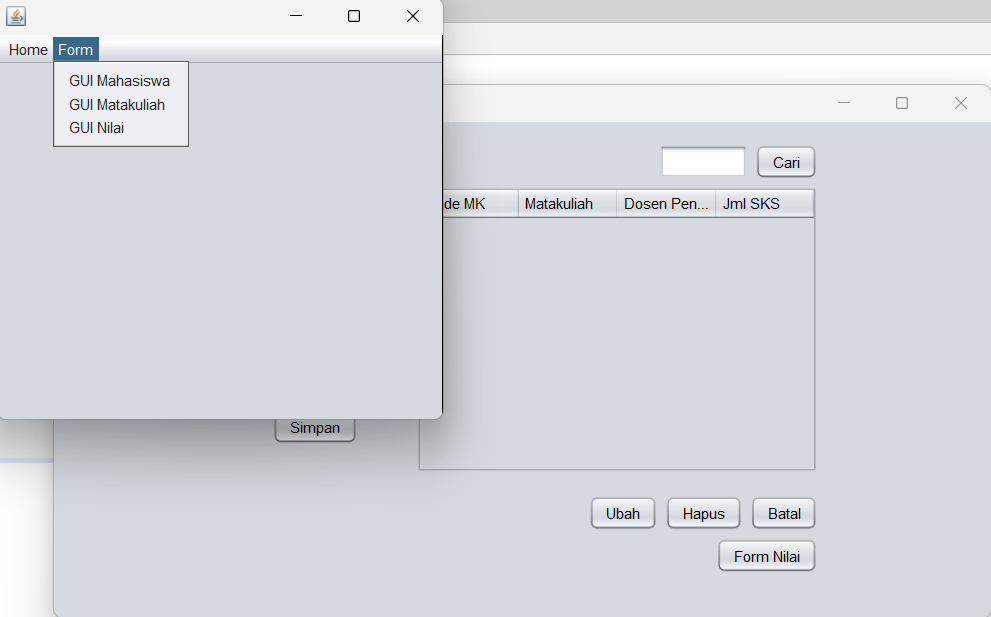
#### Properti Desain GUI\_MenuUtama.java

|  |  |  |  |
| --- | --- | --- | --- |
| **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:

****

### Hasil Tampilan GUI\_MenuUtama.java

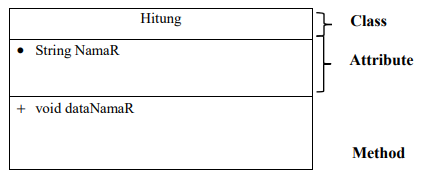
## Tugas Rumah 1 :

Implementasi Class Abstract

Judul : Informasi Wisata

Tema :Informasi Wisata

*Diagram Class* (Hitung.java):



### 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):



### Desain Gui\_Reservasi.java

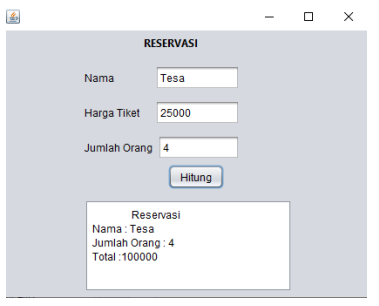
#### Properti Desain GUI\_Reservasi.java

|  |  |  |  |
| --- | --- | --- | --- |
| **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:

****

### Hasil Tampilan GUI\_Reservasi.java

Analisa:

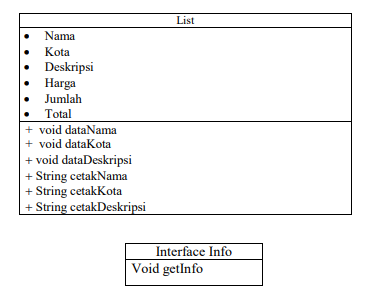
Program diatas merupakan implementasi dari class hitung yang di hubungkan gui reservasi.Tampil yang muncul nama pemesan,harga tiket dan jumlah orang.Setelah itu akan muncul di memo sesuai apa yang di inputkan.Pada class ini yang di jadikan private yaitu Harga.Pada bagian ini juga kita buat class baru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

## Tugas Rumah 2 :

Tema : Sistem Informasi

Judul : Menerapkan Konsep Interface

*Diagram Class* (Hitung.java):



### Hasil Tampilam Diagram Hitung

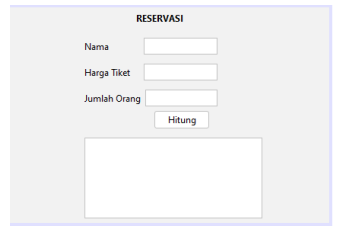
Source code Object Class (Hitung.java):

|  |
| --- |
| 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):



### Desain Gui\_Reservasi.java

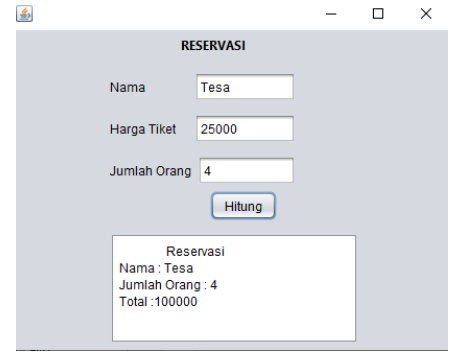
#### Properti Desain GUI\_Login.java

|  |  |  |  |
| --- | --- | --- | --- |
| **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:



### 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.

## Tugas Rumah 3 :

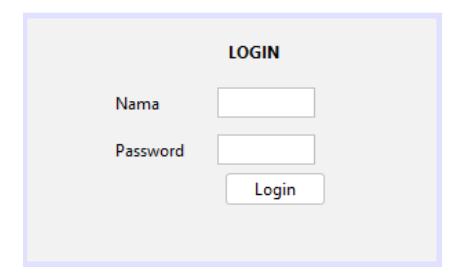
Tema : Sistem Informasi

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):



### Desain Gui\_Daftar.java

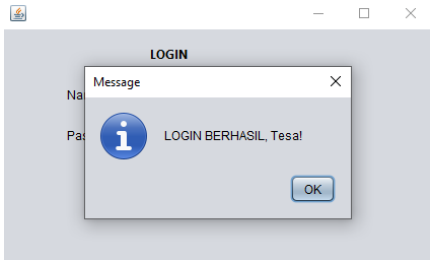
#### Properti Desain GUI\_Login.java

|  |  |  |  |
| --- | --- | --- | --- |
| **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

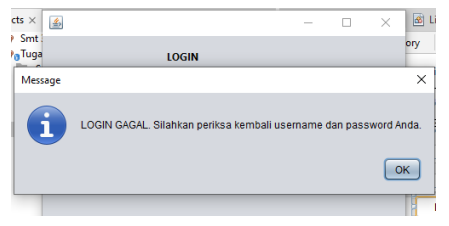
|  |
| --- |
| private void btnLoginActionPerformed(java.awt.event.ActionEvent evt) {  // TODO add your handling code here:  Login l = new Login();  String username = txtNama.getText();  String password = txtPass.getText();  boolean Authenticated = l.CekLogin(username, password);  if (Authenticated)  {  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:

****

### Hasil Tampilan Login Berhasil.java

Hasil Tampilan:

****

### 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 mengguanakan abstract.Pada gui wisata terjadi masalah yang membuat yang membuat memo tidak keluar.

