**LAPORAN PRAKTIKUM**

OBJECT ORIENTED PROGRAMMING

**SEMESTER GANJIL TAHUN AKADEMIK 2022/2023**



## Disusun oleh :

### **Nama** : Firman Frezy Pradana

**NIM** 2118112

**Prodi** : TEKNIK INFORMATIKA S1

## Kelompok 25

**LEMBAR PERSETUJUAN PRAKTIKUM OBJECT ORIENTED PROGRAMMING SEMESTER GANJIL TAHUN AKADEMIK 2022/2023**



#### Disusun Oleh

### NAMA : Firman Frezy Pradana NIM 2118131

### PRODI : TEKNIK INFORMATIKA S-1

### Mengetahui, Kepala Laboratorium

### Pemrograman Komputer

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

### Menyetujui,

### Dosen Pembimbing

## (Ahmad Fahrudi Setiawan, S.Kom, M.T.) NIP.P.1031500497

# KATA PENGANTAR

### Dengan memanjatkan puji syukur kehadirat Allah SWT, karena atas berkah rahmat dan karunia-Nya sehingga penulis dapat menyelesaikan Laporan Praktikum Mata Kuliah *Object Oriented Programming*, 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 Fahrudi Setiawan, S.Kom, M.T. selaku dosen pembimbing Praktikum *Object Oriented Programming*.
2. Bapak Yoseph Agus Pranoto, ST, MT. dan Dedy Rudhistisar,S.Kom.M.Cs selaku dosen mata kuliah *Object Oriented Programming*.
3. Bapak Ahmad Faisol, ST, MT selaku Ketua Pelaksana Praktikum *Object Oriented Programming* Program Studi Teknik Informatika ITN Malang.

### Instruktur Lab.Pemrograman Komputer Teknik Informatika yang telah memberi petunjuk kepada penulis selama pelaksanaan praktikum.

### 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 2022

### Penulis

# DAFTAR ISI

[KATA PENGANTAR i](#_TOC_250003)

[DAFTAR ISI ii](#_TOC_250002)

[DAFTAR GAMBAR iv](#_TOC_250001)

[DAFTAR TABEL viii](#_TOC_250000)

### BAB I PENDAHULUAN..................................................................................... I-1

### I.1 Latar Belakang ............................................................................................ I-1

### I.2 Rumusan Masalah ....................................................................................... I-2

### I.3 Tujuan.......................................................................................................... I-2

### I.4 Manfaat........................................................................................................ I-2 BAB II KONSEP DASAR OOP .........................................................................II-1

### II.1 Landasan Teori..........................................................................................II-1

### II.2 Langkah – Langkah Praktikum .................................................................II-3

### II.3 Tugas Praktikum ke-1: ..............................................................................II-4

### II.4 Tugas Rumah 1 : .......................................................................................II-7

### II.5 Kesimpulan .............................................................................................II-11 BAB III KONSTRUKTOR DAN INHERITANCE .......................................... III-1 III.1 Landasan Teori ...................................................................................... III-1

### III.2 Langkah – Langkah Praktikum.............................................................. III-3

### III.3 Tugas Praktikum ke-1:........................................................................... III-3

### III.4 Tugas Praktikum ke-2 :.......................................................................... III-6

### III.5 Tugas Rumah 1 : .................................................................................. III-10

### III.6 Tugas Rumah 2 : .................................................................................. III-15

### III.7 Kesimpulan .......................................................................................... III-22 BAB IV ENCAPSULASI, OVERIDDING DAN OVERLOADING ............... IV-1 IV.1 Landasan Teori ...................................................................................... IV-1

### IV.2 Langkah – Langkah Praktikum ............................................................. IV-4

### IV.3 Tugas Praktikum ke-1: .......................................................................... IV-4

### IV.4 Tugas praktikum ke-2:........................................................................... IV-8

### IV.5 Tugas Rumah 1 :.................................................................................. IV-11

### IV.6 Tugas Rumah 2 :.................................................................................. IV-22

### IV.7 Kesimpulan.......................................................................................... IV-34

### BAB V ABSTRACT, POLIMORFISME DAN EXCEPTION HANDLING V-1

### V.1 Landasan Teori .........................................................................................V-1

### V.2 Langkah – Langkah Praktikum.................................................................V-7

### V.3 Tugas Praktikum 1: ...................................................................................V-7

### V.4 Tugas Praktikum 2 : ................................................................................V-10

### V.5 Tugas praktikum 3 : ................................................................................V-13

### V.6 Tugas Rumah 1 : .....................................................................................V-14

### V.7 Tugas Rumah 2: ......................................................................................V-26

### V.8 Tugas Rumah 3: ......................................................................................V-35

### V.9 Kesimpulan .............................................................................................V-45 BAB VI INTERFACE DAN PENGENALAN DATABASE .......................... VI-1 VI.1 Landasan Teori ...................................................................................... VI-1 VI.2 VI.2. Langkah – Langkah Praktikum .................................................. VI-4

### VI.3 VI.3. Tugas Praktikum ke-1: ................................................................. VI-5

### VI.4 Tugas Praktikum ke-2: .......................................................................... VI-7

### VI.5 Tugas Praktikum ke-3: ........................................................................ VI-14

### VI.6 Tugas Praktikum ke-4: ........................................................................ VI-21

### VI.7 Tugas Rumah ke-1:.............................................................................. VI-28

### VI.8 Tugas Rumah ke-2 :............................................................................. VI-33

### VI.9 Tugas Rumah ke-3 :............................................................................. VI-40

### VI.10 Tugas Rumah ke-4 :........................................................................... VI-47

### VI.11 Kesimpulan........................................................................................ VI-55

### BAB VII Kesimpulan ..................................................................................... VII-56

### DAFTAR PUSTAKA ..................................................................................... VII-58

# DAFTAR GAMBAR

### Gambar 2.1 Desain from GUI\_Mahasiswa.java I-5

### Gambar 2.2 Tampilan hasil GUI\_Mahasiswa.java ..............................................II-6

### Gambar 2.3 Desain GUI\_Barang.java(*object* & *method*)....................................II-8 Gambar 2.4 Hasil Tampilan GUI\_DataPenerimaBeasiswa.java .......................II-10 Gambar 3.1 Desain form GUI\_Matkul .............................................................. III-4

### Gambar 3.2 Hasil tampilan GUI\_Matkul .......................................................... III-5

### Gambar 3.3 Desain GUI\_Penilaian.java............................................................ III-7

### Gambar 3.4 Hasil Tampilan GUI\_Penilaian...................................................... III-9

### Gambar 3.5 Desain *form* GUI\_Barang(*Konstrucktor*) .................................... III-11

### Gambar 3.6 Hasil Tampilan GUI\_Barang.java(*konstructor*) .......................... III-14 Gambar 3.7 Class diagram *inheritance*............................................................ III-15

### Gambar 3.8 Desain *form (*GUI\_BarangMasuk.java) ....................................... III-16

### Gambar 3.9 Hasil tampilan GUI\_BarangMasuk.............................................. III-18 Gambar 3.10 Desain *form* GUI\_ DataPenerimaBeasiswa ............................... III-19 Gambar 3.11 Tampilan GUI\_BarangKeluar.................................................... III-22

### Gambar 4.1 Desain GUI\_Penilaian.java............................................................ IV-5

### Gambar 4.2 Hasil Tampilan GUI\_Penilaian...................................................... IV-7

### Gambar 4.3 Desain from GUI\_Mahasiswa.java ................................................ IV-9

### Gambar 4.4 Tampilan hasil GUI\_Mahasiswa.java .......................................... IV-10

### Gambar 4.5 Gambar class diagram Barang(Enkapsulasi) ............................... IV-11 Gambar 4.6 Desain GUI\_Barang(*Enkapsulasi*)............................................... IV-12

### Gambar 4.7 Hasil Tampilan GUI\_Barang.java(*Enkapsulasi*) ......................... IV-14

### Gambar 4.8 class diagram Barangmasuk(Enkapsulasi)................................... IV-14

### Gambar 4.9 Desain GUI\_BarangMasuk.java(Enkapsulasi) ............................ IV-15

### Gambar 4.10 Hasil GUI\_BarangMasuk.java(*Enkapsulasi*)............................. IV-18 Gambar 4.11 Class diagram barang keluar ...................................................... IV-18 Gambar 4.12 Desain GUI\_Barang Keluar.java(*Enkapsulasi*) ......................... IV-19

### Gambar 4.13 Hasil GUI\_barangkeluar..java(*Enkapsulasi*) ............................. IV-21 Gambar 4.14 Class diagram (*Overriding*) ....................................................... IV-22

### Gambar 4.15 Desain Gui\_barang.java(*overriding*) ......................................... IV-23

### Gambar 4.16 Tampilan Gui\_Barang(*Overriding*) ........................................... IV-26

Gambar 5.1 *Upcasting* dan *Downcasting* V-2

### Gambar 5.2 Hirarki Exception.............................................................................V-5

### Gambar 5.3 Desain GUI\_Matkul.........................................................................V-7

### Gambar 5.4 Desain GUI\_Matkul(*JTable*) ...........................................................V-8

### Gambar 5.5 Hasil Tampilan GUI\_Matkul ...........................................................V-9

### Gambar 5.6 Desain *Form* GUI\_Nilai.java.........................................................V-10

### Gambar 5.7 Hasil Tampilan GUI\_Nilai.java .....................................................V-12

### Gambar 5.8 Desain GUI\_MenuUtama ..............................................................V-13

### Gambar 5.9 Hasil Running GUI\_MenuUtama ..................................................V-13

### Gambar 5.10 class diagram (*Abstract*) ..............................................................V-14

### Gambar 5.11 Desain Form GUI\_Barang(*Abstract*) ...........................................V-15

### Gambar 5.12 Hasil Tampilan GUI\_Barang(*Abstract*) .......................................V-18

### Gambar 5.13 Hasil Tampilan GUI\_BarangMasuk(*Abstract*) ............................V-19

### Gambar 5.14 Tampilan hasil GUI\_BarangMasuk(*Abstract*) .............................V-21 Gambar 5.15 Desain GUI\_Barang.java(*TryCatch*) ...........................................V-27

### Gambar 5.16 Hasil tampilan GUI\_Barang(*TryCatch*).......................................V-28

### Gambar 5.17 Desain GUI\_BarangMasuk.java (*TryCatch*) ...............................V-30

### Gambar 5.18 Hasil Tampilan GUI\_BarangMasuk(*TryCatch*) ..........................V-31 Gambar 5.19 Desain GUI\_BarangKeluar(*TryCatch*) ........................................V-33 Gambar 5.20 Hasil Tampilan GUI\_BarangKeluar(*TryCatch*) ..........................V-34 Gambar 5.21 Class Diagram *Polimorfisme* .......................................................V-35

### Gambar 5.22 Desain form GUI\_Barang.java(*Polimorfisme*) ............................V-36

### Gambar 5.23 Hasil Tampila GUI\_Barang(*Polimorfisme*) .................................V-38

### Gambar 5.24 Desain GUI\_BarangMasuk(*Polimorfisme*)..................................V-39

### Gambar 5.25 Hasil Tampilan GUI\_BarangMasuk ............................................V-41

### Gambar 5.26 Desain GUI\_BarangKeluar(*Polimorfisme*)..................................V-43

### Gambar 5.27 Hasil Tampilan GUI\_BarangKeluar ............................................V-45

### Gambar 6.1 Class diagram implements interface langsung............................... VI-2 Gambar 6.2 Interface extends interface lain ...................................................... VI-3

### Gambar 6.3 Class extends class lain .................................................................. VI-3

### Gambar 6.4 Desain Awal GUI\_Nilai................................................................. VI-5

### Gambar 6.5 Desain Baru GUI\_Nilai.................................................................. VI-5

### Gambar 6.6 Hasil Tampilan GUI\_Nilai I-7

### Gambar 6.7 Membuat *database*......................................................................... VI-7

### Gambar 6.8 Membuat tabel (tb\_mahasiswa) ..................................................... VI-7

### Gambar 6.9 Menambahkan library connector ................................................... VI-8

### Gambar 6.10 Desain *Form* GUI\_Mahasiswa .................................................. VI-11

### Gambar 6.11 Tampilan hasil *method* tampil() GUI\_Mahasiswa ..................... VI-13

### Gambar 6.12 Tampilan hasil *method insert*() GUI\_Mahasiswa ...................... VI-13

### Gambar 6.13 Tampilan hasil *method update*() GUI\_Mahasiswa .................... VI-13

### Gambar 6.14 Tampilan hasil *metod delete*() GUI\_Mahasiswa........................ VI-13

### Gambar 6.15 Tampilan hasil method batal() GUI\_Mahasiwa......................... VI-14 Gambar 6.16 Membuat tabel (tb\_matkul)........................................................ VI-14

### Gambar 6.17 Menambahkan library connector ............................................... VI-14

### Gambar 6.18 Desain *Form* GUI\_Matkul ......................................................... VI-18

### Gambar 6.19 Tampilan hasil *method* tampil() GUI\_Matkul ........................... VI-19

### Gambar 6.20 Tampilan hasil *method insert*() GUI\_Matkul ............................ VI-20

### Gambar 6.21 Tampilan hasil *method update*() GUI\_Matkul........................... VI-20

### Gambar 6.22 Tampilan hasil *method delete*() GUI\_Matkul ............................ VI-20

### Gambar 6.23 Tampilan hasil *method* batal() GUI\_Matkul.............................. VI-20 Gambar 6.24 Membuat tabel (tb\_nilai)............................................................ VI-21

### Gambar 6.25 Menambah library connector ..................................................... VI-21

### Gambar 6.26 Desain *Form* GUI\_Nilai ............................................................ VI-25

### Gambar 6.27 Tampilan hasil *metod* tampil() GUI\_Nilai ................................. VI-27

### Gambar 6.28 Tampilan hasil *method insert*() GUI\_Nilai ................................ VI-27

### Gambar 6.29 Tampilan hasil *method update*() GUI\_Nilai .............................. VI-27

### Gambar 6.30 Tampilan hasil *method delete*() GUI\_Nilai................................ VI-27

### Gambar 6.31 Tampilan hasil *method* batal() GUI\_Nilai ................................. VI-27 Gambar 6.32 Class Diagram interface pada kelas BarangKeluar.................... VI-28 Gambar 6.33 Hasil tampilan Driver Class(*interface*) ...................................... VI-32 Gambar 6.34 Membuat database baru ............................................................. VI-33

### Gambar 6.35 Struktur table tb\_barang............................................................. VI-33

### Gambar 6.36 Mennmabahkan library mysql connector................................... VI-37 Gambar 6.37 Desain GUI\_tambahBarang ....................................................... VI-37

### Gambar 6.38 Struktur table tb\_barang I-40

### Gambar 6.39 Menambahkan library Connector .............................................. VI-44

### Gambar 6.40 Desain Form Barang Masuk ...................................................... VI-44

### Gambar 6.41 Tampilan hasil Gui barnag masuk ............................................. VI-46

### Gambar 6.42 Struktur tabel barang keluar....................................................... VI-47

### Gambar 6.43 Menambahkan library Connector .............................................. VI-52

### Gambar 6.44 Desain Form Barang keluar ....................................................... VI-52

### Gambar 6.45 Hasil Tampilan Barang Keluar .................................................. VI-54

# DAFTAR TABEL

### Tabel 2.1 Properti Desain GUI\_Mahasiswa.java I-5

### Tabel 2.2 Properti Desain GUI\_DataPenerimaBeasiswa.java.............................II-8 Tabel 3.1 Properti Desain GUI\_Nilai ................................................................ III-4

### Tabel 3.2 Properti Desain GUI\_Nilai ................................................................ III-7

### Tabel 3.3 Properti Desain GUI\_ Barang.java .................................................. III-12

### Tabel 3.4 Properti Desain GUI\_DataMahasiswa java ..................................... III-16

### Tabel 3.5 Properti Desain GUI\_DataPenerimaBeasiswa.java......................... III-19 Tabel 4.1 Hak akses Enkapsulasi....................................................................... IV-3

### Tabel 4.2 Properti desain GUI\_Penilaian .......................................................... IV-5

### Tabel 4.3 Properti Desain GUI\_Nilai ................................................................ IV-9

### Tabel 4.4 Properti Desain GUI\_Barang.java .................................................. IV-12

### Tabel 4.5 Tabel Property Design GUI\_BarangMasuk..................................... IV-15

### Tabel 4.6 Properti desain GUI\_BarangKeluar.(*Enkapsulasi*) ......................... IV-19 Tabel 4.7 Property desain GUI\_barang.java.................................................... IV-24

### Tabel 4.8 Properti Desain GUI ........................................................................ IV-27

### Tabel 4.9 Properti GUI\_BarangKeluar.java ................................................... IV-31

### Tabel 5.1 Checked Exception , Uncecked Exception dan error ..........................V-6 Tabel 5.2 Properti Desain GUI\_Matkul (*JTable*) ................................................V-8

### Tabel 5.3 Properti Desain GUI\_Nilai (*Polimorfisme*) .......................................V-11

### Tabel 5.4 Property Desain GUI\_MenuUtama ...................................................V-13

### Tabel 5.5 Property Desain GUI .........................................................................V-16 Tabel 5.6 Property Desain GUI\_BarangMasuk .................................................V-19

### Tabel 5.7 Property GUI\_Barang(*TryCatch*) ......................................................V-27

### Tabel 5.8 Property Gui\_BarangMasuk(*TryCatch*) ............................................V-30

### Tabel 5.9 Property Desain GUI\_BarangKeluar .................................................V-33

Tabel 5.10 Property desain GUI\_Barang(*Polimorfisme*) V-36

### Tabel 5.11 Property GUI\_BarangMasuk(*polimorfisme*) ...................................V-39

### Tabel 5.12 Property Desain GUI\_BarangKeluar ...............................................V-43

### Tabel 6.1 Properti Desain GUI\_Nilai(*JTable*)................................................... VI-5

### Tabel 6.2 Properti Desain GUI\_Mahasiswa (*Database*) ................................. VI-11

### Tabel 6.3 Properti Desain GUI\_ Matkul (*Database*)....................................... VI-18

### Tabel 6.4 Properti Desain GUI\_ Nilai (*Database*) .......................................... VI-25 Tabel 6.5 Property GUI\_Barang ...................................................................... VI-37

### Tabel 6.6 Properti GUI\_BarangMasuk ............................................................ VI-44

### Tabel 6.7 Properti GUI\_BarangKeluar ............................................................ VI-52

**BAB I 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. Didalam pemrograman prosedural seluruh , konstanta, ataupun fungsi yang dibutuhkan didalam rogram wajib didefinisikan sementara didalam programan beorientasi objek semua kebutuhan , konstanta atau fungsi tersebut cukup dibuatkan di dalam sebuah objek.

### OOP adalah paradigma pemrograman yang cukup dominan saat ini, karena mampu solusi kaidah modern. Meskipun demikian, bukan berarti bahwa pemrograman prosedural sudah tidak layak lagi. OOP diciptakan karena dirasakan masih adanya keterbatasan pada bahasa pemrograman tradisional. Konsep pdari 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.

|  |  |
| --- | --- |
| Nama Aslab Pengajar :  Silva Anjelina Tambunan | TTD : |
| Tanggal : |

### 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

### Apa itu OOP?

### Apa tujuan dari OOP?

### Bagaimana pengaplikasian OOP?

## Tujuan

### Mampu memahami dengan konsep OOP

### Mampu menerapkan dan menguasai berdasarkan konsep OOP

### Mampu membuat program dari OOP

## Manfaat

### Dapat mengetahui konsep dasar maupun lanjutan dari OOP

### Dapat mengaplikasikan OOP

### Dapat mengerti fungsi-fungsi keyword yang terdapat dalam OOP

|  |  |
| --- | --- |
| Nama Aslab Pengajar : Silva Anjelina Tambunan | TTD : |
| Tanggal : |

**BAB II 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 perograman 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, objek itu sendiri merupakan gabungan dari beberapa *attribute* yang lebih kecil lagi. Contoh saja seperti mobil, mobil merupakan suatu objek. Mobil terbentuk dari beberapa *attribute* yang lebih kecil seperti roda, *kursi*, kemudi, mesin dll. Mobil sebagai objek yang terbentuk dari *attribute* yang saling berhubungan dan berinteraksi. Begitu juga dengan program, sebuah objek yang besar dibentuk dari beberapa *attribute* dan method, *attribute* itu saling berkomunikasi dan berinteraksi.

|  |  |
| --- | --- |
| Nama Aslab Pengajar :  Silva Anjelina Tambunan | TTD : |
| Tanggal : |

#### Class & Object

#### Class

### merupakan rancangan dari sebuah objek yang medefenisikan atribut (ciri/variabel) dan *method* (perilaku) umum dari suatu objek tertentu.

* + - * 1. *Drive Class : Class* yang menggunakan main atau dapat di

#### running.

* + - * 1. *Object Class* : *Class* yang tidak ada main, biasanya berisi *method* dan *attribute* yang nantinya akan di panggil di *Driver Class*. *Class* dideklarasikan sebagai berikut:

class NamaClass{

//deklarasi atribut

//deklarasi method

}

### Keterangan :

### *Class* adalah kata kunci yang digunakan untuk mendeklarasikan suatu kelas.

NamaClass merupakan *indentifier*.

### Atribut dan *Method* bisa berjumlah 0 atau lebih.

#### Atribut

### *Atribut* adalah data/variabel yang akan dimiliki oleh objek dari kelas tersebut. Atribut dapat memiliki hak akses *private, public* maupun *protected* (akan dijelaskan di bab 3).

*Attribute* dideklarasikan sebagai berikut:

Hak\_akses tipe\_data nama\_Atribut;

#### Method

### adalah aksi atau tindakan yang dapat dilakukan oleh objek dari suatu kelas.

*Method* terdiri dari 2 jenis, yakni *Method* void dan *Method* Non Void.

#### 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**} | | |
|  | Nama Aslab Pengajar : Silva Anjelina Tambunan | TTD : |
| Tanggal : |

* + - * 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”.

#### Object

### adalah realisasi 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();

## Langkah – Langkah Praktikum

1. Buka Aplikasi *Netbeans*.
2. Buat *Class*.

### Memberi *script* pada kelas tersebut.

1. Buat *form*.
2. Memberi *script* pada *form* tersebut.

### Menjalankan program.

## Tugas Praktikum ke-1:

### Membuat kelas Mahasiswa.java dan GUI\_Mahasiswa.java Diagram *Class* (*Class* Mahasiswa):



|  |
| --- |
| Mahasiswa |
| * String nim, nama, prodi, angktn; |
| + void dataNIM(String Nim)  + void dataNama(String Nama)  + void dataProdi(String Prodi)  + void dataAngkatan(String Angktn)  + String cetakNIM()  + String cetakNama()  + String cetakProdi()  + String cetakAngkatan() |

*Source code* (Mahasiswa.java) :

### Class Attribute

### Method

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;

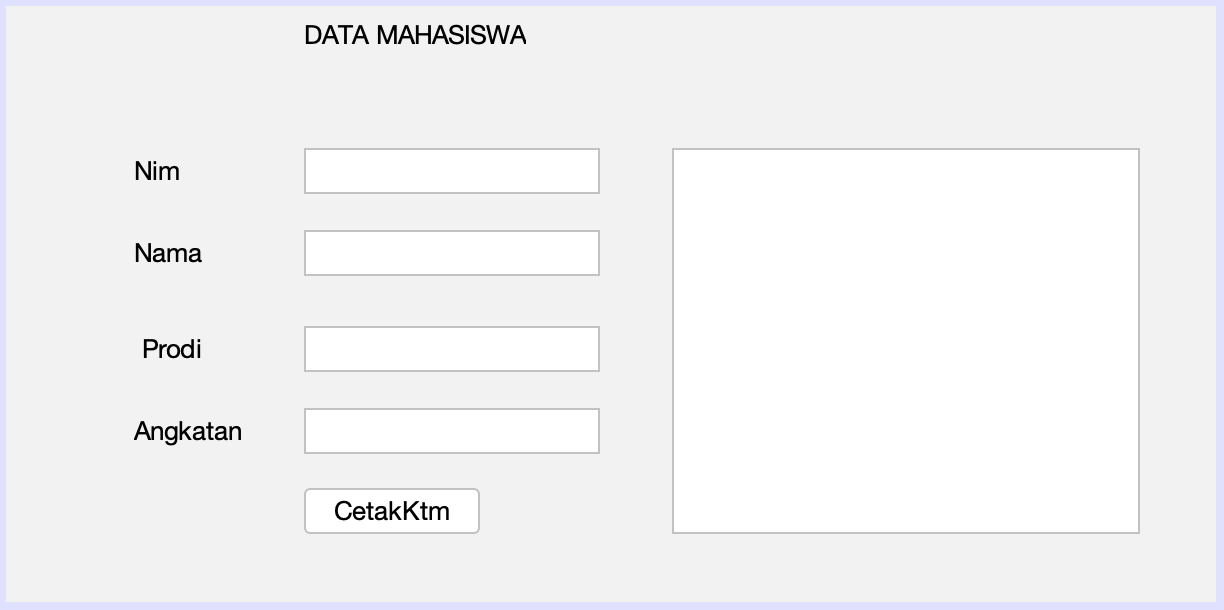
}

String cetakAngkatan(){ return angktn;

}

}

### Desain *form* (GUI\_Mahasiswa.java) :



### Gambar 2.1 Desain from GUI\_Mahasiswa.java Tabel 2.1 Properti Desain GUI\_Mahasiswa.java

|  |  |  |  |
| --- | --- | --- | --- |
| **No** | **Objek** | **Properti** | **Nilai** |
| 1 | jLabel1 | Text | DATA MAHASISWA |
| 2 | jLabel2 | Text | NIM |
| 3 | jLabel3 | Text | Nama |
| 4 | jLabel4 | Text | Prodi |
| 5 | jLabel5 | Text | Angkatan |
| 6 | jTextField1 | Name | txtNim |
| Text |  |
| 7 | jTextField2 | Name | txtNama |
| Text |  |
| 8 | jTextField3 | Name | txtProdi |
| Text |  |
| 9 | jTextField4 | Name | txtAng |
| Text |  |
| 10 | jButton1 | Name | btnKTM |
| Text | Cetak KTM |
| 11 | jTextArea | Name | memoKTM |
| Text |  |

*Source code* pada *button* Cetak KTM :

private void btnKTMActionPerformed(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()); mhs.dataProdi(txtprodi.getText()); mhs.dataAngkatan(txtangkatan.getText());

memoKTM.append("Kartu Tanda Mahasiswa\n"); memoKTM.append("

\n");

memoKTM.append("NIM : " + mhs.cetakNIM()

+"\n");

memoKTM.append("Nama : " + mhs.cetakNama() +

"\n");

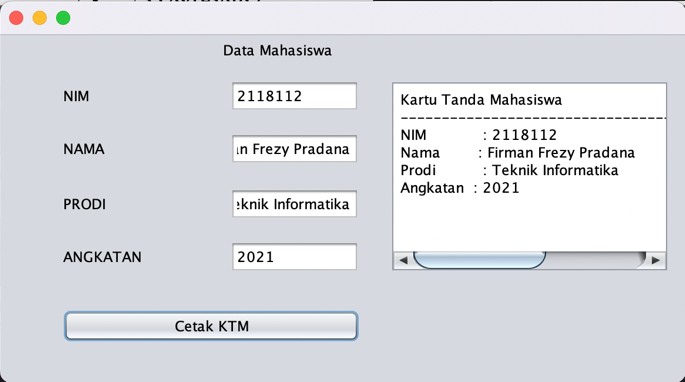
memoKTM.append("Prodi : " + mhs.cetakProdi() +"\n");

memoKTM.append("Angkatan : " + mhs.cetakAngkatan()

+ "\n");

}

### Hasil Tampilan :



### Gambar 2.2 Tampilan hasil GUI\_Mahasiswa.java

### Analisa:

### Pada Analisa *sorce code* di atas terdapat pendeklarasian *object* dari *class* Mahasiswa dengan nama “mhs”. Pada *class* Mahasiswa terdapat beberapa *method* yang akan di gunakan untuk menampilkan hasil dari inputan *user* salah satunya yaitu *method* dataNIM (String Nim) yang digunakan untuk mengambil data dari isi *textField* Nim. Sehingga pada saat *button* Cetak KTM di klik, maka method cetakNIM() pada *object* mhs akan mengembalikan nilai variable nim yang nantinya ditampilkan pada memoKTM. Proses yang sama juga terjadi pada *method* dataNama(String Nama), dataProdi(*String* Prodi) dan dataAngkatan(*String* Angktn).

## Tugas Rumah 1 :

### Membuat kelas Barang.java dan GUI\_Barang.java Judul : inventory *Sparepart* Motor

Diagram *Class*:

|  |
| --- |
| Barang |
| * String NmBarang, KdBarang, Kategori, ForMotor, Suplayer; * int Harga; |
| * Void dataNmBarang(String NmBarang) * Void dataKategori(String KdBarang) * Void dataKatgori(String Kategori) * Void dataForMotor(String FoMotor) * Void dataSuplayer(String Suplayer) * Int dataHrga(int Harga) * String CetakNmBarang() * String CetakKategori() * String CetakForMotor() * String CetakSuplayer() * Int CetakHarga() |

*Source code* pada *Class* Data\_PenerimaBeasiswa.java :

### Class Atribut

### Method

|  |  |  |
| --- | --- | --- |
| public class Barang {  String NmBarang,Kategori,ForMotor,Suplayer; int Harga,kdBarang;  int Kode(){  Random rndm = new Random(); kdBarang = rndm.nextInt(100000); return kdBarang;  }  void dataNmBarang(String NmBarang){ this.NmBarang = NmBarang;  }  void dataKategori(String Kategori){ this.Kategori = Kategori;  }  void dataForMotor(String ForMotor){ this.ForMotor = ForMotor;  }  void dataSuplayer(String Suplayer){ this.Suplayer = Suplayer;  }  int dataHarga(int Harga){ this.Harga = Harga; return Harga;  }  String CetakNmBarang(){ return NmBarang;  } | | |
|  | Nama Aslab Pengajar : Silva Anjelina Tambunan | TTD : |
| Tanggal : |

String CetakKategori(){ return Kategori;

}

String CetakForMotor(){ return ForMotor;

}

String CetakSuplayer(){ return Suplayer;

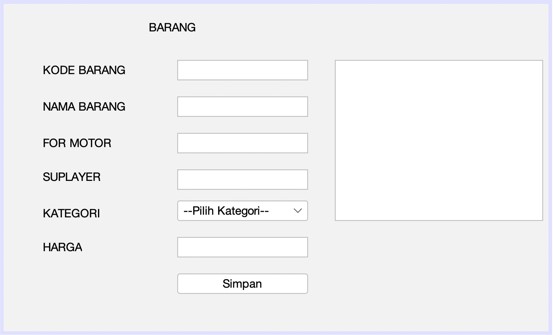
}

int CetakHarga(){ return Harga;

}

}

### Desain *form* (GUI\_Barang.java):



### Gambar 2.3 Desain GUI\_Barang.java(*object* & *method*) Tabel 2.2 Properti Desain GUI\_DataPenerimaBeasiswa.java

|  |  |  |  |
| --- | --- | --- | --- |
| **No** | **Objek** | **Properti** | **Nilai** |
| 1 | jLabel1 | Text | “BARANG” |
| 2 | jLabel2 | Text | “KODE BARANG” |
| 3 | jLabel3 | Text | “FOR MOTOR” |
| 4 | jLabel4 | Text | “ SUPLAYER” |
| 5 | jLabel5 | Text | “KATEGORI” |
| 6 | JLabel6 | Text | “HARGA” |
| 7 | jTextField1 | Name | Txt\_KdBarang |
| Text | “ “ |
| 8 | jTextField1 | Text | “ “ |
| Name | Txt\_NmBarang |
| 9 | jTextField2 | Text | “ “ |
| Name | Txt\_ForMotor |
| 10 | jTextField3 | Text | “ “ |

|  |  |
| --- | --- |
| Nama Aslab Pengajar : Silva Anjelina Tambunan | TTD : |
| Tanggal : |

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | Name | Txt\_Suplayer |
| 11 | TextField5 | Name | “Txt\_Harga” |
| text | “ “ |
| 12 | JComboBox1 | Name | Cmb\_Kategori |
| Model | --Pilih Kategori--, Oli, Part motor, Velek, Lampu Depan, Lampu Belakang, Pack Block/BB, Pack Kalter / CC, pack Kopling /KPL,  Baut, Mur |
| 13 | jButton1 | Name | Simpan |
| Text | “Simpan” |
| 14 | jTextArea | Name | Memo |
| Text | “ “ |

*Source code Button* Cetak pada GUI:

|  |  |  |
| --- | --- | --- |
| private void  SimpanActionPerformed(java.awt.event.ActionEvent evt) {  // TODO add your handling code here: int Berhasil = 0;  Barang brg = new Barang(); brg.dataKdBarang(Txt\_KdBarang.getText()); brg.dataNmBarang(Txt\_NmBarang.getText()); brg.dataForMotor(Txt\_ForMotor.getText()); brg.dataSuplayer(Txt\_KdBarang.getText());  brg.dataHarga(Integer.parseInt(Txt\_Harga.getText())); if(Cmb\_Kategori.getSelectedIndex()==0){  JOptionPane.showMessageDialog(null, "Kategori Belum Anda Masuukan");  }else if(Cmb\_Kategori.getSelectedIndex()==1){ Berhasil = 1;  brg.dataKategori("oli");  }else if(Cmb\_Kategori.getSelectedIndex()==2){ Berhasil = 1;  brg.dataKategori("Part Motor Mesin");  }else if(Cmb\_Kategori.getSelectedIndex()==3){ Berhasil = 1;  brg.dataKategori("Lampu Depan");  }else if(Cmb\_Kategori.getSelectedIndex()==4){ Berhasil = 1;  brg.dataKategori("Lampu Belakang");  }else if(Cmb\_Kategori.getSelectedIndex()==5){ Berhasil = 1; | | |
|  | Nama Aslab Pengajar : Silva Anjelina Tambunan | TTD : |
| Tanggal : |

brg.dataKategori("Baut");

}else if(Cmb\_Kategori.getSelectedIndex()==6){ Berhasil = 1;

brg.dataKategori("Mur");

}

if(Berhasil == 1){ JOptionPane.showMessageDialog(null,

"Berhasil!!!");

memo.append("

+brg.CetakKdBarang()+"\n");

memo.append(" "+brg.CetakNmBarang()+"\n");

memo.append(" "+brg.CetakForMotor()+"\n");

memo.append(" "+brg.CetakSuplayer()+"\n");

memo.append("

Kode

Barang

:

"

Nama

Barang

:

For

Motor

:

Suplayer

:

Kategori

:

"+brg.CetakKategori()+"\n");

memo.append(" Harga :Rp"+brg.CetakHarga());

}else{

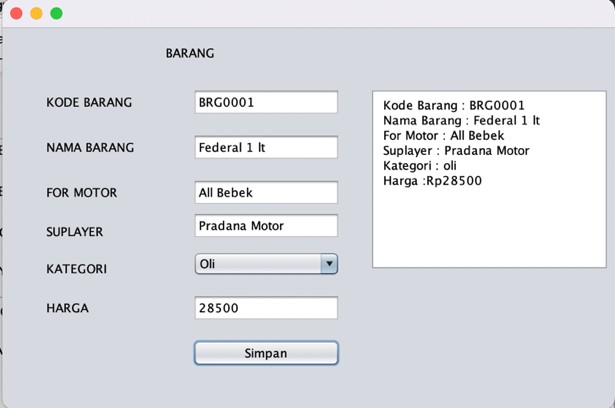
JOptionPane.showMessageDialog(null,

"Gagal!!!");

}

}

### Hasil Tampilan:



### Gambar 2.4 Hasil Tampilan GUI\_DataPenerimaBeasiswa.java Analisa:

### Di dalam *class* Barang.java berisi fungsi *void* dan non *void* yang nantinya akan di panggil di *class* GUI\_Barang.java. kegunaanya antara lain untuk mengambil data yang di *input* kan , dan menampilkannya pada layer monitor didalam *textarea*. Kemudian pada *Source code Button* didalamnya terdapat perintah untuk mengambil data dari txt\_Kdbarang, txt\_NmBarang, txt\_ForMotor, txt\_Suplayer, txt\_Kategori, txt\_harga. kemudian menampilkann didalam *textarea*.

## Kesimpulan

### *Class* merupakan rancangan dari sebuah objek yang mendefenisikan atribut (ciri/variabel) dan method (perilaku) umum dari suatu objek tertentu.

1. *Object* adalah realisasi dari *class*. Ketika kita membuat sebuah *class*

### bukan berarti kita membuat sebuah objek.

### Atribut adalah data/variabel yang akan dimiliki oleh objek dari kelas tersebut. Atribut dapat memiliki hak akses *private, public* maupun *protected.*

**BAB III KONSTRUKTOR 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.   1. Praktikan mampu mengetahui Inheritance |
| **Alat / bahan** | **:** | 1. Seperangkat *computer*. 2. Perangkat lunak: *Netbeans*. 3. Modul Praktikum *OOP* 2022. |

## Landasan Teori

### Konstruktor

*Konsktruktor* adalah *method* yang pertama kali dijalankan pada saat sebuah objek pertama kali diciptakan, yang di mana syarat penulisan *method konstruktor* ini harus sama dengan nama *class* yang di naungi. Biasanya *method* ini digunakan untuk *inisialisasi* atau mempersiapkan data untuk objek.

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;

}

}

|  |  |
| --- | --- |
| Nama Aslab Pengajar :  Silva Anjelina Tambunan | TTD : |
| Tanggal : |

### Keterangan:

### “NamaClass” merupakan nama class dapat di ganti sesuai keinginan

### “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

### Inheritance

### *Inheritance* atau Pewarisan atau Penurunan adalah konsep pemrograman di mana sebuah *class* dapat “menurunkan ” property dan method yang dimilikinya kepada *class* lain. Konsep *inheritance* digunakan untuk memanfaatkan fitur ‘*code reuse’* untuk menghindari duplikasi kode program.

### Konsep *inheritance* membuat sebuah struktur atau ‘*hierarchy’ class* dalam kode program. *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 atau heir class. Tidak semua property dan *method* dari class induk 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 diakses

### dari class anak.

### Pada pemrograman berorientasi objek atau OOP, konsep *inheritance* menjadi salah satu topik yang penting. Suatu objek diwariskan dengan menggunakan *keyword extends***.** Pada pemrograman Java setiap objek pada kenyataannya adalah turunan dari class Object. Walaupun tidak secara eksplisit (jelas), ini bisa dibuktikan dengan operator *instanceof*.

Istilah penting dalam konsep *inheritance*:

### Super Class : kelas induk yang mewariskan atribut dan method kepada turunannya.

### Sub Class atau Child Class : kelas turunan yang mewarisi atribut dan method. Sub Class dapat menambah atribut dan methodnya sendiri sebagai tambahan dari kelas yang memberi warisan.

### Reusability : menggunakan kembali atribut dan method dari super class di sub class.

## Langkah – Langkah Praktikum

### Buka Aplikasi Netbeans.

### Buat Class.

### Memberi script pada kelas tersebut.

### Buat form.

### Memberi script pada form tersebut.

### Menjalankan program

## Tugas Praktikum ke-1:

Implementasi Method *Konstrucktor* Pada *Projek* Praktikum GUI\_Matkul Diagram *Class*: (Class Data\_Matkul.java)

### Class Attribute

|  |
| --- |
| Data\_Matkul |
| * String kode\_mk, nama\_mk,dosen; |
| * Public Data\_Matkul(String kd, String mk, String dsn) |

### Method

*Source code* pada *Class* Data\_Matkul :

public class Data\_Matkul {

String kode\_mk, nama\_mk, dosen;

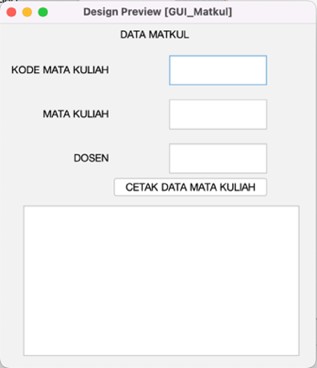
public Data\_Matkul(String kd, String mk, String dsn){ this.kode\_mk=kd;

this.dosen=dsn; this.nama\_mk=mk;

}

}

Desain *form* GUI\_Matkul:



### Gambar 3.1 Desain form GUI\_Matkul Tabel 3.1 Properti Desain GUI\_Nilai

|  |  |  |  |
| --- | --- | --- | --- |
| **No** | **Objek** | **Properti** | **Nilai** |
| 1 | jLabel1 | Text | DATA MATKUL |
| 2 | jLabel2 | Text | KODE MATA KULIAH |
| 3 | jLabel3 | Text | MATA KULIAH |
| 4 | jLabel4 | Text | Dosen Pengajar |
| 5 | jTextField1 | Name | txtkode |
| Text |  |
| 6 | jTextField2 | Name | txtmk |
| Text |  |
| 7 | jTextField3 | Name | txtdsn |
| Text |  |
| 8 | jButton1 | Name | jButton1 |
| Text | Cetak Data Mata Kuliah |
| 9 | jTextArea | Name | memo |
| Text |  |

*Source code* pada button “ Cetak Data Matakuliah :

private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {

// TODO add your handling code here: Data\_Matkul matkul =new Data\_Matkul("","",""); matkul.nama\_mk=(txtmk.getText()); matkul.dosen=(txtdsn.getText()); matkul.kode\_mk=(txtkode.getText());

memo.append("DATA MATA KULIAH\n");

memo.append("

\n");

memo.append("Nama MK matkul.nama\_mk +"\n");

memo.append("Nama Dosen matkul.dosen + "\n");

memo.append("Kode MK

+"\n");

memo.setEnabled(false);

}

: " +

: " +

: " + matkul.kode\_mk

*Source code* pada Method Konstrucktor pada GUI\_Matkul :

public GUI\_Matkul() { initComponents();

Data\_Matkul matkul = new Data\_Matkul("","",""); matkul.nama\_mk="OOP"; txtmk.setText(matkul.nama\_mk); txtmk.setEnabled(false);

matkul.dosen="yosep"; txtdsn.setText(matkul.dosen); txtdsn.setEnabled(false); matkul.kode\_mk="F137"; txtkode.setText(matkul.kode\_mk); txtkode.setEnabled(false);

}

### Hasil Tampilan :



### Gambar 3.2 Hasil tampilan GUI\_Matkul

### Analisa :

### Dalam program tersebut, semua *textfield* di set menjadi *enabled* sehingga user tidak bisa menginputkan data secara manual. Tetapi data akan diambil dari method konstruktor.dalam program tersebut terdapat *method* konstruktor dengan nama Data\_Matkul yang isinya adalah parameter yang sama dengan atribut.kemudian *method* tersebut di panggil pada GUI\_matkul untuk di isialisasi isinya yaitu yang ada pada konstruktor public.

## Tugas Praktikum ke-2 :

### Implementasi *Inheritance* Pada Projek Praktikum “Class Penilaian” Diagram Class (Penilaian.java):

### Class Attribute

|  |
| --- |
| Penilaian |
| * String NIM, kode\_mk; * Int nilaiTugas, nilaiPrak; |
| * Double nilaiAkhirPrak() * Double tampilNA() * doubleAkhir() |

### Method

*Source code* pada *Class* Penilaian :

|  |
| --- |
| Penilaian |
| * Double nilaiAkhir() |

### Class Method

public class Penilaian { String NIM,nama,kode\_mk;

int nilaiTugas, nilaiPrak; double nilaiAkhirPrak(){

return ((nilaiTugas\*0.6)+(nilaiPrak\*0.4));

}

double tampilNA(){

return nilaiAkhirPrak();

}

double nilaiAkhir(){ return 0;

}

}

*Source code* pada *Class* Penilaian :

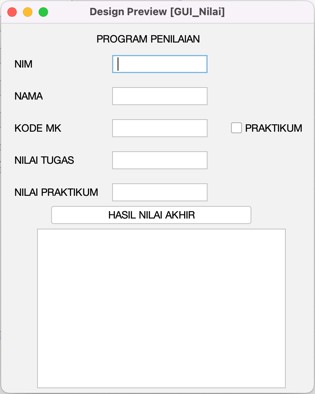
public class Penilaian\_NonPrak extends Penilaian { @Override

double nilaiAkhir(){ return (nilaiTugas);

}

}

Desain *form* GUI\_Nilai:



### Gambar 3.3 Desain GUI\_Penilaian.java Tabel 3.2 Properti Desain GUI\_Nilai

|  |  |  |  |
| --- | --- | --- | --- |
| **No** | **Objek** | **Properti** | **Nilai** |
| 1 | jLabel1 | Text | PROGRAM PENILAIAN |
| 2 | jLabel2 | Text | NIM |
| 3 | jLabel3 | Text | Nama |
| 4 | jLabel4 | Text | Kode MK |
| 5 | jLabel5 | Text | Nilai Tugas |
| 6 | jLabel6 | Text | Nilai Praktikum |
| 7 | jTextField1 | Name | txtNIM |
| Text |  |
| 8 | jTextField2 | Name | txtNama |
| Text |  |
| 9 | jTextField3 | Name | txtKodeMK |
| Text |  |
| 10 | jTextField4 | Name | txtNT |
| Text |  |
| 11 | jTextField5 | Name | txtNP |
| Text |  |
| 12 | jButton1 | Name | btnNA |

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | Text | Hasil Nilai Akhir |
| 13 | jTextArea | Name | memoNilai |
| Text |  |
| 14 | JCheckBox | Text | Praktikum |
| Name | cek\_prak |

*Source code Jbutton* pada GUI\_Nilai :

private void btnNAActionPerformed(java.awt.event.ActionEvent evt) {

// TODO add your handling code here: memoNilai.setText("");

Penilaian\_NonPrak nilai = new Penilaian\_NonPrak(); nilai.NIM = txtNIM.getText();

nilai.nama = txtNama.getText(); nilai.kode\_mk = txtKodeMK.getText(); nilai.nilaiTugas =

Integer.parseInt(txtNT.getText()); nilai.nilaiPrak =

Integer.parseInt(txtNP.getText());

if (cek\_prak.isSelected()){ memoNilai.append("Nilai Akhir Mata Kuliah

\n");

memoNilai.append("

\n");

memoNilai.append("NIM : "+nilai.NIM +"\n");

memoNilai.append("Nama : "+nilai.nama +"\n");

memoNilai.append("Kode MK : "+nilai.kode\_mk +"\n");

memoNilai.append("Nilai Tugas : "+nilai.nilaiTugas +"\n");

memoNilai.append("Nilai Praktikum : "+nilai.nilaiPrak +"\n");

memoNilai.append("

\n");

memoNilai.append("Nilai Akhir : "+nilai.tampilNA());

}else{

memoNilai.append("Nilai Akhir Mata Kuliah

\n");

memoNilai.append("

\n");

memoNilai.append("NIM : "+nilai.NIM +"\n");

memoNilai.append("Nama : "+nilai.nama +"\n");

memoNilai.append("Kode MK : "+nilai.kode\_mk +"\n");

memoNilai.append("Nilai Tugas : "+nilai.nilaiTugas +"\n");

memoNilai.append("

\n");

memoNilai.append("Nilai Akhir "+nilai.nilaiAkhir());

}

}

:

*Source code* cek box pada GUI\_Nilai:

private

void

cek\_prakActionPerformed(java.awt.event.ActionEvent evt) {

// TODO add your handling code here: if (cek\_prak.isSelected()){

txtNP.setEnabled(true);

}

else{

txtNP.setEnabled(false);

}

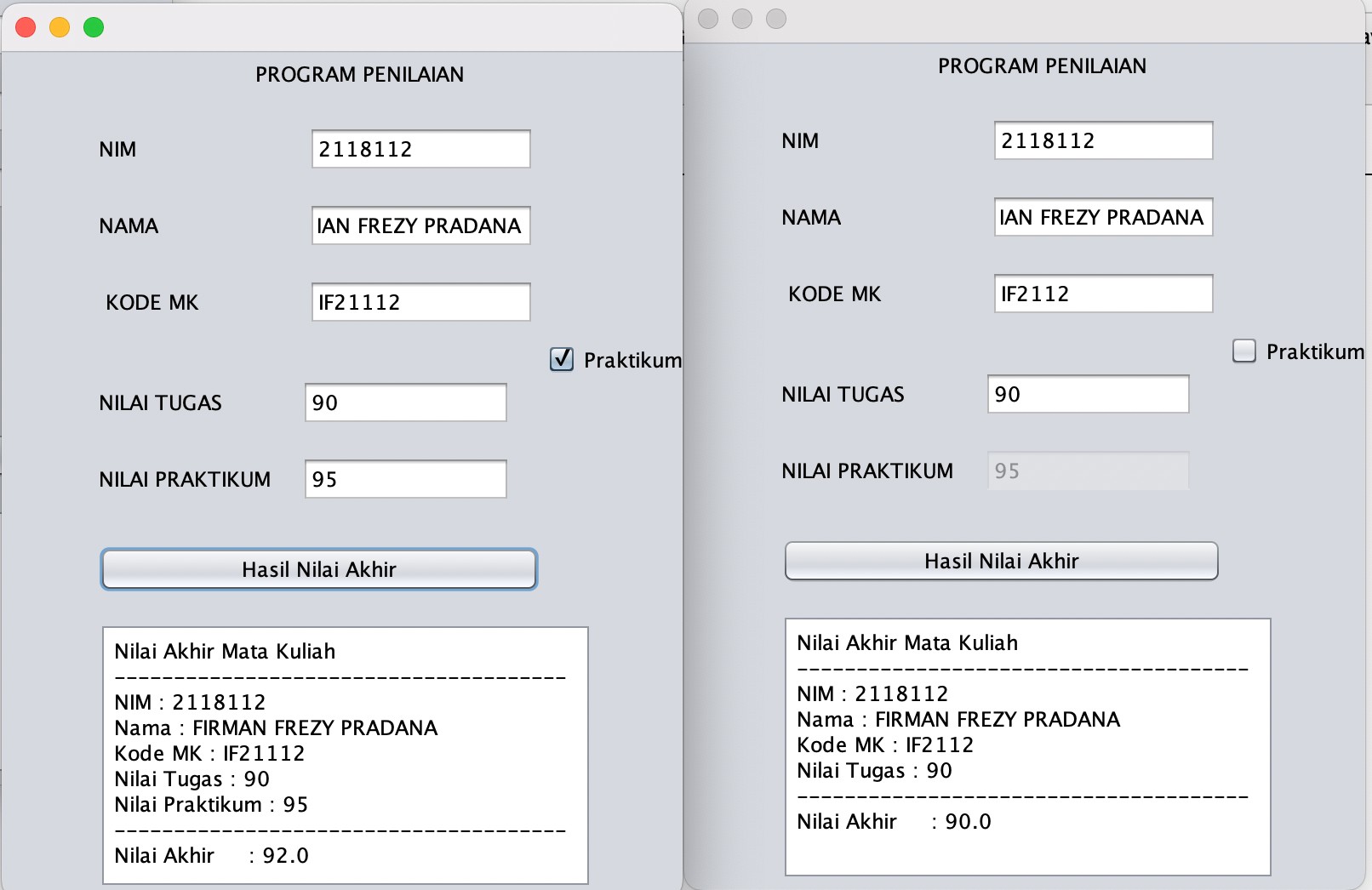
}

*Source code* method Konstructor pada GUI\_Nilai:

Private GUI\_Nilai(){ initComponents(); txt.setENableb(true)

}

### Hasil Tampilan GUI\_Nilai :



### Gambar 3.4 Hasil Tampilan GUI\_Penilaian

### Analisa

### Pada program tersebut untuk menampilkan nillai akhir. Pada kelas anak terdapat method dimana tidak dimiliki pada sub class yaitu,double nilaiakhir(), dimana didalamnya akan mengembalikan data dari atribut nilaitugas yang terdapat di class induk.berbeda Ketika matkulnya terdapat praktikum, maka kelas anak akan mendapat turunan method dari class penilaian yaitu method double nilaiakhirprak(); sesuai dengan yang di perintahkan yaitu implementasi inheritance , inheritance sendiri adalah

### Inheritance atau Pewarisan/Penurunan adalah konsep pemrograman dimana sebuah class dapat ‘menurunkan’ properti dan method yang dimilikinya kepada class lain. Pada kasus program di atas. Sudah menerapkan syarat syarat tersebut.