## 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.

# EXCEPTION DAN PENGENALAN DATABASE

|  |  |  |
| --- | --- | --- |
| **Jumlah Pertemuan** | **:** | **2 x 60 menit** |
| **Tujuan Praktikum** | **:** | 1. Praktikan mampu memahami tentang *exception* dan menerapkan *exception* ke dalam program. 2. Praktikan mampu membuat *database* dengan menggunakan MySQL. 3. Praktikan mampu mengkoneksikan *database* dengan program yang sudah dibuat sebelumnya. 4. Praktikan mampu memberi akses *insert, update, delete,* dan *searching* (memasukkan, merubah, menghapus, dan pencarian). |
| **Alat / bahan** | **:** | 1. Seperangkat *computer*. 2. Perangkat lunak: *Netbeans*. 3. Modul Praktikum *OOP* 2022. |

## Landasan Teori

1. Interface
2. 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* :

1. *Exception* bisa membawa sangat banyak informasi tentang kesalahan yang terjadi (deskripsi,stack race, lokasi baris kode dll)
2. *Exception* sudah didukung oleh banyak IDE modern yang ada di pasar
3. Mekanisme *Exception* terintegrasi secara baik dengan OOP
4. 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:

1. *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*
2. *Uncecked* *Exception*, adalah *exception* yang terjadi saat *execution* *time*. *Error* ini *terjadi* dalam lingkup internal dari aplikasi, biasanya terjadi karena salah penulisan kode. Contoh: *NullPointerException*.
3. *Error*, adalah *exception* yang diluar kendali *user* atau *programmer*. *Eror* 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*, terdappat 3 *block* kode yaitu *try*, *catch* dan *finally.*

1. *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  } |

1. *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 } |

1. *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 } |

1. Pengenalan Database
2. 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.

1. 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.

1. 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.

## 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.

## Tugas Praktikum 1 :

Membuat *Class* KeaktifanMahasiswa, Penilaian dan GUI\_Penilaian

Judul : Mendesain ulang GUI\_Penilaian + Method Batal

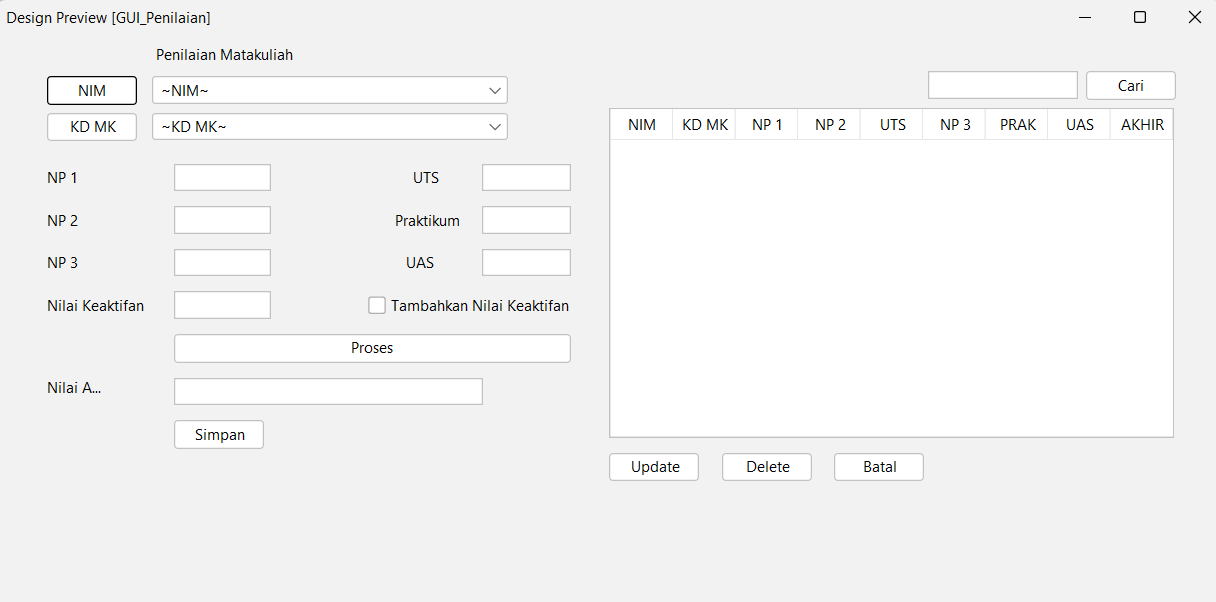
*Source code Object Class:*

|  |
| --- |
| 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) + nilaiAkhir());  }  } |

Desain *form* (GUI\_ Penilaian.java):

****

### Desain Gui\_ Penilaian.java

#### Properti Desain GUI\_Penilaia.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 |

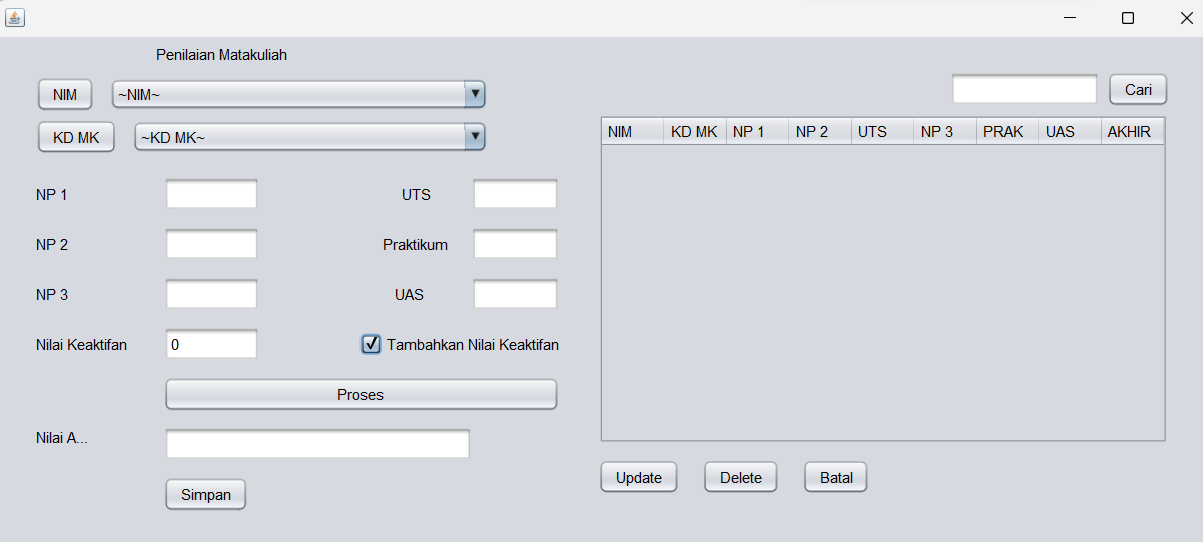
*Source code* dibawah *Method* Konstruktor pada GUI:

|  |
| --- |
| 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();  } |

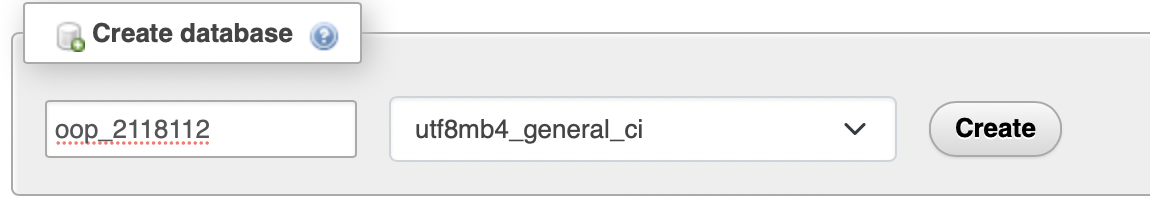
Hasil Tampilan:

****

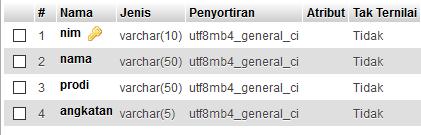
### Hasil Tampilan GUI\_Penilaian.java

## Tugas Praktikum 2 :

Mengkoneksikan GUI\_Mahasiswa Kedalam Database

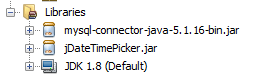
**

*Structure* tb\_mahasiswa pada database oop\_NIM



### Membuat table(tb\_mahasiswa)

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



### Menambahkan *Library Connection*

*Source code* pada GUI (import *Library*):

|  |
| --- |
| import java.sql.Connection;  import java.sql.*Driver*Manager;  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\_2218005?user=root&password=");  } catch (ClassNotFoundException ex) {  Logger.getLogger(GUI\_Mahasiswa.class.getName()).log(Level.SEVERE, null, ex);  } catch (SQLException e) {  Logger.getLogger(GUI\_Mahasiswa.class.getName()).log(Level.SEVERE, null, e);  } catch (Exception es) {  Logger.getLogger(GUI\_Mahasiswa.class.getName()).log(Level.SEVERE, 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();  } |

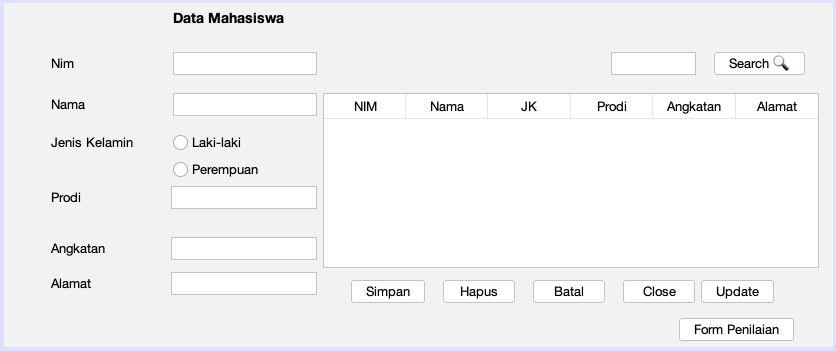
*Source code method cari :*

|  |
| --- |
| 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:



### Desain Form Gui\_Mahasiswa

#### 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();  } |

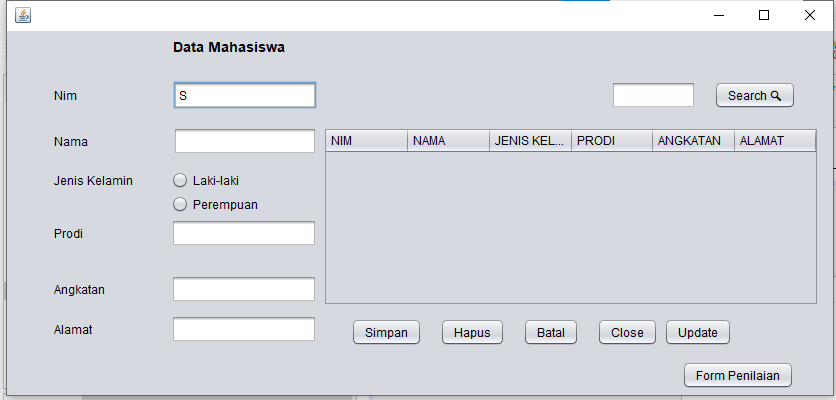
*Source code* pada tombol cari*:*

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

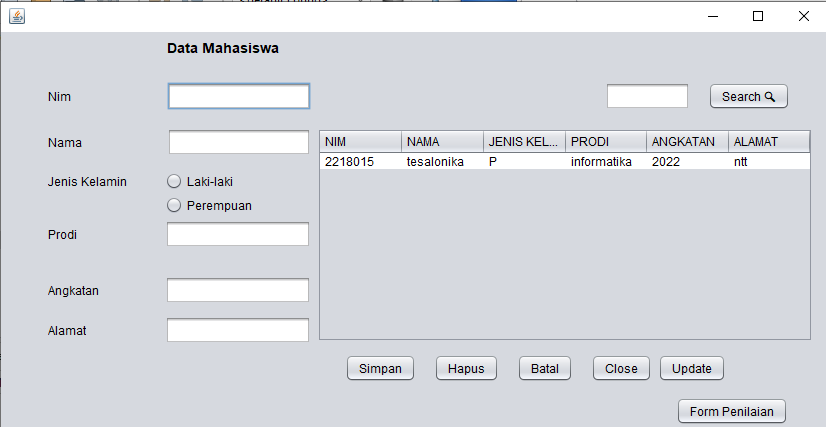
*Source code* pada tombol batal*:*

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

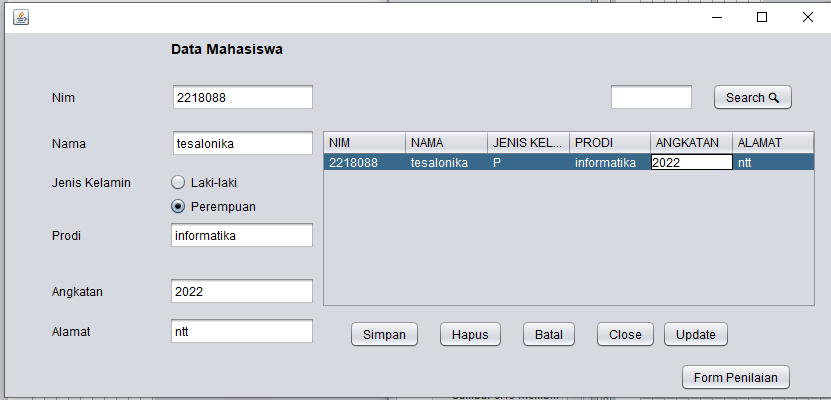
Tampilan Hasil:



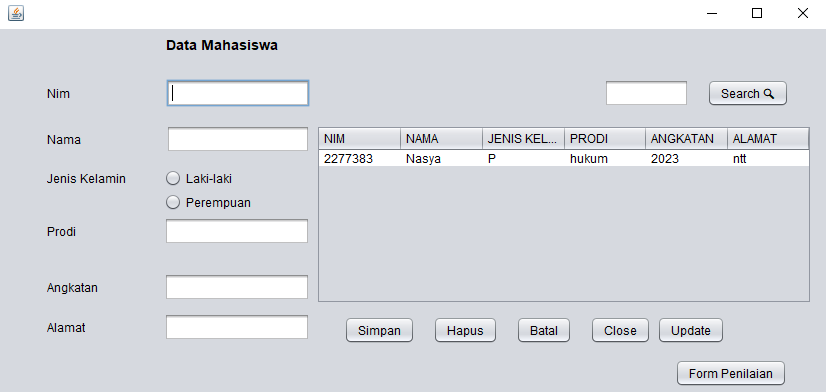
### Tampilan Hasil Method tampil() Gui\_Mahasiswa



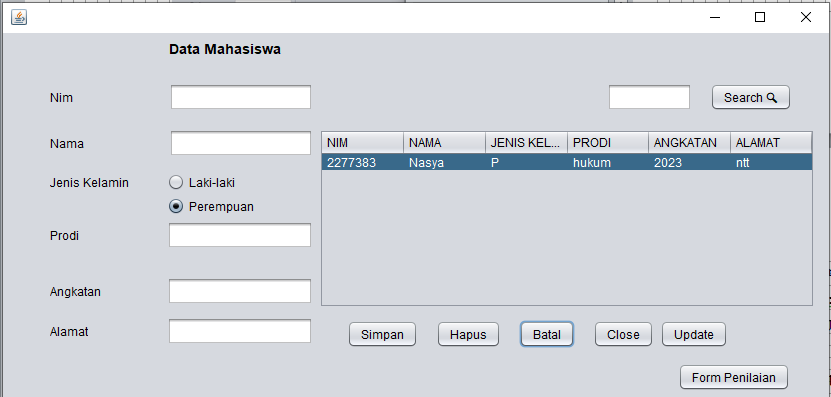
### Tampilan Hasil Method tambah() Gui\_Mahasiswa



### Tampilan Hasil Method ubah() Gui\_Mahasiswa



### Tampilan Hasil Method hapus() Gui\_Mahasiswa



### Tampilan Hasil Method batal() Gui\_Mahasiswa

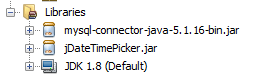
## Tugas Praktikum 3 :

Mengkoneksikan GUI\_Nilai Kedalam Database

****

### Membuat table (tb\_nilai)

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



### Menambhakan *Library Connection*

*Source code* pada GUI (import *Library*):

|  |
| --- |
| import java.sql.Connection;  import java.sql.*Driver*Manager;  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\_2218005?user=root&password=");  } catch (ClassNotFoundException ex) {  Logger.getLogger(GUI\_Penilaian.class.getName()).log(Level.SEVERE, null, ex);  } catch (SQLException e) {  Logger.getLogger(GUI\_Penilaian.class.getName()).log(Level.SEVERE, null, e);  } catch (Exception es) {  Logger.getLogger(GUI\_Penilaian.class.getName()).log(Level.SEVERE, 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();  } |

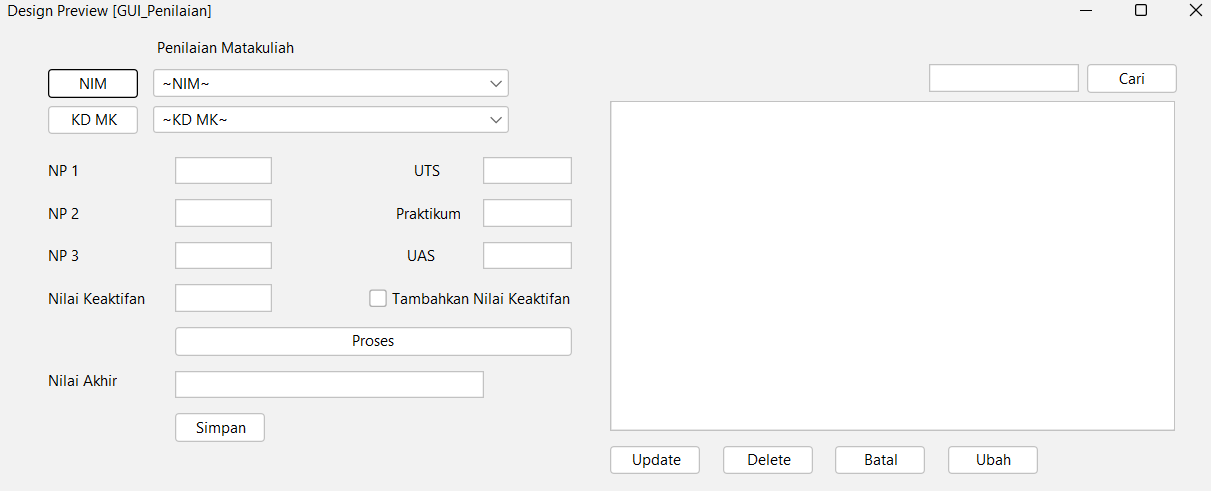
*Source code method delete :*

|  |
| --- |
| 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:

****

### Desain Form Gui\_Nilai

#### 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();  } |

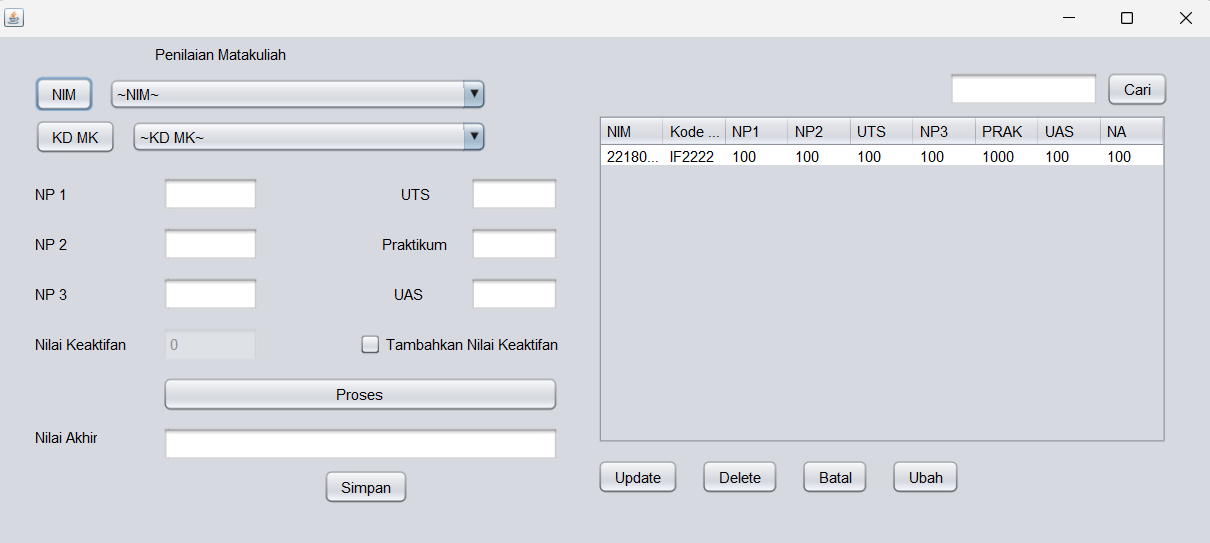
*Source code* pada tombol cari*:*

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

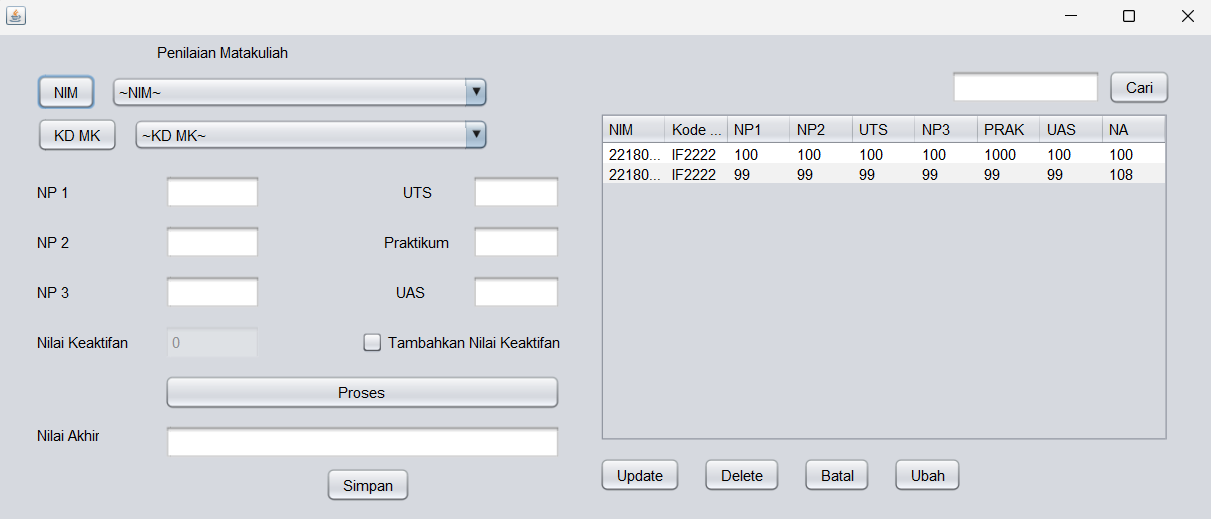
*Source code* pada tombol batal*:*

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

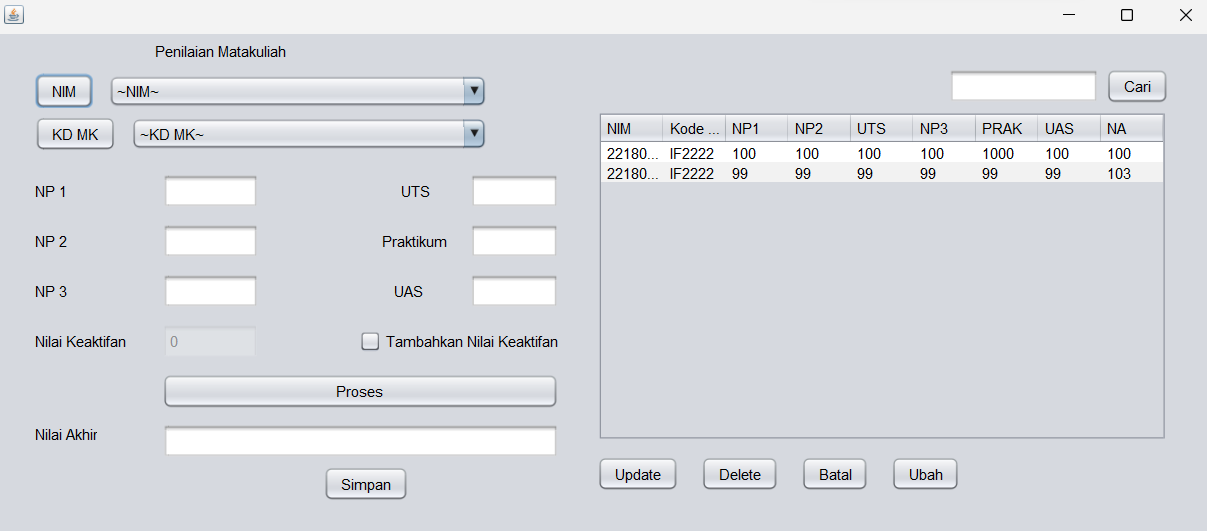
Tampilan Hasil:



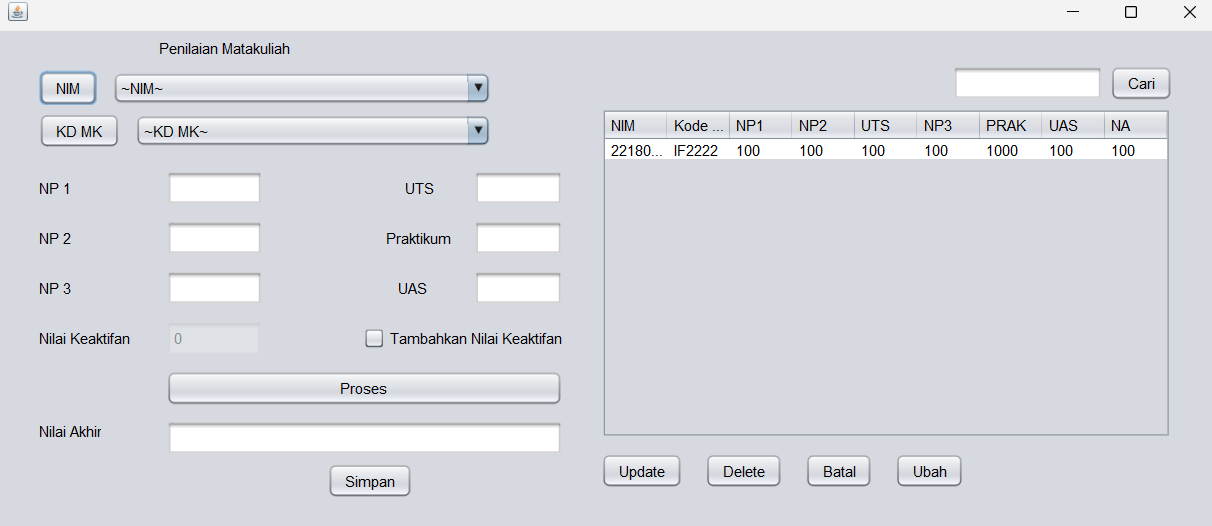
### Tampilan Hasil Method tampil() Gui\_Mahasiswa