## Tugas Rumah 1 :

### Implementasi *Method* Konstruktor pada projek Barang. Judul : Inventory Sparepart Motor

### Diagram Class:

|  |
| --- |
| Barang |
| * String NmBarang, KdBarang, Kategori, ForMotor, Suplayer; * int Harga; |
| * Barang(){   NmBarang = "Default Nama Barang";  Kode();  ForMotor = "Veza,CB,CBR 150"; Suplayer= "Default Suplayer";  Harga = 49500;  }   * Void dataNmBarang(String NmBarang) * Void dataKategori(String KdBarang) * Void dataKatgori(String Kategori) * Void dataForMotor(String FoMotor) * Void dataSuplayer(String Suplayer) * Int dataHrga(int Harga) * String CetakNmBarang() * String CetakKategori() * String CetakForMotor() * String CetakSuplayer()    + Int CetakHarga() |

*Source code* pada *object Class* (Barang.java) :

### Class Attribute

### Konstruktor

### Method

public class Barang {

String NmBarang,Kategori,ForMotor,Suplayer; int Harga,kdBarang;

int Kode(){

Random rndm = new Random(); kdBarang = rndm.nextInt(100000); return kdBarang;

}

Barang(){

NmBarang = "Default Nama Barang"; Kode();

ForMotor = "Veza,CB,CBR 150"; Suplayer= "Default Suplayer"; Harga = 49500;

}

void dataNmBarang(String NmBarang){ this.NmBarang = NmBarang;

}

void dataKategori(String Kategori){ this.Kategori = Kategori;

}

void dataForMotor(String ForMotor){ this.ForMotor = ForMotor;

}

void dataSuplayer(String Suplayer){ this.Suplayer = Suplayer;

}

int dataHarga(int Harga){ this.Harga = Harga; return Harga;

}

String CetakNmBarang(){ return NmBarang;

}

String CetakKategori(){ return Kategori;

}

String CetakForMotor(){ return ForMotor;

}

String CetakSuplayer(){ return Suplayer;

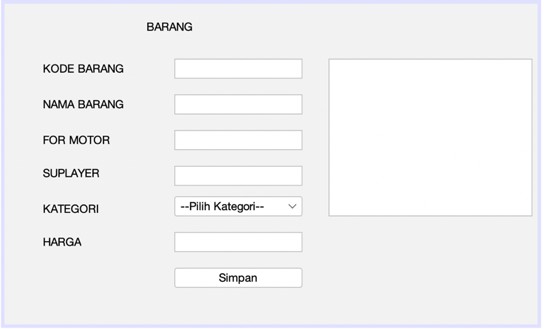
}

int CetakHarga(){ return Harga;

}

}

### Desain *form* GUI\_Barang.java :



Gambar 3.5 Desain *form* GUI\_Barang(*Konstrucktor*)

### Tabel 3.3 Properti Desain GUI\_ Barang.java

|  |  |  |  |
| --- | --- | --- | --- |
| **No** | **Objek** | **Properti** | **Nilai** |
| 1 | jLabel1 | Text | “BARANG” |
| 2 | jLabel2 | Text | “KODE BARANG” |
| 3 | jLabel3 | Text | “FOR MOTOR” |
| 4 | jLabel4 | Text | “ SUPLAYER” |
| 5 | jLabel5 | Text | “KATEGORI” |
| 6 | JLabel6 | Text | “HARGA” |
| 7 | jTextField1 | Name | Txt\_KdBarang |
| Text | “ “ |
| 8 | jTextField1 | Text | “ “ |
| Name | Txt\_NmBarang |
| 9 | jTextField2 | Text | “ “ |
| Name | Txt\_ForMotor |
| 10 | jTextField3 | Text | “ “ |
| Name | Txt\_Suplayer |
| 11 | TextField5 | Name | “Txt\_Harga” |
| text | “ “ |
| 12 | JComboBox1 | Name | Cmb\_Kategori |
| Model | --Pilih Kategori--, Oli, Part motor, Velek, Lampu Depan, Lampu Belakang, Pack Block/BB, Pack Kalter / CC, pack Kopling /KPL,  Baut, Mur |
| 13 | jButton1 | Name | Simpan |
| Text | “Simpan” |
| 14 | jTextArea | Name | Memo |
| Text | “ “ |

*Source code* pada *konsruktor* GUI\_Data\_Barang:

public Gui\_Barang() {

initComponents(); int nilai;

Barang brg = new Barang(); Txt\_KdBarang.setText(Integer.toString(brg.kdBarang));

Txt\_KdBarang.setEnabled(false); Txt\_NmBarang.setText(brg.NmBarang); Txt\_NmBarang.setEnabled(false); Txt\_ForMotor.setText(brg.ForMotor); Txt\_ForMotor.setEnabled(false); Txt\_Suplayer.setText(brg.Suplayer); Txt\_Suplayer.setEnabled(false); Txt\_Harga.setText(Integer.toString(brg.Harga)); Txt\_Harga.setEnabled(false);

}

*Source code* pada button “Simpan”:

private void SimpanActionPerformed(java.awt.event.ActionEvent evt) {

// TODO add your handling code here: int Berhasil = 0,Kode;

Barang brg = new Barang(); brg.dataNmBarang(Txt\_NmBarang.getText()); brg.dataForMotor(Txt\_ForMotor.getText()); brg.dataSuplayer(Txt\_Suplayer.getText());

brg.dataHarga(Integer.parseInt(Txt\_Harga.getText())); Kode = Integer.parseInt(Txt\_KdBarang.getText()); if(Cmb\_Kategori.getSelectedIndex()==0){

JOptionPane.showMessageDialog(null, "Kategori Belum Anda Masuukan");

}else if(Cmb\_Kategori.getSelectedIndex()==1){ Berhasil = 1;

brg.dataKategori("oli");

}else if(Cmb\_Kategori.getSelectedIndex()==2){ Berhasil = 1;

brg.dataKategori("Part Motor Mesin");

}else if(Cmb\_Kategori.getSelectedIndex()==3){ Berhasil = 1;

brg.dataKategori("Lampu Depan");

}else if(Cmb\_Kategori.getSelectedIndex()==4){ Berhasil = 1;

brg.dataKategori("Lampu Belakang");

}else if(Cmb\_Kategori.getSelectedIndex()==5){ Berhasil = 1;

brg.dataKategori("Baut");

}else if(Cmb\_Kategori.getSelectedIndex()==6){ Berhasil = 1;

brg.dataKategori("Mur");

}

if(Berhasil == 1){ JOptionPane.showMessageDialog(null,

"Berhasil!!!");

memo.append(" Kode Barang : " +Kode+"\n"); memo.append(" Nama Barang :

"+brg.CetakNmBarang()+"\n");

memo.append(" For Motor : "+brg.CetakForMotor()+"\n");

memo.append(" Suplayer : "+brg.CetakSuplayer()+"\n");

memo.append(" Kategori : "+brg.CetakKategori()+"\n");

memo.append(" Harga :Rp"+brg.CetakHarga());

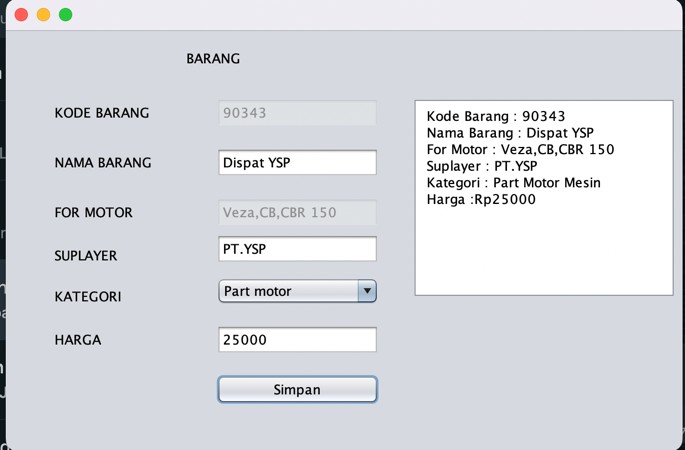
}else{

JOptionPane.showMessageDialog(null,

"Gagal!!!");

} }

### Tampilan Hasil:



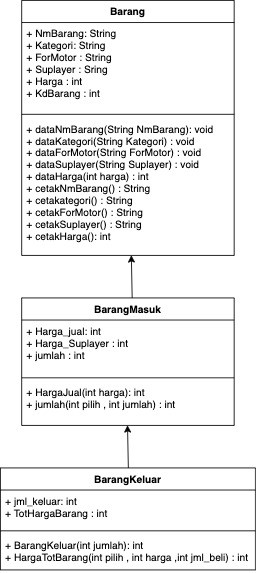
### Gambar 3.6 Hasil Tampilan GUI\_Barang.java(*konstructor*) Analisa :

### Dalam program tersebut, *textfield* Kode barang memiliki nilai otomatis dan di set *enable,* For Motor di set menjadi *enabled* sehingga *user* tidak bisa menginputkan data secara manual. Tetapi data akan diambil dari *method konstruktor*.dalam program tersebut terdapat *method* konstruktor dengan nama Barang yang isinya adalah parameter yang sama dengan atribut.kemudian *method* tersebut di panggil pada GUI\_Barang.

## Tugas Rumah 2 :

### Menerapkan Inheritance serta membuat GUI\_BarangMasuk, dan Barangkeluar.

### Diagram Class:



Gambar 3.7 Class diagram *inheritance Source code* pada *Class* BarangMasuk.java :

public class BarangMasuk extends Barang{ int harga\_jual,jumlah,harga\_suplayer; public BarangMasuk(){

harga\_suplayer = 45000;

jumlah = 200;

}

int HargaJual(int harga){ this.harga\_jual = harga; return harga;

}

int Jumlah(int pilih,int jumlah){ if(pilih == 1){

jumlah = jumlah;

}else if(pilih == 2){ jumlah = jumlah \* 12;

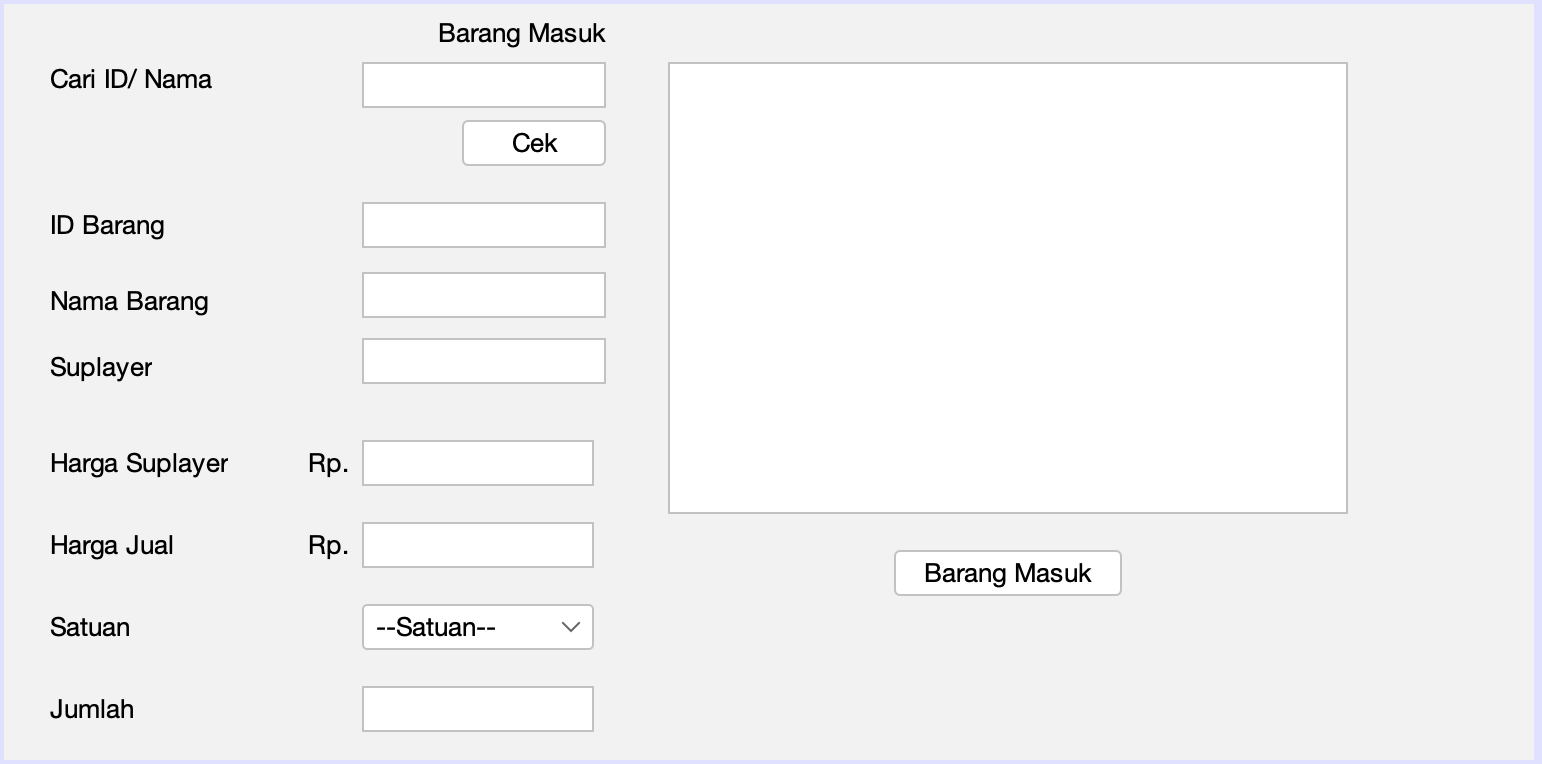
}else if(pilih == 3){

jumlah = jumlah \* 24;

}

return jumlah;} }

### Desain *form* (GUI\_BarangMasuk.java):



### Gambar 3.8 Desain *form (*GUI\_BarangMasuk.java) Tabel 3.4 Properti Desain GUI\_DataMahasiswa java

|  |  |  |  |
| --- | --- | --- | --- |
| **No** | **Objek** | **Properti** | **Nilai** |
| 1 | jLabel1 | text | "Cari ID/Nama" |
| 2 | jLabel2 | text | "ID Barang" |
| 3 | jLabel3 | text | "Nama Barang" |
| 4 | jLabel4 | text | "Suplayer" |
| 5 | jLabel5 | text | "Harga Suplayer" |
| 6 | jLabel6 | text | "Harga Jual" |
| 7 | jLabel7 | text | "Satuan" |
| 8 | jLabel8 | text | "Jumlah" |
| 9 | JTextField1 | text | " " |
| name | txt\_Search |
| 10 | JTextField2 | text | " " |
| name | ID\_Barang |
| 11 | JTextField3 | text | " " |
| name | txt\_NmBarang |
| 12 | JTextField4 | text | " " |
| name | txt\_Suplayer |
| 13 | JTextField5 | text | " " |
| name | txt\_HargaSuplayer |
| 14 | JTextField6 | text | " " |
| name | txt\_HargaJual |
| 15 | JTextField7 | text | " " |
| name | txt\_Jumlah |
| 16 | JtextArea1 | name | memo |

|  |  |  |  |
| --- | --- | --- | --- |
| 17 | JComboBox | model | --Satuan--, Unit, Kardus,  Kodi. |
| name | cmb\_satuan |
| 18 | JButton1 | text | "CEK" |
| name | btn\_Cek |
| 19 | JButton2 | Text | Barang Masuk |
| Name | BarangMasuk |

*Source code* pada button BarangMasuk:

private void BarangMasukActionPerformed(java.awt.event.ActionEvent evt)

{ int

berhasil=0;

BarangMasuk brgMasuk = new BarangMasuk(); brgMasuk.harga\_jual =

Integer.parseInt(txt\_HargaJual .getText()); brgMasuk.harga\_suplayer =

Integer.parseInt(txt\_HargaSuplayer .getText());

int jum = Integer.parseInt(txt\_Jumlah.getText()); memo.append("Kode Barang

:"+Integer.toString(brgMasuk.kdBarang)+"\n"); memo.append("Nama Barang

:"+brgMasuk.NmBarang+"\n"); memo.append("Nama Suplayer

:"+brgMasuk.Suplayer+"\n"); memo.append("Harga Suplayer

:"+brgMasuk.harga\_suplayer+"\n"); memo.append("Harga Jual

:"+brgMasuk.harga\_jual+"\n");

switch (cmb\_satuan.getSelectedIndex()) { case 0:

JOptionPane.showMessageDialog(null, "Masukkan Satuan Barang");

break; case 1:

memo.append("Jumlah Barang : "+Integer.toString(brgMasuk.Jumlah(1, jum)));

berhasil = 1; break;

case 2:

memo.append("Jumlah Barang : "+Integer.toString(brgMasuk.Jumlah(2, jum)));

berhasil = 1; break;

case 3:

berhasil = 1; memo.append("Jumlah Barang :

"+Integer.toString(brgMasuk.Jumlah(3, jum))); default:

break;

}

if(berhasil == 1){

JOptionPane.showMessageDialog(null, "Barang berhasil dimasukkan");

}

}

*Source code* pada button Cek:

private void btn\_CekActionPerformed(java.awt.event.ActionEvent evt) {

// TODO add your handling code here: BarangMasuk brgMasuk = new BarangMasuk(); if(

txt\_Search.getText().equalsIgnoreCase("12345")|| txt\_Search.getText().equalsIgnoreCase("Kampas rem ysp")){

JOptionPane.showMessageDialog(null, "Barang Ditemukan"); ID\_Barang.setText(Integer.toString(brgMasuk.kdBarang));

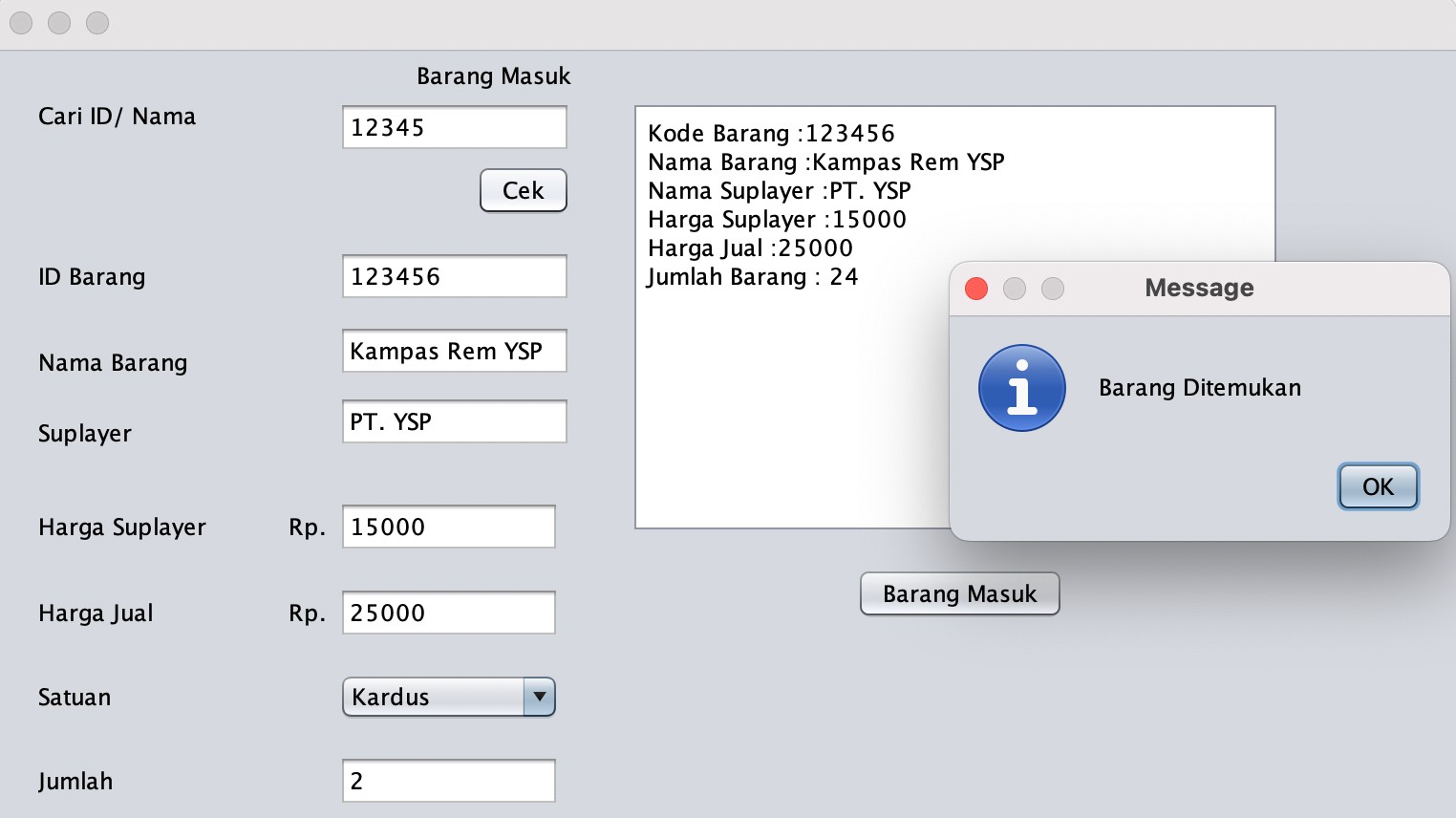
txt\_NmBarang.setText(brgMasuk.NmBarang); txt\_Suplayer.setText(brgMasuk.Suplayer);

}else{

JOptionPane.showMessageDialog(null, "Barang belum di tambahkan!!!!");

} }

### Tampilan Hasil :



### Gambar 3.9 Hasil tampilan GUI\_BarangMasuk

### Analisa :

### Barang masuk digunakan untuk menginputkan jumlah stock dari barang , maka dari itu antara barang masuk dan barang saling berelasi. Ketika *user* menginputkan ID/Nama dari Barang jika sudah ada data pada barang maka otomatis akan menampilkan barang ditemukan dan *field* akan otomatis terisi jika barang belum ada *user* harus menambahkan barang baru di GUI\_barang. Kemudian *user* diminta untuk memasukkan harga dari *suplayer*, harga jual kemudian satuan dan jumlah.

*Source code* pada *Class* BarangKeluar.java :

public class BarangKeluar extends BarangMasuk{ int jml\_keluar,TotHargaBarang;

int barangKeluar(int jumlah){ jml\_keluar = this.jumlah - jumlah; return jml\_keluar;

}

int HargaTotalBarang(int pilih,int harga,int jumlah\_beli){

if(pilih == 1){

TotHargaBarang = harga \* jumlah\_beli;

}else if(pilih == 2){

TotHargaBarang = (jumlah\_beli \* 12) \* harga;

}else if(pilih == 3){

TotHargaBarang = (jumlah\_beli \* 24) \* harga;

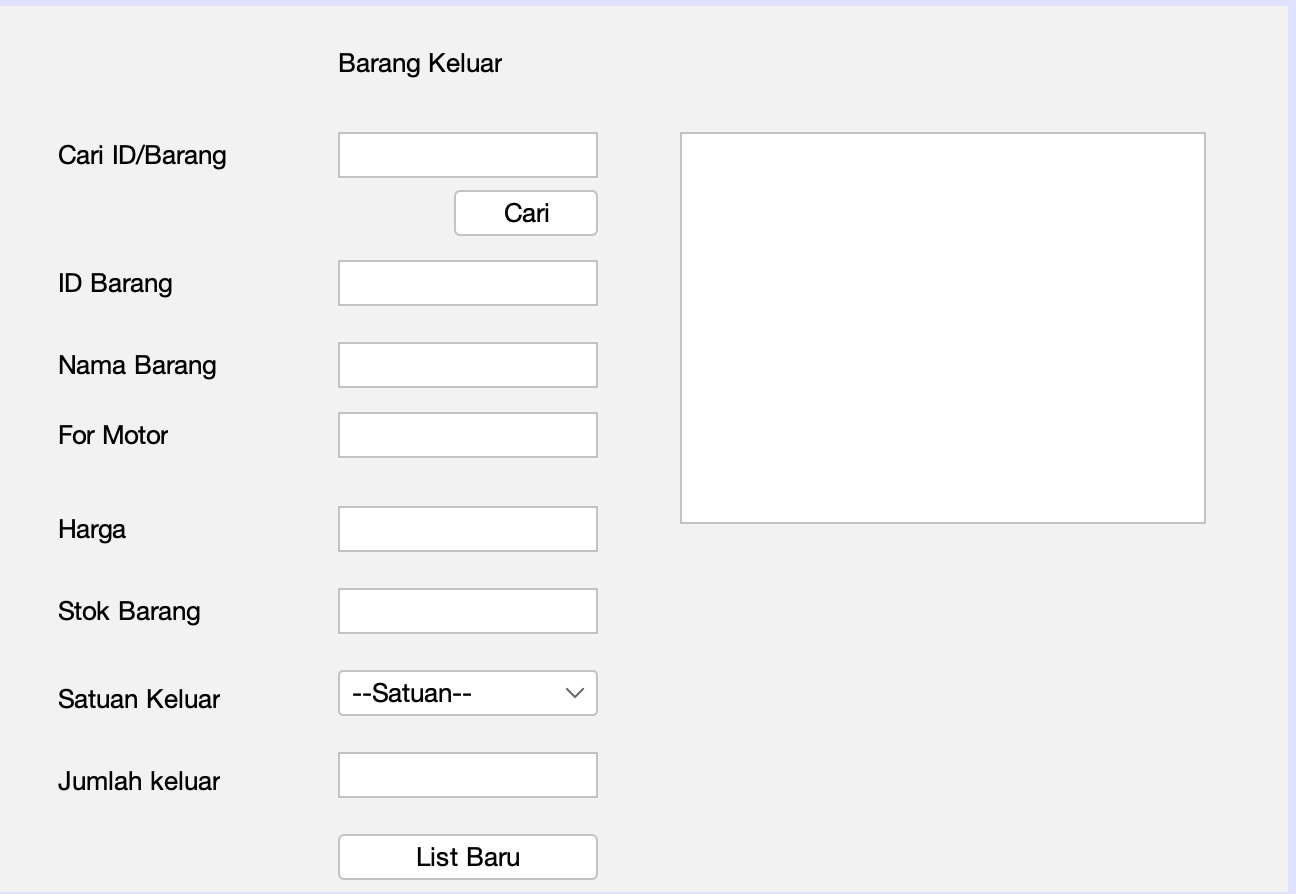
}

return TotHargaBarang;

}

}

### Desain *form* (GUI\_BarangKeluar.java):



### Gambar 3.10 Desain *form* GUI\_ DataPenerimaBeasiswa Tabel 3.5 Properti Desain GUI\_DataPenerimaBeasiswa.java

|  |  |  |  |
| --- | --- | --- | --- |
| **No** | **Objek** | **Properti** | **Nilai** |
| 1 | jLabel1 | text | "Cari ID/Nama" |
| 2 | jLabel2 | text | "ID Barang" |
| 3 | jLabel3 | text | "Nama Barang" |
| 4 | jLabel4 | text | "For Motor" |
| 5 | jLabel5 | text | "Harga" |
| 6 | jLabel6 | text | "Stock Barang" |
| 7 | jLabel7 | text | "Satuan keluar" |

|  |  |  |  |
| --- | --- | --- | --- |
| 8 | jLabel8 | text | "Jumlah Keluar" |
| 9 | jLabel8 | text | “Barang Keluar” |
| 9 | JTextField1 | text | " " |
| name | txt\_Search |
| 10 | JTextField2 | text | " " |
| name | ID\_Barang |
| 11 | JTextField3 | text | " " |
| name | txt\_NmBarang |
| 12 | JTextField4 | text | " " |
| name | txt\_ForMotor |
| 13 | JTextField5 | text | " " |
| name | txt\_Harga |
| 14 | JTextField6 | text | " " |
| name | txt\_StokBarang |
| 15 | JTextField7 | text | " " |
| name | txt\_Jml |
| 16 | JtextArea1 | name | memo |
| 17 | JComboBox | model | --Satuan--,Unit,Kardus,  Kodi. |
| name | cmb\_satuan |
| 18 | JButton1 | text | "CEK" |
| name | btn\_cek |
| 19 | JButton2 | Text | List Baru |
| Name | btn\_proses |

### Source code public (GUI\_BarangKeluar) :

public GUI\_BarangKeluar() { initComponents(); ID\_Barang.setEnabled(false); txt\_NmBarang.setEnabled(false); txt\_ForMotor.setEnabled(false); txt\_Harga.setEnabled(false); txt\_StokBarang.setEnabled(false);

}

### Source code pada Button list baru(GUI\_BarangKeluar) :

private void btn\_prosesActionPerformed(java.awt.event.ActionEvent evt)

{

// TODO add your handling code here: BarangMasuk brgMasuk = new BarangMasuk(); BarangKeluar brgKeluar = new BarangKeluar(); int ID = Integer.parseInt(ID\_Barang.getText()); String nama = txt\_NmBarang.getText();

int harga = Integer.parseInt(txt\_Harga.getText()); int stok =

Integer.parseInt(txt\_StokBarang.getText());

int jumlah = Integer.parseInt(txt\_Jml.getText()); memo.append("Kode Barang

:"+Integer.toString(ID)+"\n"); memo.append("Nama Barang :"+nama+"\n");

memo.append("Harga satuan Barang :"+harga+"\n"); switch (cmb\_Satuan.getSelectedIndex()) {

case 0:

JOptionPane.showMessageDialog(null, "Masukkan Satuan Barang");

break; case 1:

memo.append("Harga total:"+brgKeluar.HargaTotalBarang(1,harga, jumlah)+"\n");

break; case 2:

memo.append("Harga total:"+brgKeluar.HargaTotalBarang(2,harga, jumlah)+"\n");

break; case 3:

memo.append("Harga total:"+brgKeluar.HargaTotalBarang(3,harga, jumlah)+"\n");

default:

break;

}

memo.append("Stok Barang tersedia

:"+brgKeluar.barangKeluar(jumlah)+"\n");

}

### Source code pada Button cek(GUI\_BarangKeluar) :

|  |  |  |
| --- | --- | --- |
| private void  btn\_cekActionPerformed(java.awt.event.ActionEvent evt) {  // TODO add your handling code here: BarangMasuk brgMasuk = new BarangMasuk(); if(  txt\_Search.getText().equalsIgnoreCase("12345")|| txt\_Search.getText().equalsIgnoreCase("Kampas rem ysp")){  JOptionPane.showMessageDialog(null, "Barang  Ditemukan");  ID\_Barang.setText(Integer.toString(brgMasuk.kdBarang)); txt\_NmBarang.setText(brgMasuk.NmBarang); txt\_ForMotor.setText(brgMasuk.ForMotor);  txt\_Harga.setText(Integer.toString(brgMasuk.Harga)); | | |
|  | Nama Aslab Pengajar : Silva Anjelina Tambunan | TTD : |
| Tanggal : |

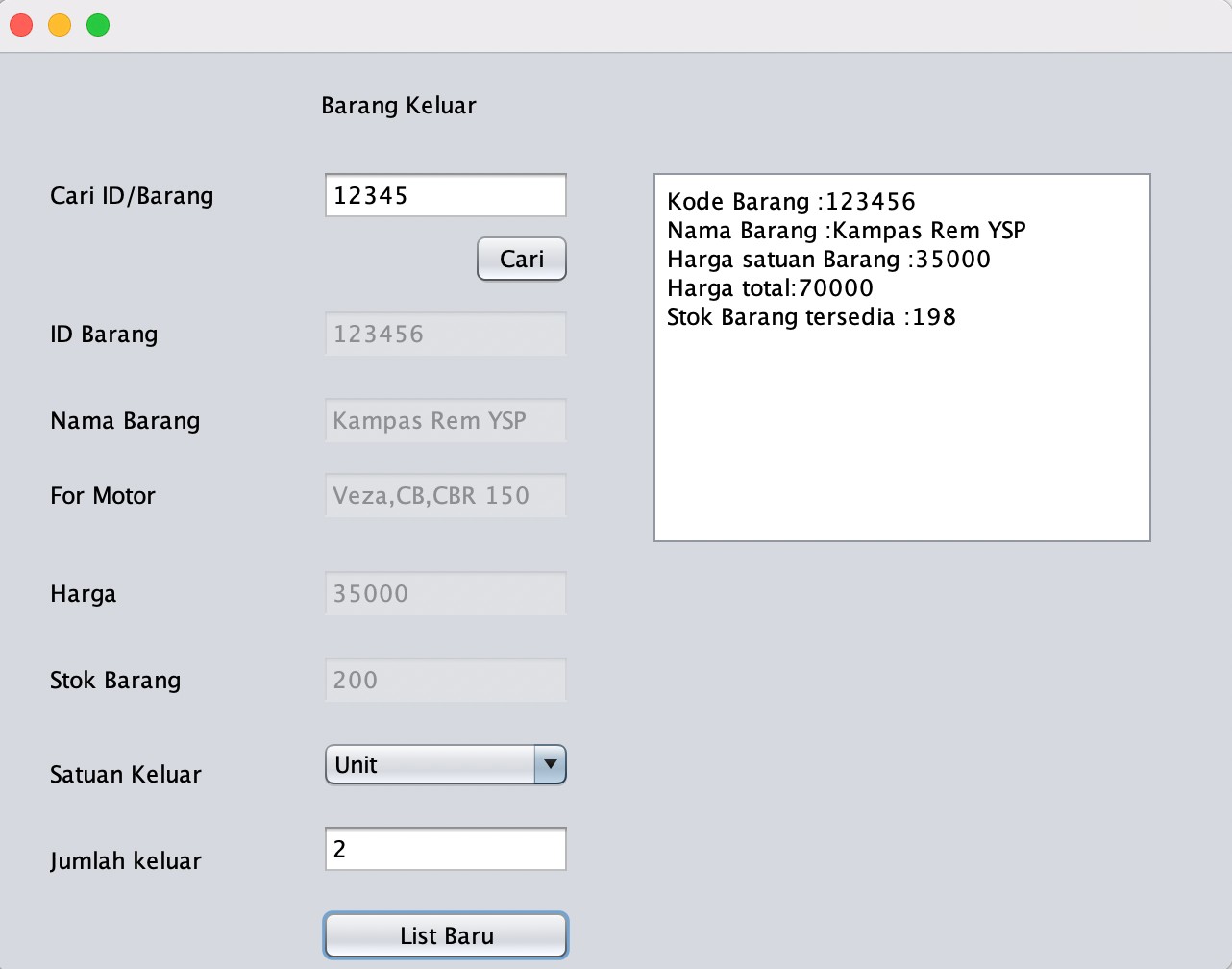
txt\_StokBarang.setText(Integer.toString(brgMasuk.jumlah));

}else{

JOptionPane.showMessageDialog(null, "Barang belum di tambahkan!!!!");

}}

### Tampilan Hasil :



### Gambar 3.11 Tampilan GUI\_BarangKeluar

### Analisa :

### Barang keluar digunakan untuk mendata barang yang keluar, oleh karena itu barang masuk dan barang keluar saling berelasi karena berhubungan dengan stock barang. Pada barang keluar ketika ID/Nama dari barang di inputkan akan otomatis mencari ketika barang ditemukan maka beberapa field otomatis akan terisi karena diambil dari data barang masuk, user tinggal menginputkan satuan dari barang yang akan dibeli dan jumlah.

## Kesimpulan

1. *Inheritance* atau Pewarisan/Penurunan adalah konsep pemrograman di mana 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.
2. *Konsktruktor* adalah *method* yang pertama kali dijalankan pada saat sebuah objek pertama kali diciptakan, yang di mana syarat penulisan *method konstruktor* ini harus sama dengan nama *class* yang di naungi.

### Bentuk sederhana *konsep* pewarisan dalam kehidupan sehari-hari misalnya terdapat class (super class) Bagun\_Datar yang memiliki *method* luas dan keliling.

|  |  |
| --- | --- |
| Nama Aslab Pengajar : Silva Anjelina Tambunan | TTD : |
| Tanggal : |

**BAB IV**

**ENCAPSULASI, OVERIDDING 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.   1. 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 *Enkapsulasi*

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

### Dengan enkapsulasi, maka programmer 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 di *publish*.

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

|  |  |
| --- | --- |
| Nama Aslab Pengajar :  Silva Anjelina Tambunan | TTD : |
| Tanggal : |

### Jenis-jenis enkapsulasi (Hak Akses)

#### 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.

### Bentuk cara deklarasi hak akses *Public*:

public String atribut1; **//pada atribut**

public int Luas(){…} **// pada methode**

#### 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.

### Bentuk cara deklarasi hak akses *Private*:

private String atribut1; **//pada atribut**

private int Luas(){…} **// pada methode**

#### 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.

Bentuk cara deklarasi hak akses *Protected*:

protected String atribut1; **//pada atribut**

protected int Luas(){…} **// pada methode**

### 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)

### Bentuk cara deklarasi hak akses *Default*:

String atribut1; **//pada atribut**

int Luas(){…} **// pada methode**

### Tabel 4.1 Hak akses Enkapsulasi

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Hak Akses** | **Public** | **Protected** | **Default** | **Private** |
| Dari kelas yang sama |  |  |  |  |
| Dari kelas lain didalam paket |  |  |  | × |
| Dari kelas lain diluar paket |  | × | × | × |
| Dari kelas turunan didalam paket |  |  |  | × |
| Dari kelas turunan diluar paket |  |  | × | × |

#### Overloading dan Overrding.

### 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. Tipe data dari parameter*.*

### Contoh:

public class contohOverloading { public void jumlah (int a, int b){

System.out.println("Jumlah 2 angka ="+ (a + b));

}

**//overloading perbedaan jumlah parameter**

public void jumlah (int a, int b, int c){ System.out.println("Jumlah 3 angka =" + (a + b

+ c));

}

**//overloading perbedaan tipe data parameter**

public void jumlah (double a, int b){

System.out.println("Jumlah 2 angka (double+int)= "+ (a + b));

}

**//overloading perbedaan urutan tipe data parameter**

public void jumlah (int b, double a){

System.out.println("Jumlah 2 angka (int+double)= "+ (a + b));

}

}

### Overriding

*Overriding method* adalah kemampuan dari *subclass* 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.

Aturan *Overriding*:

1. Parameter yang terdapat pada *method Overriding* di subclass harus sama dengan parameter yang terdapat pada *parent class*.
2. Aturan hak akses, hak akses *method Overriding* di *subclass* tidak boleh lebih ketat di bandingkan dengan hak akses *method* pada *parent class*.

## Langkah – Langkah Praktikum

### Buka Aplikasi Netbeans.

### Buat Class.

### Memberi script pada kelas tersebut.

### Buat form.

### Memberi script pada form tersebut.

### Menjalankan program.

## Tugas Praktikum ke-1:

### Implementasi Enkapsulasi Pada Projek Praktikum “Class Penilaian” Diagram Class (Penilaian.java):

### Class Attribute

|  |
| --- |
| Penilaian |
| * String NIM, kode\_mk; * Int nilaiTugas, nilaiPrak; |
| * Double nilaiAkhirPrak() * Double tampilNA() * doubleAkhir() |

### Method

|  |
| --- |
| Penilaian |
| * Double nilaiAkhir() |

### Class Method

*Source code* pada *Class* Penilaian (Super Class):

public class Penilaian { String NIM,nama,kode\_mk;

int nilaiTugas, nilaiPrak; double nilaiAkhirPrak(){

return ((nilaiTugas\*0.6)+(nilaiPrak\*0.4));

}

double tampilNA(){

return nilaiAkhirPrak();

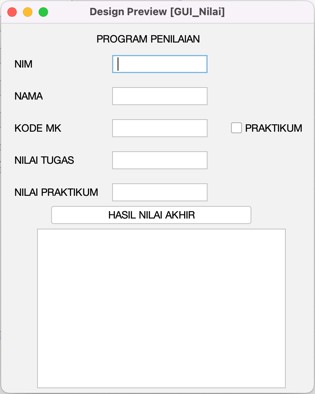
}

double nilaiAkhir(){ return 0;

}

}

Desain *form* GUI\_Nilai:



### Gambar 4.1 Desain GUI\_Penilaian.java Tabel 4.2 Properti desain GUI\_Penilaian

|  |  |  |  |
| --- | --- | --- | --- |
| **No** | **Objek** | **Properti** | **Nilai** |
| 1 | jLabel1 | Text | PROGRAM PENILAIAN |
| 2 | jLabel2 | Text | NIM |
| 3 | jLabel3 | Text | Nama |
| 4 | jLabel4 | Text | Kode MK |
| 5 | jLabel5 | Text | Nilai Tugas |
| 6 | jLabel6 | Text | Nilai Praktikum |
| 7 | jTextField1 | Name | txtNIM |
| Text |  |

|  |  |  |  |
| --- | --- | --- | --- |
| 8 | jTextField2 | Name | txtNama |
| Text |  |
| 9 | jTextField3 | Name | txtKodeMK |
| Text |  |
| 10 | jTextField4 | Name | txtNT |
| Text |  |
| 11 | jTextField5 | Name | txtNP |
| Text |  |
| 12 | jButton1 | Name | btnNA |
| Text | Hasil Nilai Akhir |
| 13 | jTextArea | Name | memoNilai |
| Text |  |
| 14 | JCheckBox | Text | Praktikum |
| Name | cek\_prak |

*Source code* Jbutton pada GUI\_Nilai :

private void btnNAActionPerformed(java.awt.event.ActionEvent evt) {

// TODO add your handling code here: memoNilai.setText("");

Penilaian\_NonPrak nilai = new Penilaian\_NonPrak(); nilai.NIM = txtNIM.getText();

nilai.nama = txtNama.getText(); nilai.kode\_mk = txtKodeMK.getText(); nilai.nilaiTugas =

Integer.parseInt(txtNT.getText()); nilai.nilaiPrak =

Integer.parseInt(txtNP.getText()); if (cek\_prak.isSelected()){

memoNilai.append("Nilai Akhir Mata Kuliah

\n");

memoNilai.append("

\n");

memoNilai.append("NIM "+nilai.NIM +"\n");

memoNilai.append("Nama "+nilai.nama +"\n");

memoNilai.append("Kode MK "+nilai.kode\_mk +"\n");

memoNilai.append("Nilai Tugas

:

:

:

:

"+nilai.nilaiTugas +"\n");

memoNilai.append("Nilai Praktikum : "+nilai.nilaiPrak +"\n");

memoNilai.append("

\n");

memoNilai.append("Nilai Akhir "+nilai.tampilNA());

:

}else{

memoNilai.append("Nilai Akhir Mata Kuliah

\n");

memoNilai.append("

\n");

memoNilai.append("NIM "+nilai.NIM +"\n");

memoNilai.append("Nama "+nilai.nama +"\n");

memoNilai.append("Kode MK "+nilai.kode\_mk +"\n");

memoNilai.append("Nilai Tugas

:

:

:

:

"+nilai.nilaiTugas +"\n");

memoNilai.append("

\n");

memoNilai.append("Nilai Akhir : "+nilai.nilaiAkhir());} }

*Source code* cek box pada GUI\_Nilai:

private

void

cek\_prakActionPerformed(java.awt.event.ActionEvent evt) {

// TODO add your handling code here: if (cek\_prak.isSelected()){

txtNP.setEnabled(true);

}

else{

txtNP.setEnabled(false);

}

}

*Source code* pada class (Penilaian\_NonPrak.java):

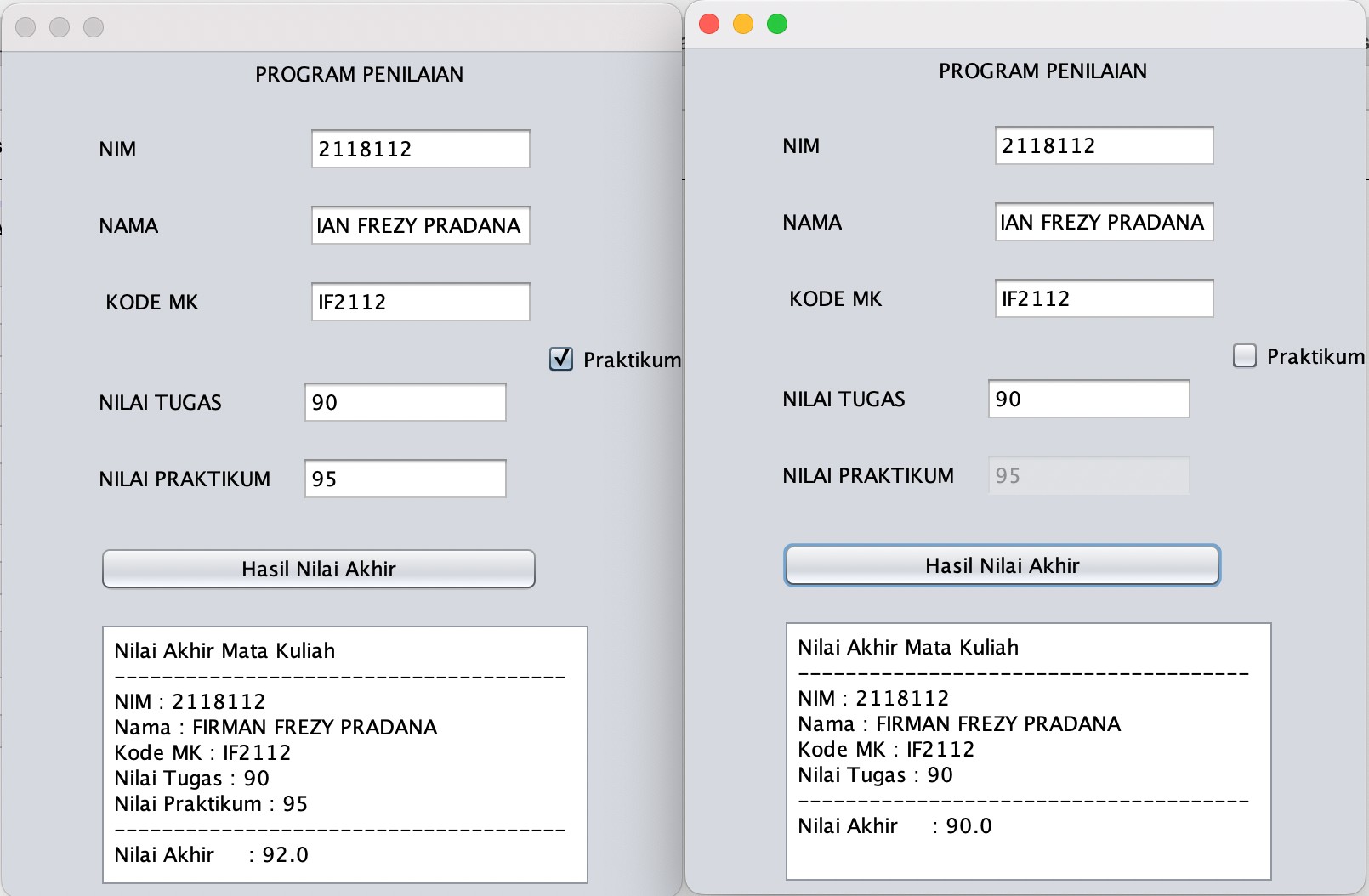
public class Penilaian\_NonPrak extends Penilaian { @Override

double nilaiAkhir(){ return (nilaiTugas);

}

}

### Hasil Tampilan GUI\_Nilai :



### Gambar 4.2 Hasil Tampilan GUI\_Penilaian

### Analisa :

### Pada program tersebut untuk menampilkan nilai akhir. Pada kelas anak terdapat *method* di mana tidak dimiliki pada sub *class* yaitu *double* nilaiakhir(), di mana di dalamnya akan mengembalikan data dari atribut nilaitugas yang terdapat di *class induk*.berbeda Ketika matkulnya terdapat praktikum, maka kelas anak akan mendapat turunan *method* dari *class* penilaian yaitu *method double* nilaiakhirprak(); sesuai dengan yang di perintahkan yaitu implementasi *inheritance* , *inheritance* sendiri adalah *Inheritance* atau Pewarisan/Penurunan adalah konsep pemrograman di mana sebuah *class* dapat ‘menurunkan’ properti dan *method* yang dimilikinya kepada *class* lain. Pada kasus program di atas. Sudah menerapkan syarat tersebut.

## Tugas praktikum ke-2:

### Mendesain ulang GUI\_Mahasiswa.java + membuat method batal() Diagram Class (Class Mahasiswa):

 Class Attribute

|  |
| --- |
| Mahasiswa |
| * String nim, nama, prodi, angktn; |
| + void dataNIM(String Nim)  + void dataNama(String Nama)  + void dataProdi(String Prodi)  + void dataAngkatan(String Angktn)  + String cetakNIM()  + String cetakNama()  + String cetakProdi()  + String cetakAngkatan() |

### Method

*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;

}

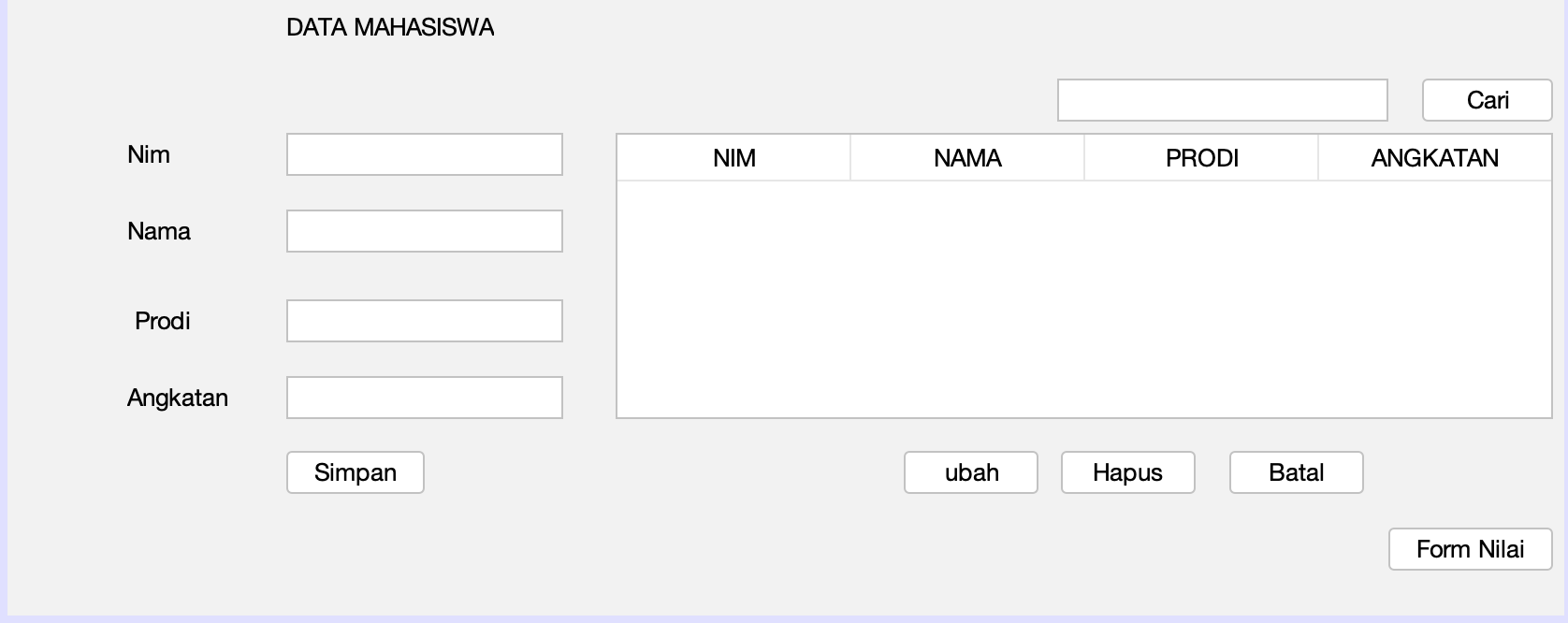
String cetakProdi(){ return prodi;

}

String cetakAngkatan(){ return angktn;

}}

### Desain *form* (GUI\_Mahasiswa.java) :



### Gambar 4.3 Desain from GUI\_Mahasiswa.java Tabel 4.3 Properti Desain GUI\_Nilai

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **No** | **Objek** | | **Properti** | **Nilai** | |
| 1 | jLabel1 | | Text | DATA MAHASISWA | |
| 2 | jLabel2 | | Text | NIM | |
| 3 | jLabel3 | | Text | Nama | |
| 4 | jLabel4 | | Text | Prodi | |
| 5 | jLabel5 | | Text | Angkatan | |
| 6 | jTextField1 | | Name | txtNim | |
| Text |  | |
| 7 | jTextField2 | | Name | txtNama | |
| Text |  | |
| 8 | jTextField3 | | Name | txtProdi | |
| Text |  | |
|  | | Nama Aslab Pengajar : Silva Anjelina Tambunan | | | TTD : |
| Tanggal : | | |

|  |  |  |  |
| --- | --- | --- | --- |
| 9 | jTextField4 | Name | txtAng |
| Text |  |
| 10 | jButton1 | Name | btnKTM |
| Text | Cetak KTM |
| 11 | jTextArea | Name | memoKTM |
| Text |  |

*Source code* pada button Cetak KTM :

private void btnKTMActionPerformed(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()); mhs.dataProdi(txtprodi.getText()); mhs.dataAngkatan(txtangkatan.getText());

memoKTM.append("Kartu Tanda Mahasiswa\n"); memoKTM.append("

\n");

memoKTM.append("NIM : " + mhs.cetakNIM()

+"\n");

memoKTM.append("Nama : " + mhs.cetakNama() +

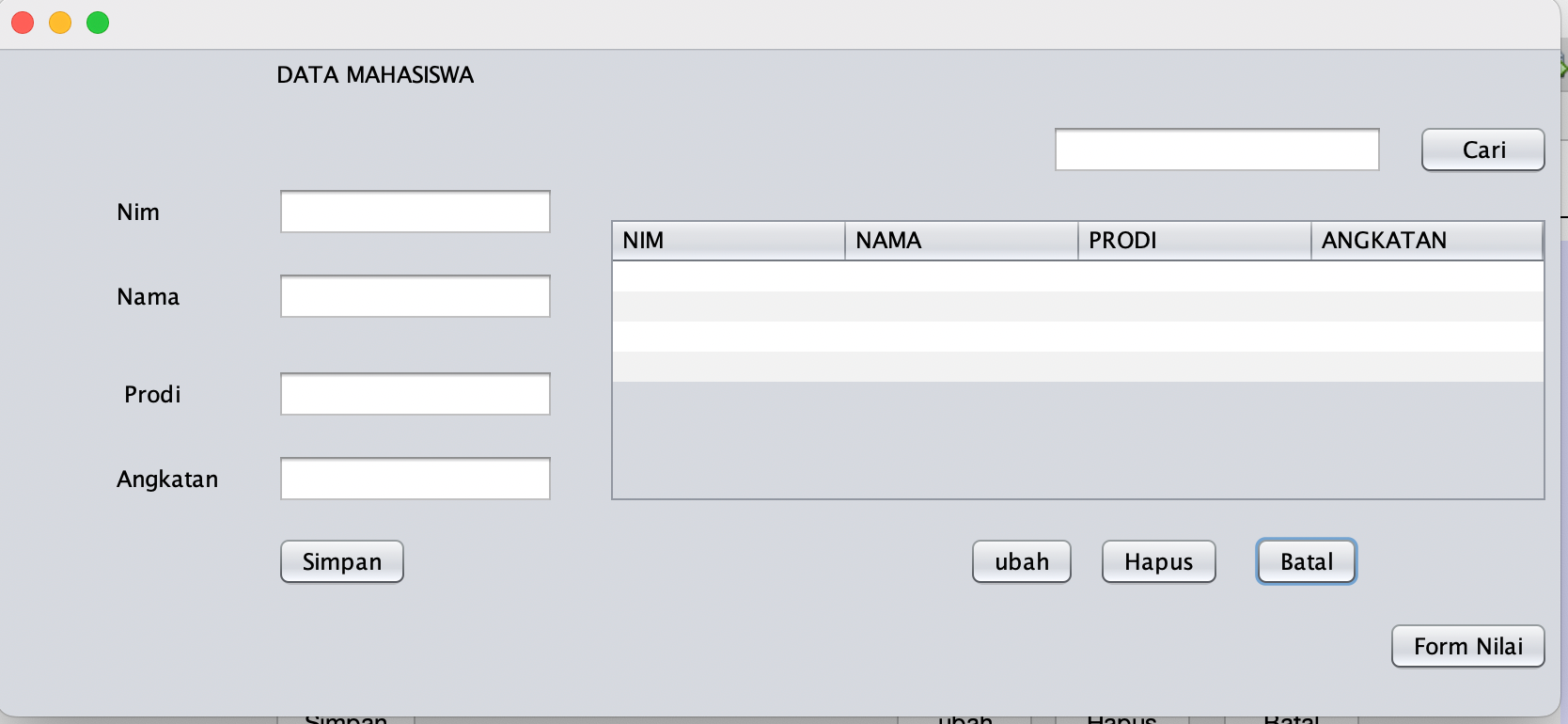
"\n");

memoKTM.append("Prodi : " + mhs.cetakProdi() +"\n");

memoKTM.append("Angkatan : " + mhs.cetakAngkatan()

+ "\n");}

### Hasil Tampilan :



### Gambar 4.4 Tampilan hasil GUI\_Mahasiswa.java

## Tugas Rumah 1 :

### Implementasi *Enkapsulasi* GUI\_Barang Judul : Inventory Barang Sparepart Motor Diagram Class :

Gambar 4.5 Gambar class diagram Barang(*Enkapsulasi*)

*Source code* pada *Super Class*(Barang.java) *:*

public class Barang {

protected int kdBarang, harga;

protected String NmBarang,Kategori,ForMotor,Suplayer; Barang(){

kdBarang = 123456 ; NmBarang = "Kampas Rem YSP";

ForMotor = "Veza,CB,CBR 150"; Suplayer= "PT. YSP";

harga = 35000;

}

public int Kode(){

Random rndm = new Random(); kdBarang = rndm.nextInt(100000); return kdBarang;

}

public String NmBarang(String NmBarang){ this.NmBarang = NmBarang;

return NmBarang;

}

public String Kategori(String Kategori){ this.Kategori = Kategori;

return Kategori;

}

public String ForMotor(String ForMotor){ this.ForMotor = ForMotor;

return ForMotor;

}

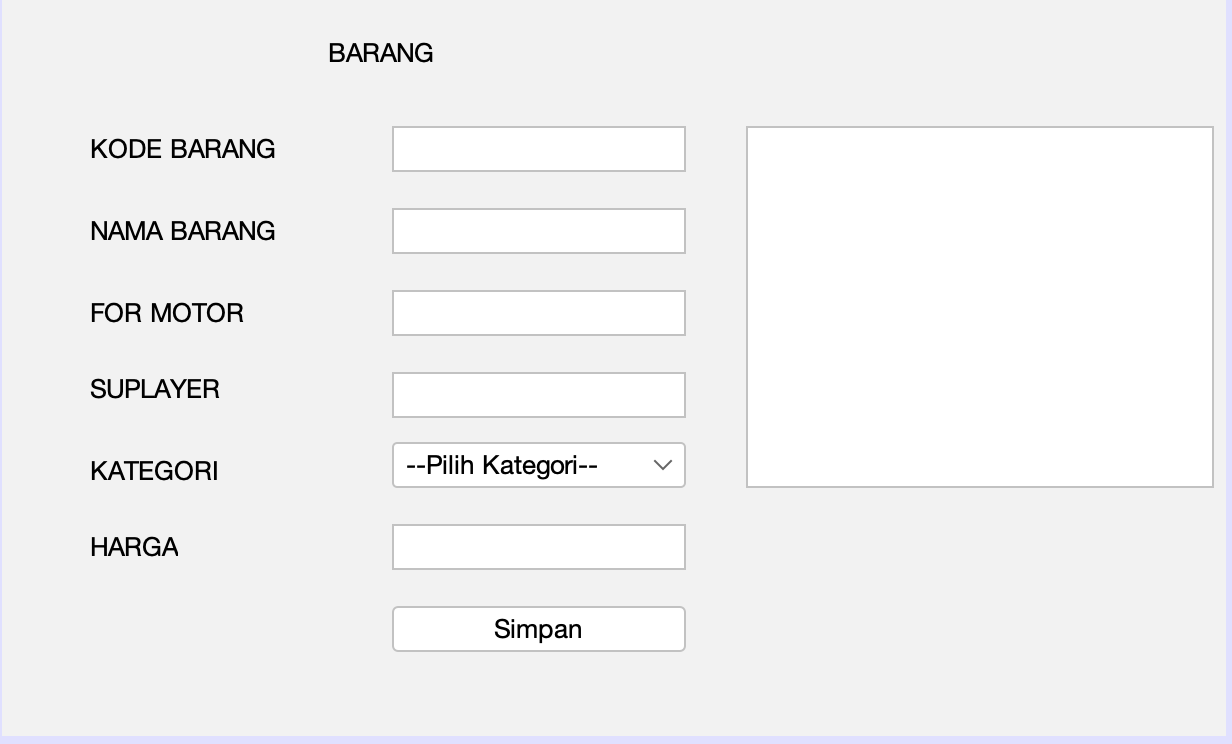
public String Suplayer(String Suplayer){ this.Suplayer = Suplayer;

return Suplayer;

}

public int Harga(int harga){ this.harga = harga; return harga;}}

### Desain *Form* (GUI\_Barang.java)



### Gambar 4.6 Desain GUI\_Barang(*Enkapsulasi*) Tabel 4.4 Properti Desain GUI\_Barang.java

|  |  |  |  |
| --- | --- | --- | --- |
| **No** | **Objek** | **Properti** | **Nilai** |
| 1 | jLabel1 | Text | “BARANG” |
| 2 | jLabel2 | Text | “KODE BARANG” |
| 3 | jLabel3 | Text | “FOR MOTOR” |
| 4 | jLabel4 | Text | “ SUPLAYER” |
| 5 | jLabel5 | Text | “KATEGORI” |
| 6 | JLabel6 | Text | “HARGA” |
| 7 | jTextField1 | Name | Txt\_KdBarang |
| Text | “ “ |
| 8 | jTextField1 | Text | “ “ |
| Name | Txt\_NmBarang |
| 9 | jTextField2 | Text | “ “ |
| Name | Txt\_ForMotor |
| 10 | jTextField3 | Text | “ “ |
| Name | Txt\_Suplayer |
| 11 | TextField5 | Name | “Txt\_Harga” |
| text | “ “ |
| 12 | JComboBox1 | Name | Cmb\_Kategori |
| Model | --Pilih Kategori--, Oli,  Part motor, Velek, |

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  | Lampu Depan, Lampu Belakang, Pack Block/BB, Pack Kalter / CC, pack Kopling /KPL,  Baut, Mur |
| 13 | jButton1 | Name | Simpan |
| Text | “Simpan” |
| 14 | jTextArea | Name | Memo |
| Text | “ “ |

### Source code pada button Simpan :

private void SimpanActionPerformed(java.awt.event.ActionEvent evt) {

// TODO add your handling code here: int Berhasil = 0,Kode;

Barang brg = new Barang(); brg.NmBarang(Txt\_NmBarang.getText()); brg.ForMotor(Txt\_ForMotor.getText()); brg.Suplayer(Txt\_Suplayer.getText()); brg.Harga(Integer.parseInt(Txt\_Harga.getText())); Kode = Integer.parseInt(Txt\_KdBarang.getText()); if(Cmb\_Kategori.getSelectedIndex()==0){

JOptionPane.showMessageDialog(null, "Kategori Belum Anda Masuukan");

}else if(Cmb\_Kategori.getSelectedIndex()==1){ Berhasil = 1;

brg.Kategori("oli");

}else if(Cmb\_Kategori.getSelectedIndex()==2){ Berhasil = 1;

brg.Kategori("Part Motor Mesin");

}else if(Cmb\_Kategori.getSelectedIndex()==3){ Berhasil = 1;

brg.Kategori("Lampu Depan");

}else if(Cmb\_Kategori.getSelectedIndex()==4){ Berhasil = 1;

brg.Kategori("Lampu Belakang");

}else if(Cmb\_Kategori.getSelectedIndex()==5){ Berhasil = 1;

brg.Kategori("Baut");

}else if(Cmb\_Kategori.getSelectedIndex()==6){ Berhasil = 1;

brg.Kategori("Mur");

}

if(Berhasil == 1){ JOptionPane.showMessageDialog(null,

"Berhasil!!!");

memo.append(" Kode Barang : " +Kode+"\n"); memo.append(" Nama Barang :

"+brg.NmBarang+"\n");

memo.append(" For Motor : "+brg.ForMotor+"\n");

memo.append(" Suplayer : "+brg.Suplayer+"\n"); memo.append(" Kategori : "+brg.Kategori+"\n"); memo.append(" Harga :Rp"+brg.harga);

}else{

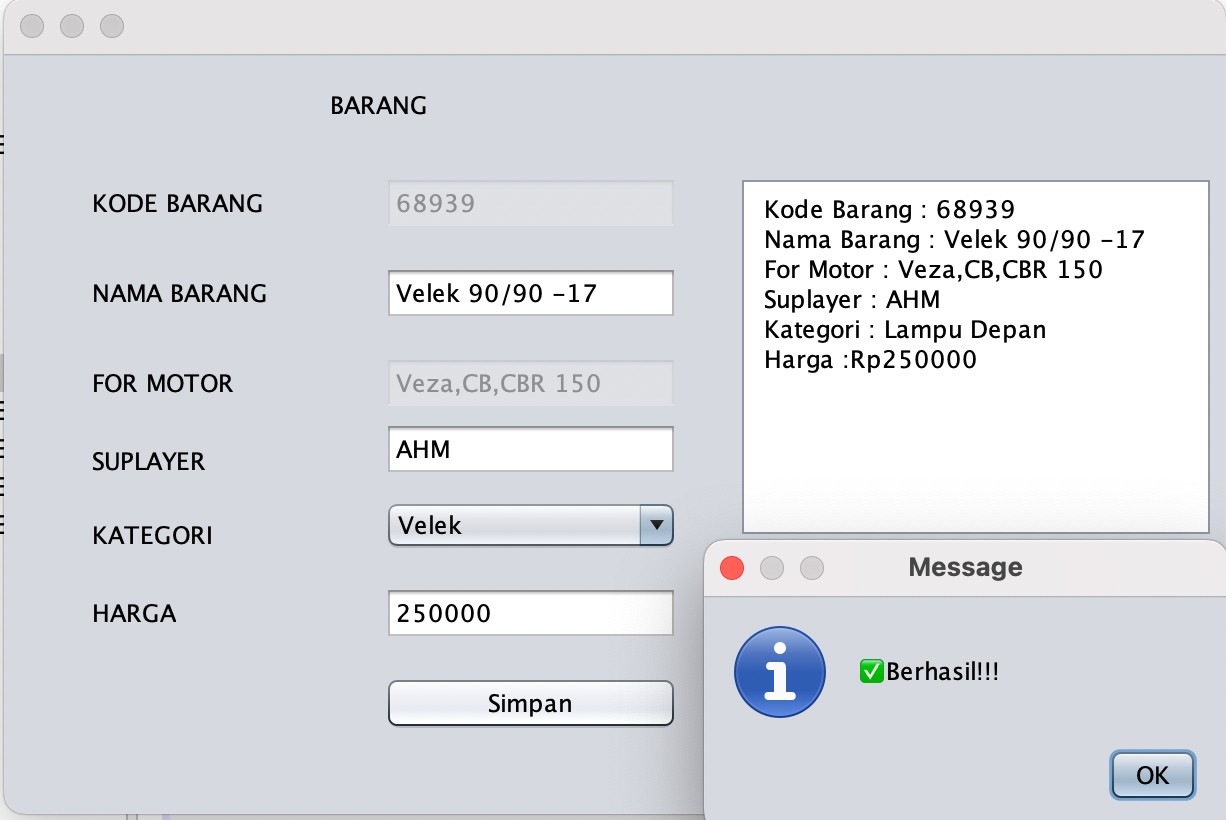
JOptionPane.showMessageDialog(null,

"Gagal!!!");

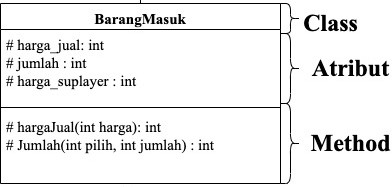
}

}

### Tampilan Hasil:



Gambar 4.7 Hasil Tampilan GUI\_Barang.java(*Enkapsulasi*) Diagaram *Class*:



Gambar 4.8 class diagram Barangmasuk(*Enkapsulasi*)

*Source code* pada *Class* BarangMasuk.java:

public class BarangMasuk extends Barang{ protected int harga\_jual;

protected int jumlah; protected int harga\_suplayer; public BarangMasuk(){

harga\_suplayer = 45000;

jumlah = 200;

}

protected int HargaJual(int harga){ this.harga\_jual = harga; return harga;

}

protected int Jumlah(int pilih,int jumlah){ if(pilih == 1){

jumlah = jumlah;

}else if(pilih == 2){ jumlah = jumlah \* 12;

}else if(pilih == 3){ jumlah = jumlah \* 24;

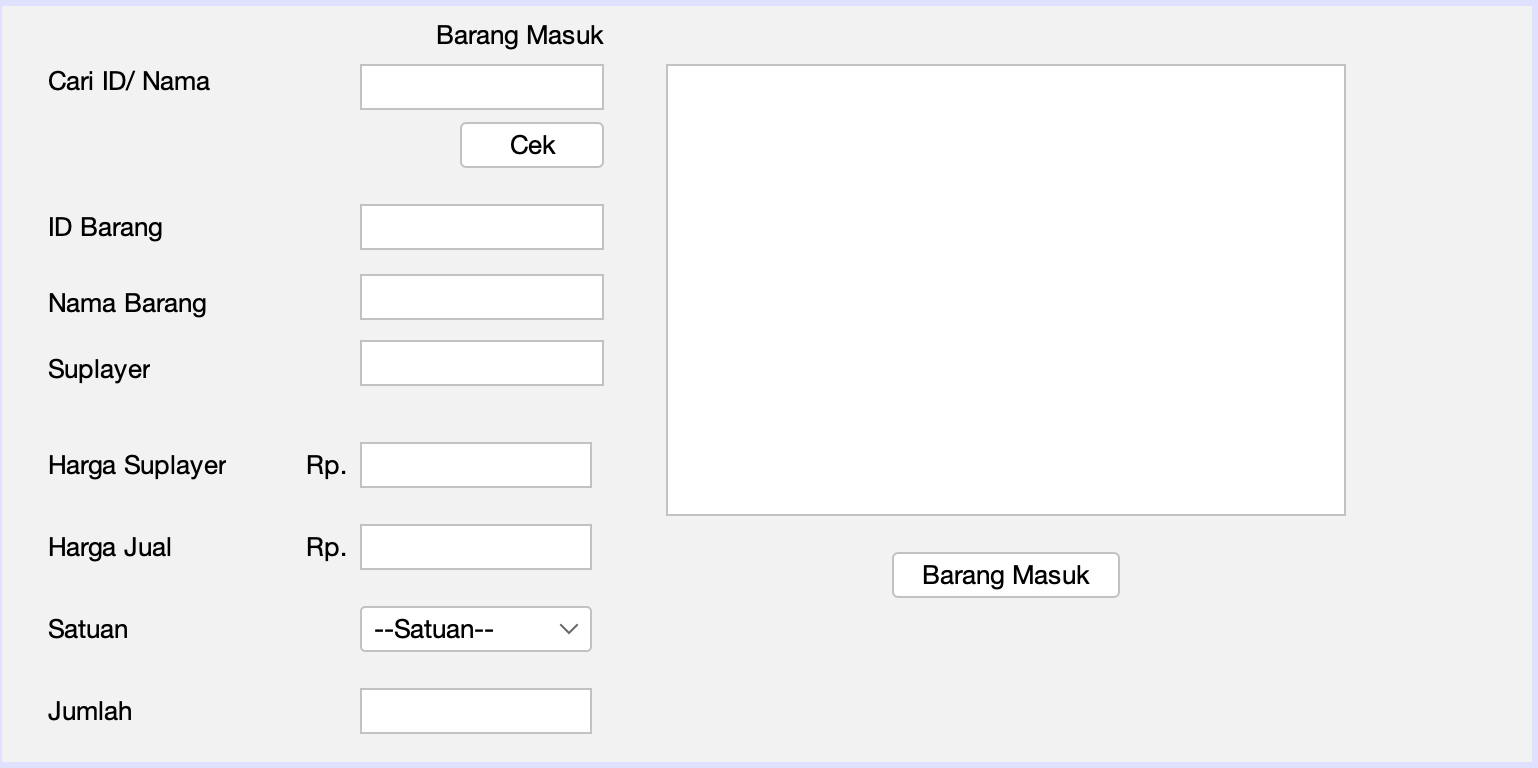
}

return jumlah;

}

}

### Desain *form* (GUI\_.BarangMasuk.java) :



### Gambar 4.9 Desain GUI\_BarangMasuk.java(*Enkapsulasi*) Tabel 4.5 Tabel Property Design GUI\_BarangMasuk

|  |  |  |  |
| --- | --- | --- | --- |
| **No** | **Objek** | **Properti** | **Nilai** |
| 1 | jLabel1 | Text | “Barang Masuk” |
| 2 | jLabel2 | Text | “Cari ID/Nama” |
| 3 | jLabel3 | Text | “ID Barang” |
| 4 | jLabel4 | Text | “Nama Barang” |
| 5 | jLabel5 | Text | “Suplayer” |
| 6 | jLabel6 | Text | “Harga Suplayer” |
| 7 | jLabel7 | Text | “Harga Jual” |
| 8 | jLabel8 | Text | “Satuan” |
| 9. | jLabel9 | Text | “Jumlah” |
| 10 | jTextField1 | Name | txt\_Search |
| Text |  |

|  |  |  |  |
| --- | --- | --- | --- |
| 11 | jTextField2 | Name | ID\_Barang |
| Text |  |
| 12 | jTextField3 | Name | txt\_NmBarang |
| Text |  |
| 13 | jTextField4 | Name | txt\_Suplayer |
| Text |  |
| 14 | jTextField6 | Name | txt\_HargaSuplayer |
| Text |  |
| 15 | jTextField7 | name | txt\_HargaJual |
| text |  |
| 16 | jTextField8 | neme | txt\_Jumlah |
| text |  |
| 17 | JComboBox | text |  |
| Name | cmb\_satuan |
| 18 | jButton1 | Name | btnCetak |
| Text | Cek |
| 19 | jButton2 | Neme | BarangMasuk |
| text | “BarangMasuk” |
| 20 | jTextArea | Name | memo |
| Text |  |

*Source code* pada *button* Cek :

private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {

// TODO add your handling code here: BarangMasuk brgMasuk = new BarangMasuk(); if(

txt\_Search.getText().equalsIgnoreCase("12345")|| txt\_Search.getText().equalsIgnoreCase("Kampas rem ysp")){

JOptionPane.showMessageDialog(null, "Barang

Ditemukan");

ID\_Barang.setText(Integer.toString(brgMasuk.kdBarang)); txt\_NmBarang.setText(brgMasuk.NmBarang); txt\_Suplayer.setText(brgMasuk.Suplayer);

}else{

JOptionPane.showMessageDialog(null, "Barang belum di tambahkan!!!!");

} }

*Source code* pada *button* Barang Masuk.

private void BarangMasukActionPerformed(java.awt.event.ActionEvent evt)

{

// TODO add your handling code here: int berhasil=0;

BarangMasuk brgMasuk = new BarangMasuk(); brgMasuk.harga\_jual =

Integer.parseInt(txt\_HargaJual .getText()); brgMasuk.harga\_suplayer =

Integer.parseInt(txt\_HargaSuplayer .getText());

int jum = Integer.parseInt(txt\_Jumlah.getText()); memo.append("Kode Barang

:"+Integer.toString(brgMasuk.kdBarang)+"\n"); memo.append("Nama Barang

:"+brgMasuk.NmBarang+"\n"); memo.append("Nama Suplayer

:"+brgMasuk.Suplayer+"\n"); memo.append("Harga Suplayer

:"+brgMasuk.harga\_suplayer+"\n"); memo.append("Harga Jual

:"+brgMasuk.harga\_jual+"\n");

switch (cmb\_satuan.getSelectedIndex()) { case 0:

JOptionPane.showMessageDialog(null, "Masukkan Satuan Barang");

break; case 1:

memo.append("Jumlah Barang : "+Integer.toString(brgMasuk.Jumlah(1, jum)));

berhasil = 1; break;

case 2:

memo.append("Jumlah Barang : "+Integer.toString(brgMasuk.Jumlah(2, jum)));

berhasil = 1; break;

case 3:

berhasil = 1; memo.append("Jumlah Barang :

"+Integer.toString(brgMasuk.Jumlah(3, jum))); default:

break;

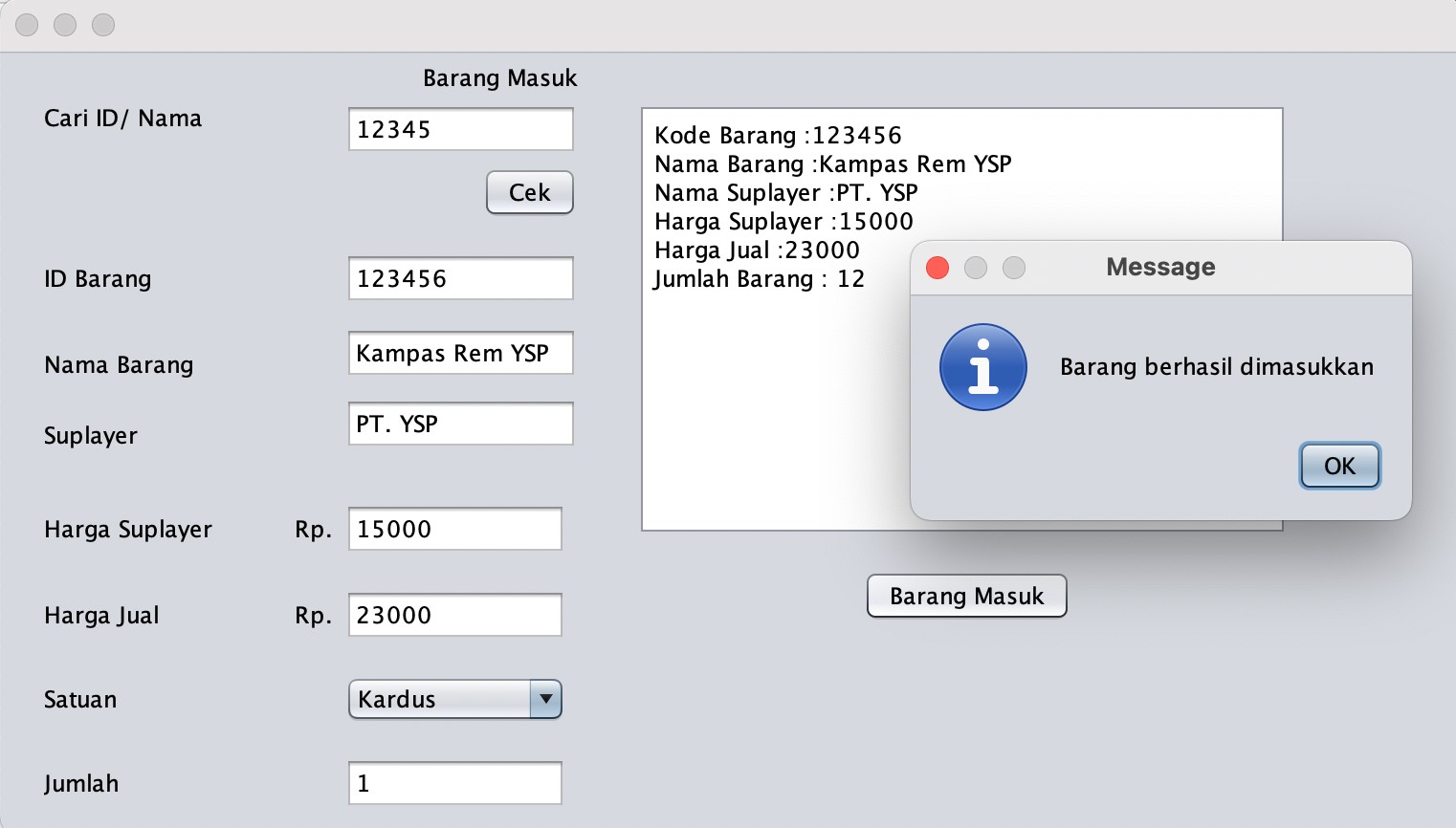
}

if(berhasil == 1){

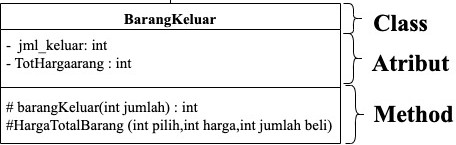
JOptionPane.showMessageDialog(null, "Barang berhasil dimasukkan");

} }

### Tampilan Hasil :



Gambar 4.10 Hasil GUI\_BarangMasuk.java(*Enkapsulasi*) Diagram *class* Barang Keluar:



### Gambar 4.11 Class diagram barang keluar

*Source code* pada class Barang Keluar :

public class BarangKeluar extends BarangMasuk{ private int jml\_keluar;

private int TotHargaBarang;

protected int barangKeluar(int jumlah){ jml\_keluar = this.jumlah - jumlah; return jml\_keluar;

}

protected int HargaTotalBarang(int pilih,int harga,int jumlah\_beli){

if(pilih == 1){

TotHargaBarang = harga \* jumlah\_beli;

}else if(pilih == 2){

TotHargaBarang = (jumlah\_beli \* 12) \* harga;

}else if(pilih == 3){

TotHargaBarang = (jumlah\_beli \* 24) \* harga;

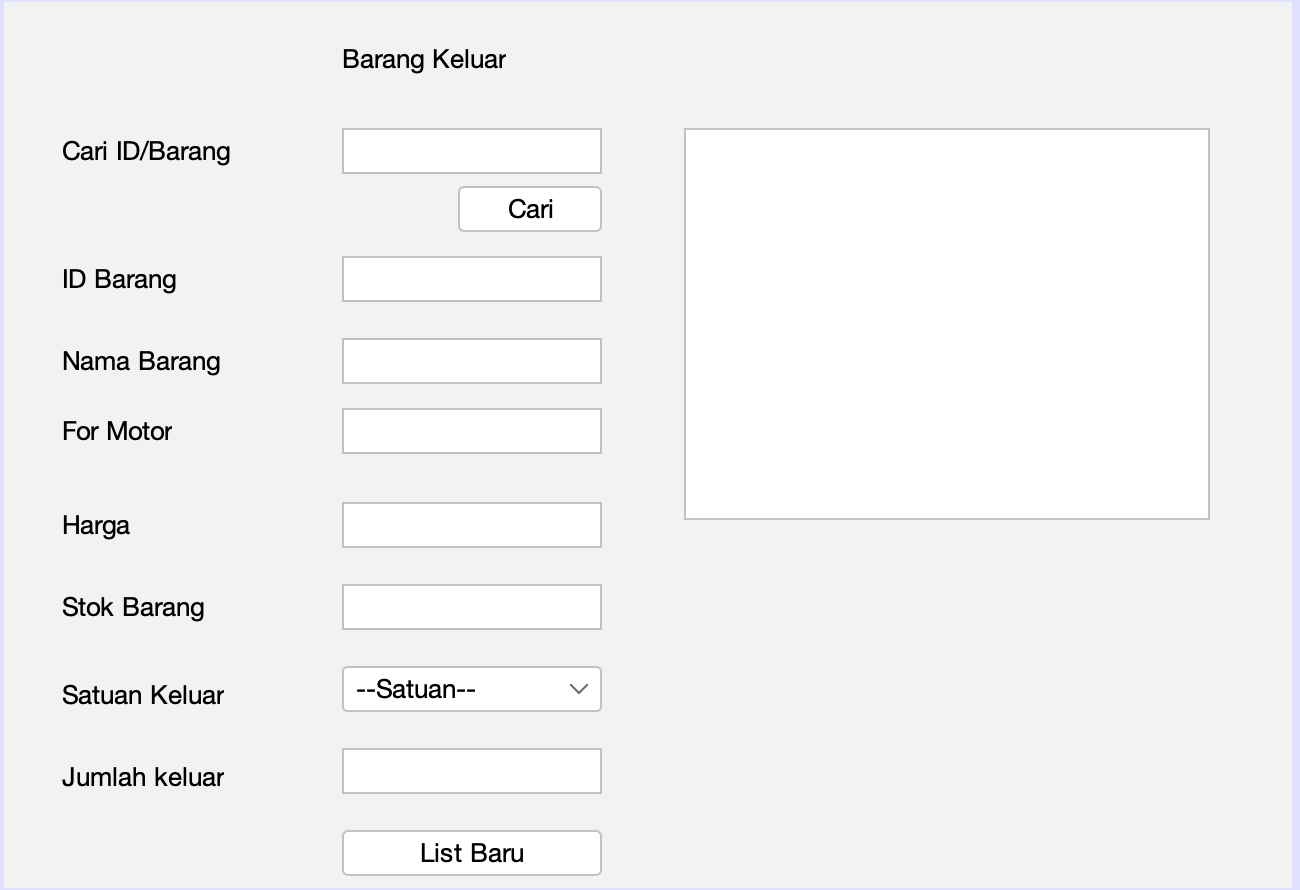
}

return TotHargaBarang;

}

}

### Desain *Form* (GUI\_BarangKeluar.java) :



### Gambar 4.12 Desain GUI\_Barang Keluar.java(*Enkapsulasi*) Tabel 4.6 Properti desain GUI\_BarangKeluar.(*Enkapsulasi*)

|  |  |  |  |
| --- | --- | --- | --- |
| **No** | **Objek** | **Properti** | **Nilai** |
| 1 | jLabel1 | Text | “Barang Keluar” |
| 2 | jLabel2 | Text | “Cari ID/Nama” |
| 3 | jLabel3 | Text | “ID Barang” |
| 4 | jLabel4 | Text | “Nama Barang” |
| 5 | jLabel5 | Text | “For Motor” |
| 6 | jLabel6 | Text | “Harga” |
| 7 | jLabel7 | Text | “Stock Barang” |
| 8 | jLabel8 | Text | “Satuan Barang” |
| 9. | jLabel9 | Text | “Jumlah keluar” |
| 10 | jTextField1 | Name | txt\_Search |
| Text |  |
| 11 | jTextField2 | Name | ID\_Barang |
| Text |  |
| 12 | jTextField3 | Name | txt\_NmBarang |
| Text |  |
| 13 | jTextField4 | Name | txt\_forMotor |
| Text |  |
| 14 | jTextField6 | Name | txt\_Harga |

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | Text |  |
| 15 | jTextField7 | name | txt\_StokBarang |
| text |  |
| 16 | jTextField8 | neme | txt\_Jml |
| text |  |
| 17 | JComboBox | Model | --Satuan--, Unit, Kardus,  kodi |
| Name | cmb\_satuan |
| 18 | jButton1 | Name | btn\_cari |
| Text | Cari |
| 19 | jButton2 | Name | btn\_proses |
| Text | “List baru” |
| 20 | jTextArea | Name | memo |
| Text |  |

*Source code* button Cari:

private

void

btn\_cariActionPerformed(java.awt.event.ActionEvent evt) {

// TODO add your handling code here: BarangMasuk brgMasuk = new BarangMasuk(); if(

txt\_Search.getText().equalsIgnoreCase("12345")|| txt\_Search.getText().equalsIgnoreCase("Kampas rem ysp")){

JOptionPane.showMessageDialog(null, "Barang

Ditemukan");

ID\_Barang.setText(Integer.toString(brgMasuk.kdBarang)); txt\_NmBarang.setText(brgMasuk.NmBarang); txt\_ForMotor.setText(brgMasuk.ForMotor);

txt\_Harga.setText(Integer.toString(brgMasuk.harga)); txt\_StokBarang.setText(Integer.toString(brgMasuk.jumlah));

}else{

JOptionPane.showMessageDialog(null, "Barang belum di tambahkan!!!!");

}

}

*Source code* button list baru :

private void

btn\_prosesActionPerformed(java.awt.event.ActionEvent evt)

{

// TODO add your handling code here: BarangMasuk brgMasuk = new BarangMasuk();

BarangKeluar brgKeluar = new BarangKeluar(); int ID = Integer.parseInt(ID\_Barang.getText()); String nama = txt\_NmBarang.getText();

int harga = Integer.parseInt(txt\_Harga.getText()); int stok =

Integer.parseInt(txt\_StokBarang.getText());

int jumlah = Integer.parseInt(txt\_Jml.getText()); memo.append("Kode Barang

:"+Integer.toString(ID)+"\n"); memo.append("Nama Barang :"+nama+"\n");

memo.append("jumlah Barang :"+jumlah+"\n"); memo.append("Harga satuan Barang :"+harga+"\n"); switch (cmb\_Satuan.getSelectedIndex()) {

case 0:

JOptionPane.showMessageDialog(null, "Masukkan Satuan Barang");

break; case 1:

memo.append("Harga total:"+brgKeluar.HargaTotalBarang(1,harga, jumlah)+"\n");

break; case 2:

memo.append("Harga total:"+brgKeluar.HargaTotalBarang(2,harga, jumlah)+"\n");

break; case 3:

memo.append("Harga total:"+brgKeluar.HargaTotalBarang(3,harga, jumlah)+"\n");

default:

break;

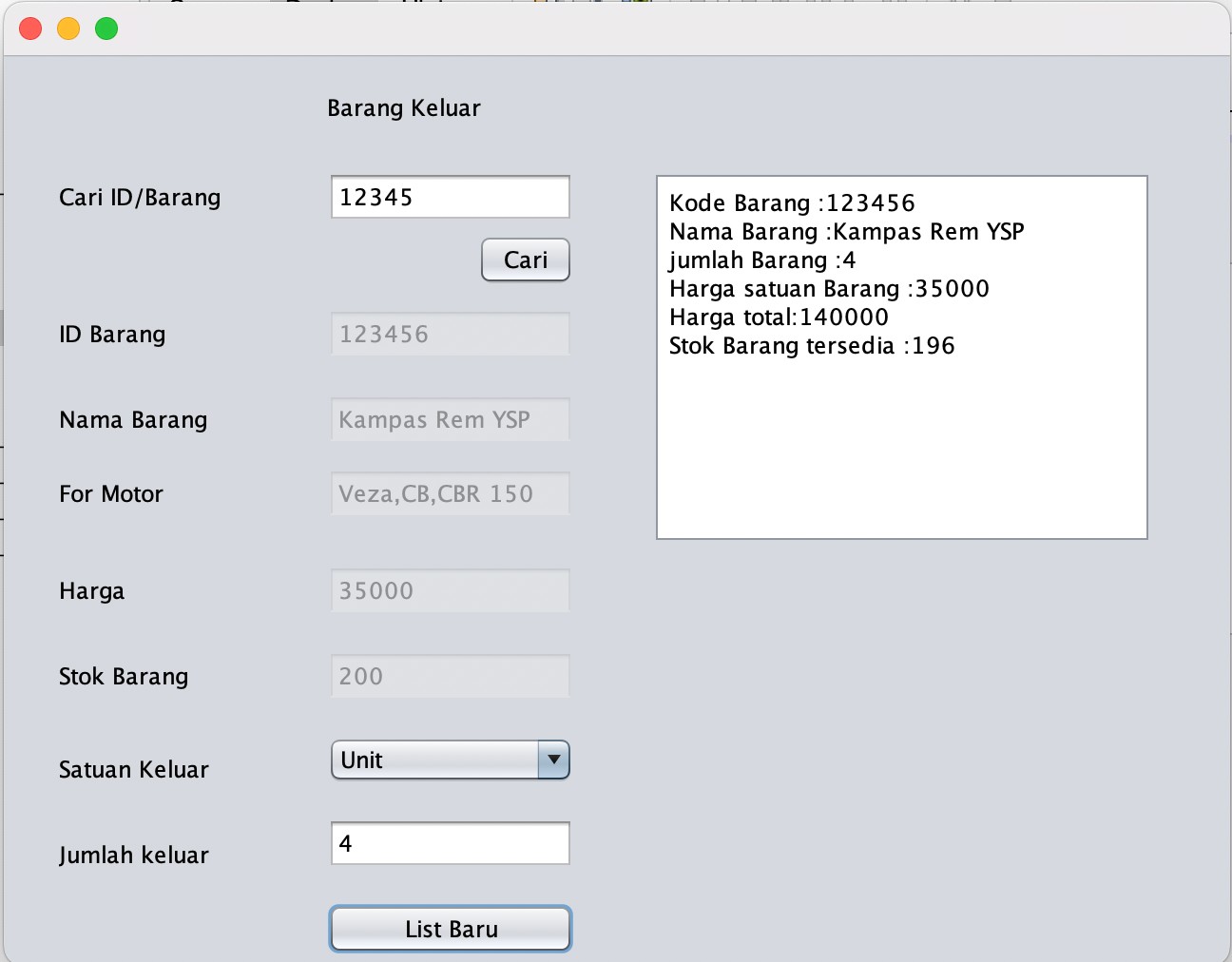
}

memo.append("Stok Barang tersedia

:"+brgKeluar.barangKeluar(jumlah)+"\n");

}

### Tampilan Hasil :



Gambar 4.13 Hasil GUI\_barangkeluar..java(*Enkapsulasi*)

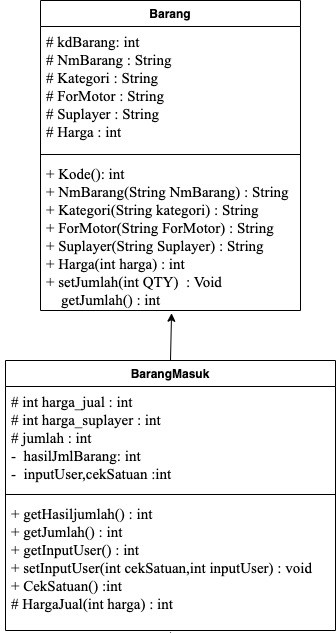
### Analisa :

Pada semua *class* di atas terdapat beberapa *attribute* yang menggunakan hak akses *public*, *private*, *protected* . Hak akses *private* hanya memperbolehkan pengaksesan data hanya dari dalam *class* tersebut saja diluar class tersebut maka *attribute* tersebut tidak dikenali. Oleh karena itu memerlukan *method setter* dan *getter* . *Method setter* digunakan untuk memberikan nilai pada attribute sedangkan *getter* digunakan untuk mengambil nilai yang telah diset pada *method setter*. Berbeda dengan hak akses lainnya yaitu *public* dan *protected* kedua hak akses tersebut bisa diakses pada di luar maupun di dalam *class*.