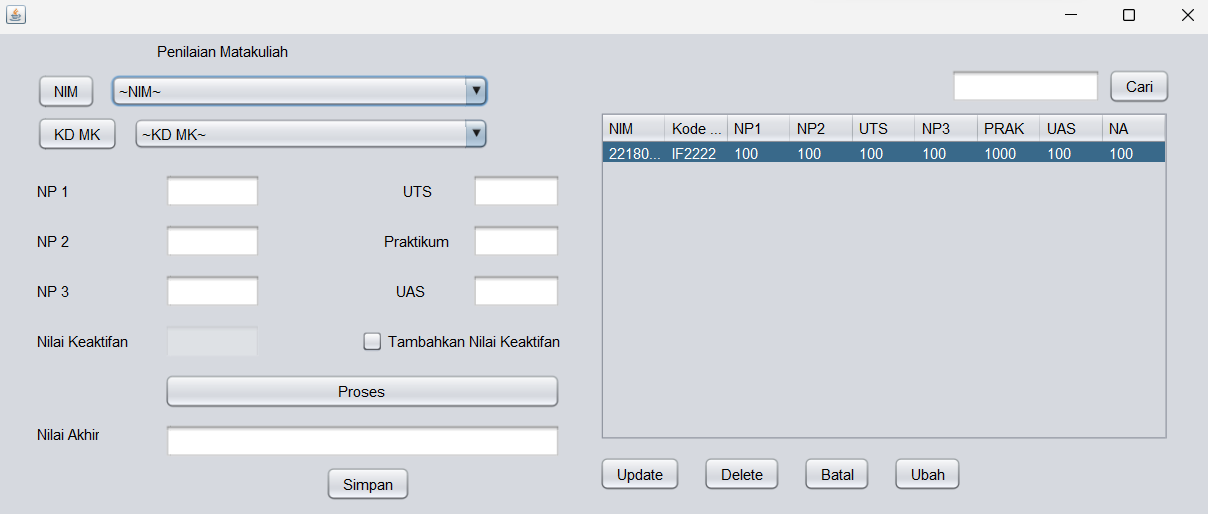
### Tampilan Hasil Method tambah() Gui\_Mahasiswa



### Tampilan Hasil Method ubah() Gui\_Mahasiswa



### Tampilan Hasil Method hapus() Gui\_Mahasiswa



### Tampilan Hasil Method batal() Gui\_Mahasiswa

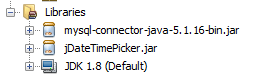
## Tugas Praktikum 4 :

Mengkoneksikan GUI\_Matkul Kedalam Database

****

### Membuat table (tb\_matkul)

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



### Menambhakan *Library Connection*

*Source code* pada GUI (import *Library*):

|  |
| --- |
| import java.sql.Connection;  import java.sql.*Driver*Manager;  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\_2218005?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();  } |

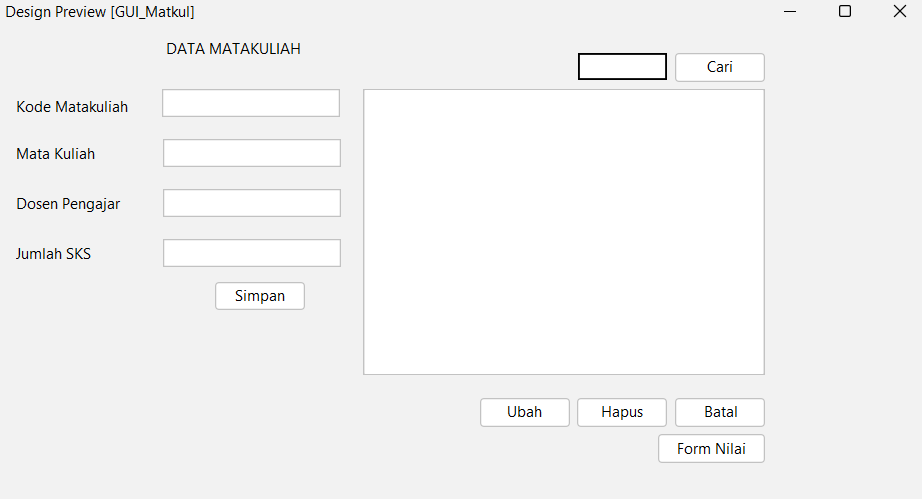
*Source code method hapus :*

|  |
| --- |
| 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:

****

### Desain Form Gui\_Matkul

#### 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();  } |

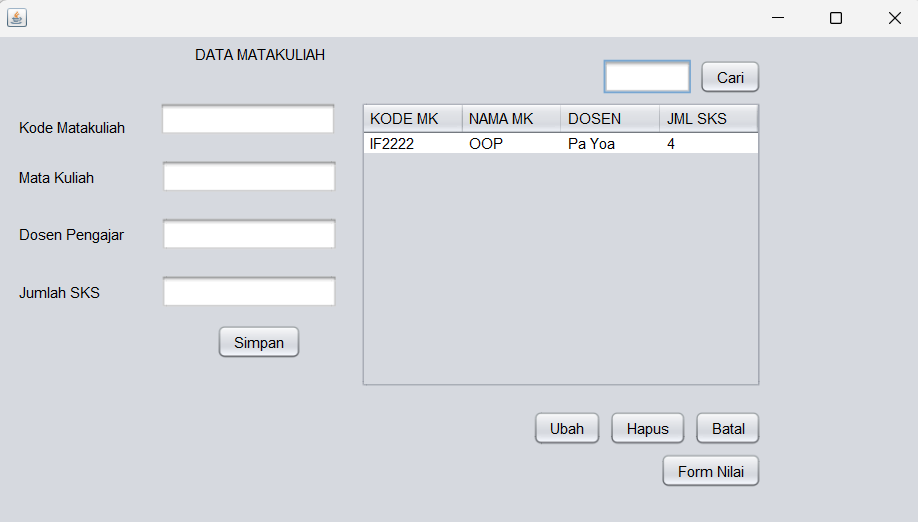
*Source code* pada tombol cari*:*

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

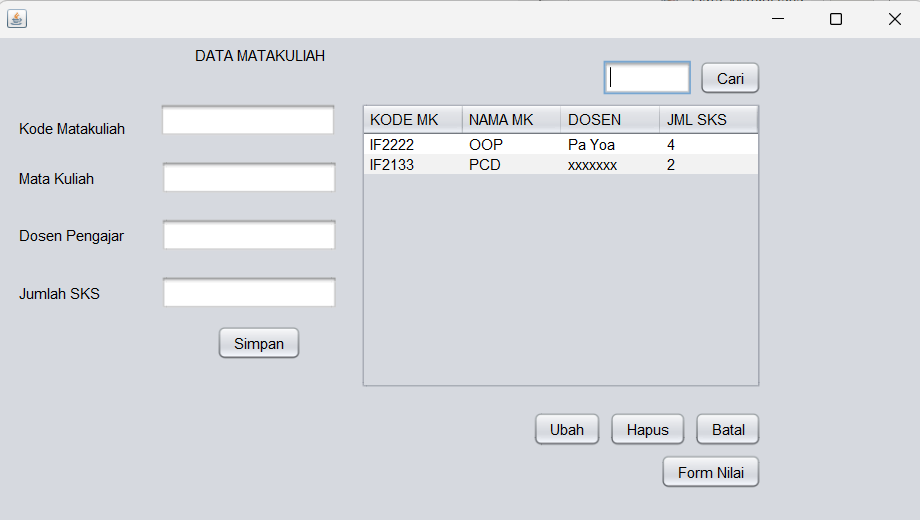
*Source code* pada tombol batal*:*

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

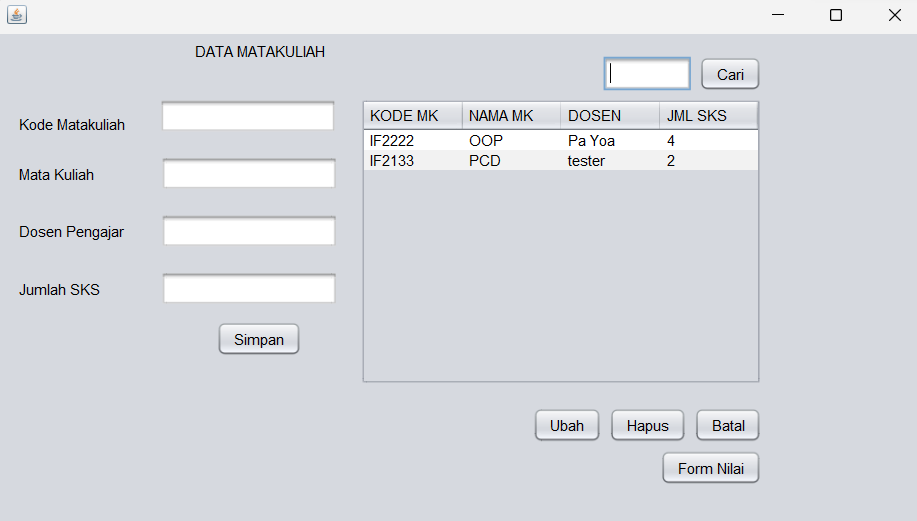
Tampilan Hasil:



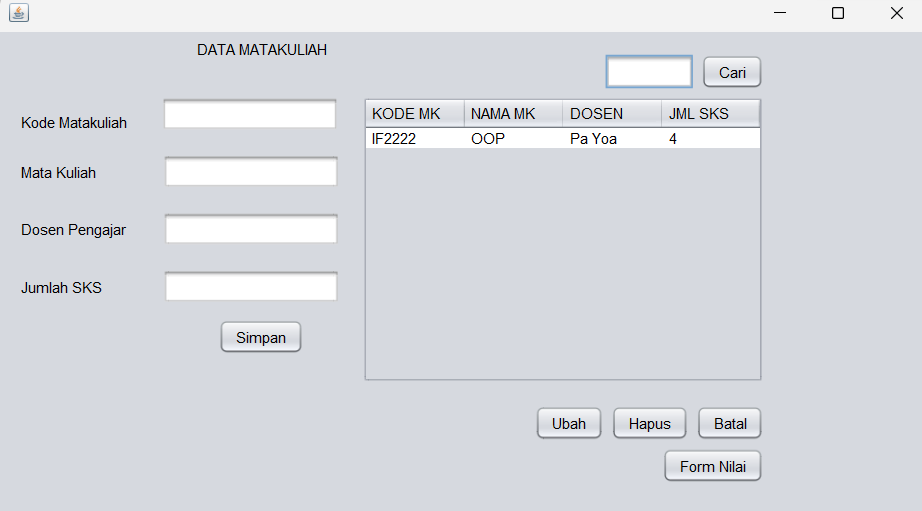
### Tampilan Hasil Method tampil() Gui\_Mahasiswa



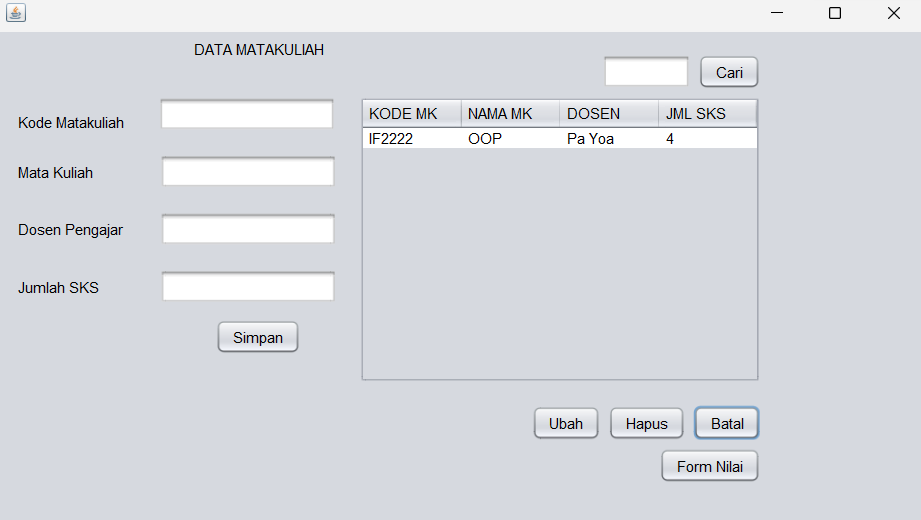
### Tampilan Hasil Method tambah() Gui\_Mahasiswa



### Tampilan Hasil Method ubah() Gui\_Mahasiswa



### Tampilan Hasil Method hapus() Gui\_Mahasiswa



### Tampilan Hasil Method batal() Gui\_Mahasiswa

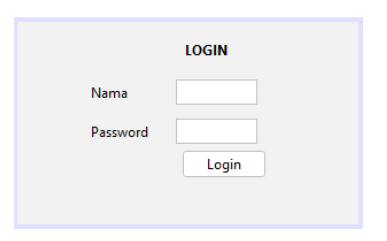
## Tugas Rumah 1 :

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):



### Tampilan Desain Daftar.java

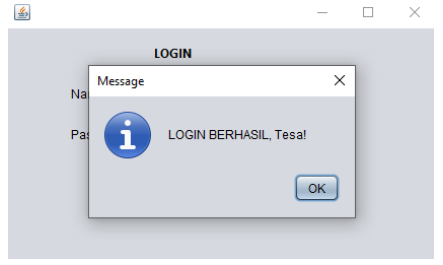
#### Properti Desain GUI\_ Matkul (Database)

|  |  |  |  |
| --- | --- | --- | --- |
| **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 bttnLogin :