## Tugas Rumah 2 :

Judul : Implementasi *Overloading* dan *Overriding* Pada Projek Individu Tema : Inventory Barang Sparepart Motor

### Diagram Class:



Gambar 4.14 Class diagram (*Overriding*)

*Source code* pada class (Barang.java)

public class Barang {

protected int kdBarang ,harga,jumlah;

protected String NmBarang,Kategori, ForMotor,Suplayer; int Kode(){

Random rndm = new Random(); kdBarang = rndm.nextInt(100000); return kdBarang;

}

void setJumlah(int QTY) { this.jumlah = QTY;

}

int getJumlah() { return jumlah;

}

Barang(){

kdBarang = 123456 ; NmBarang = "Kampas Rem YSP";

ForMotor = "Veza,CB,CBR 150"; Suplayer= "PT. YSP";

harga = 35000;

}

public String NmBarang(String NmBarang){ this.NmBarang = NmBarang;

return NmBarang;

}

public String Kategori(String Kategori){ this.Kategori = Kategori;

return Kategori;

}

public String ForMotor(String ForMotor){ this.ForMotor = ForMotor;

return ForMotor;

}

public String Suplayer(String Suplayer){ this.Suplayer = Suplayer;

return Suplayer;

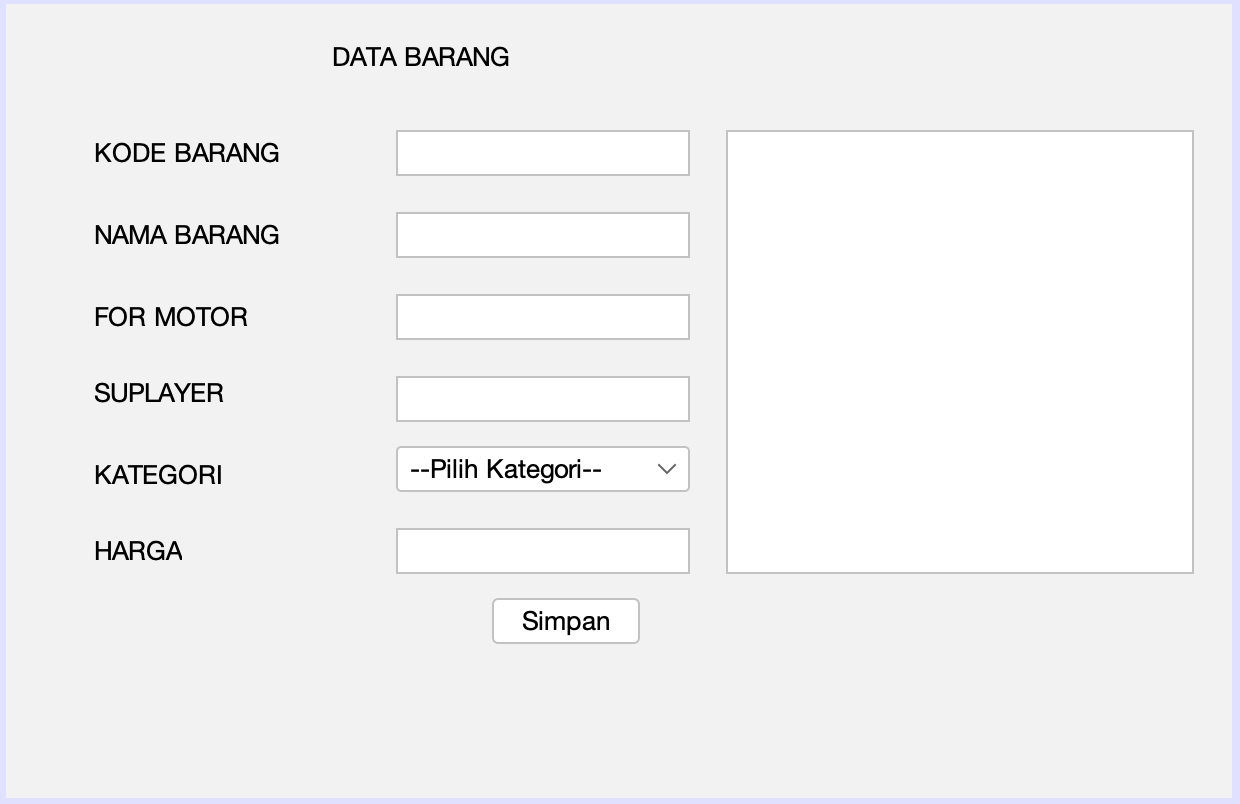
}

public int Harga(int harga){ this.harga = harga; return harga;

}

}

### Desain Gui\_barang.java



Gambar 4.15 Desain Gui\_barang.java(*overriding*)

### Tabel 4.7 Property desain GUI\_barang.java

|  |  |  |  |
| --- | --- | --- | --- |
| **No** | **Objek** | **Properti** | **Nilai** |
| 1 | jLabel1 | text | "Cari ID/Nama" |
| 2 | jLabel2 | text | "ID Barang" |
| 3 | jLabel3 | text | "Nama Barang" |
| 4 | jLabel4 | text | "Suplayer" |
| 5 | jLabel5 | text | "Harga Suplayer" |
| 6 | jLabel6 | text | "Harga Jual" |
| 7 | jLabel7 | text | "Satuan" |
| 8 | jLabel8 | text | "Jumlah" |
| 9 | jLabel9 | text | “QTY” |
| 10 | JTextField1 | text | " " |
| name | txt\_Search |
| 11 | JTextField2 | text | " " |
| name | ID\_Barang |
| 12 | JTextField3 | text | " " |
| name | txt\_NmBarang |
| 13 | JTextField4 | text | " " |
| name | txt\_Suplayer |
| 14 | JTextField5 | text | " " |
| name | txt\_HargaSuplayer |
| 15 | JTextField6 | text | " " |
| name | txt\_HargaJual |
| 16 | JTextField7 | text | " " |
| name | txt\_Jumlah |
| 17 | JTextField7 | text |  |
| name | txtQTY |
| 18 | JtextArea1 | name | memo |
| 19 | JComboBox | model | Satuan,Unit, Kardus, Kodi. |
| name | cmb\_satuan |
| 20 | JButton1 | text | "CEK" |

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | name | btn\_Cek |
| 23 | JButton2 | Text | Barang Masuk |
| Name | BarangMasuk |

*Source code* pada button simpan

private void SimpanActionPerformed(java.awt.event.ActionEvent evt) {

// TODO add your handling code here: int Berhasil = 0,Kode;

Barang brg = new Barang(); brg.NmBarang(Txt\_NmBarang.getText()); brg.ForMotor(Txt\_ForMotor.getText()); brg.Suplayer(Txt\_Suplayer.getText()); brg.Harga(Integer.parseInt(Txt\_Harga.getText())); Kode = Integer.parseInt(Txt\_KdBarang.getText()); if(Cmb\_Kategori.getSelectedIndex()==0){

JOptionPane.showMessageDialog(null, "Kategori Belum Anda Masuukan");

}else if(Cmb\_Kategori.getSelectedIndex()==1){ Berhasil = 1;

brg.Kategori("oli");

}else if(Cmb\_Kategori.getSelectedIndex()==2){ Berhasil = 1;

brg.Kategori("Part Motor Mesin");

}else if(Cmb\_Kategori.getSelectedIndex()==3){ Berhasil = 1;

brg.Kategori("Lampu Depan");

}else if(Cmb\_Kategori.getSelectedIndex()==4){ Berhasil = 1;

brg.Kategori("Lampu Belakang");

}else if(Cmb\_Kategori.getSelectedIndex()==5){ Berhasil = 1;

brg.Kategori("Baut");

}else if(Cmb\_Kategori.getSelectedIndex()==6){ Berhasil = 1;

brg.Kategori("Mur");

}

//

if(Berhasil == 1){ JOptionPane.showMessageDialog(null,

"Berhasil!!!");

memo.append(" Kode Barang : " +Kode+"\n"); memo.append(" Nama Barang :

"+brg.NmBarang+"\n");

memo.append(" For Motor : "+brg.ForMotor+"\n");

memo.append(" Suplayer : "+brg.Suplayer+"\n"); memo.append(" Kategori : "+brg.Kategori+"\n"); memo.append(" Harga :Rp"+brg.harga);

}else{

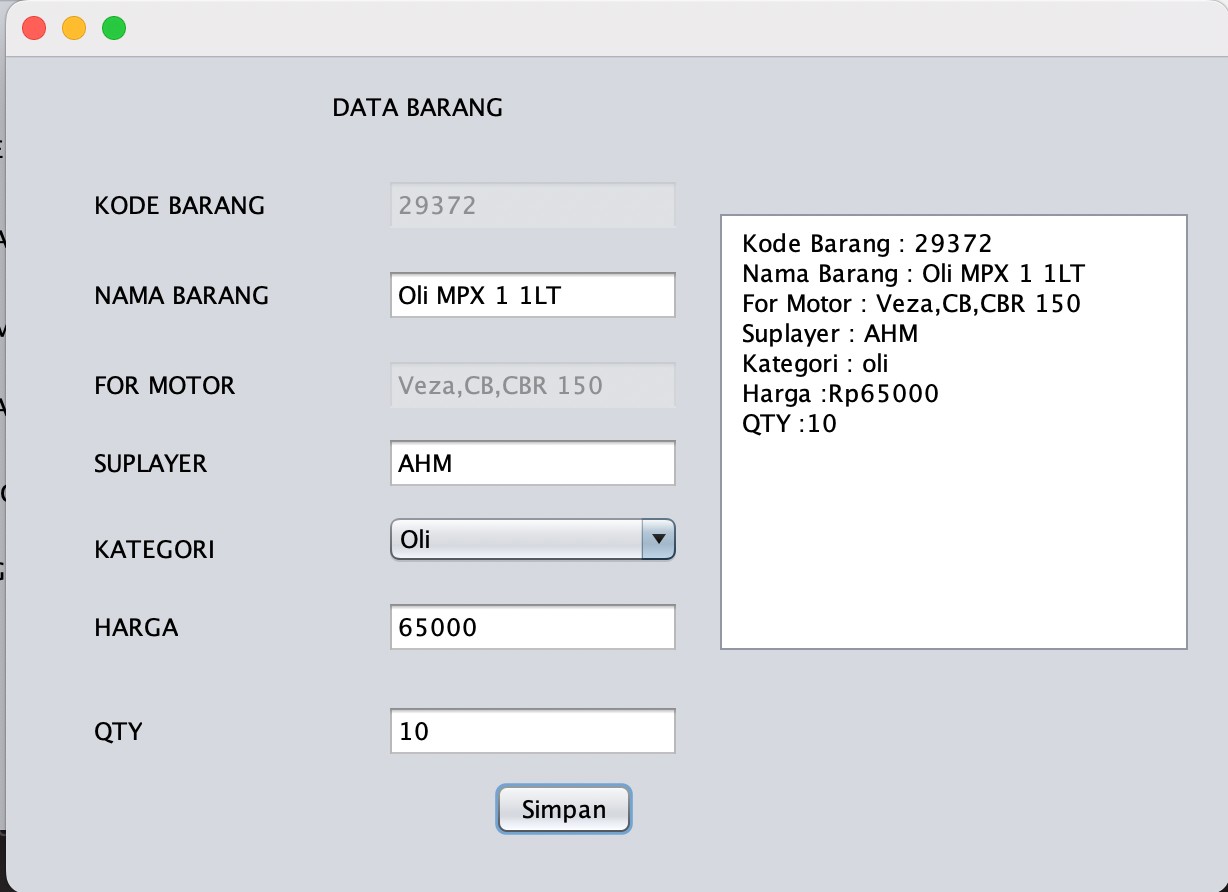
JOptionPane.showMessageDialog(null,

"Gagal!!!");

}

}

### Hasil running



### Gambar 4.16 Tampilan Gui\_Barang(*Overriding*) Source code pada class(Barangmasuk.java)

public class BarangMasuk extends Barang{ protected int harga\_jual, harga\_suplayer; private int jumlah , inputUser,cekSatuan; public BarangMasuk(){

harga\_suplayer = 45000;

jumlah = 200;

}

public int getHasiljumlah() { return jumlah;

}

//getter hasil jml barang public int getJumlah() {

return jumlah;

}

// setter dan getter input barang public int getInputUser() {

return inputUser;

}

public void setInputUser(int cekSatuan,int inputUser)

{

this.inputUser = inputUser; this.cekSatuan = cekSatuan; CekSatuan();

}

public int CekSatuan() { if(cekSatuan == 1){

jumlah = inputUser;

}else if(cekSatuan == 2){ jumlah = inputUser \* 12;

}else if(cekSatuan == 3){ jumlah = inputUser \* 24;

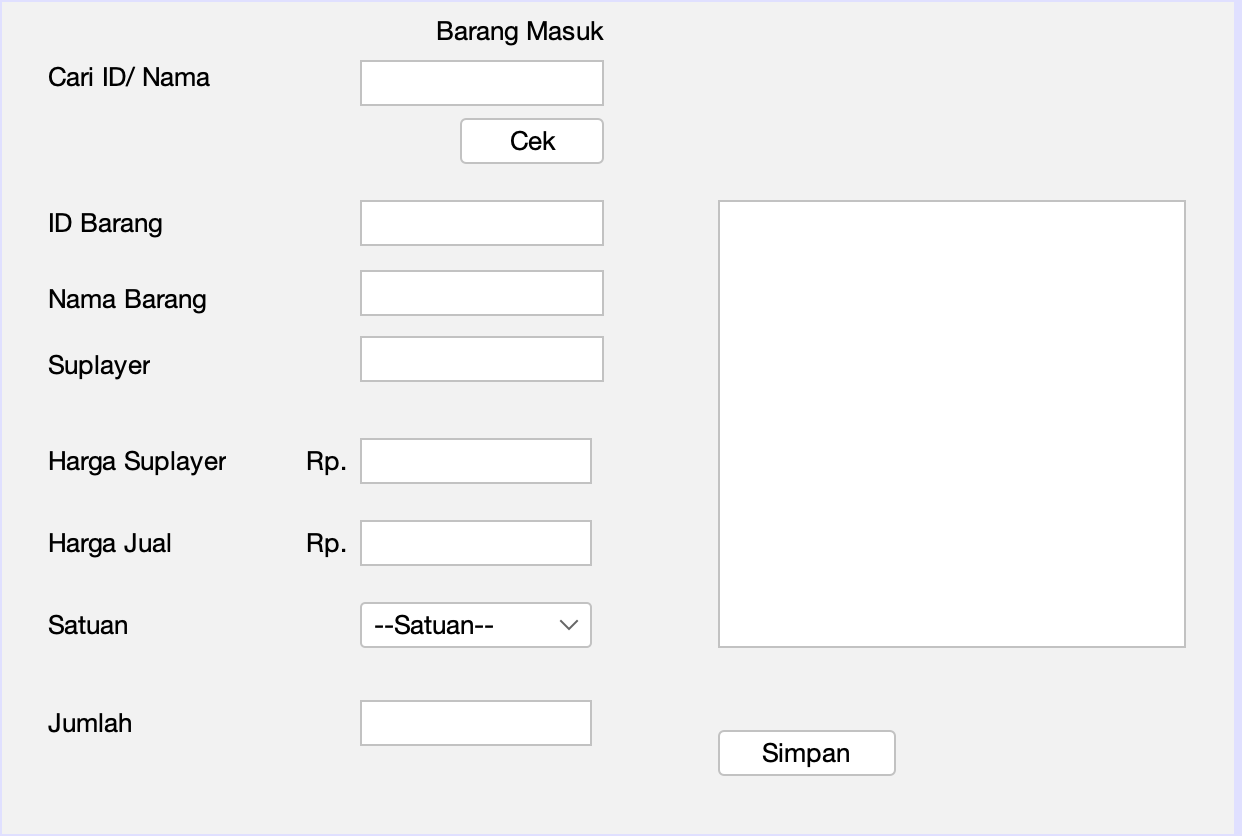
}

return jumlah;}

protected int HargaJual(int harga){ this.harga\_jual = harga;return harga;} }

### Desain Form (GUI\_BarangMasuk.java):

|  |  |  |  |
| --- | --- | --- | --- |
| **No** | **Objek** | **Properti** | **Nilai** |
| 1 | jLabel1 | Text | “Barang Masuk” |
| 2 | jLabel2 | Text | “Cari ID/Nama” |
| 3 | jLabel3 | Text | “ID Barang” |
| 4 | jLabel4 | Text | “Nama Barang” |
| 5 | jLabel5 | Text | “Suplayer” |
| 6 | jLabel6 | Text | “Harga Suplayer” |
| 7 | jLabel7 | Text | “Harga Jual” |
| 8 | jLabel8 | Text | “Satuan” |
| 9. | jLabel9 | Text | “Jumlah” |
| 10 | jTextField1 | Name | txt\_Search |
| Text |  |
| 11 | jTextField2 | Name | ID\_Barang |
| Text |  |
| 12 | jTextField3 | Name | txt\_NmBarang |
| Text |  |
| 14 | jTextField4 | Name | txt\_Suplayer |
| Text |  |
| 15 | jTextField6 | Name | txt\_HargaSuplayer |



### Gambar 4. 1 Desain GUI\_barangMasuk(*Overriding*) Tabel 4.8 Properti Desain GUI

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | Text |  |
| 16 | jTextField7 | name | txt\_HargaJual |
| text |  |
| 17 | jTextField8 | neme | txt\_Jumlah |
| text |  |
| 18 | JComboBox | text |  |
| Name | cmb\_satuan |
| 19 | jButton1 | Name | btnCetak |
| Text | Cek |
| 20 | jButton2 | Neme | BarangMasuk |
| text | “BarangMasuk” |
| 21 | jTextArea | Name | memo |
| Text |  |

*Sorce code* button BarangMasuk

|  |  |  |
| --- | --- | --- |
| private void BarangMasukActionPerformed(java.awt.event.ActionEvent evt)  {  // TODO add your handling code here: int berhasil=0,cek;  BarangMasuk brgMasuk = new BarangMasuk(); brgMasuk.harga\_jual =  Integer.parseInt(txt\_HargaJual .getText()); brgMasuk.harga\_suplayer =  Integer.parseInt(txt\_HargaSuplayer .getText());  int jum = Integer.parseInt(txt\_Jumlah.getText()); memo.append("Kode Barang  :"+Integer.toString(brgMasuk.kdBarang)+"\n"); memo.append("Nama Barang  :"+brgMasuk.NmBarang+"\n"); memo.append("Nama Suplayer  :"+brgMasuk.Suplayer+"\n"); memo.append("Harga Suplayer  :"+brgMasuk.harga\_suplayer+"\n"); memo.append("Harga Jual  :"+brgMasuk.harga\_jual+"\n"); int pil =1 ;  if(cmb\_satuan.getSelectedIndex() == 0){ JOptionPane.showMessageDialog(null, "Masukkan  Satuan Barang");  }else if(cmb\_satuan.getSelectedIndex() == 1){ brgMasuk.setInputUser(1, jum); memo.append("Jumlah Barang :  "+Integer.toString(brgMasuk.getHasiljumlah()));  }else if(cmb\_satuan.getSelectedIndex() == 2){ brgMasuk.setInputUser(2, jum); | | |
|  | Nama Aslab Pengajar : Silva Anjelina Tambunan | TTD : |
| Tanggal : |

memo.append("Jumlah Barang : "+Integer.toString(brgMasuk.getHasiljumlah()));

}else if(cmb\_satuan.getSelectedIndex() == 3){ brgMasuk.setInputUser(3, jum); memo.append("Jumlah Barang :

"+Integer.toString(brgMasuk.getHasiljumlah()));

}

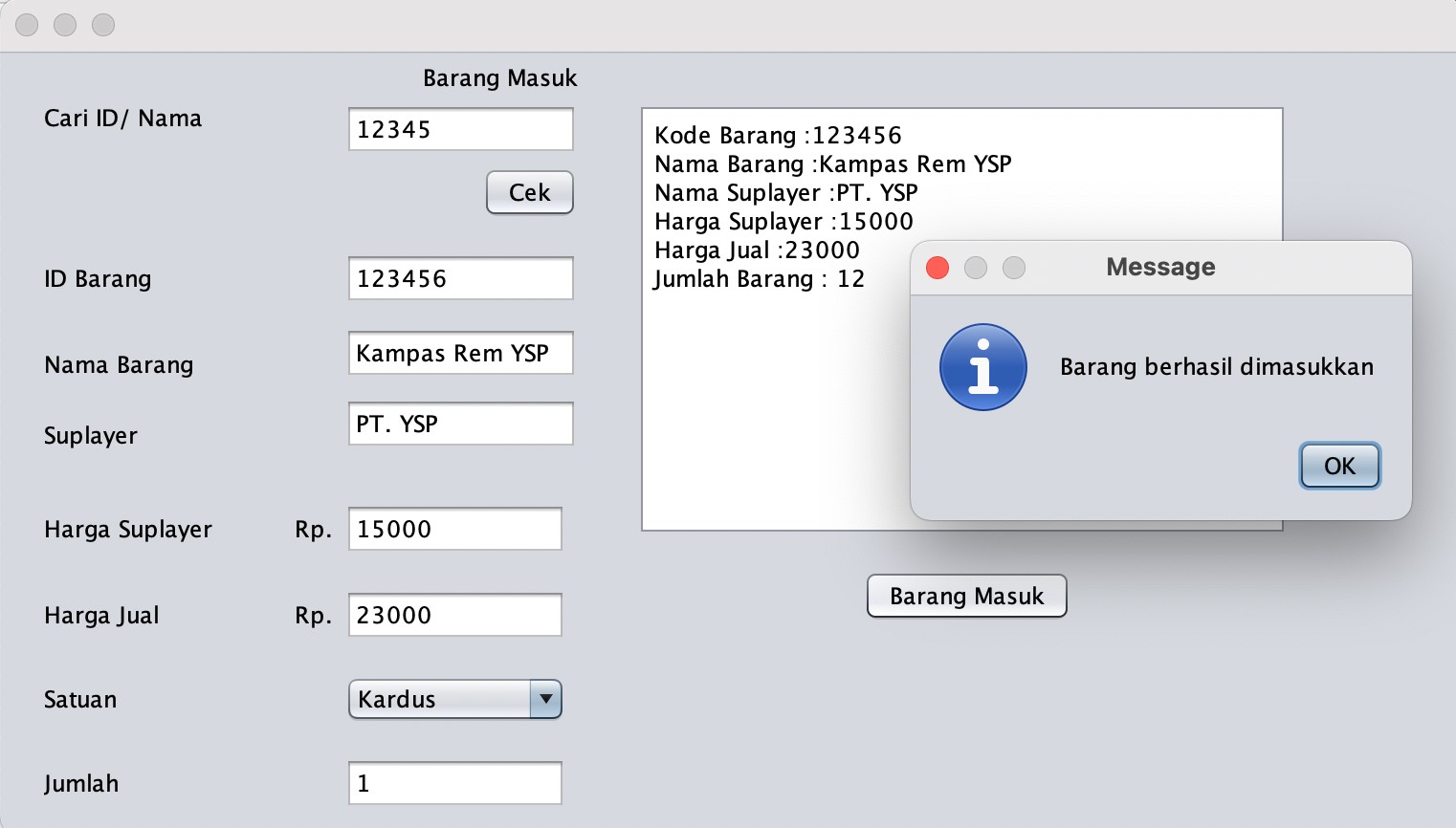
if(berhasil == 1){

JOptionPane.showMessageDialog(null, "Barang berhasil dimasukkan");

}

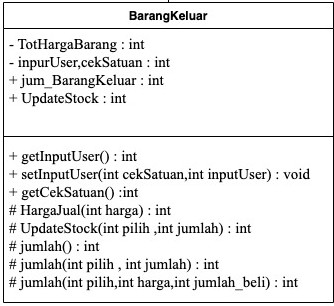
}

### Hasil Running :



### Gambar 4. 2 Hasil Tampilan GUI\_Barangmasuk (*Overriding*)

#### Class diagram overloading :



Gambar 4. 3 class diagram barangKeluar(*Overloading*)

*Source code* pada *class* (BarangKeluar.java):

public class BarangKeluar extends BarangMasuk{ private int TotHargaBarang;

private int inpurUser,cekSatuan; public int jum\_BarangKeluar; public int UpdateStock;

public int getInpurUser() { return inpurUser;

}

public int getCekSatuan() { return cekSatuan;

}

public void setInpurUser(int inpurUser,int cekSatuan)

{

this.inpurUser = inpurUser; this.cekSatuan = cekSatuan;

}

protected int UpdateStock(int pilih ,int jumlah){ if(pilih == 1){

UpdateStock= getJumlah() - jumlah;

}else if(pilih == 2){

UpdateStock = getJumlah() - (jumlah \* 12);

}else if(pilih == 3){

UpdateStock = getJumlah()- (jumlah \* 24);

}

jumlah();

return UpdateStock;

}

protected int jumlah(){ return UpdateStock;

}

protected int jumlah(int pilih , int jumlah){ if(pilih == 1){

jum\_BarangKeluar= jumlah;

}else if(pilih == 2){ jum\_BarangKeluar = jumlah \* 12;

}else if(pilih == 3){ jum\_BarangKeluar = jumlah \* 24;

}

return jum\_BarangKeluar;

}

protected int jumlah(int pilih,int harga,int jumlah\_beli){

if(pilih == 1){

TotHargaBarang = harga \* jumlah\_beli;

}else if(pilih == 2){

TotHargaBarang = (jumlah\_beli \* 12) \* harga;

}else if(pilih == 3){

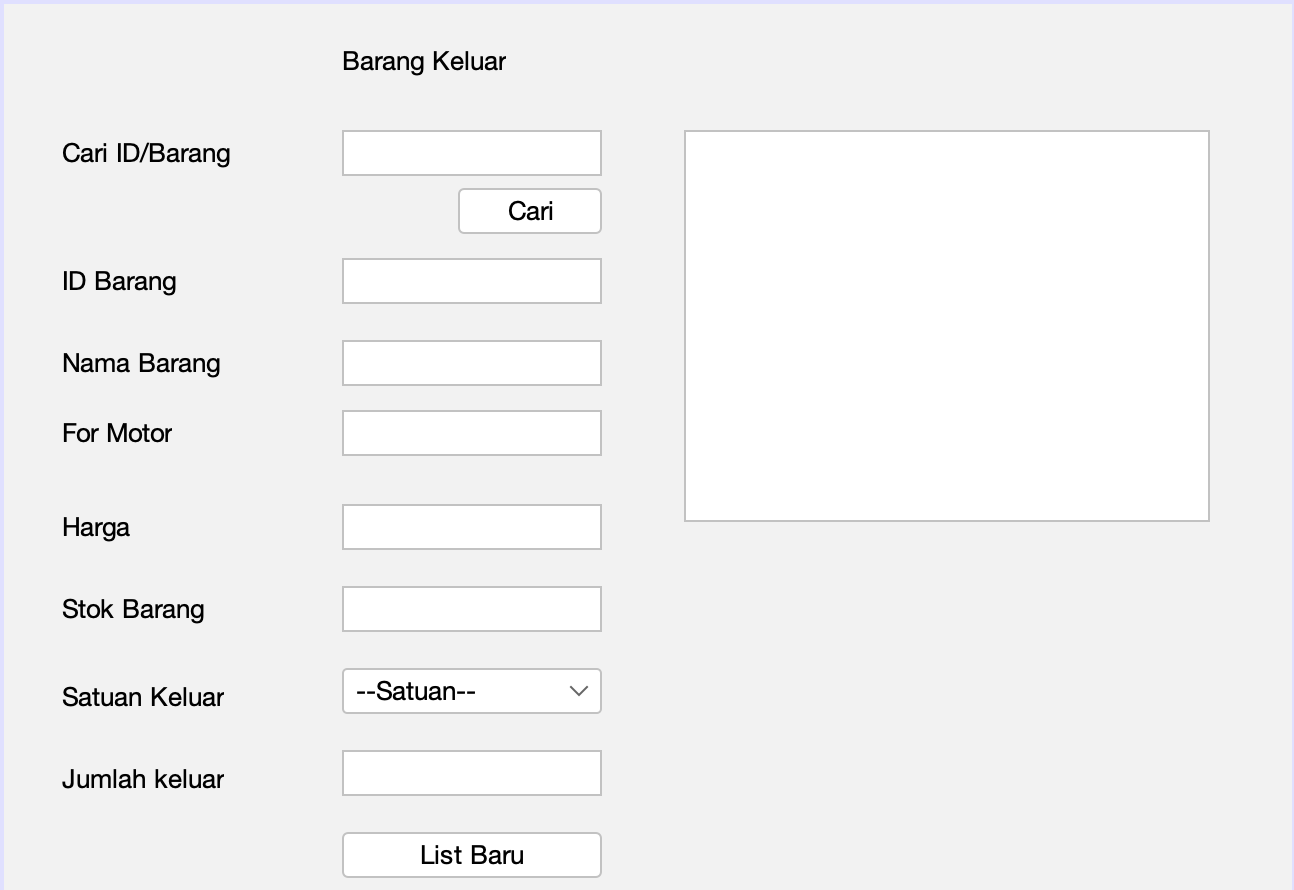
TotHargaBarang = (jumlah\_beli \* 24) \* harga;

}

return TotHargaBarang;

}}

### Desain *form* (GUI\_BarangKeluar.java) :



### Gambar 4. 4 Properti Desain BarangKeluar(*Overloading*) Tabel 4.9 Properti GUI\_BarangKeluar.java

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **No** | **Objek** | | **Properti** | **Nilai** | |
| 1 | jLabel1 | | Text | “Barang Keluar” | |
| 2 | jLabel2 | | Text | “Cari ID/Nama” | |
| 3 | jLabel3 | | Text | “ID Barang” | |
| 4 | jLabel4 | | Text | “Nama Barang” | |
| 5 | jLabel5 | | Text | “For Motor” | |
| 6 | jLabel6 | | Text | “Harga” | |
| 7 | jLabel7 | | Text | “Stock Barang” | |
| 8 | jLabel8 | | Text | “Satuan Barang” | |
| 9. | jLabel9 | | Text | “Jumlah keluar” | |
| 10 | jTextField1 | | Name | txt\_Search | |
| Text |  | |
| 11 | jTextField2 | | Name | ID\_Barang | |
| Text |  | |
| 12 | jTextField3 | | Name | txt\_NmBarang | |
| Text |  | |
| 13 | jTextField4 | | Name | txt\_forMotor | |
| Text |  | |
| 14 | jTextField6 | | Name | txt\_Harga | |
|  | | Nama Aslab Pengajar : Silva Anjelina Tambunan | | | TTD : |
| Tanggal : | | |

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | Text |  |
| 15 | jTextField7 | name | txt\_StokBarang |
| text |  |
| 16 | jTextField8 | neme | txt\_Jml |
| text |  |
| 17 | JComboBox | Model | --Satuan--,Unit,Kardus, kodi |
| Name | cmb\_satuan |
| 18 | jButton1 | Name | btn\_cari |
| Text | Cari |
| 19 | jButton2 | Name | btn\_proses |
| text | “List baru” |
| 20 | jTextArea | Name | memo |
| Text |  |

*Source code* pada (Button cari)

private void btn\_searchActionPerformed(java.awt.event.ActionEvent evt)

{

// TODO add your handling code here: BarangMasuk brgMasuk = new BarangMasuk(); if(

txt\_Search.getText().equalsIgnoreCase("12345")|| txt\_Search.getText().equalsIgnoreCase("Kampas rem ysp")){

JOptionPane.showMessageDialog(null, "Barang

Ditemukan");

ID\_Barang.setText(Integer.toString(brgMasuk.kdBarang)); txt\_NmBarang.setText(brgMasuk.NmBarang); txt\_ForMotor.setText(brgMasuk.ForMotor);

txt\_Harga.setText(Integer.toString(brgMasuk.harga));

txt\_StokBarang.setText(Integer.toString(brgMasuk.getJumlah ()));

}else{

JOptionPane.showMessageDialog(null, "Barang belum di tambahkan!!!!");

}

}

### Source code pada button(list baru)

private void btn\_listBaruActionPerformed(java.awt.event.ActionEvent evt) {

// TODO add your handling code here: BarangMasuk brgMasuk = new BarangMasuk();

BarangKeluar brgKeluar = new BarangKeluar(); int ID = Integer.parseInt(ID\_Barang.getText()); String nama = txt\_NmBarang.getText();

int harga = Integer.parseInt(txt\_Harga.getText()); int stok =

Integer.parseInt(txt\_StokBarang.getText());

int jumlah = Integer.parseInt(txt\_Jml.getText()); memo.append("Kode Barang

:"+Integer.toString(ID)+"\n"); memo.append("Nama Barang :"+nama+"\n"); if(cmb\_Satuan.getSelectedIndex()==1){

JOptionPane.showMessageDialog(null, "Masukkan Satuan Barang");

}else if(cmb\_Satuan.getSelectedIndex()==1){ memo.append("jumlah Barang

:"+brgKeluar.jumlah(1, jumlah)+"\n");

memo.append("Harga total:"+brgKeluar.jumlah(1,harga, jumlah)+"\n");

brgKeluar.UpdateStock(1, jumlah);

}else if(cmb\_Satuan.getSelectedIndex()==2){ memo.append("jumlah Barang

:"+brgKeluar.jumlah(2, jumlah)+"\n");

memo.append("Harga total:"+brgKeluar.jumlah(2,harga, jumlah)+"\n");

brgKeluar.UpdateStock(2, jumlah);

}else if(cmb\_Satuan.getSelectedIndex()==3){ memo.append("jumlah Barang

:"+brgKeluar.jumlah(3, jumlah)+"\n");

memo.append("Harga total:"+brgKeluar.jumlah(3,harga, jumlah)+"\n");

brgKeluar.UpdateStock(3, jumlah);

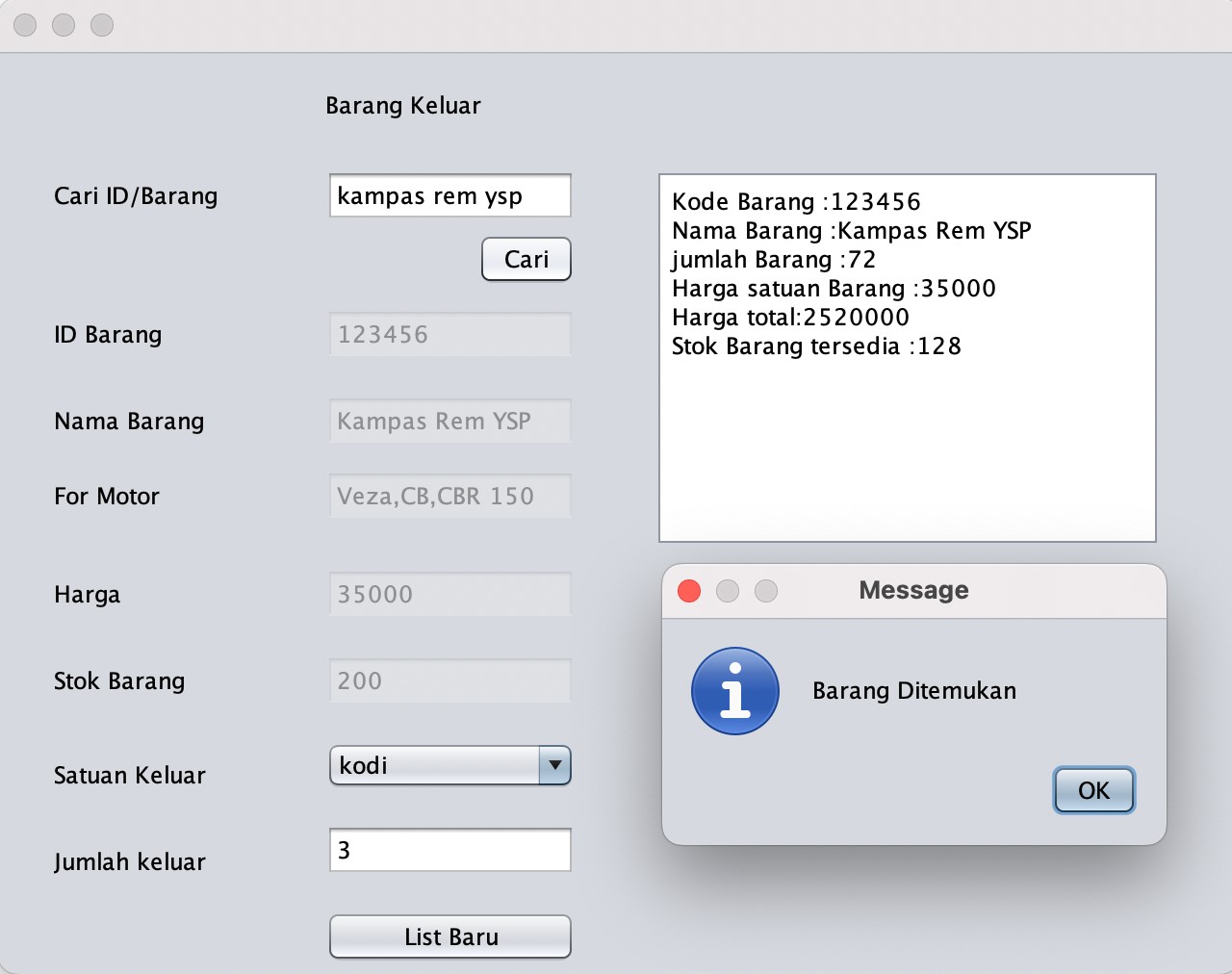
}

memo.append("Stok Barang tersedia

:"+brgKeluar.jumlah()+"\n");

}

### Tampilan Hasil :



Gambar 4. 5 Hasil GUI\_BarangMasuk.java(*Enkapsulasi*)

### Analisa :

### Pada source code class barang dan barang masuk terdapat method overriding yaitu getJumlah pada method barang dan pada method barangMasuk methos tersebut digunakan untuk mengambil nilai jumlah, dan barang keluar terdapat overloading yaitu pada method jumlah didefinisikan 3 kali namun berbeda parameter. Hal tersebut berarti pada method overloading memiliki kemampuan untuk memperbolehkan suatu class memiliki nama method yang sama namun berbeda pada parameter

## Kesimpulan

### *Inheritance* atau Pewarisan/Penurunan adalah konsep pemrograman dimana sebuah class dapat ‘menurunkan’ property dan method yang dimilikinya kepada class lain.

### *Method Overloading* adalah sebuah kemampuan yang membolehkan sebuah class mempunyai 2 atau lebih method dengan nama yang sama, namun parameter yang berbeda.

### *Overriding* method adalah kemampuan dari subclass untuk memodifikasi method dari superclass-nya, dengan cara mendefinisikan kembali method superclass-nya.

**BAB V**

**ABSTRACT, POLIMORFISME DAN EXCEPTION HANDLING**

|  |  |  |
| --- | --- | --- |
| **Jumlah**  **Pertemuan** | **:** | **2 x 60 menit** |
| **Tujuan Praktikum** | **:** | 1. Praktikan dapat mengetahui fungsi atau kegunaan kelas Abstrak dan kelas *Interface* . 2. Praktikan mampu membuat kelas Abstrak dan kelas *Interface* pada *java*. 3. Praktikan mampu memahami dan   mengimplementasikan *polimorfisme* ke dalam program. |
| **Alat / bahan** | **:** | 1. Seperangkat *computer*. 2. Perangkat lunak: *Netbeans*. 3. Modul Praktikum *OOP* 2022. |

## Landasan Teori

### Pengertian 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 *abstract class* umumnya akan memiliki *abstract method*.

### Berikut bentuk deklarasi Class Abstract

Akses\_modifier **abstract class** namaClassAbstrak

{

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

}

*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:

|  |  |
| --- | --- |
| Nama Aslab Pengajar :  Silva Anjelina Tambunan | TTD : |
| Tanggal : |

### Berikut bentuk deklarasi Class Abstract

abstract tipe\_data namaMethodAbstract(parameter); //non- void

abstract void namaMethodAbstract(parameter); // void

*Abstract class* di gunakan 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.

### *Abstract* class tidak dapat diinstansiasi (menj`adi 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*). Ada yang Namanya upcasting dan downcasting

### *Upcasting* adalah salah satu cara untuk *casting* atau mengkonversi objek ,dari tipe Sub *class* atau *Class* Turunan , ke tipe Super Class atau *Class* Parent (ke arah atas pada hirarki pewarisan). Downcasting adalah salah satu cara untuk *casting* atau mengkonversi objek ,dari tipe Super *Class* atau *Class Parent*, ke tipe Sub *class* atau *Class* Turunan (ke arah bawah pada hirarki pewarisan).

Gambar 5.1 *Upcasting* dan *Downcasting*

* + - 1. Sebuah *class* dapat dideklarasikan sebagai *abstract class* meskipun

### *class* tersebut tidak memiliki abstract method.

* + - 1. *Abstract method* tidak boleh mempunyai *body method* dan demikian juga sebaliknya bahwa *method* yang tidak ditulis *body* methodnya maka harus dideklarasikan sebagai *abstract method.*
    1. Pengertian Dasar *Polimorfisme*

*Poly* artinya banyak, *morfisme* artinya bentuk. *Polimorfisme* dalam OOP adalah sebuah prinsip di mana *class* dapat memiliki banyak **“bentuk”** *method* yang berbeda- beda meskipun namanya sama. “Bentuk” di sini dapat kita artikan: isinya berbeda (*overriding*), parameternya berbeda dan tipe datanya berbeda (*overlading*).

### *Polimorfisme* merupakan salah satu hal esensial dalam pemograman berorientasi objek karena alasan berikut: yakni *polimorfisme* mengizinkan kelas induk untuk mendefinisikan sebuah *method* general (bersifat umum) untuk semua kelas turunannya.

*Polymorphism* pada java dibagi menjadi dua macam yaitu :

* + - 1. Static Polymorphism (*Polimorfisme statis*)

### *Polimorfisme* statis menggunakan mekanisme method overloading dalam penerapannya dan dilakukan disaat compile Time (menjalankan program). Method overloading terjadi pada sebuah class yang memiliki nama method yang sama tapi memiliki parameter dan tipe data yang berbeda.

* + - 1. Dynamic Polymorphism (*Polimorfisme dinamik*)

### *Polimorfisme* dinamik menggunakan mekanisme 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.

### Exception

### Pengertian Dasar Exception

### kita mengetahui bahwa tiada program yang sempurna , tiada platform yang sempurna dimana suatu program dijalankan dan juga tiada pengguna program yang sempurna. Oleh karenanya diperlukan mekanisme yang membantu menangani error atau kesalahan yang terjadi. Baik saat pembuatan maupun implementasi program.

### Misalnya saja dalam dunia nyata, suatu program atau software memerlukan input berupa angka tetapi pengguna memasukkan data karakter/huruf yang tidak dapat diproses oleh program. hal semacam ini perlu diinformasikan kepada pengguna dan program/software perlu melakukan sesuatu berkaitan dengan hal tersebut.sehingga java menyediakan mekanisme dalam pemrograman untuk menangani hal – hal tersebut yang disebut *Exception*

### Exception *Handling* merupakan mekanisme yang paling diperlukan dalam menangani error yang terjadi pada saat runtime (program berjalan) atau yang lebih dikenal dengan sebutan runtime error.

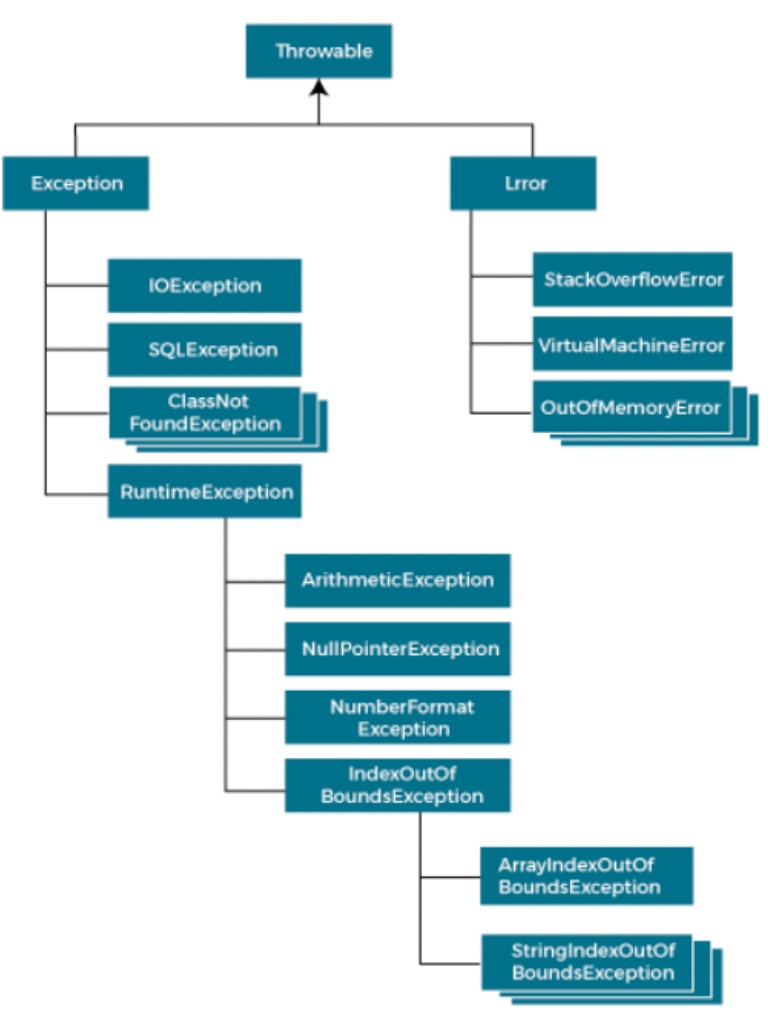
### Secara umum, adanya kesalahan / error yang terjadi pada program pada saat *runtime* dapat menyebabkan program berhenti atau *hang*. Untuk itulah diperlukan mekanisme untuk memastikan bahwa program tetap dapat berjalan meskipun terdapat kesalahan yang terjadi. Keuntungan penggunaan *Exception Handling* :

### *Exception* bisa membawa sangat banyak informasi tentang kesalahan yang terjadi (deskripsi,stack trace, lokasi baris kode dll)

### *Exception* sudah didukung oleh banyak IDE modern yang ada di pasar

### Mekanisme *Exception* terintegrasi secara baik dengan OOP

### Hirarki Exception



### Gambar 5.2 Hirarki Exception

Dalam Java . *Exception* merupakan objek dari subkelas yang diturunkan (*inheritance*) dari kelas *Throwable*. Kelas *Throwable* terdapat dalam *package* java.lang.object yang merupakan *default library* dari Java.

### 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*.

### *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.

### Tabel 5.1 Checked Exception , Uncecked Exception dan error

|  |  |  |
| --- | --- | --- |
| *Checked Exception* | *Uncecked Exception* | Error |
| Kelas yang secara  langsung mewarisi kelas Throwable | Kelas yang mewarisi RuntimeException | Tidak dapat dipulihkan |
| Diperiksa pada waktu kompilasi | tidak diperiksa pada waktu kompilasi, tetapi diperiksa pada waktu proses selesai |  |

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

### 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 {

} catch (ExceptionType nama\_variabel) {

}

//Protected code

//Catch

### 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

}

### 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

}

## Langkah – Langkah Praktikum

### Buka Aplikasi Dev C++.

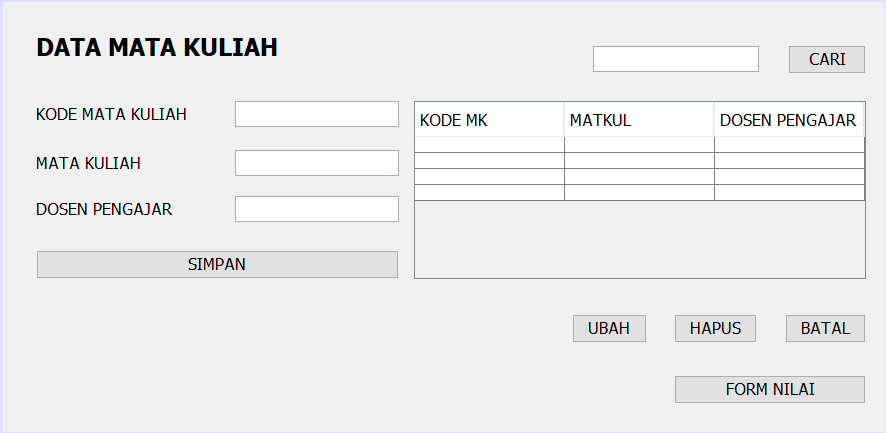
1. Buka Modul Praktikum *Algoritma* dan *Pemrogramman*.
2. Menjalankan *script* dari modul.

## Tugas Praktikum 1:

### Mendesain ulang GUI\_Matkul + membuat *method* Batal() Desain lama *form* (GUI\_Matkul.java) :

### Gambar 5.3 Desain GUI\_Matkul

### Desain baru *form* (GUI\_Matkul.java) :



### Gambar 5.4 Desain GUI\_Matkul(*JTable*) Tabel 5.2 Properti Desain GUI\_Matkul (*JTable*)

|  |  |  |  |
| --- | --- | --- | --- |
| **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 | jTextField1 | Text |  |
| Name | txtKodeMK |
| 6 | jTextField2 | Text |  |
| Name | txtMK |
| 7 | jTextField3 | Text |  |
| Name | txtDosen |
| 8 | jTextField4 | Text |  |
| Name | txtCari |
| 9 | jButton1 | Text | btnSimpan |
| Name | Simpan |
| 10 | jButton2 | Text | btnUbah |
| Name | Ubah |
| 12 | jButton3 | Text | btnHapus |
| Name | Hapus |
| 13 | jButton4 | Text | btnBatal |
| Name | Batal |
| 14 | jButton5 | Text | btnCari |
| Name | Cari |
| 15 | jButton6 | Text | btnNilai |

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | Name | Form Nilai |
| 16 | JTable | Name | tabel\_data |

*Source code* pada *method* batal():

public void batal(){ txtkode\_mk.setText(""); txtmatakuliah.setText(""); txtdosen.setText("");

}

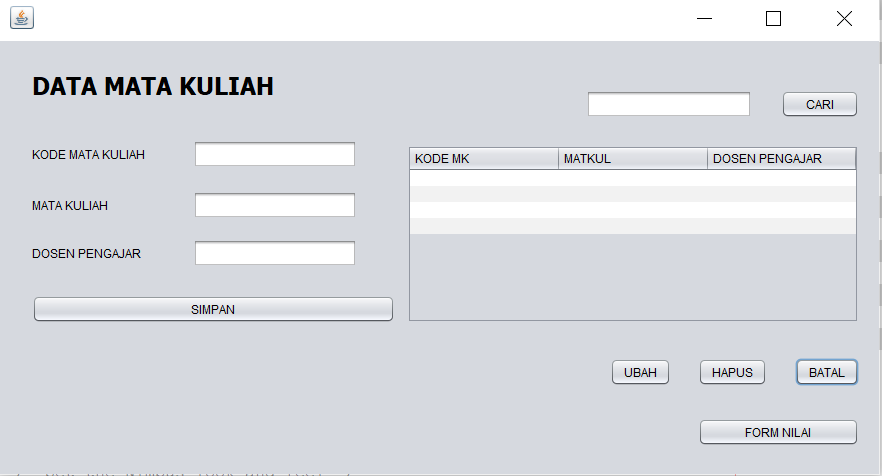
*Source code* pada *button* Batal:

private void btnBatalActionPerformed(java.awt.event.ActionEvent evt) {

batal();

}

### Tampilan Hasil:



### Gambar 5.5 Hasil Tampilan GUI\_Matkul

### Analisa :

### Pada program tersebut terdapat method *konstruktor* dengan nama batal yang berfungsi untuk reset ulang bagian *textfield* yang telah di isi inputan oleh *user* menjadi kosong .Pada *button* batal juga terdapat pemanggilan *method* tersebut di mana di dalamnya hanya ditulis batal().

## Tugas Praktikum 2 :

Menerapkan *Polimorfisme* pada GUI\_Nilai Diagram Class:

|  |
| --- |
| Penilaian |
| * String NIM, kode\_mk; * int nilaiTugas, nilaiPrak |
| # double nilaiAkhirPrak();  + double tampilNA()  + abstract double nilaiAkhir() |

* double nilaiAkhir()

Penilaian\_NonPrak

*Source code* pada *Super Class* Penilaian.java :

public abstract class Penilaian { String NIM, kode\_mk;

int nilaiTugas, nilaiPrak; private double nilaiAkhirPrak(){

return ((nilaiTugas\*0.6)+(nilaiPrak\*0.4));

}

double tampilNA(){

return nilaiAkhirPrak();

}

abstract double nilaiAkhir();

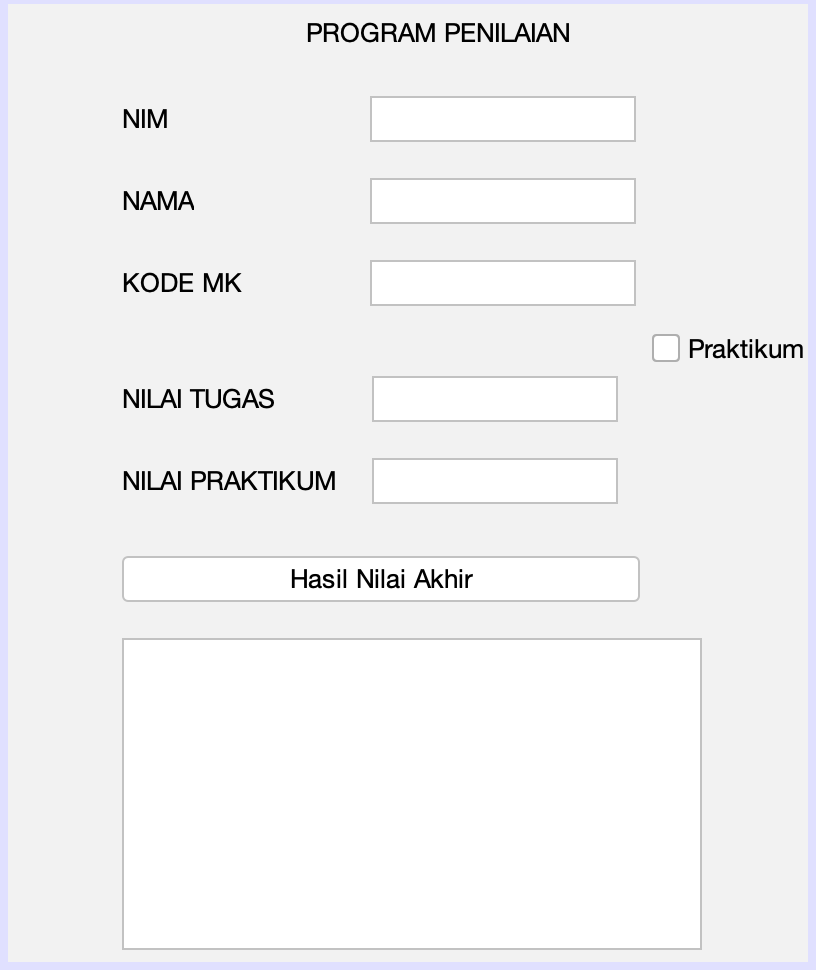
}

*Source code* pada *Sub Class* Penilaian\_NonPrak.java :

Public class Penilaian\_NonPrak extends Penilaian{ @Override

double nilaiAkhir() { return (nilaiTugas);}}

### Desain *form* (GUI\_Nilai.java) :



### Gambar 5.6 Desain *Form* GUI\_Nilai.java

Tabel 5.3 Properti Desain GUI\_Nilai (*Polimorfisme*)

|  |  |  |  |
| --- | --- | --- | --- |
| **No** | **Objek** | **Properti** | **Nilai** |
| 1 | jLabel1 | Text | PROGRAM PENILAIAN |
| 2 | jLabel2 | Text | NIM |
| 3 | jLabel3 | Text | Kode MK |
| 4 | jLabel4 | Text | Nilai Tugas |
| 5 | jLabel5 | Text | Nilai Praktikum |
| 6 | jTextField1 | Name | txtNIM |
| Text |  |
| 7 | jTextField2 | Name | txtKodeMK |
| Text |  |
| 8 | jTextField3 | Name | txtNT |
| Text |  |
| 9 | jTextField4 | Name | txtNP |
| Text |  |
| 10 | jButton1 | Name | btnNA |
| Text | Hasil Nilai Akhir |
| 11 | jTextArea | Name | memoNilai |
| Text |  |
| 12 | JCheckBox | Name | cek\_prak |
| Text | Praktikum |

*Source code* pada *Button* Hasil Nilai Akhir.java :

memoNilai.setText(""); Penilaian nilai; // instansiasi

//objek nilai dari super class diinisialisasi dengan sub class

nilai = new Penilaian\_NonPrak();

nilai.NIM = txtNIM.getText(); nilai.nama = txtNama.getText(); nilai.kode\_mk = txtKodeMK.getText();

nilai.setNT(Integer.parseInt(txtNT.getText())); nilai.setNP(Integer.parseInt(txtNP.getText()));

if (cek\_prak.isSelected()){ memoNilai.append("Nilai Akhir Mata Kuliah \n");

memoNilai.append("

\n");

memoNilai.append("NIM : "+nilai.NIM +"\n"); memoNilai.append("Nama : "+nilai.nama +"\n"); memoNilai.append("Kode MK : "+nilai.kode\_mk +"\n"); memoNilai.append("Nilai Tugas : "+nilai.getNT() +"\n"); memoNilai.append("Nilai Praktikum : "+nilai.getNP()

+"\n");

memoNilai.append("

\n");

memoNilai.append("Nilai Akhir :"+nilai.nilaiAkhirPrak());

}else {

memoNilai.append("Nilai Akhir Mata Kuliah \n"); memoNilai.append("

\n");

memoNilai.append("NIM : "+nilai.NIM +"\n"); memoNilai.append("Nama : "+nilai.nama +"\n"); memoNilai.append("Kode MK : "+nilai.kode\_mk +"\n"); memoNilai.append("Nilai Tugas : "+nilai.getNT() +"\n"); memoNilai.append("

\n");

memoNilai.append("Nilai Akhir : "+nilai.nilaiAkhir());

}

### Tampilan Hasil:

### Gambar 5.7 Hasil Tampilan GUI\_Nilai.java

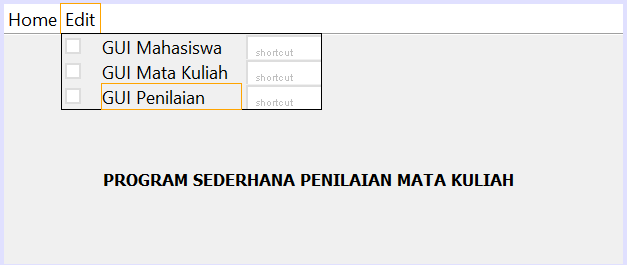
### Analisa :

### Pada program tersebut jika Ketika *user* mencentang *checkbox* maka akan diminta untuk menginputkan nilai praktikum, sebaliknya jika tidak dicentang progam akan membaca hanya terdapat nilai tugas sebagai nilai akhirnya. Di *source code* terdapat percabangan Boolean untuk *checkbox* dimana jika user mencentang atau bernilai 1 atau true maka dijumlahkanlah nilai tugas dan nilai praktikum. Jika *false* atau tidak dicentang maka hanya nilai tugas saja sebagai nilai akhirnya.

## Tugas praktikum 3 :

### Membuat GUI\_MenuUtama

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



### Gambar 5.8 Desain GUI\_MenuUtama Tabel 5.4 Property Desain GUI\_MenuUtama

|  |  |  |  |
| --- | --- | --- | --- |
| **No** | **Objek** | **Properti** | **Nilai** |
| 1 | jLabel1 | Text | PROGRAM SEDERHANA  PENILAIAN MATA KULIAH |
| 2 | jMenuBar | jMenu | [1] = Home  [2] = Form |
| jMenuItem | [1] = GUI Mahasiswa  [2] = GUI Mata Kuliah  [3] = GUI Nilai |

### Source code pada menu item GUI\_mahasiswa :

GUI\_Mahasiswa mhs = new GUI\_Mahasiswa(); mhs.show();

### Source code pada menu item Gui\_Matakuliah

GUI\_Matkul mk = new GUI\_Matkul(); mk.show();

### Source code pada menu item GUI\_Nilai:

GUI\_Nilai nilai = new GUI\_Nilai(); nilai.show();

### Tampilan GUI :

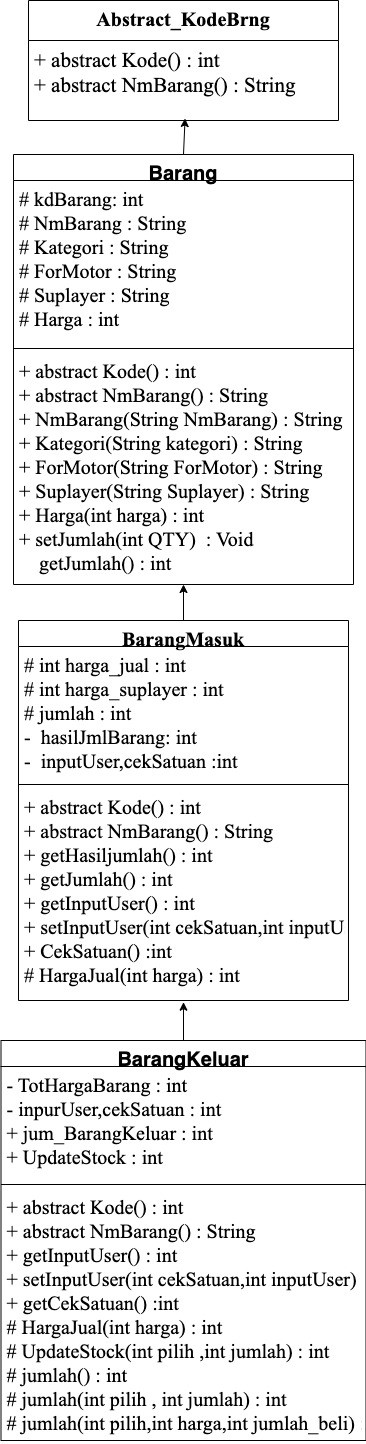


### Gambar 5.9 Hasil Running GUI\_MenuUtama

## Tugas Rumah 1 :

### Judul : Implementasi Class *Abstract* pada Class *Abstract*\_kodeBrng dan Class Barang, class barang masuk

### Diagram Class:



### Gambar 5.10 class diagram (*Abstract*) Source Code pada Abstract Class Abstract\_kodeBarang :

public abstract class Abstract\_KodeBrng { abstract int Kode();

abstract String NmBarang(String NmBarang);

}

*Source Code* pada *Super Class* Barang :

public class Barang extends Abstract\_KodeBrng{ protected int kdBarang, harga;

protected String NmBarang, Kategori, ForMotor,Suplayer; @Override

public int Kode(){

Random rndm = new Random(); kdBarang = rndm.nextInt(100000); return kdBarang;

}

@Override

public String NmBarang(String NmBarang){ this.NmBarang = NmBarang;

return NmBarang;

}

Barang(){

kdBarang = 123456 ; NmBarang = "Kampas Rem YSP";

ForMotor = "Veza,CB,CBR 150"; Suplayer= "PT. YSP";

harga = 35000;

}

public String Kategori(String Kategori){ this.Kategori = Kategori;

return Kategori;

}

public String ForMotor(String ForMotor){ this.ForMotor = ForMotor;

return ForMotor;

}

public String Suplayer(String Suplayer){ this.Suplayer = Suplayer;

return Suplayer;

}

public int Harga(int harga){ this.harga = harga; return harga;

}}

### Desain Form GUI\_Barang.java:



### Gambar 5.11 Desain Form GUI\_Barang(*Abstract*)

### Tabel 5.5 Property Desain GUI

|  |  |  |  |
| --- | --- | --- | --- |
| **No** | **Objek** | **Properti** | **Nilai** |
| 1 | jLabel1 | Text | “BARANG” |
| 2 | jLabel2 | Text | “KODE BARANG” |
| 3 | jLabel3 | Text | “FOR MOTOR” |
| 4 | jLabel4 | Text | “ SUPLAYER” |
| 5 | jLabel5 | Text | “KATEGORI” |
| 6 | JLabel6 | Text | “HARGA” |
| 7 | jTextField1 | Name | Txt\_KdBarang |
| Text | “ |
| 8 | jTextField2 | Text | “ “ |
| Name | Txt\_NmBarang |
| 9 | JTextField3 | Text | “ “ |
| Name | Txt\_ForMotor |
| 10 | JTextField3 | Text | “ “ |
| Suplayer | Txt\_Suplayer |
| 11 | TextField5 | Name | “Txt\_Harga” |
| text | “ “ |
| 12 | JComboBox1 | Name | Cmb\_Kategori |
| Model | --Pilih Kategori--, Oli, Part motor, Velek, Lampu Depan, Lampu Belakang, Pack Block/BB, Pack Kalter / CC, pack Kopling /KPL,  Baut, Mur |
| 13 | jButton1 | Name | Simpan |
| Text | “Simpan” |
| 14 | jTextArea | Name | Memo |
| Text | “ “ |

*Source code* button simpan

private void SimpanActionPerformed(java.awt.event.ActionEvent evt) {

int Berhasil = 0,Kode; Barang brg = new Barang();

brg.NmBarang(Txt\_NmBarang.getText()); brg.ForMotor(Txt\_ForMotor.getText()); brg.Suplayer(Txt\_Suplayer.getText()); brg.Harga(Integer.parseInt(Txt\_Harga.getText())); Kode = Integer.parseInt(Txt\_KdBarang.getText()); if(Cmb\_Kategori.getSelectedIndex()==0){

JOptionPane.showMessageDialog(null, "Kategori Belum Anda Masuukan");

}else if(Cmb\_Kategori.getSelectedIndex()==1){ Berhasil = 1;

brg.Kategori("oli");

}else if(Cmb\_Kategori.getSelectedIndex()==2){ Berhasil = 1;

brg.Kategori("Part Motor Mesin");

}else if(Cmb\_Kategori.getSelectedIndex()==3){ Berhasil = 1;

brg.Kategori("Lampu Depan");

}else if(Cmb\_Kategori.getSelectedIndex()==4){ Berhasil = 1;

brg.Kategori("Lampu Belakang");

}else if(Cmb\_Kategori.getSelectedIndex()==5){ Berhasil = 1;

brg.Kategori("Baut");

}else if(Cmb\_Kategori.getSelectedIndex()==6){ Berhasil = 1;

brg.Kategori("Mur");

}

if(Berhasil == 1){ JOptionPane.showMessageDialog(null,

"Berhasil!!!");

memo.append(" Kode Barang : " +Kode+"\n"); memo.append(" Nama Barang :

"+brg.NmBarang+"\n");

memo.append(" For Motor : "+brg.ForMotor+"\n"); memo.append(" Suplayer : "+brg.Suplayer+"\n"); memo.append(" Kategori : "+brg.Kategori+"\n"); memo.append(" Harga :Rp"+brg.harga);

}else{

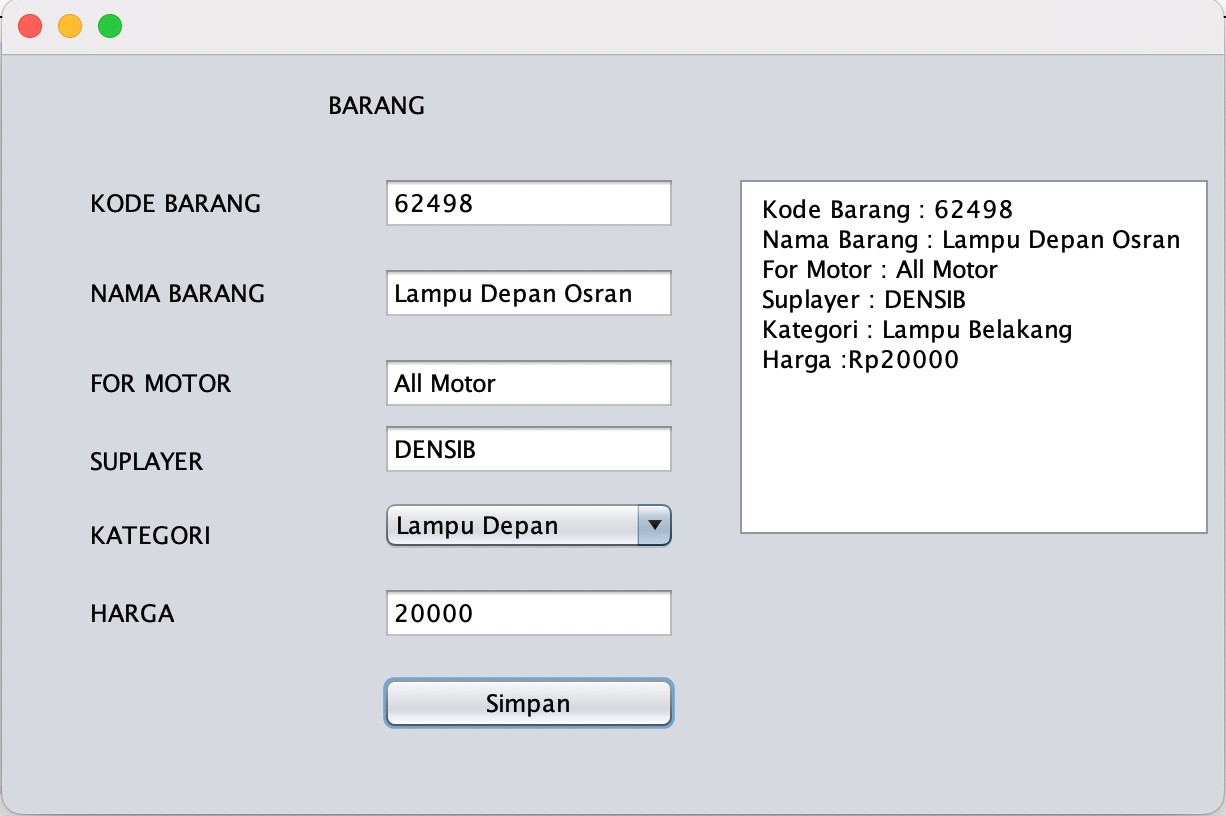
JOptionPane.showMessageDialog(null,

"Gagal!!!");

}

}

### Tampilan hasil :



### Gambar 5.12 Hasil Tampilan GUI\_Barang(*Abstract*)

*Source code* class BarangMasuk

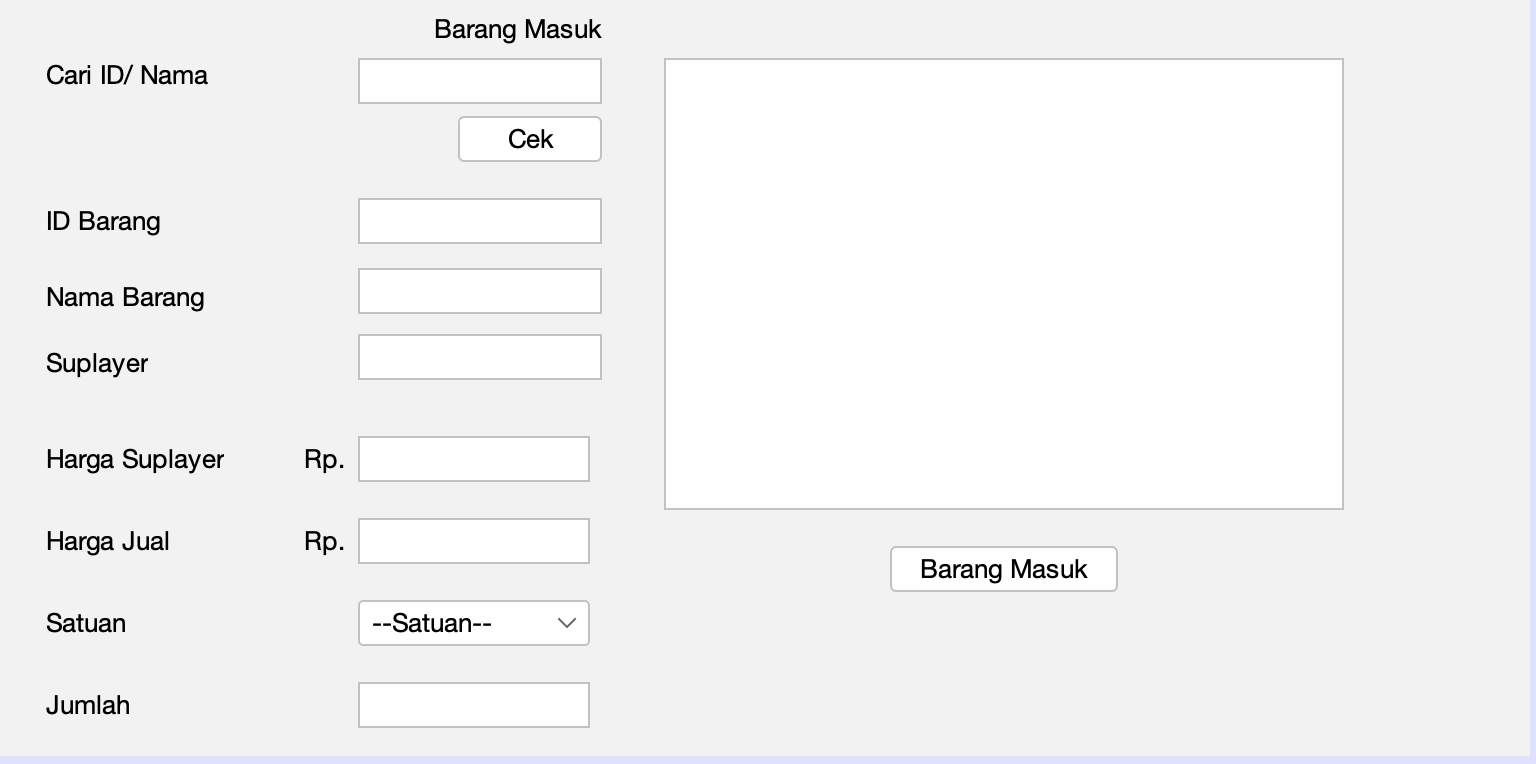
|  |  |  |
| --- | --- | --- |
| public class BarangMasuk extends Barang{ protected int harga\_jual, harga\_suplayer;  private int jumlah , hasilJmlBarang,inputUser,cekSatuan; @Override  public int Kode(){ return kdBarang;  }  @Override  public String NmBarang(String NmBarang){ this.NmBarang = NmBarang;  return NmBarang;  }  public BarangMasuk(){ harga\_suplayer = 45000;  jumlah = 200;  }  public int getHasiljumlah() { return jumlah;  }  //getter hasil jml barang public int getJumlah() {  return jumlah;  }  public int getInputUser() { return inputUser;  }  public void setInputUser(int cekSatuan,int inputUser) { this.inputUser = inputUser;  this.cekSatuan = cekSatuan; CekSatuan();  }  public int CekSatuan() { if(cekSatuan == 1){  jumlah = inputUser;  }else if(cekSatuan == 2){ jumlah = inputUser \* 12;  }else if(cekSatuan == 3){ jumlah = inputUser \* 24;  }  return jumlah; | | |
|  | Nama Aslab Pengajar : Silva Anjelina Tambunan | TTD : |
| Tanggal : |

}

protected int HargaJual(int harga){ this.harga\_jual = harga; return harga;

} }

### Desain Form Gui\_BarangMasuk



### Gambar 5.13 Hasil Tampilan GUI\_BarangMasuk(*Abstract*) Tabel 5.6 Property Desain GUI\_BarangMasuk

|  |  |  |  |
| --- | --- | --- | --- |
| **No** | **Objek** | **Properti** | **Nilai** |
| 1 | jLabel1 | Text | “Barang Masuk” |
| 2 | jLabel2 | Text | “Cari ID/Nama” |
| 3 | jLabel3 | Text | “ID Barang” |
| 4 | jLabel4 | Text | “Nama Barang” |
| 5 | jLabel5 | Text | “Suplayer” |
| 6 | jLabel6 | Text | “Harga Suplayer” |
| 7 | jLabel7 | Text | “Harga Jual” |
| 8 | jLabel8 | Text | “Satuan” |
| 9. | jLabel9 | Text | “Jumlah” |
| 9 | jTextField1 | Name | txt\_Search |
| Text |  |
| 10 | jTextField2 | Name | ID\_Barang |
| Text |  |
| 11 | jTextField3 | Name | txt\_NmBarang |
| Text |  |
| 12 | jTextField4 | Name | txt\_Suplayer |
| Text |  |

|  |  |  |  |
| --- | --- | --- | --- |
| 13 | jTextField6 | Name | txt\_HargaSuplayer |
| Text |  |
| 14 | jTextField7 | name | txt\_HargaJual |
| text |  |
| 15 | jTextField8 | neme | txt\_Jumlah |
| text |  |
| 16 | JComboBox | text |  |
| Name | cmb\_satuan |
| 17 | jButton1 | Name | btnCetak |
| Text | Cek |
| 18 | jButton2 | Neme | BarangMasuk |
| text | “BarangMasuk” |
| 17 | jTextArea | Name | memo |
| Text |  |

*Source code* Button Simpan

private void BarangMasukActionPerformed(java.awt.event.ActionEvent evt)

{

// TODO add your handling code here: int berhasil=0,cek;

BarangMasuk brgMasuk = new BarangMasuk(); ID\_Barang.setText(Integer.toString(brgMasuk.Kode()));

brgMasuk.NmBarang(txt\_NmBarang.getText()); brgMasuk.harga\_jual =

Integer.parseInt(txt\_HargaJual .getText()); brgMasuk.harga\_suplayer =

Integer.parseInt(txt\_HargaSuplayer .getText());

int jum = Integer.parseInt(txt\_Jumlah.getText()); memo.append("Kode Barang

:"+Integer.toString(brgMasuk.kdBarang)+"\n"); memo.append("Nama Barang

:"+brgMasuk.NmBarang+"\n"); memo.append("Nama Suplayer

:"+brgMasuk.Suplayer+"\n"); memo.append("Harga Suplayer

:"+brgMasuk.harga\_suplayer+"\n"); memo.append("Harga Jual

:"+brgMasuk.harga\_jual+"\n"); int pil =1 ;

if(cmb\_satuan.getSelectedIndex() == 0){ JOptionPane.showMessageDialog(null, "Masukkan

Satuan Barang");

}else if(cmb\_satuan.getSelectedIndex() == 1){ brgMasuk.setInputUser(1, jum);

memo.append("Jumlah Barang : "+Integer.toString(brgMasuk.getHasiljumlah()));

}else if(cmb\_satuan.getSelectedIndex() == 2){ brgMasuk.setInputUser(2, jum); memo.append("Jumlah Barang :

"+Integer.toString(brgMasuk.getHasiljumlah()));

}else if(cmb\_satuan.getSelectedIndex() == 3){ brgMasuk.setInputUser(3, jum); memo.append("Jumlah Barang :

"+Integer.toString(brgMasuk.getHasiljumlah()));

}

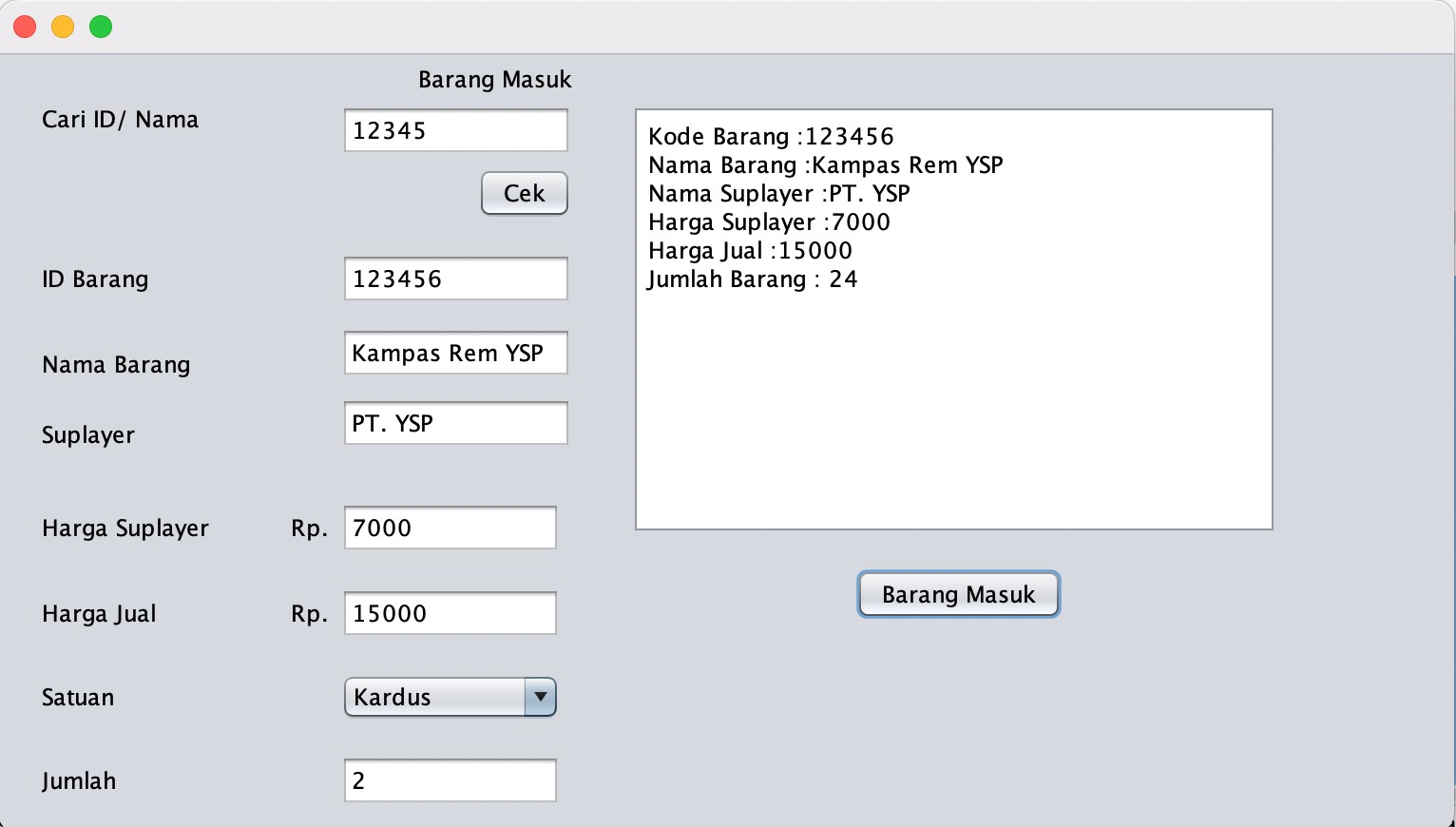
if(berhasil == 1){

JOptionPane.showMessageDialog(null, "Barang berhasil dimasukkan");

}

}

### Tampilan Hasil :



### Gambar 5.14 Tampilan hasil GUI\_BarangMasuk(*Abstract*)

*Source code* class BarangKeluar

public class BarangKeluar extends BarangMasuk{ private int TotHargaBarang;

private int inpurUser,cekSatuan; int jum\_BarangKeluar;

int UpdateStock;

@Override

public int Kode(){ return kdBarang;

}

@Override

public String NmBarang(String NmBarang){ this.NmBarang = NmBarang;

return NmBarang;

}

public int getInpurUser() { return inpurUser;

}

public int getCekSatuan() { return cekSatuan;

}

public void setInpurUser(int inpurUser,int cekSatuan) { this.inpurUser = inpurUser;

this.cekSatuan = cekSatuan;

}

protected int UpdateStock(int pilih ,int jumlah){ if(pilih == 1){

UpdateStock= getJumlah() - jumlah;

}else if(pilih == 2){

UpdateStock = getJumlah() - (jumlah \* 12);

}else if(pilih == 3){

UpdateStock = getJumlah()- (jumlah \* 24);

}

jumlah();

return UpdateStock;

}

protected int jumlah(){ return UpdateStock;

}

protected int jumlah(int pilih , int jumlah){ if(pilih == 1){

jum\_BarangKeluar= jumlah;

}else if(pilih == 2){ jum\_BarangKeluar = jumlah \* 12;

}else if(pilih == 3){ jum\_BarangKeluar = jumlah \* 24;

}

return jum\_BarangKeluar;

}

protected int jumlah(int pilih,int harga,int jumlah\_beli){

if(pilih == 1){

TotHargaBarang = harga \* jumlah\_beli;

}else if(pilih == 2){

TotHargaBarang = (jumlah\_beli \* 12) \* harga;

}else if(pilih == 3){

TotHargaBarang = (jumlah\_beli \* 24) \* harga;

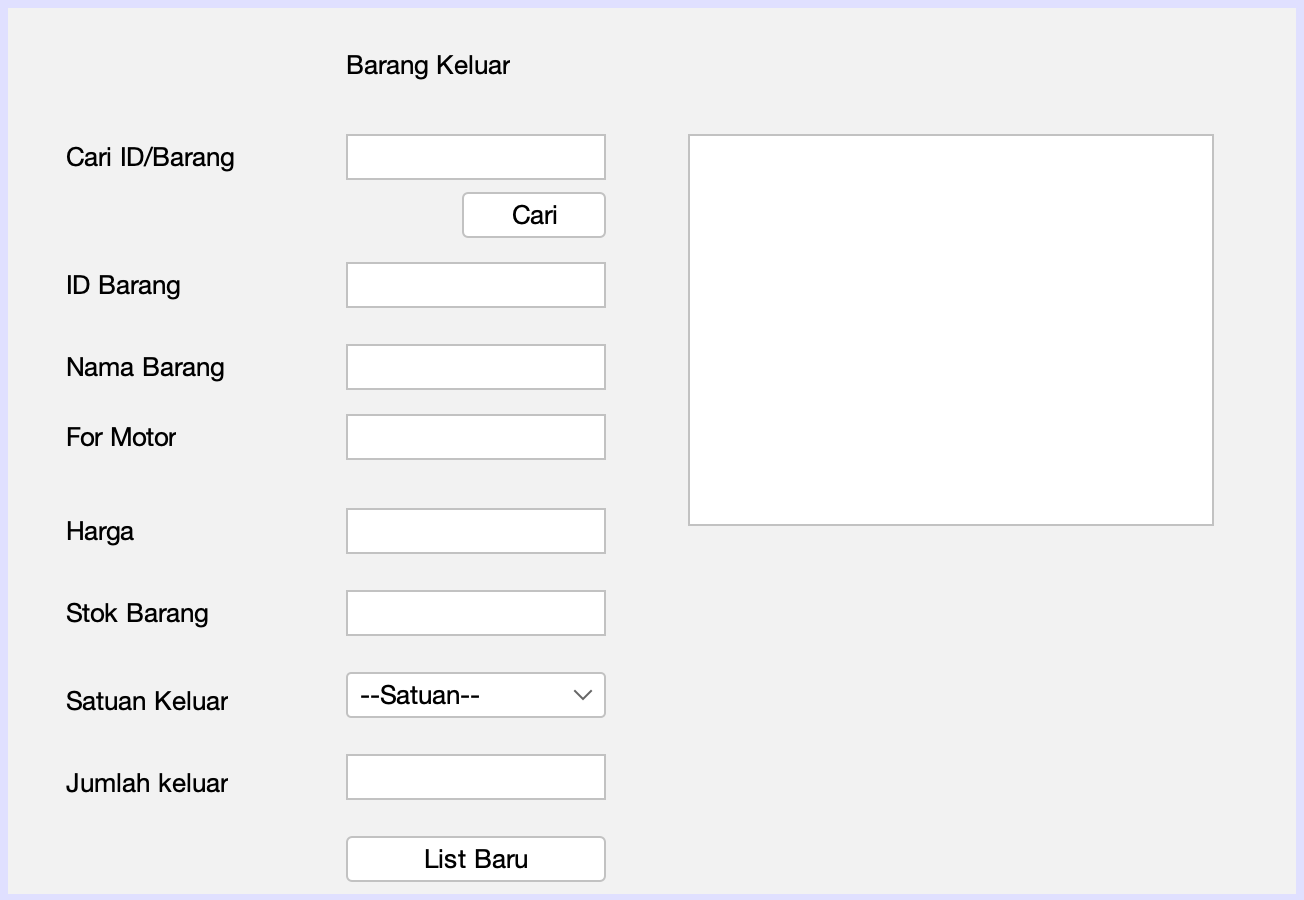
}

return TotHargaBarang;

}

}

### Desain Form BarangKeluar



### Gambar 5. 1 Desain Form GU\_BarangKeluar(*Abstract*)

### Tabel 5. 1 Property GUI\_BarangMasuk

|  |  |  |  |
| --- | --- | --- | --- |
| **No** | **Objek** | **Properti** | **Nilai** |
| 1 | jLabel1 | Text | “Barang Keluar” |
| 2 | jLabel2 | Text | “Cari ID/Nama” |
| 3 | jLabel3 | Text | “ID Barang” |
| 4 | jLabel4 | Text | “Nama Barang” |
| 5 | jLabel5 | Text | “For Motor” |
| 6 | jLabel6 | Text | “Harga” |
| 7 | jLabel7 | Text | “Stock Barang” |
| 8 | jLabel8 | Text | “Satuan Barang” |
| 9. | jLabel9 | Text | “Jumlah keluar” |
| 9 | jTextField1 | Name | txt\_Search |
| Text |  |
| 10 | jTextField2 | Name | ID\_Barang |
| Text |  |
| 11 | jTextField3 | Name | txt\_NmBarang |
| Text |  |
| 12 | jTextField4 | Name | txt\_forMotor |
| Text |  |
| 13 | jTextField6 | Name | txt\_Harga |
| Text |  |
| 14 | jTextField7 | name | txt\_StokBarang |
| text |  |
| 15 | jTextField8 | neme | txt\_Jml |
| text |  |
| 16 | JComboBox | Model | --Satuan-- Unit Kardus  kodi |
| Name | cmb\_satuan |
| 17 | jButton1 | Name | btn\_cari |

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | Text | Cari |
| 18 | jButton2 | Name | btn\_proses |
| text | “List baru” |
| 17 | jTextArea | Name | memo |
| Text |  |

*Source code* button (List Baru)

private void prosesActionPerformed(java.awt.event.ActionEvent evt) {

// TODO add your handling code here: BarangKeluar brgKeluar = new BarangKeluar();

ID\_Barang.setText(Integer.toString(brgKeluar.Kode())); brgKeluar.NmBarang(txt\_NmBarang.getText()); int ID = Integer.parseInt(ID\_Barang.getText()); String nama = txt\_NmBarang.getText();

int harga = Integer.parseInt(txt\_Harga.getText()); int stok =

Integer.parseInt(txt\_StokBarang.getText());

int jumlah = Integer.parseInt(txt\_Jml.getText()); memo.append("Kode Barang

:"+Integer.toString(ID)+"\n"); memo.append("Nama Barang :"+nama+"\n"); if(cmb\_Satuan.getSelectedIndex()==1){

JOptionPane.showMessageDialog(null, "Masukkan Satuan Barang");

}else if(cmb\_Satuan.getSelectedIndex()==1){ memo.append("jumlah Barang

:"+brgKeluar.jumlah(1, jumlah)+"\n");

memo.append("Harga total:"+brgKeluar.jumlah(1,harga, jumlah)+"\n");

brgKeluar.UpdateStock(1, jumlah);

}else if(cmb\_Satuan.getSelectedIndex()==2){ memo.append("jumlah Barang

:"+brgKeluar.jumlah(2, jumlah)+"\n");

memo.append("Harga total:"+brgKeluar.jumlah(2,harga, jumlah)+"\n");

brgKeluar.UpdateStock(2, jumlah);

}else if(cmb\_Satuan.getSelectedIndex()==3){ memo.append("jumlah Barang

:"+brgKeluar.jumlah(3, jumlah)+"\n");

memo.append("Harga total:"+brgKeluar.jumlah(3,harga, jumlah)+"\n");

brgKeluar.UpdateStock(3, jumlah);

}

memo.append("Stok Barang tersedia

:"+brgKeluar.jumlah()+"\n");

}

*Source code* Button cari

private

void

jButton1ActionPerformed(java.awt.event.ActionEvent evt) {

// TODO add your handling code here: BarangMasuk brgMasuk = new BarangMasuk(); if(

txt\_Search.getText().equalsIgnoreCase("12345")|| txt\_Search.getText().equalsIgnoreCase("Kampas rem ysp")){

JOptionPane.showMessageDialog(null, "Barang Ditemukan"); ID\_Barang.setText(Integer.toString(brgMasuk.kdBarang));

txt\_NmBarang.setText(brgMasuk.NmBarang); txt\_ForMotor.setText(brgMasuk.ForMotor);

txt\_Harga.setText(Integer.toString(brgMasuk.harga)); txt\_StokBarang.setText(Integer.toString(brgMasuk.getJumlah(

)));

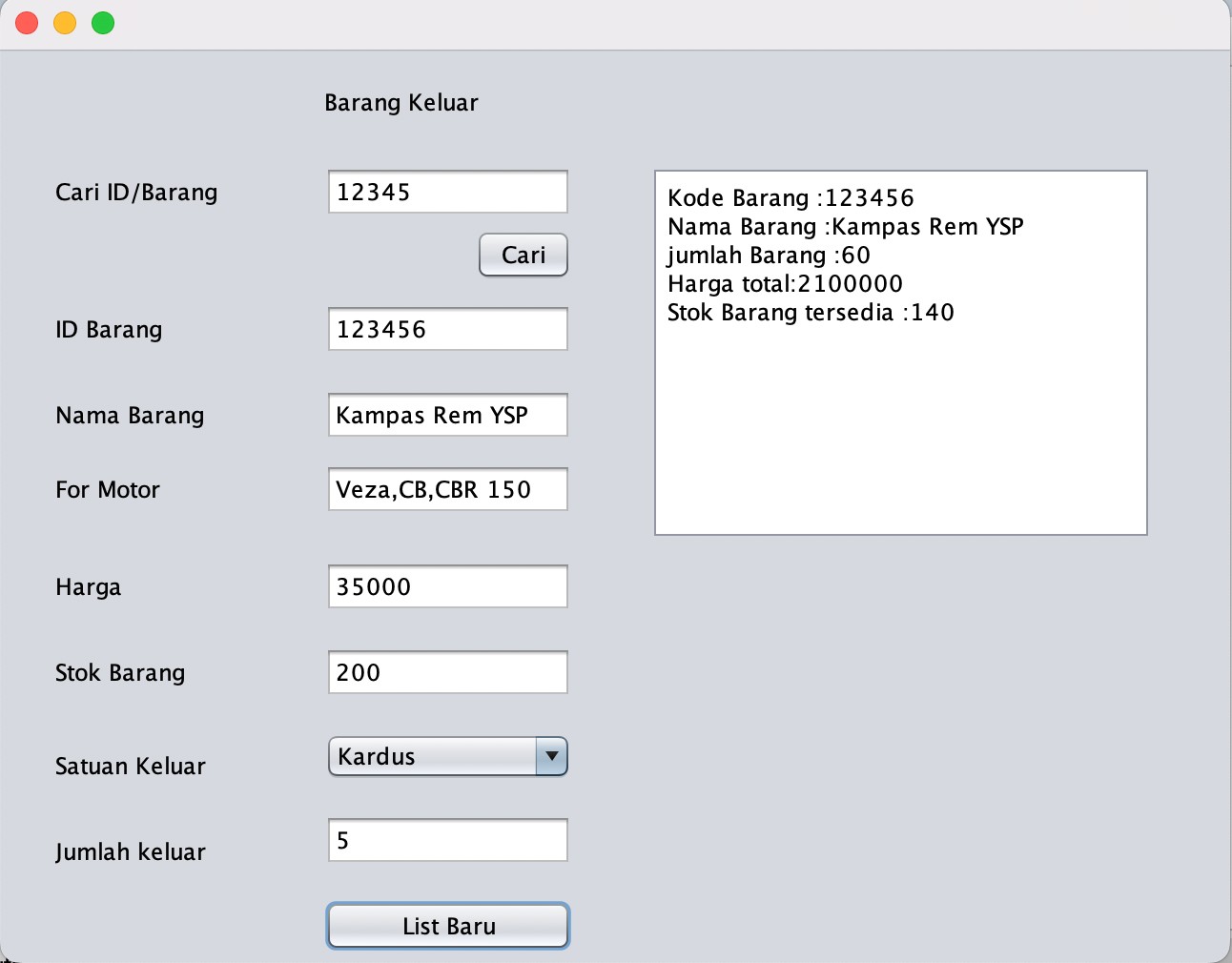
}else{

JOptionPane.showMessageDialog(null, "Barang belum di tambahkan!!!!");

}

}

### Tampilan Hasil :



### Gambar 5. 2 Tampilan Hasil Gui (*Abstract*)

### Analisa :

### Pada program di atas *method abstract* terletak pada class abstract\_kodeBrng. Disini method *abstract* yaitu pada kode() dan pada NmBarang(). Kedua method abstract tersebut akan mewarisi sifat/ method pada class-class yang ada di bagian bawahnya. Jadi pada class diagaram diatas parent class memiliki class *abstract* yang mana akan diturunkan pada kelas- kelas anak.