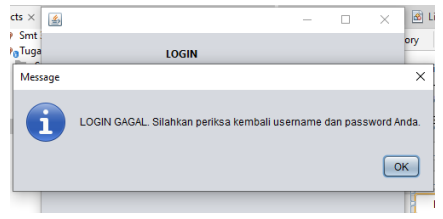
|  |
| --- |
| private void btnLoginActionPerformed(java.awt.event.ActionEvent evt) {  // TODO add your handling code here:  Login l = new Login();  String username = txtNama.getText();  String password = txtPass.getText();  boolean Authenticated = l.CekLogin(username, password);  if (Authenticated)  {  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:

****

### Tampilan Login Berhasil.java……….

Tampilan Hasil:

****

### 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.

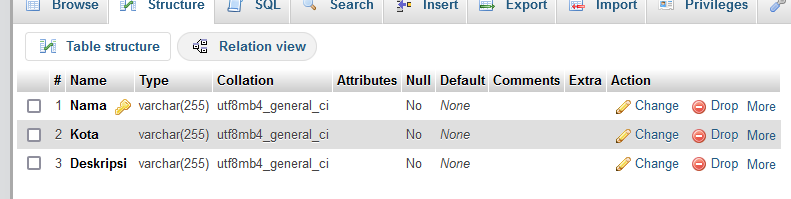
## Tugas Rumah 2 :

Menerapkan CRUD Pada GUI\_Wisata.java



### Membuat database oop\_2218015

*Structure tb\_* list pada database oop\_2218015



### Membuat table(tb\_list)

*Source code* pada GUI:

|  |
| --- |
| import java.sql.Connection;  import java.sql.*Driver*Manager;  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()); |

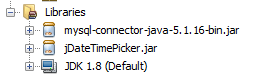
*Source code method* ubah *:*

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

*Source code method* hapus *:*

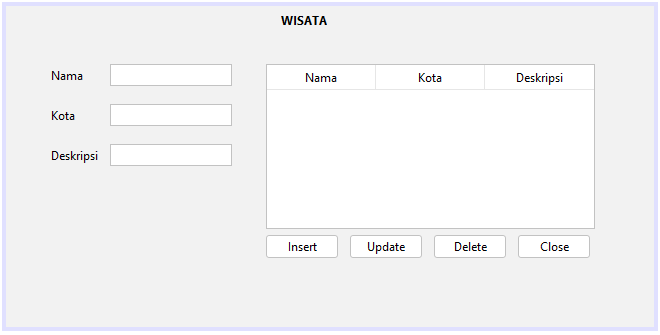
|  |
| --- |
| String sql = "DELETE FROM list WHERE nama=?";  PreparedStatement preparedStatement = connection.prepareStatement(sql); } |

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



### Menambahkan *Library Connection*

Desain *form* GUI\_Wisata:

****

### Desain Form GUI\_Wisata

#### 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);  }  } |

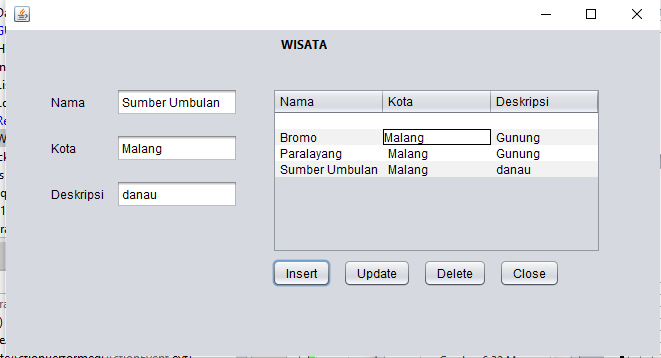
*Source code* pada tombol Ubah*:*

|  |
| --- |
| 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:

****

### 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.

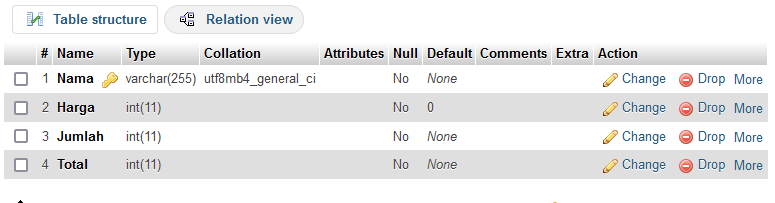
## Tugas Rumah 3 :

Menerapkan CRUD Pada GUI\_Reservasi.java

Membuat Database oop\_2218015



*Structure* tb\_Reservasi pada database oop\_2218015



### Membuat table(tb\_Reservasi)

*Source code* pada GUI:

|  |
| --- |
| import java.sql.Connection;  import java.sql.*Driver*Manager;  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); |

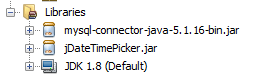
*Source code method* ubah *:*

|  |
| --- |
| 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...**



### Menambahkan *Library Connection*

Desain *form* GUI\_Reservasi.java

****

### Desain Form GUI\_Reservasi

#### 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();  }  } |

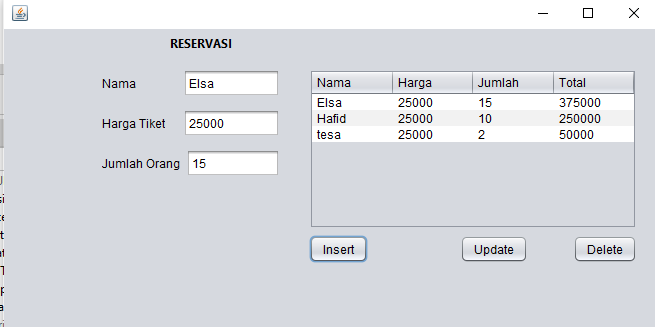
*Source code* pada tombol Ubah*:*

|  |
| --- |
| 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:

****

### Tampilan Hasil Method Tampi() GUI\_.......

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 dalete untuk keluar dari tampilan gui tersebut.

# KESIMPULAN

1. Class merupakan rancangan dari sebuah objek yang mendefenisikan 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.

**LEMBAR ASISTENSI PRAKTIKUM OBJECT ORIENTED PROGRAMING**

**SEMESTER GENAP TAHUN AKADEMIK 2023/2024**

Foto latar merah

Almamater

**Nama : Tesalonika Dua Nurak**

**NIM : 2218015**

**Kelompok : 8**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **No.** | **Tanggal** | **Asistensi** | | | | **Paraf** |
| **Konsep** | | **Program** | **Hasil Akhir** |
| 1 |  | **Instruktur** | Konsep Dasar OOP  *Konstructor*, *Inheritance*  *Enkapsulasi*  *Overloading*, *Overriding*  *Abstract* , *Polimorfisme*  *Interface*  CRUD Java  Excception |  |  |  |
| 2 |  | **Dosen** |  |  |  |  |
| **Batas Akhir:** | | | | | | |

**Asisten,**

**(………………………………………….)**

**NIM :**

Malang, 2024

**Dosen Pembimbing,**

**(……………………………………………..)  
NIP.P :**