## Tugas Rumah 2:

Menambahkan *Button Exception Handling* pada GUI\_Barang

*Source code* pada *Class* Barang.java:

public class Barang extends Abstract\_KodeBrng{

protected String kdBarang,NmBarang,Kategori,ForMotor, Suplayer;

int harga\_jual;

int harga\_suplayer; @Override

public String Kode(String kd){ this.kdBarang = kd; return kd;

}

@Override

public String NmBarang(String NmBarang){ this.NmBarang = NmBarang;

return NmBarang;

}

@Override

public String ForMotor(String ForMotor){ this.ForMotor = ForMotor;

return ForMotor;

}

Barang(){

kdBarang = "KZL345356467464" ;

NmBarang = "Kampas Rem YSP"; ForMotor = "Veza,CB,CBR 150"; Suplayer= "PT. YSP";

}

public String Kategori(String Kategori){ this.Kategori = Kategori;

return Kategori;

}

public String Suplayer(String Suplayer){ this.Suplayer = Suplayer;

return Suplayer;

}

}

*Source code* pada GUI\_Barang.java:

private void btn\_cekActionPerformed(java.awt.event.ActionEvent evt) {

// TODO add your handling code here: try{

Barang brg;

brg= new Barang(); brg.Kode(Txt\_KdBarang.getText()); brg.Kode(Txt\_KdBarang.getText()); brg.NmBarang(Txt\_NmBarang.getText()); brg.Suplayer(Txt\_Suplayer.getText()); if(Cmb\_Kategori.getSelectedIndex()==0) throw

new Exception();

}catch(Exception e){ JOptionPane.showConfirmDialog(rootPane, "Input

Field Salah","WARNING",2);}}

### Desain form GUI\_Barang:



### Gambar 5.15 Desain GUI\_Barang.java(*TryCatch*) Tabel 5.7 Property GUI\_Barang(*TryCatch*)

|  |  |  |  |
| --- | --- | --- | --- |
| **No** | **Objek** | **Properti** | **Nilai** |
| 1 | jLabel1 | Text | “TAMBAH BARANG” |
| 2 | jLabel2 | Text | “KODE BARANG” |
| 3 | jLabel3 | Text | “NAMA BARANG” |
| 4 | jLabel4 | Text | “KATEGORI” |
| 5 | jLabel5 | Text | “SUPLAYER” |
| 6 | jLabel6 | Text | “FOR MOTOR” |
| 7 | JtextField1 | Text | “ ” |
| Name | Txt\_KdBarang |
| 8 | JtextField2 | Text | “ ” |
| Name | Txt\_NmBarang |
| 9 | JtextField3 | Text | “ ” |
| Name | Txt\_Suplayer |
| 10 | JtextArea4 | Text | “ ” |
| Name | txt\_ForMotor |
| 11 | JComboBox5 | Name | Cmb\_Kategori |
| Model | --Pilih Kategori--, OLI, BAN,  LAMPU, INNER |
| 12 | JTable | Name | Tabel |
| Title | Kode, Nm Barang, Kategori,  Suplayer, For Motor |
| 13 | JButton | Name | btn\_simpan |
| Title | “Simpan ” |
| 14 | JButton2 | Name | btn\_cek |

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | Title | “Cek” |
| 15 | JButton3 | Name | btn\_ubah |
| Title | “Ubah” |
| 16 | JButton4 | Name | btn\_hapus |
| Title | “Hapus” |
| 17 | JButton5 | Name | btn\_batal |
| Title | “Batal” |

### Tampilan Hasil:



Gambar 5.16 Hasil tampilan GUI\_Barang(*TryCatch*) Menambahkan *Button Exception Handling* pada GUI\_BarangMasuk *Source code* pada *Class* BarangMasuk.java:

public class BarangMasuk extends Barang{ protected int harga\_jual;

protected int harga\_suplayer; private int jumlah;

private int inputUser,cekSatuan; private String tanggal; @Override

public String Kode(String kd){ this.kdBarang = kd; return kd;

}

@Override

public String NmBarang(String NmBarang){ this.NmBarang = NmBarang;

return NmBarang;

}

@Override

public String ForMotor(String ForMotor){ this.ForMotor = ForMotor;

return ForMotor;

}

public BarangMasuk(){ harga\_suplayer = 45000;

jumlah = 200;

}

//getter hasil jml barang public int getJumlah() {

return jumlah;

}

// setter dan getter input barang public int getInputUser() {

return inputUser;

}

public void setInputUser(int cekSatuan,int inputUser) { this.inputUser = inputUser;

this.cekSatuan = cekSatuan; CekSatuan();

}

public int CekSatuan() { if(cekSatuan == 1){

jumlah = inputUser;

}else if(cekSatuan == 2){ jumlah = inputUser \* 12;

}else if(cekSatuan == 3){ jumlah = inputUser \* 24;

}

return jumlah;

}

protected int HargaJual(int harga){ this.harga\_jual = harga; return harga;

}

protected int HargaSuplayer(int hargaspy){ this.harga\_suplayer = hargaspy; return hargaspy;

}

public String getTanggal() { return tanggal;

}

public void setTanggal(String tanggal) { this.tanggal = tanggal;

}}

*Source code* pada *GUI\_*BarangMasuk.java:

private void btn\_cekActionPerformed(java.awt.event.ActionEvent evt) {

// TODO add your handling code here: try{

Barang brg;

brg = new BarangMasuk();

//penerapan polimor downcasting BarangMasuk brgmsk = (BarangMasuk) brg; brgmsk.Kode(ID\_Barang.getText()); String tampilan = "yyyy-MM-dd"; SimpleDateFormat format = new

SimpleDateFormat(tampilan);

brgmsk.setTanggal(String.valueOf(format.format(txt\_tanggal. getDate())));

brg.NmBarang(txt\_NmBarang.getText()); brg.Suplayer(txt\_Suplayer.getText());

brgmsk.HargaSuplayer(Integer.parseInt(txt\_HargaSuplayer.get Text()));

brgmsk.HargaJual(Integer.parseInt(txt\_HargaJual.getText()))

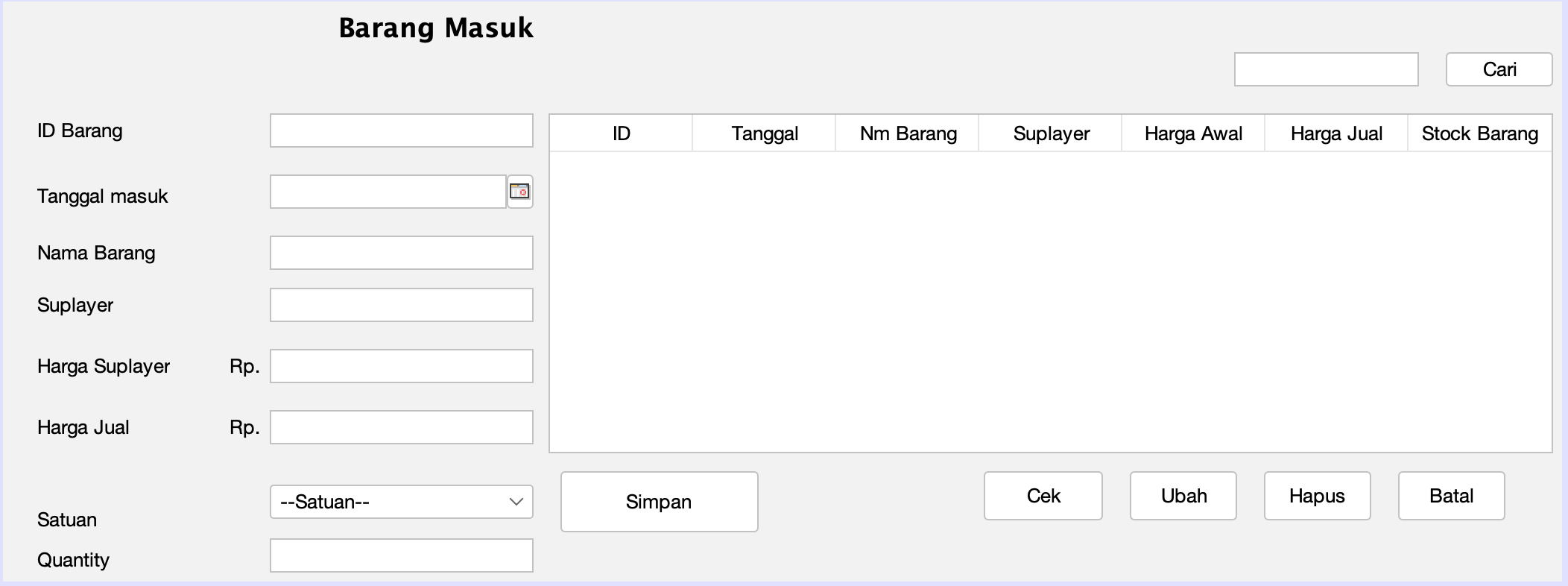
;

if(cmb\_satuan.getSelectedIndex() == 0) throw new Exception();

}catch(Exception e){ JOptionPane.showConfirmDialog(rootPane, "Input

Field Salah","WARNING",2);}}

### Desain form GUI\_BarangMasuk:

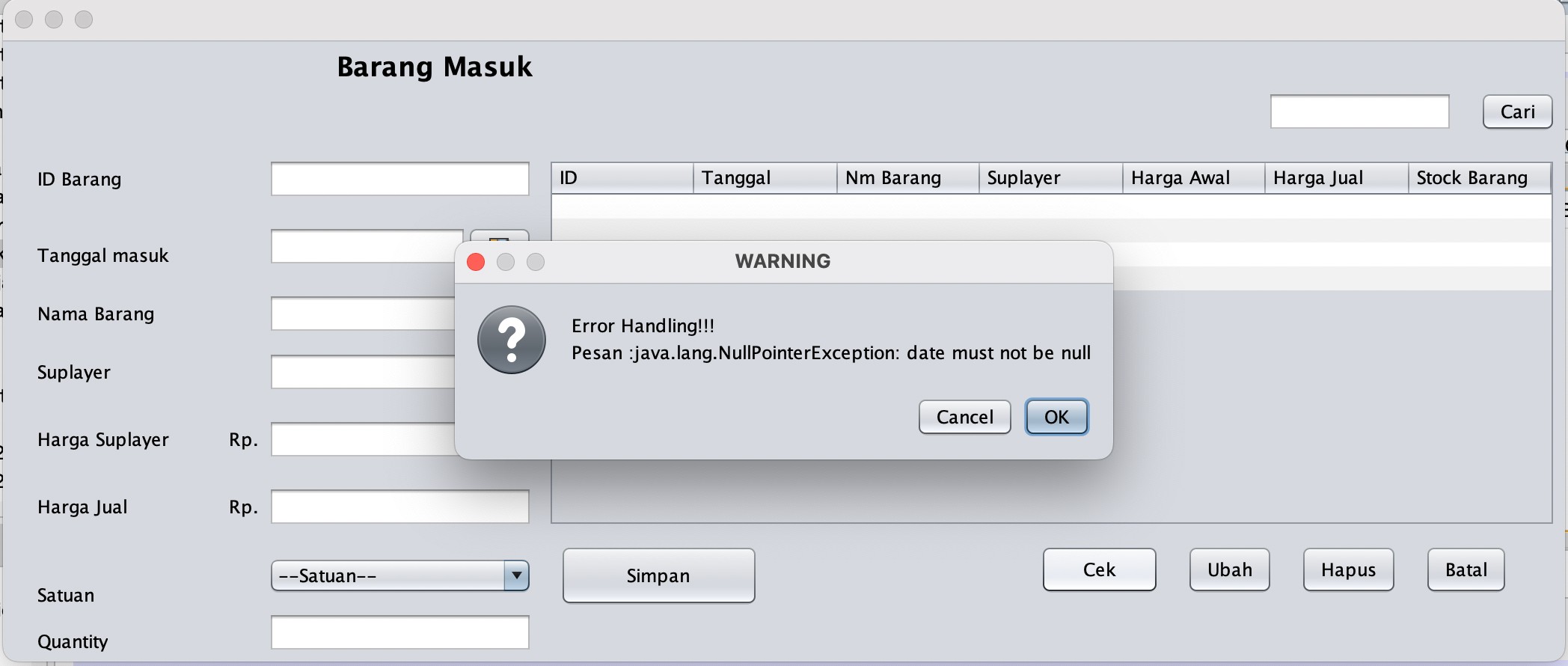


### Gambar 5.17 Desain GUI\_BarangMasuk.java (*TryCatch*) Tabel 5.8 Property Gui\_BarangMasuk(*TryCatch*)

|  |  |  |  |
| --- | --- | --- | --- |
| **No** | **Objek** | **Properti** | **Nilai** |
| 1 | jLabel1 | Text | “BARANG MASUK” |
| 2 | jLabel2 | Text | “ID Barang” |
| 3 | jLabel3 | Text | “Nama Barang” |
| 4 | jLabel4 | Text | “Harga Suplayer” |
| 5 | jLabel5 | Text | “Suplayer” |
| 6 | jLabel6 | Text | “Harga Suplayer” |
| 7 | jLabel7 | Text | “Harga Jual” |
| 8 | jLabel8 | Text | “Satuan” |
| 9 | jLabel9 | Text | “Quantity” |
| 10 | jCalender | Text | “ ” |
| Name | txt\_tanggal |
| 11 | JtextField1 | Text | “ ” |
| Name | ID\_Barang |
| 12 | JtextField2 | Text | “ ” |
| Name | txt\_NmBarang |
| 13 | JtextField3 | Text | “ ” |
| Name | txt\_Suplayer |
| 14 | JtextField4 | Text | “ ” |
| Name | txt\_HargaSuplayer |
| 15 | JtextField5 | Text | “ ” |
| Name | txt\_HargaJual |
| 16 | JtextField6 | Text | “ ” |
| Name | txt\_Jumlah |
| 17 | JtextField7 | Text | “ ” |

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | Name | txt\_Search |
| 18 | JComboBox | Model | --Satuan--, Unit, Kardus,  kodi |
| Name | cmb\_satuan |
| 19 | JButton1 | Text | “ Simpan” |
| Name | btn\_Simpan |
| 20 | JButton2 | Text | “ Cek” |
| Name | btn\_cek |
| 21 | JButton3 | Text | “ Ubah“ |
| Name | btn\_ubah |
| 22 | JButton4 | Text | “Hapus” |
| Name | btn\_Hapus |
| 23 | JButton | Text | “”Cari |
| Name | btn\_Search |
| 24 | JTabel | Title | ID,Tanggal,Nm Barang,  Suplayer, Harga Awal, Harga Jual, Stock Barang |
| Name | tabel\_BarangMasuk |

### Tampilan Hasil :



Gambar 5.18 Hasil Tampilan GUI\_BarangMasuk(*TryCatch*) Menambahkan *Button Exception Handling* pada GUI\_BarangKeluar *Source code* pada *Class* BarangKeluar.java:

public class BarangKeluar extends BarangMasuk{ private int TotHargaBarang,inpurUser,cekSatuan; int jum\_BarangKeluar, UpdateStock;

private String tanggalKeluar; public String getTanggalKeluar() {

return tanggalKeluar;

}

public void setTanggalKeluar(String tanggalKeluar) { this.tanggalKeluar = tanggalKeluar;

}

@Override

public String Kode(String kd){ this.kdBarang = kd; return kd;

}

@Override

public String NmBarang(String NmBarang){ this.NmBarang = NmBarang;

return NmBarang;

}

@Override

public String ForMotor(String ForMotor){ this.ForMotor = ForMotor;

return ForMotor;

}

public int getInpurUser() { return inpurUser;

}

public int getCekSatuan() { return cekSatuan;

}

public void setInpurUser(int inpurUser,int cekSatuan) { this.inpurUser = inpurUser;

this.cekSatuan = cekSatuan;

}

protected int UpdateStock(int pilih ,int jumlah){ if(pilih == 1){

UpdateStock= getJumlah() - jumlah;

}else if(pilih == 2){

UpdateStock = getJumlah() - (jumlah \* 12);

}else if(pilih == 3){

UpdateStock = getJumlah()- (jumlah \* 24);

}

jumlah();

return UpdateStock;

}

protected int jumlah(){ return UpdateStock;

}

protected int jumlah(int pilih , int jumlah){ if(pilih == 1){

jum\_BarangKeluar= jumlah;

}else if(pilih == 2){ jum\_BarangKeluar = jumlah \* 12;

}else if(pilih == 3){ jum\_BarangKeluar = jumlah \* 24;

}

return jum\_BarangKeluar;

}

protected int jumlah(int pilih,int harga,int jumlah\_beli){

if(pilih == 1){

TotHargaBarang = harga \* jumlah\_beli;

}else if(pilih == 2){

TotHargaBarang = (jumlah\_beli \* 12) \* harga;

}else if(pilih == 3){

TotHargaBarang = (jumlah\_beli \* 24) \* harga;

}

return TotHargaBarang;

}

}

*Source code* pada *GUI\_*BarangKeluar.java:

private void jButton5ActionPerformed(java.awt.event.ActionEvent evt) {

// TODO add your handling code here: try{

Barang brg;

brg = new BarangKeluar();

BarangKeluar brgklr = (BarangKeluar) brg; brgklr.Kode(ID\_Barang.getText());

String tampilan = "yyyy-MM-dd"; SimpleDateFormat format = new

SimpleDateFormat(tampilan); brgklr.setTanggalKeluar(String.valueOf(format.format(txt\_Ta nggal.getDate())));

brg.NmBarang(txt\_NmBarang.getText()); brg.ForMotor(txt\_ForMotor.getText()); brg = new BarangMasuk();

BarangMasuk brgmsk = (BarangMasuk) brg; brgmsk.HargaJual(Integer.parseInt(txt\_Harga.getText()));

if(cmb\_Satuan.getSelectedIndex() == 0) throw new Exception();

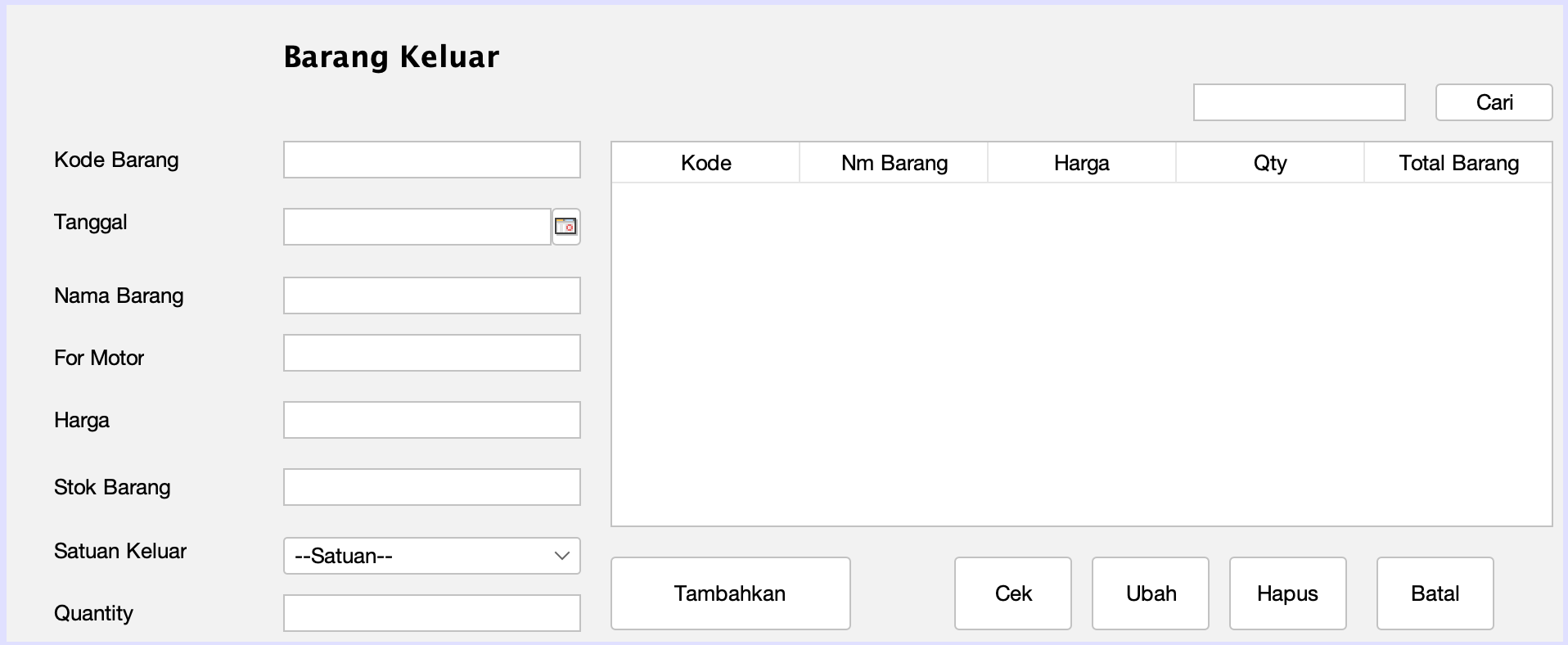
}catch(Exception e){ JOptionPane.showConfirmDialog(rootPane, "Error

Handling!!!\nPesan :"+e,"WARNING",2);

}

}

### Desain Form Gui\_BarangKeluar



### Gambar 5.19 Desain GUI\_BarangKeluar(*TryCatch*) Tabel 5.9 Property Desain GUI\_BarangKeluar

|  |  |  |  |
| --- | --- | --- | --- |
| **No** | **Objek** | **Properti** | **Nilai** |
| 1 | jLabel1 | Text | “BARANG KELUAR” |
| 2 | jLabel2 | Text | “Kode Barang” |
| 3 | jLabel3 | Text | “Tanggal” |
| 4 | jLabel4 | Text | “Nama Barang” |
| 5 | jLabel5 | Text | “For Motor” |
| 6 | jLabel6 | Text | “Harga” |
| 7 | jLabel7 | Text | “Stock Barang” |
| 8 | jLabel8 | Text | “Satua Barang” |
| 9 | jLabel9 | Text | “Quantity” |

|  |  |  |  |
| --- | --- | --- | --- |
| 10 | jCalender | Text | “ ” |
| Name | txt\_Tanggal |
| 11 | JtextField1 | Text | “ ” |
| Name | ID\_Barang |
| 12 | JtextField2 | Text | “ ” |
| Name | txt\_NmBarang |
| 13 | JtextField3 | Text | “ ” |
| Name | txt\_ForMotor |
| 14 | JtextField4 | Text | “ ” |
| Name | txt\_Harga |
| 15 | JtextField5 | Text | “ ” |
| Name | txt\_StokBarang |
| 16 | JtextField6 | Text | “ ” |
| Name | txt\_Jml |
| 17 | JtextField7 | Text | “ ” |
| Name | txt\_Search |
| 18 | JComboBox | Model | --Satuan--, Unit,Kardus, kodi |
| Name | cmb\_Satuan |
| 19 | JButton1 | Text | “ Tambahkan” |
| Name | btn\_Tambahkan |
| 20 | JButton2 | Text | “ Cek” |
| Name | btn\_cek |
| 21 | JButton3 | Text | “ Ubah“ |
| Name | btn\_ubah |
| 22 | JButton4 | Text | “Hapus” |
| Name | btn\_Hapus |
| 23 | JButton5 | Text | “”Cari |
| Name | btn\_Search |
| 24 | JTabel | Title | Kode,Nm Barang, Harga, QTY, Total Barang Barang |
| Name | tabel\_BarangKeluar |

### Tampilan Hasil :

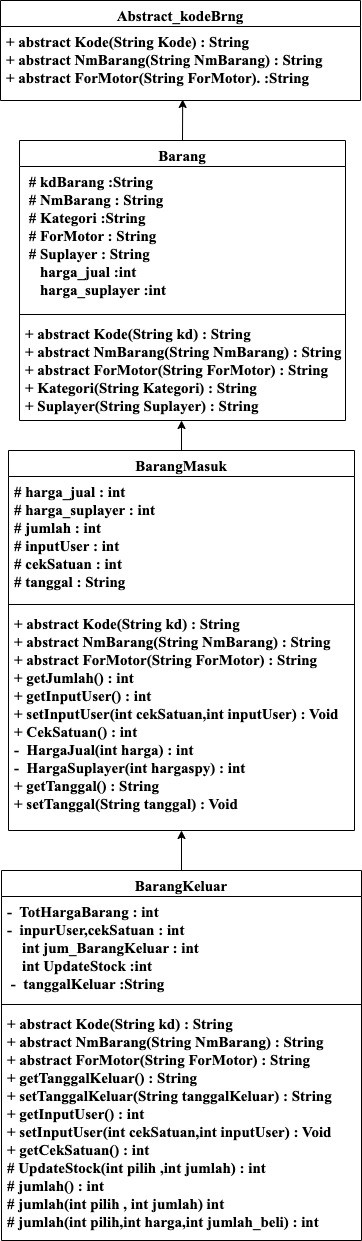


### Gambar 5.20 Hasil Tampilan GUI\_BarangKeluar(*TryCatch*)

### Analisa:

### Pada program di atas *try catch* digunakan untuk menghandle *error*, *error* dari program di atas *handle error* ketika program dijalankan , Ketika *user* salah menginputkan nilai atau user belum menginputkan nilai maka akan menampilkan warning dan peringatan eror , kesalahan *error* juga akan ditampilkan pada pop *up box error*. Ini berguna Ketika user lupa belum memasukkan inputan pada *textfield*.

## Tugas Rumah 3:

Menerapkan *Polimorfisme* pada GUI\_Barang Diagram Class:

Gambar 5.21 Class Diagram *Polimorfisme*

*Source code* pada *Super Class* Barang.java:

public class Barang extends Abstract\_KodeBrng{

protected String kdBarang,NmBarang,Kategori,ForMotor, Suplayer;

int harga\_jual,harga\_suplayer; @Override

public String Kode(String kd){ this.kdBarang = kd; return kd;

}

@Override

public String NmBarang(String NmBarang){ this.NmBarang = NmBarang;

return NmBarang;

}

@Override

public String ForMotor(String ForMotor){ this.ForMotor = ForMotor;

return ForMotor;

}

Barang(){

kdBarang = "KZL345356467464" ;

NmBarang = "Kampas Rem YSP"; ForMotor = "Veza,CB,CBR 150"; Suplayer= "PT. YSP";

}

public String Kategori(String Kategori){ this.Kategori = Kategori;

return Kategori;

}

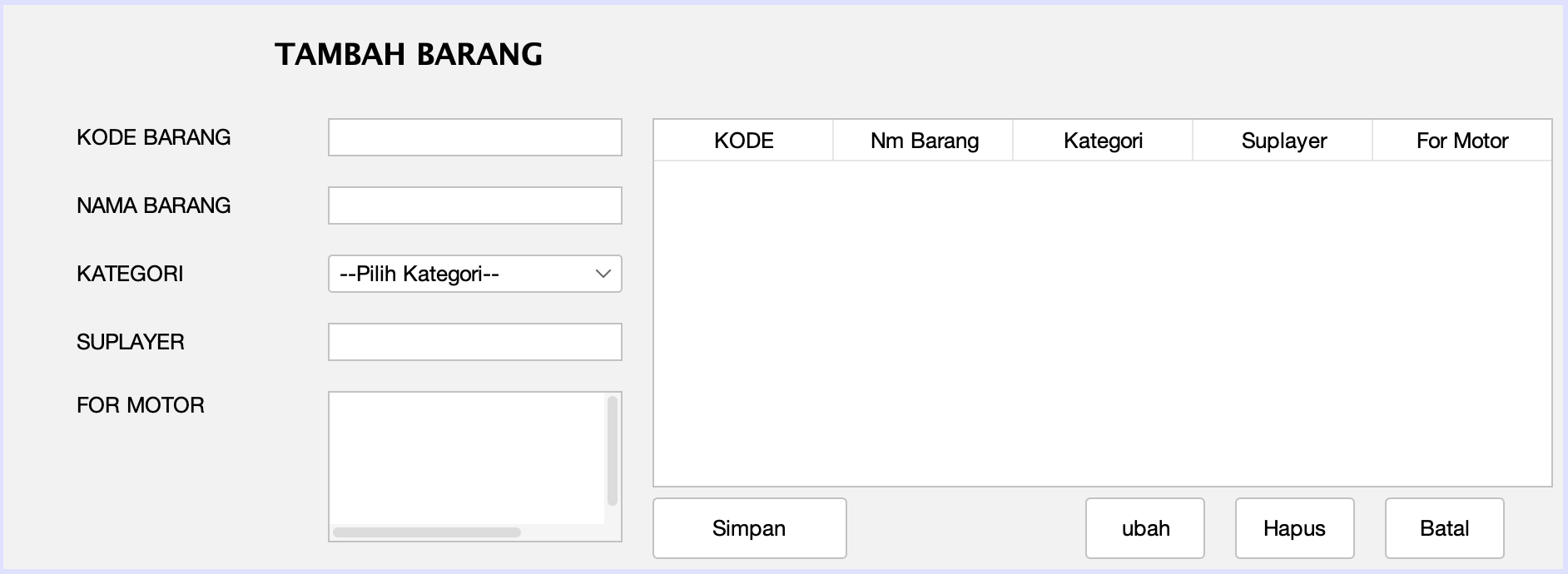
public String Suplayer(String Suplayer){ this.Suplayer = Suplayer;

return Suplayer;

}

}

### Desain form GUI\_Barang:



Gambar 5.22 Desain form GUI\_Barang.java(*Polimorfisme*) Tabel 5.10 Property desain GUI\_Barang(*Polimorfisme*)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **No** | **Objek** | | **Properti** | **Nilai** | |
| 1 | jLabel1 | | Text | “TAMBAH BARANG” | |
| 2 | jLabel2 | | Text | “KODE BARANG” | |
| 3 | jLabel3 | | Text | “NAMA BARANG” | |
|  | | Nama Aslab Pengajar : Silva Anjelina Tambunan | | | TTD : |
| Tanggal : | | |

|  |  |  |  |
| --- | --- | --- | --- |
| 4 | jLabel4 | Text | “KATEGORI” |
| 5 | jLabel5 | Text | “SUPLAYER” |
| 6 | jLabel6 | Text | “FOR MOTOR” |
| 7 | JtextField1 | Text | “ ” |
| Name | Txt\_KdBarang |
| 8 | JtextField2 | Text | “ ” |
| Name | Txt\_NmBarang |
| 9 | JtextField3 | Text | “ ” |
| Name | Txt\_Suplayer |
| 10 | JtextArea4 | Text | “ ” |
| Name | txt\_ForMotor |
| 11 | JComboBox5 | Name | Cmb\_Kategori |
| Model | --Pilih Kategori--, OLI, BAN,  LAMPU, INNER |
| 12 | JTable | Name | Tabel |
| Title | Kode, Nm Barang, Kategori,  Suplayer, For Motor |
| 13 | JButton1 | Name | btn\_simpan |
| Title | “Simpan ” |
| 14 | JButton2 | Name | btn\_ubah |
| Title | “Ubah” |
| 15 | JButton3 | Name | btn\_hapus |
| Title | “Hapus” |
| 16 | JButton4 | Name | btn\_batal |
| Title | “Batal” |

*Source code* pada *button* Simpan:

private void SimpanActionPerformed(java.awt.event.ActionEvent evt) {

// TODO add your handling code here: int Berhasil = 0,Kode;

Barang brg;

brg= new Barang(); brg.Kode(Txt\_KdBarang.getText()); brg.NmBarang(Txt\_NmBarang.getText()); brg.Suplayer(Txt\_Suplayer.getText());

if(Cmb\_Kategori.getSelectedIndex()==0){ JOptionPane.showMessageDialog(null, "Kategori

Belum Anda Masuukan");

}else if(Cmb\_Kategori.getSelectedIndex()==1){ Berhasil = 1;

brg.Kategori("OLI");

}else if(Cmb\_Kategori.getSelectedIndex()==2){ Berhasil = 1;

brg.Kategori("BAN");

}else if(Cmb\_Kategori.getSelectedIndex()==3){ Berhasil = 1;

brg.Kategori("LAMPU");

}

brg.ForMotor(txt\_ForMotor.getText());}

### Tampilan Hasil:

Gambar 5.23 Hasil Tampila GUI\_Barang(*Polimorfisme*) Menerapkan *Polimorfisme* pada GUI\_BarangMasuk

*Source code* pada *Sub Class* BarangMasuk:

public class BarangMasuk extends Barang{ protected int harga\_jual,harga\_suplayer; private int jumlah,inputUser,cekSatuan; private String tanggal;

@Override

public String Kode(String kd){ this.kdBarang = kd; return kd;

}

@Override

public String NmBarang(String NmBarang){ this.NmBarang = NmBarang;

return NmBarang;

}

@Override

public String ForMotor(String ForMotor){ this.ForMotor = ForMotor;

return ForMotor;

}

public BarangMasuk(){ harga\_suplayer = 45000;

jumlah = 200;

}

//getter hasil jml barang public int getJumlah() {

return jumlah;

}

// setter dan getter input barang public int getInputUser() {

return inputUser;}

public void setInputUser(int cekSatuan,int inputUser) { this.inputUser = inputUser;

this.cekSatuan = cekSatuan; CekSatuan();

}

public int CekSatuan() { if(cekSatuan == 1){

jumlah = inputUser;

}else if(cekSatuan == 2){ jumlah = inputUser \* 12;

}else if(cekSatuan == 3){ jumlah = inputUser \* 24;

}

return jumlah;

}

protected int HargaJual(int harga){ this.harga\_jual = harga; return harga;

}

protected int HargaSuplayer(int hargaspy){ this.harga\_suplayer = hargaspy; return hargaspy;

}

public String getTanggal() { return tanggal;

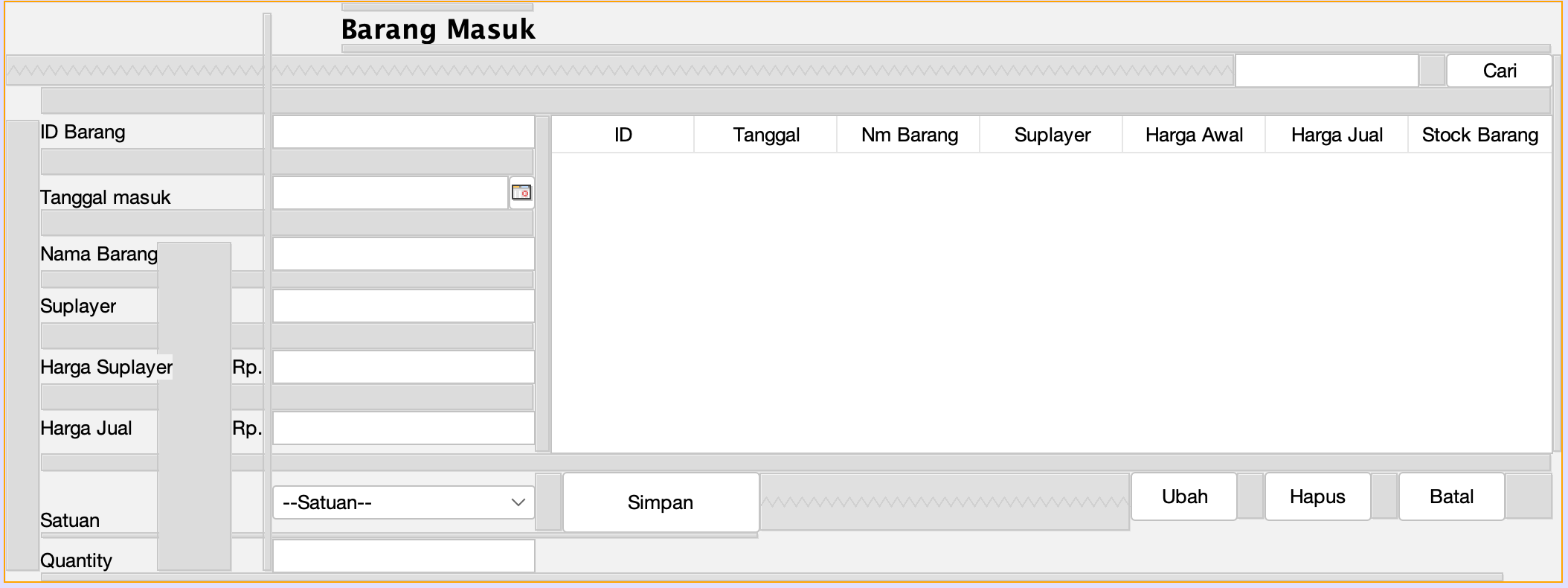
}

public void setTanggal(String tanggal) { this.tanggal = tanggal;

}

}

### Desain form GUI\_BarangMasuk:



Gambar 5.24 Desain GUI\_BarangMasuk(*Polimorfisme*) Tabel 5.11 Property GUI\_BarangMasuk(*polimorfisme*)

|  |  |  |  |
| --- | --- | --- | --- |
| **No** | **Objek** | **Properti** | **Nilai** |
| 1 | jLabel1 | Text | “BARANG MASUK” |
| 2 | jLabel2 | Text | “ID Barang” |
| 3 | jLabel3 | Text | “Nama Barang” |
| 4 | jLabel4 | Text | “Harga Suplayer” |
| 5 | jLabel5 | Text | “Suplayer” |
| 6 | jLabel6 | Text | “Harga Suplayer” |

|  |  |
| --- | --- |
| Nama Aslab Pengajar : Silva Anjelina Tambunan | TTD : |
| Tanggal : |

|  |  |  |  |
| --- | --- | --- | --- |
| 7 | jLabel7 | Text | “Harga Jual” |
| 8 | jLabel8 | Text | “Satuan” |
| 9 | jLabel9 | Text | “Quantity” |
| 10 | jCalender | Text | “ ” |
| Name | txt\_tanggal |
| 11 | JtextField1 | Text | “ ” |
| Name | ID\_Barang |
| 12 | JtextField2 | Text | “ ” |
| Name | txt\_NmBarang |
| 13 | JtextField3 | Text | “ ” |
| Name | txt\_Suplayer |
| 14 | JtextField4 | Text | “ ” |
| Name | txt\_HargaSuplayer |
| 15 | JtextField5 | Text | “ ” |
| Name | txt\_HargaJual |
| 16 | JtextField6 | Text | “ ” |
| Name | txt\_Jumlah |
| 17 | JtextField7 | Text | “ ” |
| Name | txt\_Search |
| 18 | JComboBox | Model | --Satuan--, Unit, Kardus,  kodi |
| Name | cmb\_satuan |
| 19 | JButton1 | Text | “ Simpan” |
| Name | btn\_Simpan |
| 20 | JButton3 | Text | “ Ubah“ |
| Name | btn\_ubah |
| 21 | JButton4 | Text | “Hapus” |
| Name | btn\_Hapus |
| 22 | JButton | Text | “”Cari |
| Name | btn\_Search |
| 23 | JTabel | Title | ID,Tanggal,Nm Barang, Suplayer, Harga Awal,  Harga Jual, Stock Barang |
| Name | tabel\_BarangMasuk |

*Source code* pada *button* Simpan:

private void BarangMasukActionPerformed(java.awt.event.ActionEvent evt)

{

// TODO add your handling code here: int berhasil=0,cek;

//implementasi Polimor Upcasting Barang brg;

brg = new BarangMasuk();

//penerapan polimor downcasting BarangMasuk brgmsk = (BarangMasuk) brg; brgmsk.Kode(ID\_Barang.getText()); String tampilan = "yyyy-MM-dd";

SimpleDateFormat format = new SimpleDateFormat(tampilan);

brgmsk.setTanggal(String.valueOf(format.format(txt\_tanggal. getDate())));

brg.NmBarang(txt\_NmBarang.getText()); brg.Suplayer(txt\_Suplayer.getText());

brgmsk.HargaSuplayer(Integer.parseInt(txt\_HargaSuplayer.get Text()));

brgmsk.HargaJual(Integer.parseInt(txt\_HargaJual.getText()))

;

int jum = Integer.parseInt(txt\_Jumlah.getText()); int pil =1 ;

if(cmb\_satuan.getSelectedIndex() == 0){ JOptionPane.showMessageDialog(null, "Masukkan

Satuan Barang");

}else if(cmb\_satuan.getSelectedIndex() == 1){ brgmsk.setInputUser(1, jum);

// memo.append("Jumlah Barang : "+Integer.toString(brgmsk.getJumlah()));

}else if(cmb\_satuan.getSelectedIndex() == 2){ brgmsk.setInputUser(2, jum);

// memo.append("Jumlah Barang : "+Integer.toString(brgmsk.getJumlah()));

}else if(cmb\_satuan.getSelectedIndex() == 3){ brgmsk.setInputUser(3, jum);

// memo.append("Jumlah Barang : "+Integer.toString(brgmsk.getJumlah()));

}

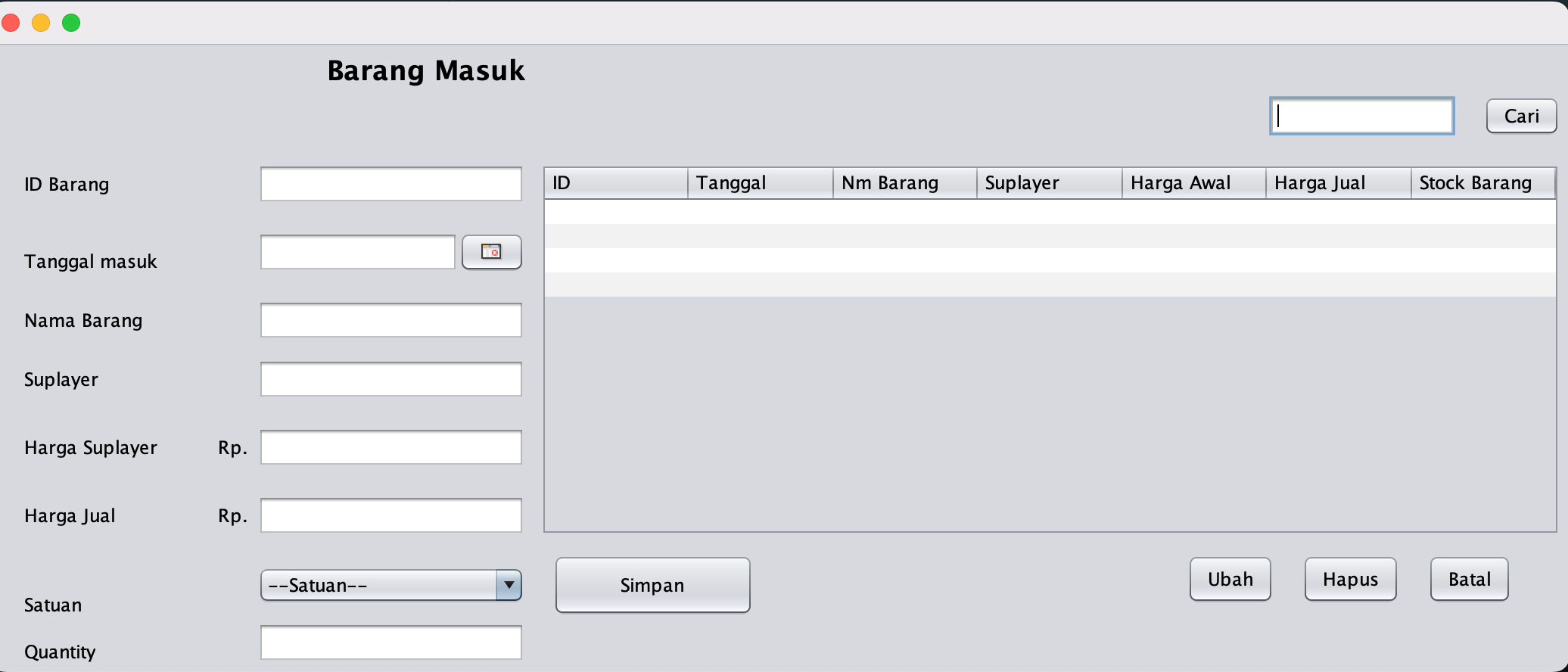
if(berhasil == 1){

JOptionPane.showMessageDialog(null, "Barang berhasil dimasukkan");

}

}

### Tampilan Hasil:



### Gambar 5.25 Hasil Tampilan GUI\_BarangMasuk Menerapkan *Polimorfisme* pada GUI\_BarangKeluar

*Source code* pada *Sub Class* Barang\_keluar:

public class BarangKeluar extends BarangMasuk{ private int TotHargaBarang,inpurUser,cekSatuan; int jum\_BarangKeluar,UpdateStock;

private String tanggalKeluar;

public String getTanggalKeluar() { return tanggalKeluar;

}

public void setTanggalKeluar(String tanggalKeluar) { this.tanggalKeluar = tanggalKeluar;

}

@Override

public String Kode(String kd){ this.kdBarang = kd; return kd;

}

@Override

public String NmBarang(String NmBarang){ this.NmBarang = NmBarang;

return NmBarang;

}

@Override

public String ForMotor(String ForMotor){ this.ForMotor = ForMotor;

return ForMotor;

}

public int getInpurUser() { return inpurUser;

}

public int getCekSatuan() { return cekSatuan;

}

public void setInpurUser(int inpurUser,int cekSatuan) { this.inpurUser = inpurUser;

this.cekSatuan = cekSatuan;

}

protected int UpdateStock(int pilih ,int jumlah){ if(pilih == 1){

UpdateStock= getJumlah() - jumlah;

}else if(pilih == 2){

UpdateStock = getJumlah() - (jumlah \* 12);

}else if(pilih == 3){

UpdateStock = getJumlah()- (jumlah \* 24);

}

jumlah();

return UpdateStock;

}

protected int jumlah(){ return UpdateStock;

}

protected int jumlah(int pilih , int jumlah){ if(pilih == 1){

jum\_BarangKeluar= jumlah;

}else if(pilih == 2){ jum\_BarangKeluar = jumlah \* 12;

}else if(pilih == 3){ jum\_BarangKeluar = jumlah \* 24;

}

return jum\_BarangKeluar;

}

protected int jumlah(int pilih,int harga,int jumlah\_beli){

if(pilih == 1){

TotHargaBarang = harga \* jumlah\_beli;

}else if(pilih == 2){

TotHargaBarang = (jumlah\_beli \* 12) \* harga;

}else if(pilih == 3){

TotHargaBarang = (jumlah\_beli \* 24) \* harga;

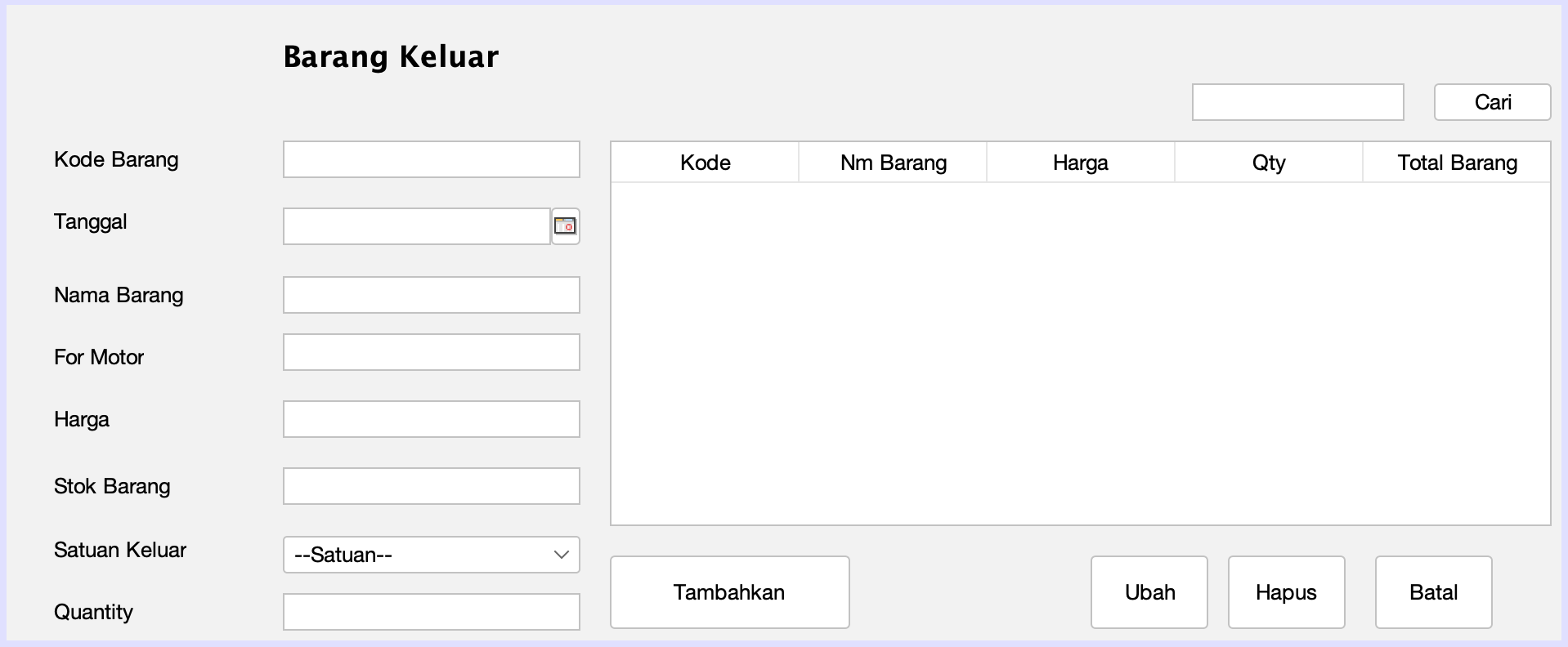
}

return TotHargaBarang;

}

}

### Desain form GUI\_Barangkeluar:



### Gambar 5.26 Desain GUI\_BarangKeluar(*Polimorfisme*) Tabel 5.12 Property Desain GUI\_BarangKeluar

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **No** | **Objek** | | **Properti** | **Nilai** | |
| 1 | jLabel1 | | Text | “BARANG KELUAR” | |
| 2 | jLabel2 | | Text | “Kode Barang” | |
| 3 | jLabel3 | | Text | “Tanggal” | |
| 4 | jLabel4 | | Text | “Nama Barang” | |
| 5 | jLabel5 | | Text | “For Motor” | |
| 6 | jLabel6 | | Text | “Harga” | |
| 7 | jLabel7 | | Text | “Stock Barang” | |
| 8 | jLabel8 | | Text | “Satua Barang” | |
| 9 | jLabel9 | | Text | “Quantity” | |
| 10 | jCalender | | Text | “ ” | |
| Name | txt\_Tanggal | |
| 11 | JtextField1 | | Text | “ ” | |
| Name | ID\_Barang | |
| 12 | JtextField2 | | Text | “ ” | |
| Name | txt\_NmBarang | |
| 13 | JtextField3 | | Text | “ ” | |
| Name | txt\_ForMotor | |
| 14 | JtextField4 | | Text | “ ” | |
| Name | txt\_Harga | |
| 15 | JtextField5 | | Text | “ ” | |
| Name | txt\_StokBarang | |
| 16 | JtextField6 | | Text | “ ” | |
|  | | Nama Aslab Pengajar : Silva Anjelina Tambunan | | | TTD : |
| Tanggal : | | |

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | Name | txt\_Jml |
| 17 | JtextField7 | Text | “ ” |
| Name | txt\_Search |
| 18 | JComboBox | Model | --Satuan--, Unit,Kardus, kodi |
| Name | cmb\_Satuan |
| 19 | JButton | Text | “ Tambahkan” |
| Name | btn\_Tambahkan |
| 20 | JButton | Text | “ Cek” |
| Name | btn\_cek |
| 21 | JButton | Text | “ Ubah“ |
| Name | btn\_ubah |
| 22 | JButton | Text | “Hapus” |
| Name | btn\_Hapus |
| 23 | JButton | Text | “”Cari |
| Name | btn\_Search |
| 24 | JTabel | Title | Kode,Nm Barang, Harga,  QTY, Total Barang Barang |
| Name | tabel\_BarangKeluar |

*Source code* pada *button* Tambahkan:

private void prosesActionPerformed(java.awt.event.ActionEvent evt) {

// TODO add your handling code here:

// penerapan polimorfisme downcasting Barang brg;

brg = new BarangKeluar();

BarangKeluar brgklr = (BarangKeluar) brg; brgklr.Kode(ID\_Barang.getText());

String tampilan = "yyyy-MM-dd"; SimpleDateFormat format = new

SimpleDateFormat(tampilan);

brgklr.setTanggalKeluar(String.valueOf(format.format(txt\_Ta nggal.getDate())));

brg.NmBarang(txt\_NmBarang.getText()); brg.ForMotor(txt\_ForMotor.getText()); brg = new BarangMasuk();

BarangMasuk brgmsk = (BarangMasuk) brg;

brgmsk.HargaJual(Integer.parseInt(txt\_Harga.getText())); int harga = Integer.parseInt(txt\_Harga.getText()); int stok =

Integer.parseInt(txt\_StokBarang.getText());

int jumlah = Integer.parseInt(txt\_Jml.getText()); if(cmb\_Satuan.getSelectedIndex()==1){

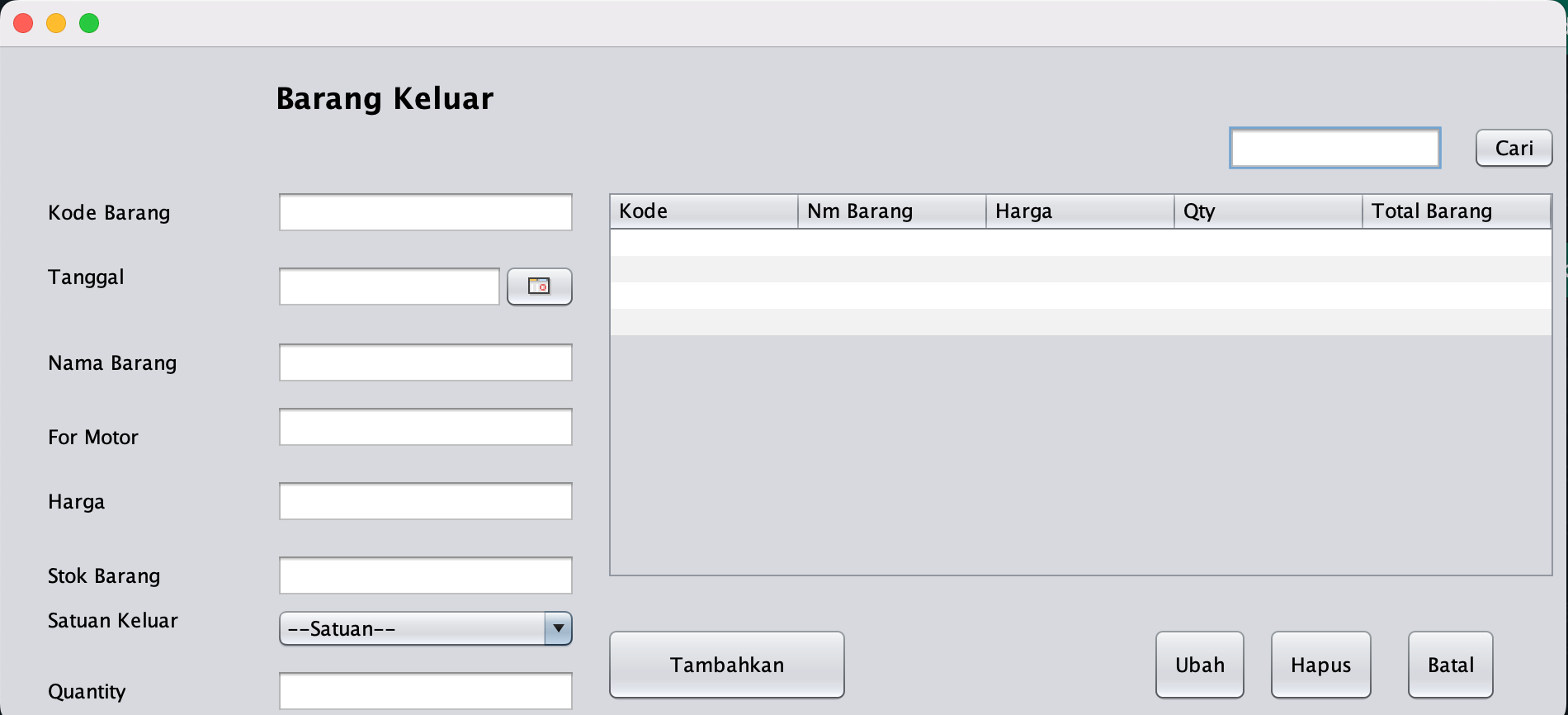
JOptionPane.showMessageDialog(null, "Masukkan Satuan Barang");

}else if(cmb\_Satuan.getSelectedIndex()==1){ brgklr.UpdateStock(1, jumlah);

}else if(cmb\_Satuan.getSelectedIndex()==2){ brgklr.UpdateStock(2, jumlah);

}else if(cmb\_Satuan.getSelectedIndex()==3){ brgklr.UpdateStock(3, jumlah);} }

### Tampilan Hasil:



### Gambar 5.27 Hasil Tampilan GUI\_BarangKeluar

### Analisa :

### Pada program diatas polimorfisme diterapkan pada gui, polimorfisme disini digunakan untuk memanggil isi dan method dari class dengan bermacam-macam bentuk. Pada polimorfisme dikenal dengan upcasting dan downcasting pada program diatas terdapat beberapa penerapan yang mana digunakan untuk mengkonversidari sub class atau class turunan ke tipe super class, dan terdapat downcasting untuk mengkonversi tipe super class ke tipe turunan.

## Kesimpulan

1. *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*).
2. *Polymorphism* sering dikaitkan dengan penggunaan lebih dari satu *method* dengan nama sama. Penggunaan *method* dengan nama sama dapat diterapkan dengan *method overloading* dan *method overriding*. Peran *polymorphism* sebenarnya tidak terbatas hanya pada hal tersebut. Ada keterkaitan antara *polymorphism* dan *inheritance* (turunan).

### 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. Secara umum, *Exception Handling* dapat dilakukan menggunakan *keyword try-catch*

**BAB VI**

**INTERFACE DAN PENGENALAN DATABASE**

|  |  |  |
| --- | --- | --- |
| **Jumlah Pertemuan** | **:** | **2 x 60 menit** |
| **Tujuan Praktikum** | **:** | 1. Praktikan mampu memahami tentang *exception*   dan menerapkan *exception* ke dalam program.   1. Praktikan mampu membuat *database* dengan menggunakan MySQL. 2. Praktikan mampu mengkoneksikan *database*   dengan program yang sudah dibuat sebelumnya.   1. 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. Pengertian dari *Interface*

*Interface* merupakan *prototype* atau template untuk sebuah class, deskripsi ini hampir serupa dengan *class* abstrak. *Interface* merupakan *class* memiliki struktur yang istimewa. Ketika menulis *method* di *interface* maka badan method kosong (*abstract method*), *class–class* yang mengimplementasikan interface wajib meng-*Overide method* untuk mendefinisikan implementasi metode – metode ini.

* + 1. Deklarasii *Interface*

### *Interface* secara struktur serupa dengan class. Isi dari interface adalah method abstract, artinya method hanya dideklarasikan dan tanpa badan method. Deklarasi method pada interface serupa dengan method pada class *abstract*

### *Variable* pada *interface* akan memiliki aturan static dan final sedangkan method akan memiliki aturan *public* dan *abstract*. Hal ini terjadi setelah proses kompilasi seperti yang terlihat pada gambar.

|  |  |
| --- | --- |
| Nama Aslab Pengajar :  Silva Anjelina Tambunan | TTD : |
| Tanggal : |

### *Sintax* 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.

* + - 1. Variable selalu bersifat *public, final* dan *static.*

### Variabel atau *attribute* yang bersifat final adalah *variable* yang tidak bisa dirubah nilainya pada kelas yang mengimplementasi *interface* tersebut.

* + - * 1. *Static* adalah pemanggilan *variable* atau *attribute* tanpa perlu dibuat objek terlebih dahulu.
      1. Bentuk Deklarasi *Interface*:

hak\_akses interface nama\_interface{

static final type\_data nama\_variabel = nilai; static final type\_data nama\_variabel = nilai;

abstract type\_method nama\_method(parameter); abstract type\_method nama\_method(parameter);

}

* + 1. Struktur *Extends* dan *Implements* pada java :

### Jika sebuah *class* tidak diperkenankan melakukan pewarisan (*extends*) lebih dari 1 *class*, maka *interface* diperbolehkan melakukan hal tersebut. Namun sebuah *class* diperbolehkan melakukan implementasi terhadap lebih dari 1 *interface*. Implements bertujuan untuk menyamakan struktur dari *class* yang mengimplementasi *interface*, sedangkan *exteds* bertujuan untuk memperluas isi sebuah *class*.

### Sebuah class dapat implements 2 interface secara langsung

public class nama\_class implements interface\_1, interface\_2

### Gambar 6.1 Class diagram implements interface langsung

### Sebuah Interface dapat extends dari interface lain :

public interface nama\_interface extends interface\_1



### Gambar 6.2 Interface extends interface lain 3.Sebuah Class dapat extends dari kelas lain :

public class nama\_class extends class\_1



### Gambar 6.3 Class extends class lain

* + 1. Pengenalan *Database*
       1. Basis Data(*Database*)

### Basis data (*database*) adalah kumpulan data yang disimpan secara sistematis di dalam komputer yang dapat diolah atau dimanipulasi menggunakan perangkat lunak (program aplikasi) untuk menghasilkan informasi. Pendefinisian basis data meliputi spesifikasi berupa tipe data, struktur data dan juga batasan-batasan pada data yang akan disimpan.

### XAMPP

### XAMPP adalah perangkat lunak bebas, yang mendukung banyak sistem operasi, merupakan kompilasi dari beberapa program. 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. Nama XAMPP merupakan singkatan dari X (empat sistem operasi apapun), *Apache*, MySQL, PHP dan Perl. Program ini tersedia dalam GNU General Public License dan

### bebas, merupakan *web server* yang mudah digunakan yang dapat melayani tampilan halaman *web* yang dinamis.

Dua (2) *Software* yang digunakan dalam XAMPP:

### phpMyAdmin(MySQL)

### Adalah perangkat lunak bebas yang ditulis dalam bahasa pemrograman *PHP* yang digunakan untuk menangani administrasi *MySQL* melalui *website* Jejaring Jagat Jembar. p*hpMyAdmin* mendukung berbagai operasi *MySQL,* diantaranya.

### Server HTTP Apache atau Server Web/WWW Apache Adalah *server web* yang dapat dijalankan di banyak sistem

### operasi yang berguna untuk melayani dan memfungsikan situs *web.* Protokol yang digunakan untuk melayani fasilitas *web*/*www* ini menggunakan *HTTP.*

## VI.2. Langkah – Langkah Praktikum

1. Buka Aplikasi *Netbeans*.
2. Buat *Class*.

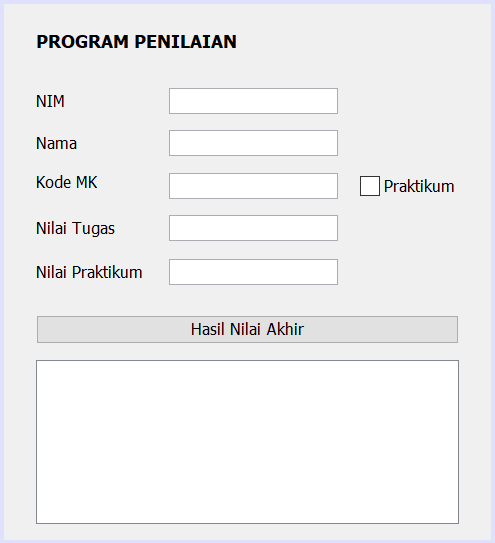
### Memberi *script* pada kelas tersebut.

1. Buat *form*.
2. Memberi *script* pada *form* tersebut.

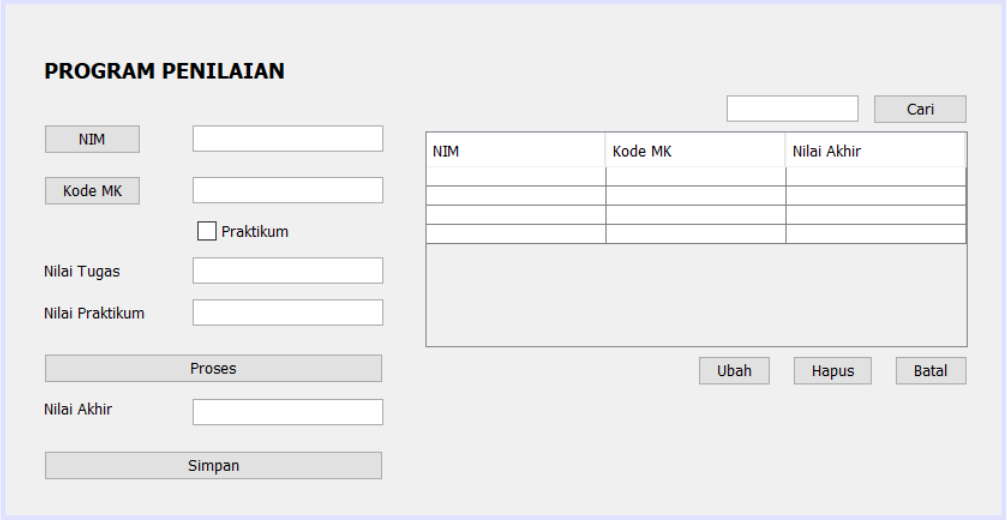
### Menjalankan program.

## VI.3. Tugas Praktikum ke-1:

### Mendesain ulang GUI\_Nilai + membuat method Batal() Desain lama *form* (GUI\_Nilai.java) :



### Gambar 6.4 Desain Awal GUI\_Nilai Desain baru *form* (GUI\_Nilai.java) :



### Gambar 6.5 Desain Baru GUI\_Nilai Tabel 6.1 Properti Desain GUI\_Nilai(*JTable*)

|  |  |  |  |
| --- | --- | --- | --- |
| **No** | **Objek** | **Properti** | **Nilai** |
| 1 | jLabel1 | Text | PROGRAM PENILAIAN |
| 2 | jLabel4 | Text | Nilai Tugas |
| 3 | jLabel5 | Text | Nilai Praktikum |
| 4 | JLabel6 | Text | Nilai Akhir |
| 5 | jTextField1 | Name | txtNIM |
| Text |  |
| 6 | jTextField2 | Name | txtKodeMK |
| Text |  |

|  |  |  |  |
| --- | --- | --- | --- |
| 7 | jTextField3 | Name | txtNT |
| Text |  |
| 8 | jTextField4 | Name | txtNP |
| Text |  |
| 9 | jTextField5 | Name | txtNA |
| Text |  |
| 10 | jTextField6 | Name | txtNA |
| Text |  |
| 11 | jButton1 | Name | btnNA |
| Text | Proses |
| 12 | JCheckBox | Text | Praktikum |
| Name | cek\_prak |
| 13 | JButton2 | Name | btnSimpan |
| Text | Simpan |
| 14 | JButton3 | Name | btnUbah |
| Text | Ubah |
| 15 | JButton4 | Name | btnHapus |
| Text | Hapus |
| 16 | JButton5 | Name | btnBatal |
| Text | Batal |
| 17 | JButton6 | Name | btnCari |
| Text | Cari |
| 18 | JButton7 | Name | btnNIM |
| Text | NIM |
| 19 | JButton8 | Name | btnNIM |
| Text | NIM |
| 20 | JTable | Name | tabel\_data |

*Source code* pada *method* batal():

public void batal(){

txtNIM.setText(""); txtKodeMK.setText(""); txtNT.setText("");

txtNP.setText("");

txtNA.setText("");

}

*Source code* pada *button* Batal:

private void btnBatalActionPerformed(java.awt.event.ActionEvent evt) {

// TODO add your handling code here: batal();}

### Tampilan Hasil:

### Gambar 6.6 Hasil Tampilan GUI\_Nilai

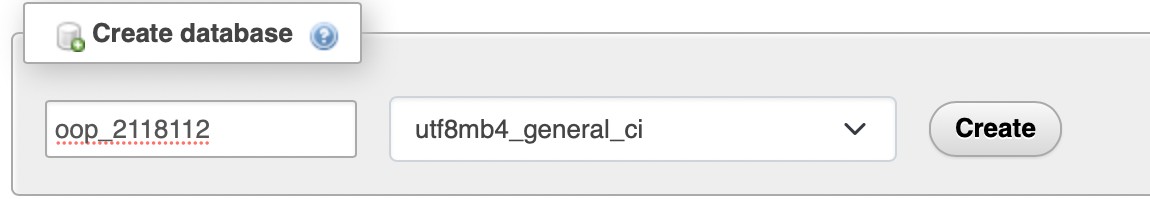
### Analisa :

Pada GUI\_Nilai, pertama – tama diubah dahulu tampilannya sehingga menjadi seperti diatas. Lalu pada *konstruktor* GUI ditambahkan *source code* untuk me*setting* isi dari *textfield* menjadi kosong. Lalu pada *button* batal, *method* tersebut dipanggil, agar ketika *user* klik *button* batal maka isi dari setiap *textfield* akan ksosong.

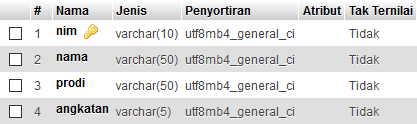
## Tugas Praktikum ke-2:

### Membuat *database* “Penilaian” dan table “tb\_mahasiswa”, meng-koneksikan GUI\_Mahasiswa dengan *database*.

Membuat *database* oop\_21118112



Gambar 6.7 Membuat *database Structure* tb\_mahasiswa pada *database* oop\_2118112



### Gambar 6.8 Membuat tabel (tb\_mahasiswa)

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



### Gambar 6.9 Menambahkan library connector

*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.jdbc.Driver"); conn=

DriverManager.getConnection("jdbc:mysql://localhost/oop\_211 8112?user=root&password=");

}

catch (ClassNotFoundException ex){

Logger.getLogger(GUI\_Mahasiswa.class.getName()).log(Level.S EVERE,null, ex);

}

catch (SQLException e) {

Logger.getLogger(GUI\_Mahasiswa.class.getName()).log(Level.S EVERE,null, e);

}

catch (Exception es) {

Logger.getLogger(GUI\_Mahasiswa.class.getName()).log(Level.S EVERE,null, es);

}

}

*Source code* di *method* tampil*:*

|  |  |  |
| --- | --- | --- |
| public void tampil() {  DefaultTableModel tabelhead DefaultTableModel();  tabelhead.addColumn("NIM"); | | = new |
|  | Nama Aslab Pengajar : Silva Anjelina Tambunan | TTD : |
| Tanggal : |

tabelhead.addColumn("NAMA"); tabelhead.addColumn("PRODI"); tabelhead.addColumn("ANGKATAN"); 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(1),res.getString(2),res.getString(3)

, res.getString(4),});

}

tabel\_data.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 Prodi = txtProdi.getText(); String Ang = txtAng.getText(); try {

koneksi();

Statement statement = conn.createStatement(); statement.executeUpdate("INSERT INTO

tb\_mahasiswa(nim, nama, prodi, angkatan)"

+"VALUES('"+Nim+"','"+Nama+"','"+Prodi+"','"+Ang+"')");

statement.close();

JOptionPane.showMessageDialog(null, Memasukan Data Barang!");

} catch (Exception e) { JOptionPane.showMessageDialog(null,

Kesalahan Input!");

}

refresh();

}

"Berhasil

"Terjadi

#### Source code method update :

|  |  |  |
| --- | --- | --- |
| public void update(){  String Nim = txtNim.getText(); String Nama = txtNama.getText(); String Prodi = txtProdi.getText(); String Ang = txtAng.getText(); String Kodelama = nim1;  try { | | |
|  | Nama Aslab Pengajar : Silva Anjelina Tambunan | TTD : |
| Tanggal : |

Statement statement = conn.createStatement();

statement.executeUpdate("UPDATE SET nim='"+Nim+"'," + "nama='"+Nama+"'"

+

",prodi='"+Prodi+"',angkatan='"+Ang+"'WHERE '"+Kodelama+"'");

statement.close(); conn.close();

tb\_mahasiswa

nim

=

JOptionPane.showMessageDialog(null, "Update Data Mahasiswa!");

} 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\_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");

}

}

#### Source code method cari :

public void cari(){ try{

try (Statement statement = conn.createStatement())

{

String sql="SELECT \* FROM tb\_mahasiswa WHERE

`nim` LIKE '%"+txtCari.getText()+"%'";

ResultSet rs=statement.executeQuery(sql);

//menampilkan data dari sql query

if(rs.next()) // .next() = scanner method

{

txtNim.setText(rs.getString(1)); txtNama.setText(rs.getString(2)); txtProdi.setText(rs.getString(3)); txtAng.setText(rs.getString(4));

}

else{

JOptionPane.showMessageDialog(null, "Data yang Anda cari tidak ada");

}

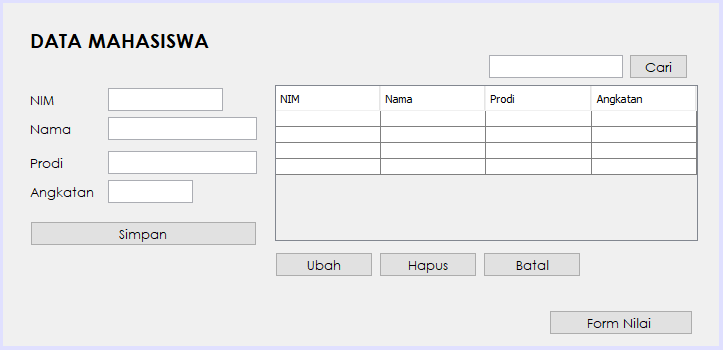
}

}catch (Exception ex){ System.out.println("Error."+ex);

}

}

### Desain *form* GUI Mahasiswa:



### Gambar 6.10 Desain *Form* GUI\_Mahasiswa Tabel 6.2 Properti Desain GUI\_Mahasiswa (*Database*)

|  |  |  |  |
| --- | --- | --- | --- |
| **No** | **Objek** | **Properti** | **Nilai** |
| 1 | jLabel1 | Text | DATA MAHASISWA |
| 2 | jLabel2 | Text | NIM |
| 3 | jLabel3 | Text | Nama |
| 4 | jLabel4 | Text | Prodi |
| 5 | jLabel5 | Text | Angkatan |
| 6 | jTextField1 | Name | txtNim |
| Text |  |
| 7 | jTextField2 | Name | txtNama |
| Text |  |
| 8 | jTextField3 | Name | txtProdi |
| Text |  |
| 9 | jTextField4 | Name | txtAng |
| 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 | jButton5 | Name | btnNilai |
| Text | Form Nilai |
| 17 | JTable | Name | tabel\_data |

*Source code* pada tombol tambah*:*

private

void

btnSimpanActionPerformed(java.awt.event.ActionEvent evt) {

// TODO add your handling code here: insert();

}

*Source code* pada tombol Ubah*:*

private void btnUbahActionPerformed(java.awt.event.ActionEvent evt) {

// TODO add your handling code here: update();

}

*Source code* pada tombol hapus*:*

private void btnUbahActionPerformed(java.awt.event.ActionEvent evt) {

// TODO add your handling code here: update();

}

*Source code* pada tombol cari*:*

private

void

btnCariActionPerformed(java.awt.event.ActionEvent evt) {

// TODO add your handling code here: cari();

}

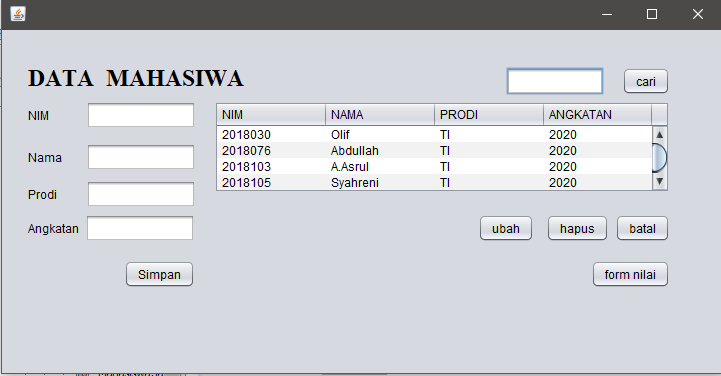
*Source code* pada tombol batal*:*

private void btnBatalActionPerformed(java.awt.event.ActionEvent evt) {

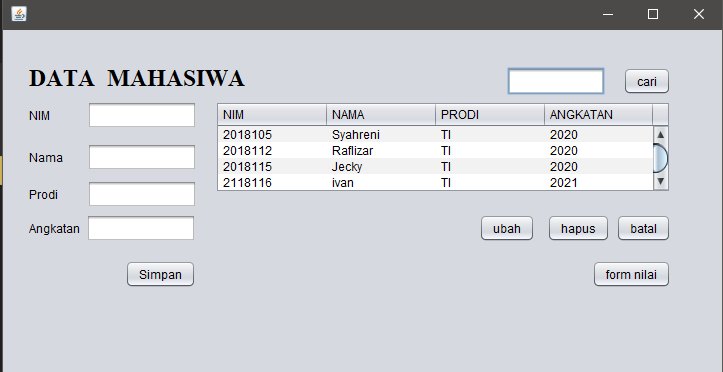
// TODO add your handling code here: batal();

}

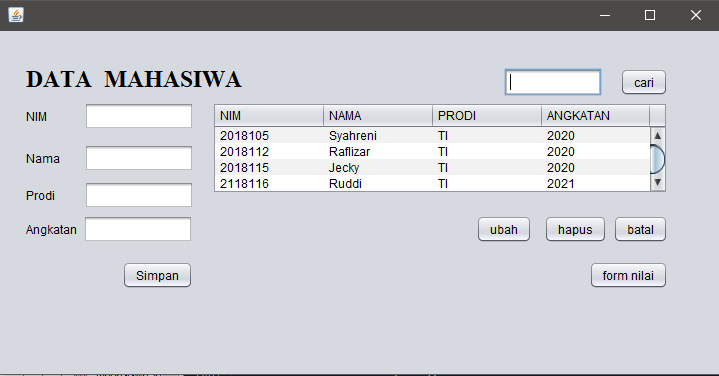
### Tampilan Hasil:



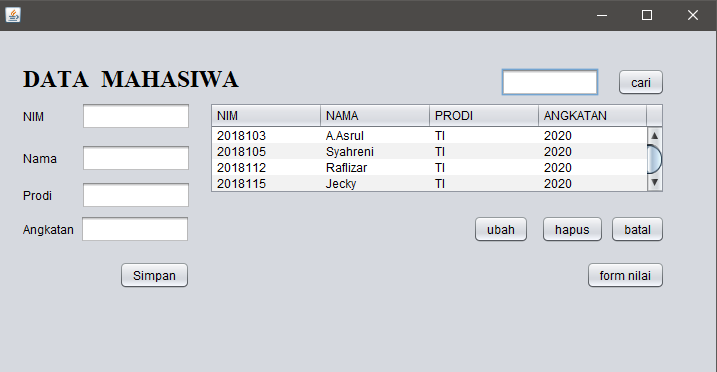
### Gambar 6.11 Tampilan hasil *method* tampil() GUI\_Mahasiswa



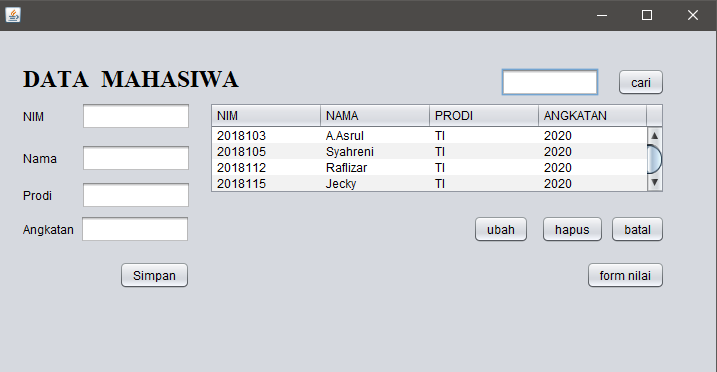
Gambar 6.12 Tampilan hasil *method insert*() GUI\_Mahasiswa



Gambar 6.13 Tampilan hasil *method update*() GUI\_Mahasiswa



Gambar 6.14 Tampilan hasil *metod delete*() GUI\_Mahasiswa



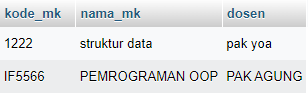
### Gambar 6.15 Tampilan hasil method batal() GUI\_Mahasiwa

### Analisa:

### Pada GUI\_Mahasiswa kali ini telah terubung dengan *database* oop\_2118112, yang berisi tabel tb\_mahasiwa. Data-data yang telah diinputkan ke dalam GUI\_Mahasiswa akan disimpan di dalam *database* tersebut. Lalu pada GUI\_Mahasiswa data-data yang telah terseimpan akan ditampilkan pada tabel yang telah dibuat. Pada GUI\_Mahasiswa telah diberi beberapa *button* untuk memodifikasi isi data.

## Tugas Praktikum ke-3:

### Mengkoneksikan GUI\_Matkul Kedalam *Database*



### Gambar 6.16 Membuat tabel (tb\_matkul)

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



### Gambar 6.17 Menambahkan library connector

*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* m*ethod* koneksi *:*

public void koneksi() throws SQLException { try {

conn=null; Class.forName("com.mysql.jdbc.Driver"); conn=

DriverManager.getConnection("jdbc:mysql://localhost/oop\_191 8111?user=root&password=");

}

catch (ClassNotFoundException ex){

Logger.getLogger(GUI\_Matkul.class.getName()).log(Level.SEVE RE,null, ex);

}

catch (SQLException e) {

Logger.getLogger(GUI\_Matkul.class.getName()).log(Level.SEVE RE,null, e);

}

catch (Exception es) {

Logger.getLogger(GUI\_Matkul.class.getName()).log(Level.SEVE RE,null, es);

}

}

*Source code* di *method* tampil*:*

public void tampil() {

DefaultTableModel DefaultTableModel();

tabelhead

=

new

tabelhead.addColumn("KODE MK"); tabelhead.addColumn("NAMA MK"); tabelhead.addColumn("DOSEN"); 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(1),res.getString(2),res.getString(3)

,});

}

tabel\_data.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 = txtKodeMK.getText(); String MK = txtMK.getText(); String Dosen = txtDosen.getText(); try {

koneksi();

Statement statement = conn.createStatement(); statement.executeUpdate("INSERT INTO

tb\_matkul(kode\_mk, nama\_mk, dosen)"

+"VALUES('"+Kode+"','"+MK+"','"+Dosen+"')");

statement.close(); JOptionPane.showMessageDialog(null,

Memasukan Data Matakuliah!");

} catch (Exception e) { JOptionPane.showMessageDialog(null,

Kesalahan Input!");

}

refresh();

}

"Berhasil

"Terjadi

#### Source code method update :

public void update(){

String Kode = txtKodeMK.getText(); String MK = txtMK.getText(); String Dosen = txtDosen.getText(); try {

Statement statement = conn.createStatement(); statement.executeUpdate("UPDATE tb\_matkul SET

kode\_mk='"+Kode+"'," + "nama\_mk='"+MK+"'"

+ ",dosen='"+Dosen+"'WHERE kode\_mk =

'"+Kode+"'");

statement.close(); conn.close();

JOptionPane.showMessageDialog(null, "Update Data MataKuliah!");

} 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\_matkul WHERE kode\_mk='"+txtKodeMK.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 '%"+txtCari.getText()+"%'";

ResultSet rs=statement.executeQuery(sql);

//menampilkan data dari sql query

if(rs.next()) // .next() = scanner method

{

txtKodeMK.setText(rs.getString(1)); txtMK.setText(rs.getString(2)); txtDosen.setText(rs.getString(3));

}

else{

JOptionPane.showMessageDialog(null, "Data yang Anda cari tidak ada");

}

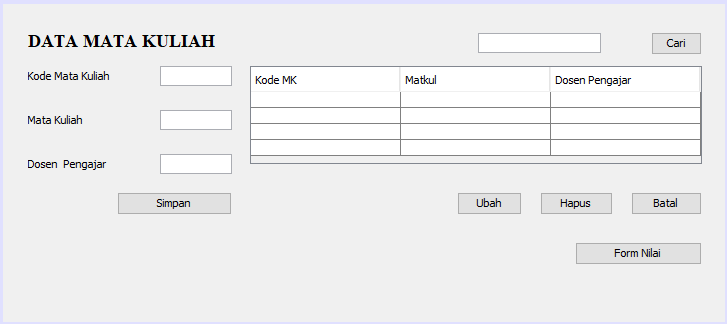
}

}catch (Exception ex){ System.out.println("Error."+ex);

}

}

Desain *form* GUI Matkul:



### Gambar 6.18 Desain *Form* GUI\_Matkul Tabel 6.3 Properti Desain GUI\_ Matkul (*Database*)

|  |  |  |  |
| --- | --- | --- | --- |
| **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 | jTextField1 | Name | txtKodeMK |
| Text |  |
| 6 | jTextField2 | Name | txtMK |
| Text |  |
| 7 | jTextField3 | Name | txtDosen |
| Text |  |
| 8 | jTextField5 | Name | txtCari |
| Text |  |
| 9 | jButton1 | Name | btnSimpan |
| Text | Simpan |
| 10 | jButton2 | Name | btnUbah |
| Text | Ubah |
| 11 | jButton3 | Name | btnHapus |
| Text | Hapus |
| 12 | jButton4 | Name | btnBatal |
| Text | Batal |
| 13 | jButton5 | Name | btnCari |
| Text | Cari |
| 14 | jButton5 | Name | btnNilai |
| Text | Form Nilai |
| 15 | JTable | Name | tabel\_data |

*Source code* pada *tombol* tambah*:*

private

void

btnSimpanActionPerformed(java.awt.event.ActionEvent evt) {

// TODO add your handling code here: insert();

}

*Source code* pada tombol Ubah*:*

private void btnUbahActionPerformed(java.awt.event.ActionEvent evt) {

// TODO add your handling code here: update();

}

*Source code pada* tombol hapus*:*

private

void

btnHapusActionPerformed(java.awt.event.ActionEvent evt) {

// TODO add your handling code here: delete();

}

*Source code pada* tombol cari*:*

private

void

btnCariActionPerformed(java.awt.event.ActionEvent evt) {

// TODO add your handling code here: cari();

}

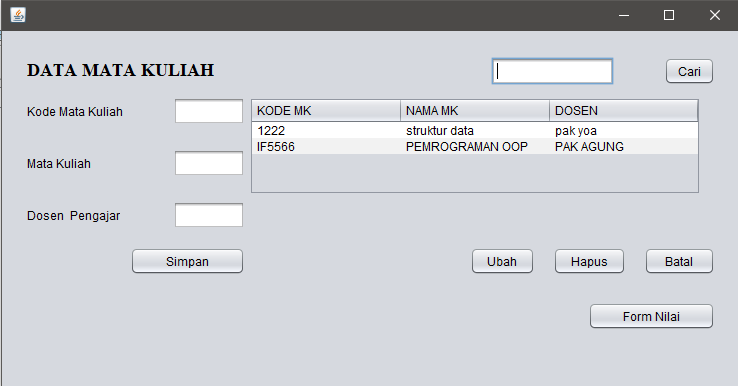
*Source code pada* tombol batal*:*

private void btnBatalActionPerformed(java.awt.event.ActionEvent evt) {

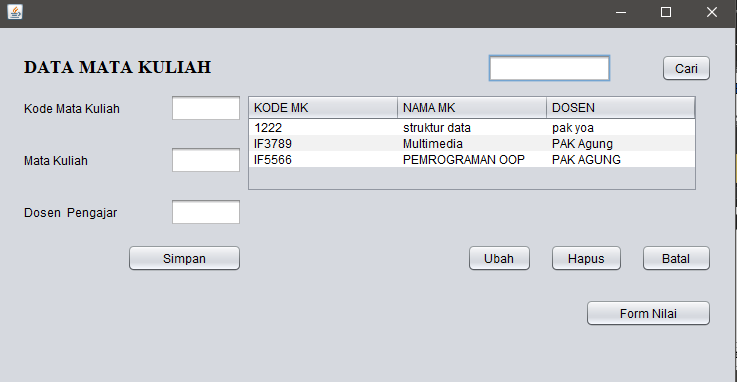
// TODO add your handling code here: batal();

}

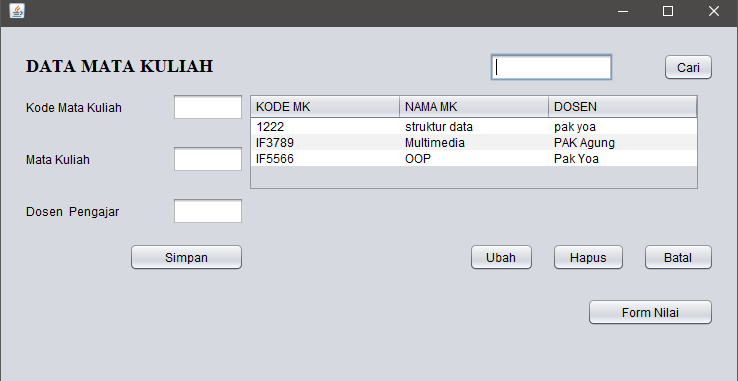
### Tampilan Hasil :



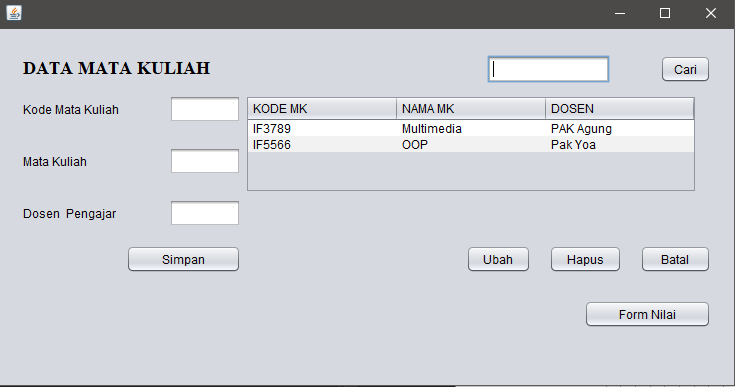
### Gambar 6.19 Tampilan hasil *method* tampil() GUI\_Matkul



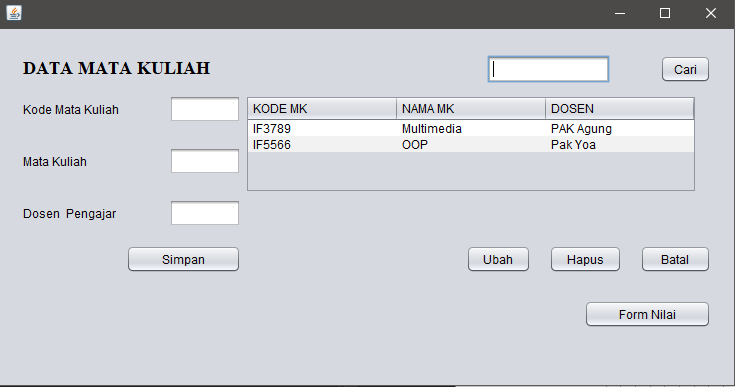
Gambar 6.20 Tampilan hasil *method insert*() GUI\_Matkul



Gambar 6.21 Tampilan hasil *method update*() GUI\_Matkul



Gambar 6.22 Tampilan hasil *method delete*() GUI\_Matkul



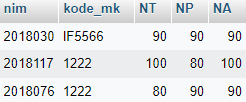
### Gambar 6.23 Tampilan hasil *method* batal() GUI\_Matkul

### Analisa:

### Pada GUI\_Matkul ini telah dihubungkan dengan *database* oop\_2118112. Kini GUI\_Matkul telah dapat menyimpan data-data yang dimasukkan ke GUI\_Matkul ke dalam *database*. Pada GUI\_Matkul terdapat tabel yang berfungsi untuk menampilkan data-data yang ada pada *database* oop\_211812, sehingga pengguna dapat melihat secara langsung data-data yang ada pada *database*.

## Tugas Praktikum ke-4:

### Mengkoneksikan GUI\_Nilai Kedalam *Database*



### Gambar 6.24 Membuat tabel (tb\_nilai)

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



### Gambar 6.25 Menambah library connector

*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.jdbc.Driver"); conn=

DriverManager.getConnection("jdbc:mysql://localhost/oop\_191 8111?user=root&password=");

}catch (ClassNotFoundException ex){

Logger.getLogger(GUI\_Nilai.class.getName()).log(Level.SEVER E,null, ex);

}catch (SQLException e) {

Logger.getLogger(GUI\_Nilai.class.getName()).log(Level.SEVER E,null, e);

}catch (Exception es) {

Logger.getLogger(GUI\_Nilai.class.getName()).log(Level.SEVER E,null, es);

}

}

*Source code* di *method* tampil*:*

public void tampil() {

DefaultTableModel DefaultTableModel();

tabelhead

=

new

tabelhead.addColumn("NIM"); tabelhead.addColumn("Kode MK"); tabelhead.addColumn("NT"); tabelhead.addColumn("NP"); 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(1),res.getString(2),res.getString(3)

,res.getString(4), res.getString(5),});

}

tabel\_data.setModel(tabelhead);

} catch (Exception e) { JOptionPane.showMessageDialog(null, "BELUM

TERKONEKSI");

}

}

#### Source code method refresh :

public void refresh(){

new GUI\_Nilai().setVisible(true); this.setVisible(false);

}

#### Source code method insert :

public void insert(){

String Nim = (String) txtNIM.getSelectedItem(); String KodeMK = (String)

txtKodeMK.getSelectedItem();

String NT = txtNT.getText(); String NP = txtNP.getText(); String NA = txtNA.getText(); try {

koneksi();

Statement statement = conn.createStatement(); statement.executeUpdate("INSERT INTO

tb\_nilai(nim, kode\_mk, n\_tugas, n\_prak, n\_akhir)"

+"VALUES('"+Nim+"','"+KodeMK+"','"+NT+"','"+NP+"','"+NA+"'

)");

statement.close();

JOptionPane.showMessageDialog(null, Memasukan Data Nilai!");

} catch (Exception e) { JOptionPane.showMessageDialog(null,

Kesalahan Input!");

}

refresh();

}

"Berhasil

"Terjadi

#### Source code method update :

|  |  |  |
| --- | --- | --- |
| public void update(){  String Nim = (String) txtNIM.getSelectedItem(); String KodeMK = (String)  txtKodeMK.getSelectedItem();  String NT = txtNT.getText(); String NP = txtNP.getText(); String NA = txtNA.getText();  String nim\_lama = nim1; String kode\_lama = kd\_mk1;  try {  Statement statement = conn.createStatement(); statement.executeUpdate("UPDATE tb\_nilai SET  nim='"+Nim+"'," + "kode\_mk='"+KodeMK+"'"  +",n\_tugas='"+NT+"',n\_prak='"+NP+"',n\_akhir='"+NA+"' WHERE nim ='"+nim\_lama+"' AND kode\_mk='"+kode\_lama+"'");  statement.close(); conn.close();  JOptionPane.showMessageDialog(null, "Update  Data Nilai!"); | | |
|  | Nama Aslab Pengajar : Silva Anjelina Tambunan | TTD : |
| Tanggal : |

} 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='"+txtNIM.getSelectedItem()+"' AND kode\_mk='"+txtKodeMK.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 '%"+txtCari.getText()+"%'";

ResultSet rs = statement.executeQuery(sql);

//menampilkan data dari sql query if(rs.next()) // .next() = scanner method

{

txtNIM.setSelectedItem(rs.getString(1)); txtKodeMK.setSelectedItem(rs.getString(2));

txtNT.setText(rs.getString(3));

txtNP.setText(rs.getString(4)); txtNA.setText(rs.getString(5));

}else{

JOptionPane.showMessageDialog(null, "Data yang Anda cari tidak ada");

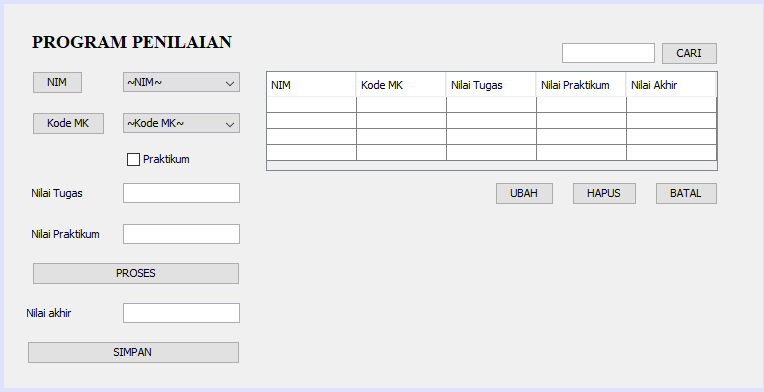
}

}

}catch (Exception ex){ System.out.println("Error."+ex);}}

|  |  |
| --- | --- |
| Nama Aslab Pengajar : Silva Anjelina Tambunan | TTD : |
| Tanggal : |

Desain *form* GUI\_Nilai:



### Gambar 6.26 Desain *Form* GUI\_Nilai Tabel 6.4 Properti Desain GUI\_ Nilai (*Database*)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **No** | **Objek** | | **Properti** | **Nilai** | |
| 1 | jLabel1 | | Text | PROGRAM PENILAIAN | |
| 2 | jLabel4 | | Text | Nilai Tugas | |
| 3 | jLabel5 | | Text | Nilai Praktikum | |
| 4 | JLabel6 | | Text | Nilai Akhir | |
| 5 | jComboBox1 | | Name | txtNIM | |
| Text |  | |
| 6 | jComboBox2 | | Name | txtKodeMK | |
| Text |  | |
| 7 | jTextField3 | | Name | txtNT | |
| Text |  | |
| 8 | jTextField4 | | Name | txtNP | |
| Text |  | |
| 9 | jTextField5 | | Name | txtNA | |
| Text |  | |
| 10 | jTextField6 | | Name | txtNA | |
| Text |  | |
| 11 | jButton1 | | Name | btnNA | |
| Text | Proses | |
| 12 | JCheckBox | | Text | Praktikum | |
| Name | cek\_prak | |
| 13 | JButton2 | | Name | btnSimpan | |
| Text | Simpan | |
| 14 | JButton3 | | Name | btnUbah | |
|  | | Nama Aslab Pengajar : Silva Anjelina Tambunan | | | TTD : |
| Tanggal : | | |

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | Text | Ubah |
| 15 | JButton4 | Name | btnHapus |
| Text | Hapus |
| 16 | JButton5 | Name | btnBatal |
| Text | Batal |
| 17 | JButton6 | Name | btnCari |
| Text | Cari |
| 18 | JButton7 | Name | btnNIM |
| Text | NIM |
| 19 | JButton8 | Name | btnNIM |
| Text | NIM |
| 20 | JTable | Name | tabel\_data |

*Source code* pada tombol tambah*:*

private

void

btnSimpanActionPerformed(java.awt.event.ActionEvent evt) {

// TODO add your handling code here: insert();

}

*Source code* pada tombol Ubah*:*

private void btnUbahActionPerformed(java.awt.event.ActionEvent evt) {

// TODO add your handling code here: update();

}

*Source code* pada tombol hapus*:*

private void btnHapusActionPerformed(java.awt.event.ActionEvent evt) {

// TODO add your handling code here: delete();

}

*Source code* pada tombol cari*:*

private void btnCariActionPerformed(java.awt.event.ActionEvent evt) {

// TODO add your handling code here:

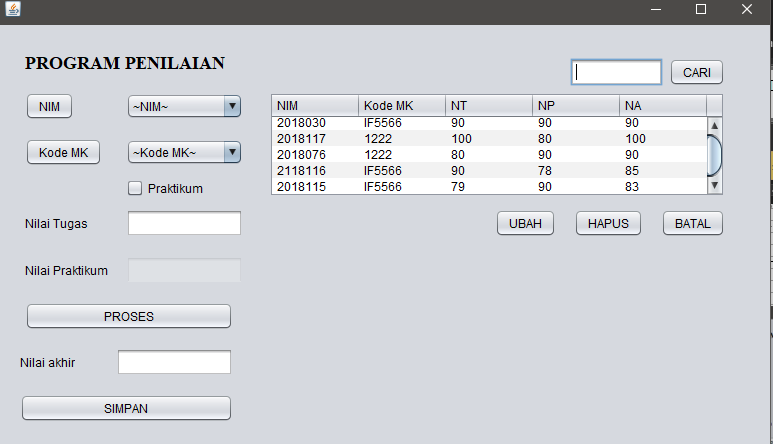
cari();}

*Source code* pada tombol batal*:*

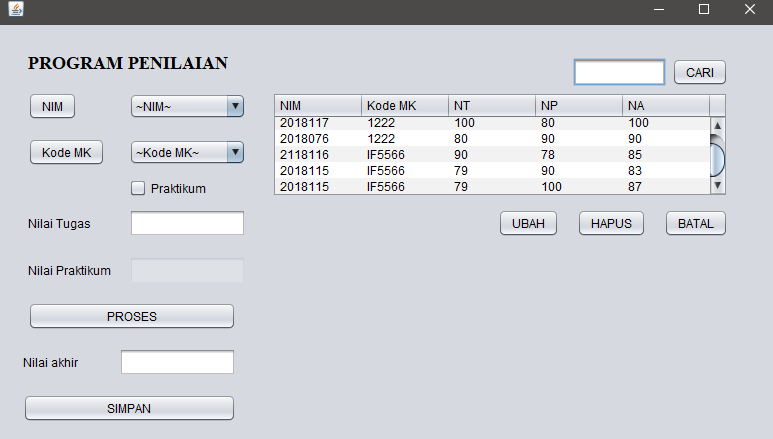
|  |  |  |
| --- | --- | --- |
| private void btnBatalActionPerformed(java.awt.event.ActionEvent evt) {  // TODO add your handling code here: batal();  } | | |
|  | Nama Aslab Pengajar : Silva Anjelina Tambunan | TTD : |
| Tanggal : |

### Tampilan Hasil :

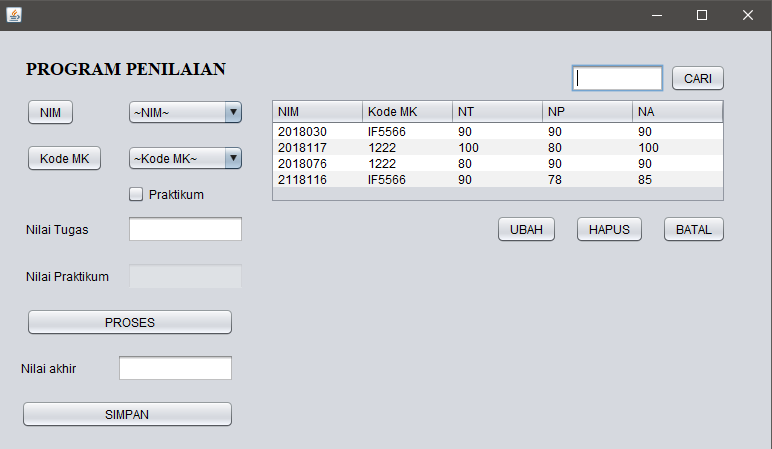
### Gambar 6.27 Tampilan hasil *metod* tampil() GUI\_Nilai



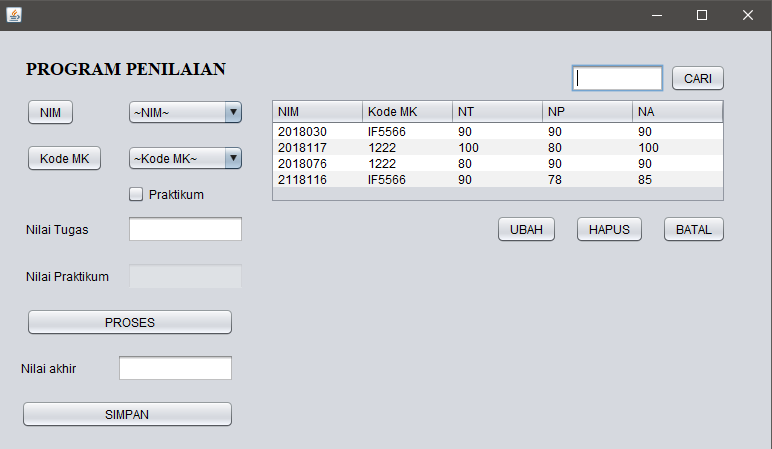
Gambar 6.28 Tampilan hasil *method insert*() GUI\_Nilai



Gambar 6.29 Tampilan hasil *method update*() GUI\_Nilai



Gambar 6.30 Tampilan hasil *method delete*() GUI\_Nilai



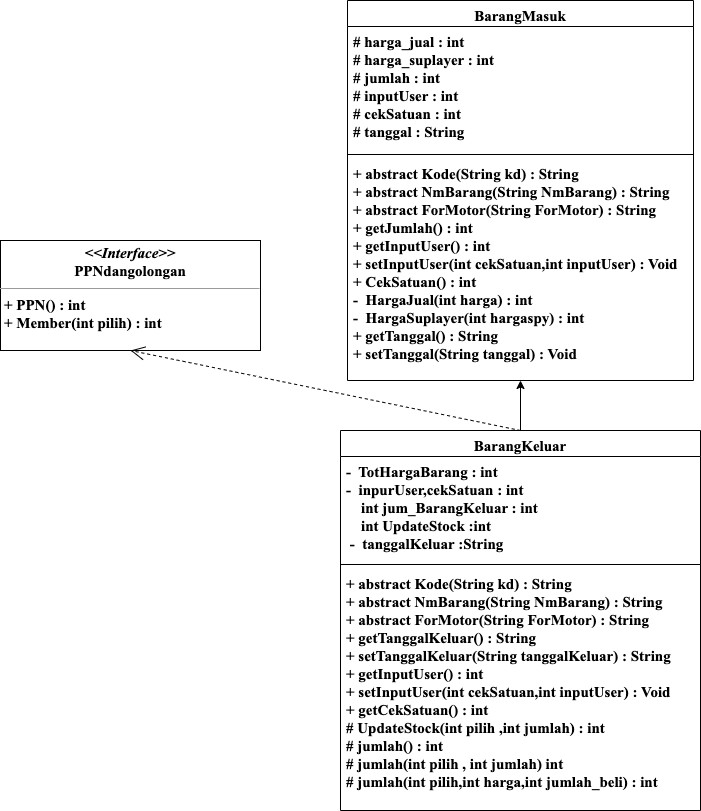
### Gambar 6.31 Tampilan hasil *method* batal() GUI\_Nilai

### Analisa:

### Pada GUI\_Nilai telah terhubung dengan *database* oop\_2118112. Sekarang data-data yang dimasukkan pada GUI\_Nilai akan langsung tersimpan pada *database*. Tabel pada GUI\_Nilai berfungsi untuk menampilkan data secara langusng ke pengguna yang diambil dari data *database*.

## Tugas Rumah ke-1:

Menerapkan *Interface* pada class BarangKeluar.java Diagram *Class*:



### Gambar 6.32 Class Diagram interface pada kelas BarangKeluar

*Source code* pada *Super Class* BarangMasuk :

public class BarangMasuk extends Barang{ protected int harga\_jual,harga\_suplayer; private int jumlah,inputUser,cekSatuan; private String tanggal;

@Override

public String Kode(String kd){ this.kdBarang = kd; return kd;

}

@Override

public String NmBarang(String NmBarang){ this.NmBarang = NmBarang;

return NmBarang;

}

@Override

public String ForMotor(String ForMotor){ this.ForMotor = ForMotor;

return ForMotor;

}

public BarangMasuk(){ harga\_suplayer = 45000;

jumlah = 200;

}

//getter hasil jml barang public int getJumlah() {

return jumlah;

}

// setter dan getter input barang public int getInputUser() {

return inputUser;

}

public void setInputUser(int cekSatuan,int inputUser) { this.inputUser = inputUser;

this.cekSatuan = cekSatuan; CekSatuan();

}

public int CekSatuan() { if(cekSatuan == 1){

jumlah = inputUser;

}else if(cekSatuan == 2){ jumlah = inputUser \* 12;

}else if(cekSatuan == 3){ jumlah = inputUser \* 24;

}

return jumlah;

}

protected int HargaJual(int harga){ this.harga\_jual = harga; return harga;

}

protected int HargaSuplayer(int hargaspy){ this.harga\_suplayer = hargaspy; return hargaspy;

}

public String getTanggal() { return tanggal;

}

public void setTanggal(String tanggal) { this.tanggal = tanggal;

}

}

*Souce code* pada *Sub Class* BarangKeluar :

|  |  |  |
| --- | --- | --- |
| public class BarangKeluar extends BarangMasuk implements PPNdangolongan{  public int TotHargaBarang,inpurUser,cekSatuan; int jum\_BarangKeluar,UpdateStock,pajak,member; private String tanggalKeluar;  public String getTanggalKeluar() { | | |
|  | Nama Aslab Pengajar : Silva Anjelina Tambunan | TTD : |
| Tanggal : |

return tanggalKeluar;

}

public void setTanggalKeluar(String tanggalKeluar) { this.tanggalKeluar = tanggalKeluar;

}

@Override

public String Kode(String kd){ this.kdBarang = kd; return kd;

}

@Override

public String NmBarang(String NmBarang){ this.NmBarang = NmBarang;

return NmBarang;

}

@Override

public String ForMotor(String ForMotor){ this.ForMotor = ForMotor;

return ForMotor;

}

public int getInpurUser() { return inpurUser;

}

public int getCekSatuan() { return cekSatuan;

}

public void setInpurUser(int inpurUser,int cekSatuan) { this.inpurUser = inpurUser;

this.cekSatuan = cekSatuan;

}

protected int UpdateStock(int pilih ,int jumlah){ if(pilih == 1){

UpdateStock= getJumlah() - jumlah;

}else if(pilih == 2){

UpdateStock = getJumlah() - (jumlah \* 12);

}else if(pilih == 3){

UpdateStock = getJumlah()- (jumlah \* 24);

}

jumlah();

return UpdateStock;

}

protected int jumlah(){ return UpdateStock;

}

protected int jumlah(int pilih , int jumlah){ if(pilih == 1){

jum\_BarangKeluar= jumlah;

}else if(pilih == 2){ jum\_BarangKeluar = jumlah \* 12;

}else if(pilih == 3){ jum\_BarangKeluar = jumlah \* 24;

}

return jum\_BarangKeluar;

}

protected int jumlah(int pilih,int harga,int jumlah\_beli){

if(pilih == 1){

TotHargaBarang = harga \* jumlah\_beli;

}else if(pilih == 2){

TotHargaBarang = (jumlah\_beli \* 12) \* harga;

}else if(pilih == 3){

TotHargaBarang = (jumlah\_beli \* 24) \* harga;

}

return TotHargaBarang;

}

@Override

public int PPN() {

return (TotHargaBarang + 2000);

}

@Override

public int Member(int pilih) { if(pilih == 0){

member = (int) (PPN() - (TotHargaBarang \*0.1));

}

return member;

}

}

*Source code* pada *Interface class* PPNdangolongan :

public interface PPNdangolongan { int PPN();

int Member(int pilih);

}

#### Source code pada Driver class :

public class Main {

public static void main(String[] args) { Barang brg;

brg = new BarangKeluar();

BarangKeluar brgklr = (BarangKeluar) brg; brgklr.Kode("KVB08695-35663-6456");

String tampilan = "dd-MM-yyyy"; SimpleDateFormat format = new

SimpleDateFormat(tampilan);

brgklr.setTanggalKeluar(String.valueOf(format.format(01-01- 1970)));

brg.NmBarang("Vanbel Set Beat FI"); brg.ForMotor("Beat FI");

brg = new BarangMasuk();

BarangMasuk brgmsk = (BarangMasuk) brg; brgmsk.HargaJual(145000); brgklr.jumlah(1, 4);

brgklr.jumlah(1, 145000, 4); System.out.println("Kode Barang

:"+brgklr.kdBarang);

System.out.println("Nama Barang

:"+brgklr.NmBarang);

System.out.println("Tanggal Keluar : "+brgklr.getTanggalKeluar());

System.out.println("Harga :" + brgmsk.harga\_jual);

System.out.println("QTY :"

+brgklr.jum\_BarangKeluar);

System.out.println("Total Harga

:"+brgklr.TotHargaBarang);

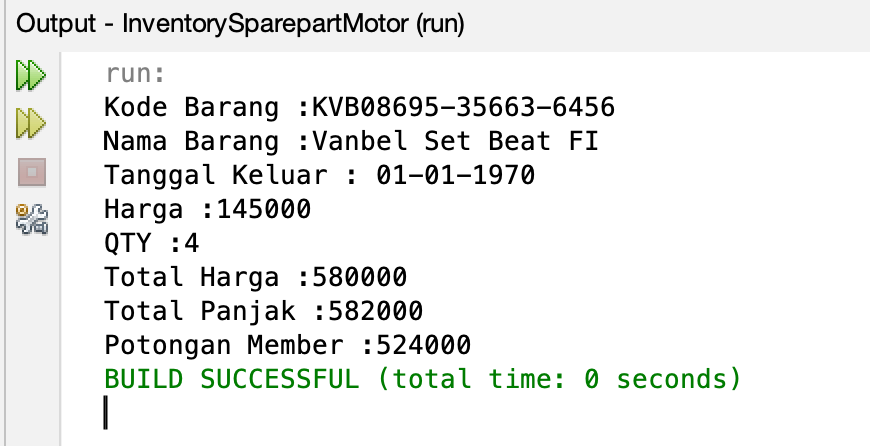
System.out.println("Total Panjak :"+brgklr.PPN()); System.out.println("Potongan Member

:"+brgklr.Member(0));

}

}

### Tampilan Hasil :



### Gambar 6.33 Hasil tampilan Driver Class(*interface*)

### Analisa:

### Pada program diatas interface digunakan untuk menghitung harga pajak dan harga member. Interface pada program berada pada class barang keluar. Interface disini tidak dapat dibuat object melainkan hanya method yang akan diwariskan Ketika class anak tidak memiliki method yang tidak didapat dari kelas induk. Untuk menghitung pajak harga total ditambah 2000 dan harga member harga hasil ppn dikurangi 10%.

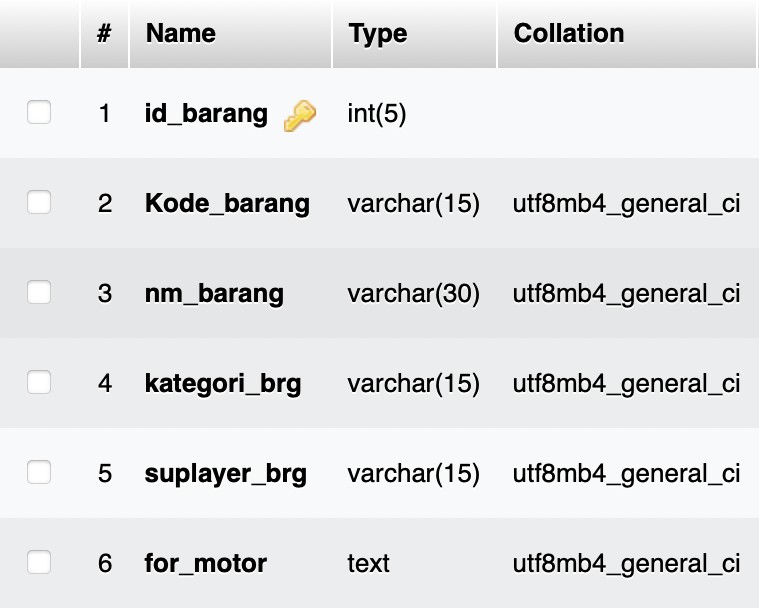
## Tugas Rumah ke-2 :

### Menerapkan CRUD Pada GUI\_Barang Membuat *Database* inventorySparepartMotor



### Gambar 6.34 Membuat database baru

*Structure* tb\_barang pada *database* InventorySparepartMotor



### Gambar 6.35 Struktur table tb\_barang

*Source code import* pada GUI:

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

import java.sql.PreparedStatement; 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\_Barang*:*

public Gui\_Barang() {

initComponents(); tampil(); tampil\_kategori(); tampil\_suplayer();

}

*Source code method* koneksi *:*

public void koneksi() throws SQLException { try {

conn=null; Class.forName("com.mysql.jdbc.Driver"); conn=

DriverManager.getConnection("jdbc:mysql://localhost/Invetor ySparepartMotor?user=root&password=");

}catch (ClassNotFoundException ex){

Logger.getLogger(Gui\_Barang.class.getName()).log(Level.SEVE RE,null, ex);

}

catch (SQLException e) {

Logger.getLogger(Gui\_Barang.class.getName()).log(Level.SEVE RE,null, e);

}

catch (Exception es) {

Logger.getLogger(Gui\_Barang.class.getName()).log(Level.SEVE RE,null, es);

}

}

*Source code* di *method* tampil*:*

public void tampil() {

DefaultTableModel tabelhead = new DefaultTableModel();

tabelhead.addColumn("KODE"); tabelhead.addColumn("Nm Barang"); tabelhead.addColumn("Kategori"); tabelhead.addColumn("suplayer"); tabelhead.addColumn("For Motor");

try { koneksi();

String sql = "SELECT \* FROM tb\_barang"; 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),});

}

tabel\_data.setModel(tabelhead);

} catch (Exception e) { JOptionPane.showMessageDialog(null, "BELUM

TERKONEKSI");

}

}

#### Source code method refresh :

public void refresh(){

new Gui\_Barang().setVisible(true); this.setVisible(false);

}

#### Source code method insert :

public void insert(){

String Kategori

= (String)

Cmb\_Kategori.getSelectedItem(); String Suplayer = (String)

cmb\_suplayer.getSelectedItem();

String Kode = Txt\_KdBarang.getText(); String Nama= Txt\_NmBarang.getText(); String ForMotor = txt\_ForMotor.getText(); try {

koneksi();

Statement statement = conn.createStatement(); statement.executeUpdate("INSERT INTO

tb\_barang(Kode\_barang, nm\_barang, kategori\_brg, suplayer\_brg,for\_motor)"

+"VALUES('"+Kode+"','"+Nama+"','"+Kategori+"','"+Suplayer+" ','"+ForMotor+"')");

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 Kategori = (String) Cmb\_Kategori.getSelectedItem();

String Suplayer = (String) cmb\_suplayer.getSelectedItem();

String Kode = Txt\_KdBarang.getText(); String Nama= Txt\_NmBarang.getText(); String ForMotor = txt\_ForMotor.getText();

String kode\_lama = kode1; try {

Statement statement = conn.createStatement(); statement.executeUpdate("UPDATE tb\_barang SET

kode\_barang='"+Kode+"'," + "nm\_barang='"+Nama+"'"

+",kategori\_brg='"+Kategori+"',suplayer\_brg='"+Suplayer+"', for\_motor='"+ForMotor+"' WHERE kode\_barang='"+kode\_lama+"'");

statement.close(); conn.close();

JOptionPane.showMessageDialog(null, "Update Data Nilai!");

} catch (Exception e) { System.out.println("Error : "+e);

}

refresh();

}

#### Source code method cari :

public void cari(){ try{

try (Statement statement = conn.createStatement())

{

String sql="SELECT \* FROM tb\_barang WHERE

`Kode\_barang` LIKE '%"+txt\_search.getText()+"%'"; ResultSet rs=statement.executeQuery(sql);

//menampilkan data dari sql query

if(rs.next()) // .next() = scanner method

{

Txt\_KdBarang.setText(rs.getString(3)); Txt\_NmBarang.setText(rs.getString(2));

Cmb\_Kategori.setSelectedItem(rs.getString(4)); cmb\_suplayer.setSelectedItem(rs.getString(5));

}

else{

JOptionPane.showMessageDialog(null, "Data yang Anda cari tidak ada");

}

}

}catch (Exception ex){ System.out.println("Error."+ex);

}

}

#### 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\_barang WHERE Kode\_barang='"+Txt\_KdBarang.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");

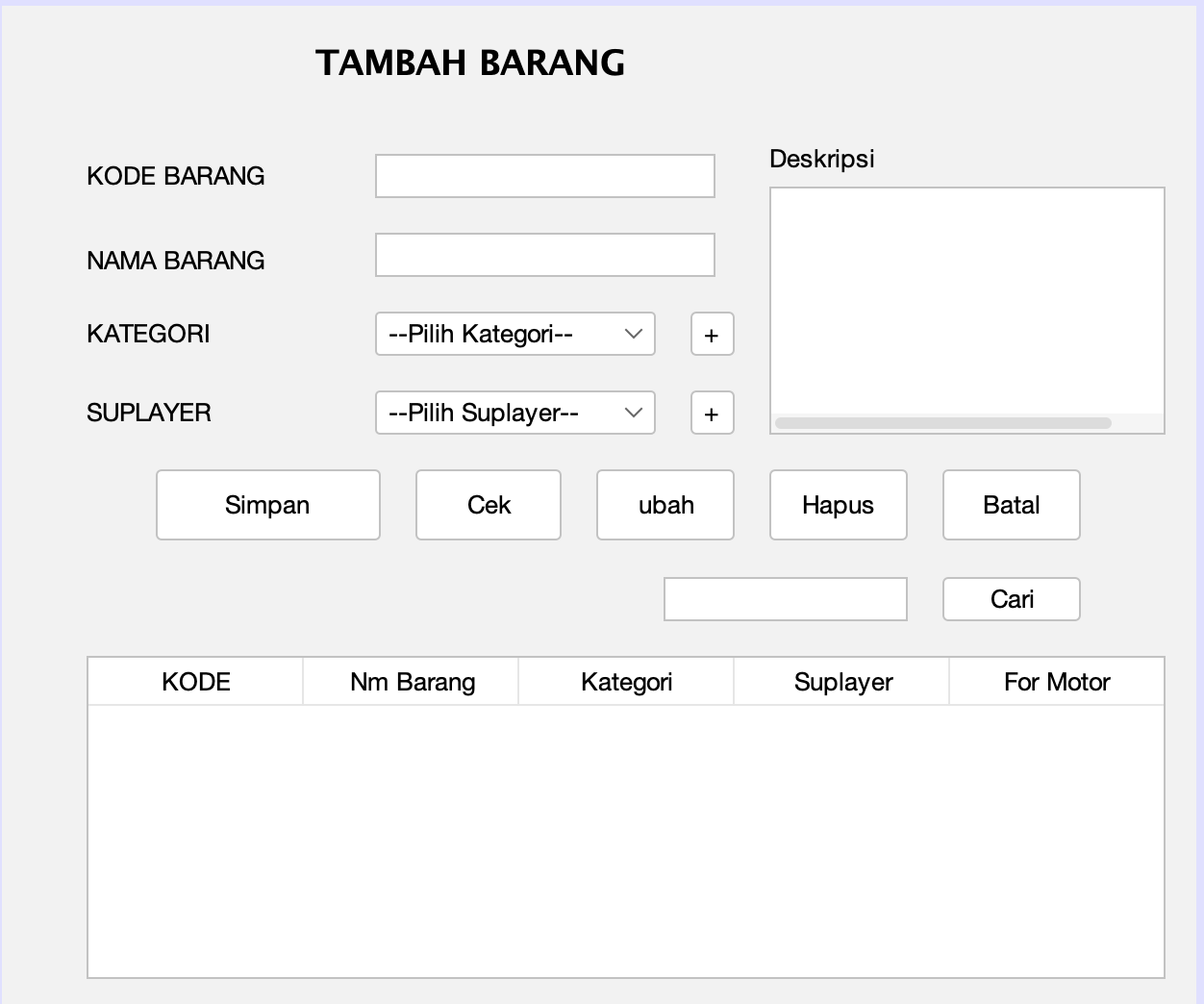
}

refresh();}}

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



### Gambar 6.36 Mennmabahkan library mysql connector Desain *form* GUI\_Barang :



### Gambar 6.37 Desain GUI\_tambahBarang Tabel 6.5 Property GUI\_Barang

|  |  |  |  |
| --- | --- | --- | --- |
| **No** | **Objek** | **Properti** | **Nilai** |
| 1 | jLabel1 | Text | TAMBAH BARANG |
| 2 | jLabel2 | Text | KODE BARANG |
| 3 | jLabel3 | Text | NAMA BARANG |
| 4 | jLabel4 | Text | KATEGORI |
| 5 | jLabel5 | Text | SUPLAYER |
| 6 | jLabel6 | Text | Deskripsi |
| 7 | JtextField1 | Text | “ ” |
| Name | Txt\_KdBarang |
| 8 | JtextField2 | Text | “ “ |
| Name | Txt\_NmBarang |
| 9 | JtextField3 | Text | “ “ |
| Name | txt\_search |
| 10 | JtextArea | Name | txt\_Deskripsi |
| 11 | combobox | Name | Cmb\_Kategori |
| Model | --Pilih Kategori-- |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 12 | combobox | | Name | | cmb\_suplayer | |
|  | | Model | | --Pilih Suplayer-- | |
| 13 | Jbutton1 | | Name | | btn\_simpan | |
| Text | | Simpan | |
| 14 | Jbutton2 | | Name | | btn\_cek | |
| Text | | Cek | |
| 15 | Jbuttno3 | | Name | | btn\_update | |
| Text | | Update | |
| 16 | Jbutton 4 | | Name | | btn\_delete | |
| Text | | Hapus | |
| 17 | Jbutton 5 | | Name | | btn\_batal | |
| Text | | Batal | |
| 18 | Jbutton6 | | Name | | btn\_cari | |
| Text | | Cari | |
| 19 | Jbutton7 | | Name | | btn\_Kategori | |
| Text | | +(add) | |
| 20 | Jbutton8 | | Name | | btn\_suplayer | |
| Text | | +(add) | |
| Jtabel | | Name | | tabel\_data | |  |

*Source* code tombol cari

cari();

*Source* code tombol add(+) kategori

GUI\_Kategori ktg = new GUI\_Kategori(); ktg.show();

*Source* code tombol add(+) suplayer

GUI\_Suplayer spy = new GUI\_Suplayer(); spy.show();

*Source* code tombol simpan

insert();

*Source* code tombol cek

private void btn\_cekActionPerformed(java.awt.event.ActionEvent evt) {

// TODO add your handling code here: try{

Txt\_KdBarang.getText(); Txt\_NmBarang.getText(); if(Cmb\_Kategori.getSelectedIndex()==0) throw

new Exception();

if(cmb\_suplayer.getSelectedIndex()==0) throw new Exception();

}catch(Exception e){ JOptionPane.showConfirmDialog(rootPane, "Input

Anda Salah","WARNING",2);

}

}

*Source* code tombol ubah

update();

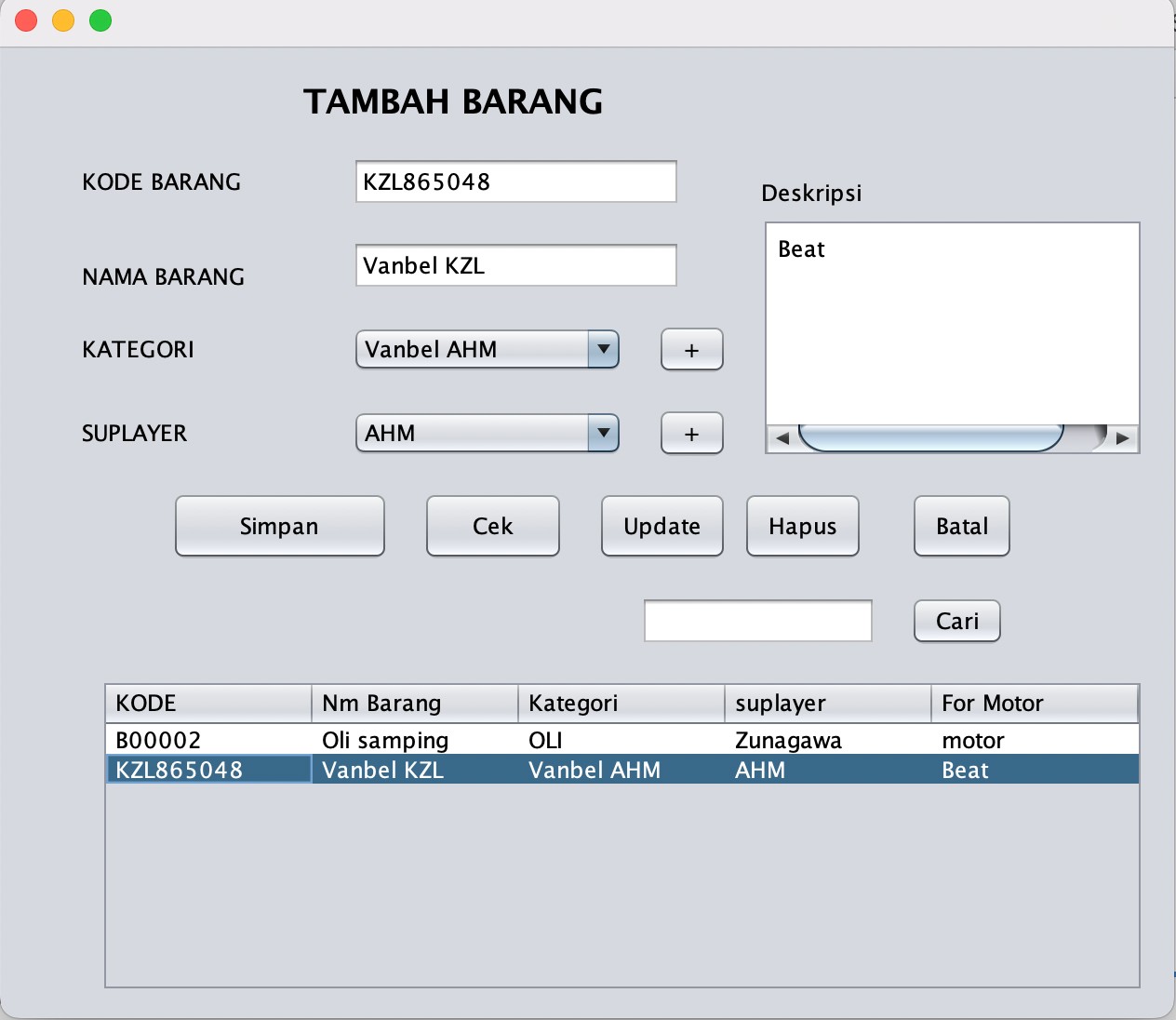
*Source* code tombol hapus

delete();

*Source* code tombol batal

batal();

### Tampilan Hasil:



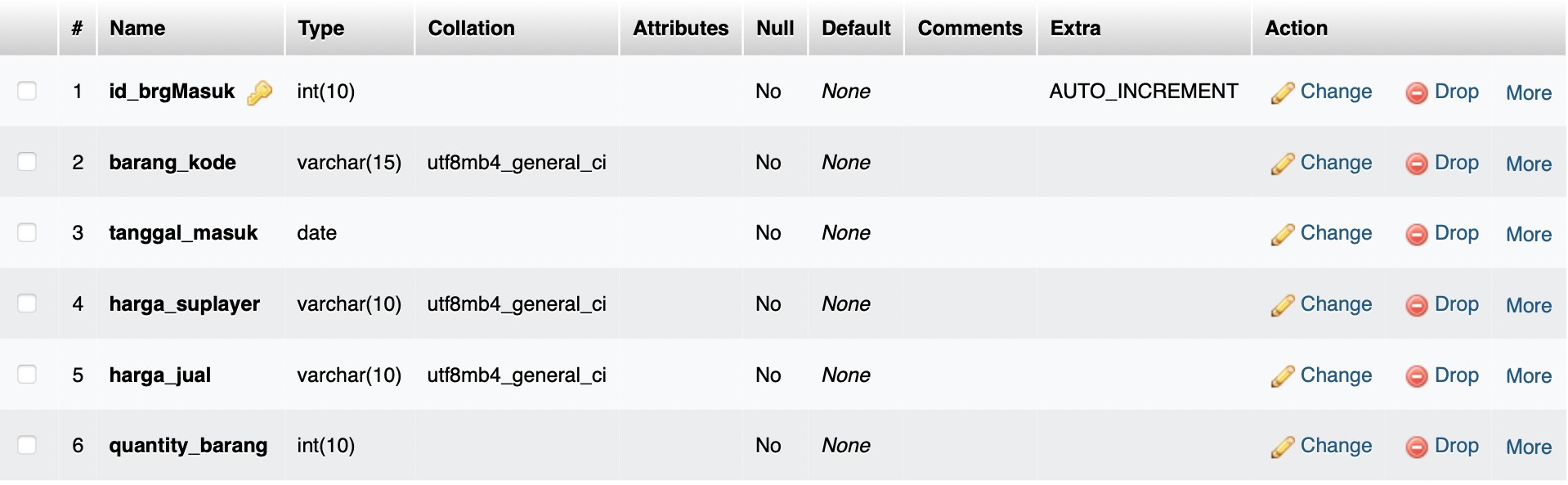
### Analisa :

### Pada program diatas digunakan untuk menambhkan data barang. *User* diminta menginputkan data kemudian pilih tombol cek error, jika tidak ada data yang error maka pilih tombol simpan untuk memasukkan data ke dalam database. Kemudian pilih tombol update untuk mengubah data, dan pilih tombol hapus untuk menghapus data dari database. Lalu tombol cari digunakan untuk mencari data yang kemudian akan ditampilkan pada textfield atau combo box.

## Tugas Rumah ke-3 :

### Menerapkan CRUD Pada GUI\_BarangMasuk

### Strukture tb\_barangmasukpada database inventorySparepartMotor



### Gambar 6.38 Struktur table tb\_barang Source code pada GUI

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

import java.text.SimpleDateFormat; 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\_BarangMasuk

public Connection conn; public void batal(){

cmb\_barang.setSelectedItem(0); txt\_HargaSuplayer.setText(" "); txt\_HargaJual.setText(" "); txt\_Jumlah.setText(" ");

}

*Source code method* koneksi *:*

public void koneksi() throws SQLException { try {

conn=null; Class.forName("com.mysql.jdbc.Driver"); conn=

DriverManager.getConnection("jdbc:mysql://localhost/Inveto rySparepartMotor?user=root&password=");

}

catch (ClassNotFoundException ex){

Logger.getLogger(GUI\_BarangMasuk.class.getName()).log(Leve l.SEVERE,null, ex);

}

catch (SQLException e) {

Logger.getLogger(GUI\_BarangMasuk.class.getName()).log(Leve l.SEVERE,null, e);

}

catch (Exception es) {

Logger.getLogger(GUI\_BarangMasuk.class.getName()).log(Leve l.SEVERE,null, es);

}

}

*Source code* di *method* tampil*:*

public void tampil() {

DefaultTableModel tabelhead = new DefaultTableModel();

tabelhead.addColumn("KODE"); tabelhead.addColumn("TanggaL Masuk"); tabelhead.addColumn("Nm Barang"); tabelhead.addColumn("Suplayer"); tabelhead.addColumn("Harga Awal"); tabelhead.addColumn("Harga Jual"); tabelhead.addColumn("Stock Barang"); try {

koneksi();

String sql = "SELECT \* FROM tb\_barangMasuk INNER JOIN tb\_barang ON tb\_barang.Kode\_barang = tb\_barangMasuk.barang\_kode;";

Statement stat = conn.createStatement(); ResultSet res = stat.executeQuery(sql); while (res.next()) {

tabelhead.addRow(new Object[]{res.getString(2), res.getString(3), res.getString(9),

res.getString(11), res.getString(4), res.getString(5),res.getString(6),});

}

tabel\_data.setModel(tabelhead);

} catch (Exception e) { JOptionPane.showMessageDialog(null, "BELUM

TERKONEKSI");

}

}

#### Source code method refresh :

public void refresh(){

new GUI\_BarangMasuk ().setVisible(true); this.setVisible(false);

}

*Source code method* insert*:*

public void insert(){

int Quntity=0;

String KdBarang = (String) cmb\_barang.getSelectedItem();

SimpleDateFormat date = new SimpleDateFormat("yyyy-MM-dd");

String tanggal = date.format(Calender.getDate()); int harga\_Spy =

Integer.parseInt(txt\_HargaSuplayer.getText()); int harga\_Jual =

Integer.parseInt(txt\_HargaJual.getText());

int qty = Integer.parseInt(txt\_Jumlah.getText()); if(cmb\_satuan.getSelectedIndex() == 1){

Quntity = qty;

}else if(cmb\_satuan.getSelectedIndex() == 2){ Quntity = qty \* 12;

}else if(cmb\_satuan.getSelectedIndex() == 3){ Quntity = qty \* 24;

}

try { koneksi();

Statement statement = conn.createStatement(); statement.executeUpdate("INSERT INTO

tb\_barangMasuk (barang\_kode, tanggal\_masuk, harga\_suplayer,harga\_jual, quantity\_barang)"

+"VALUES('"+KdBarang

+"','"+tanggal

+"','"+harga\_Spy+"','"+harga\_Jual+"','"+Quntity+"')"); statement.close(); JOptionPane.showMessageDialog(null, "Berhasil

Memasukan Data Barang!");

} catch (Exception e) { JOptionPane.showMessageDialog(null, "Terjadi

Kesalahan Input!");

}

refresh();

}

#### Source code method ubah :

public void update(){

String HargaSuplayer = txt\_HargaSuplayer.getText();

String HargaJual = txt\_HargaJual.getText(); String juml = txt\_Jumlah.getText(); String Kodelama = KdBarang1;

try {

Statement statement = conn.createStatement();

statement.executeUpdate("UPDATE tb\_barangMasuk SET harga\_suplayer='"+HargaSuplayer+"'," + "harga\_jual='"+HargaJual+"'"

+ ",quantity\_barang='"+juml+"'WHERE barang\_kode = '"+Kodelama+"'");

statement.close(); conn.close();

JOptionPane.showMessageDialog(null, "Update Data Barang Masuk!");

} 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\_barangMasuk WHERE barang\_kode='"+cmb\_barang.getSelectedItem()+"'";

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\_barangMasuk WHERE

`barang\_kode` LIKE '%"+txt\_Search.getText()+"%'"; ResultSet rs=statement.executeQuery(sql);

//menampilkan data dari sql query

if(rs.next()) // .next() = scanner method

{

cmb\_barang.setSelectedItem(rs.getString(2)); txt\_HargaSuplayer.setText(rs.getString(4));

txt\_HargaJual.setText(rs.getString(5)); txt\_Jumlah.setText(rs.getString(6));

}

else{

JOptionPane.showMessageDialog(null, "Data yang Anda cari tidak ada");

}

}

}catch (Exception ex){ System.out.println("Error."+ex);

}

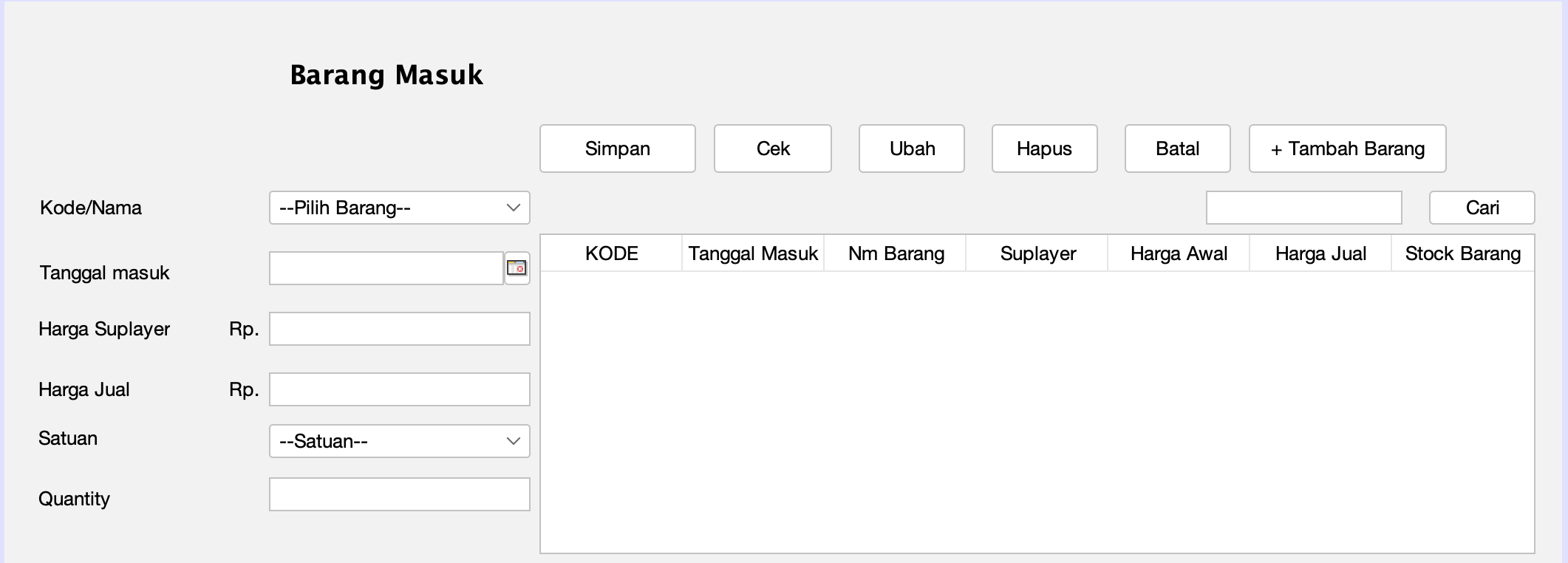
}

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



### Gambar 6.39 Menambahkan library Connector

### Desain Form



### Gambar 6.40 Desain Form Barang Masuk Tabel 6.6 Properti GUI\_BarangMasuk

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **No** | **Objek** | | **Properti** | **Nilai** | |
| 1 | Jlabel1 | | Text | Barang Masuk | |
| 2 | Jlabel2 | | Text | Kode/Nama | |
| 3 | Jlabel3 | | Text | Tanggal Masuk | |
| 4 | Jlabel4 | | Text | Harga Suplayer | |
| 5 | Jlabel5 | | Text | Harga Jual | |
| 6 | Jlabel6 | | Text | Satuan | |
| 7 | Jlabel7 | | Text | Quantiry | |
| 8 | Jlabel8 | | Text | Rp | |
| 9 | Jlabel9 | | Text | Rp | |
| 10 | Combobox1 | | Name | cmb\_barang | |
| model | --Pilih Barang-- | |
| 11 | Combobox2 | | Name | cmb\_satuan | |
| model | --Satuan--  ,Unit,Kardus,kodi | |
| 12 | JDateChooser | | Name | Calender | |
| 13 | JTextField1 | | Name | txt\_HargaSuplayer | |
| Text | “ “ | |
| 14 | JTextField2 | | Name | txt\_HargaJual | |
| Text | “ “ | |
| 15 | JTextField3 | | Name | txt\_Jumlah | |
|  | | Nama Aslab Pengajar : Silva Anjelina Tambunan | | | TTD : |
| Tanggal : | | |

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | Text | “ “ |
| 16 | JTextField4 | Name | txt\_Search |
| Text | “ “ |
| 17 | JTabel | Name | tabel\_data |
| 18 | Jbottom1 | Name | btn\_simpan |
| Text | Simpan |
| 19 | Jbottom2 | Name | btn\_cek |
| Text | Cek |
| 20 | Jbottom3 | Name | btn\_update |
| Text | Update |
| 21 | Jbottom4 | Name | btn\_delete |
| Text | Hapus |
| 22 | Jbottom5 | Name | btn\_batal |
| Text | Batal |
| 23 | Jbottom6 | Name | Next\_TabahBarang |
| Text | + Tambah Barang |
| 24 | Jbottom7 | Name | btn\_cari |
| Text | Cari |

### Source code button Simpan

insert();

### Source code button Ubah

update();

### Source code button Hapus

insert();

### Source code button Cari

cari();

### Source code button Batal

batal();

### Source code pada Tabel

int tabel = tabel\_data.getSelectedRow(); //0 KdBarang1 = tabel\_data.getValueAt(tabel,

0).toString();

tanggal1 = tabel\_data.getValueAt(tabel, 1).toString();

harga\_Spy1 = Integer.parseInt(tabel\_data.getValueAt(tabel, 4).toString());

harga\_Jual1 = Integer.parseInt(tabel\_data.getValueAt(tabel, 5).toString());

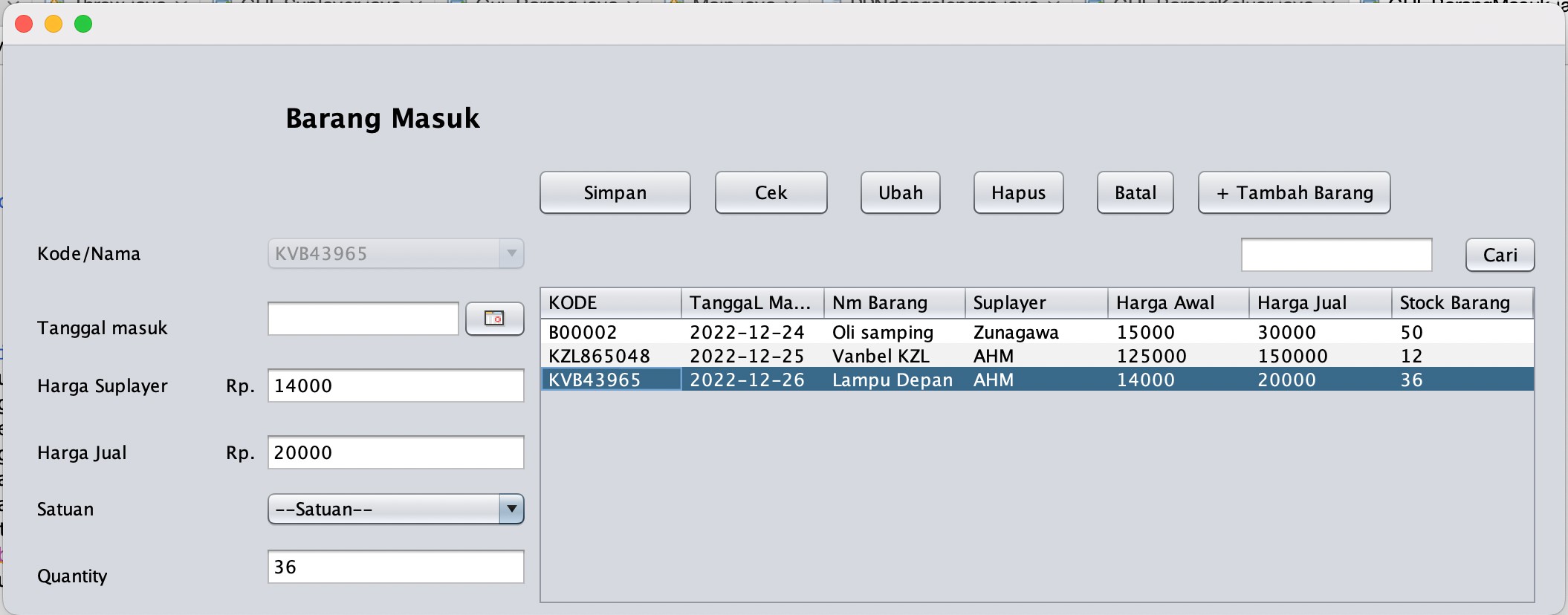
Qty1 = Integer.parseInt(tabel\_data.getValueAt(tabel, 6).toString());

itempilih();

### Source code pada Tambah Barang

Gui\_Barang brg = new Gui\_Barang(); brg.show();

### Tampilan Hasil :



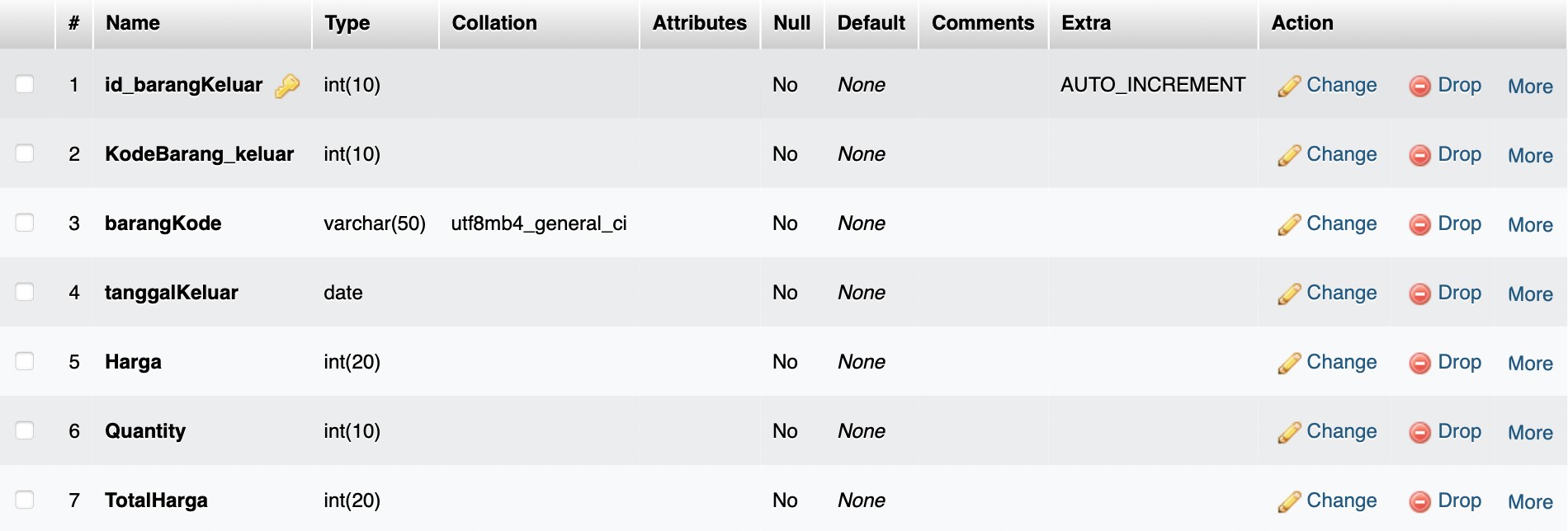
### Gambar 6.41 Tampilan hasil Gui barnag masuk

### Analisa :

### Pada program GUI di atas digunakan untuk menginputkan jumlah barang masuk, beserta harga dari barang tersebut. *User* diminta menginputkan data kemudian pilih tombol cek error, jika tidak ada data yang error maka pilih tombol tampilkan untuk memasukkan data ke dalam database. Kemudian pilih tombol Edit untuk mengubah data, dan pilih tombol hapus untuk menghapus data dari *database*. Lalu tombol cari digunakan untuk mencari data yang kemudian akan ditampilkan pada *textfield* atau *combo box*. Untuk tombol + tambah barang digunakan untuk tombol kembali ke tambah barang.

## Tugas Rumah ke-4 :

### Menerapkan CRUD Pada GUI\_BarangKeluar Membuat *Database* inventorySparepartMotor



### Gambar 6.42 Struktur tabel barang keluar

*Source code* pada GUI

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

import javax.swing.table.DefaultTableModel; import java.util.Random;

*Source code* Konstruktor GUI\_BarangKeluar

public GUI\_BarangKeluar() { initComponents(); tampil\_barang(); tampil();

txt\_NamaBarang.setEnabled(false); txt\_deskripsi.setEnabled(false); txt\_harga.setEnabled(false); txt\_stock.setEnabled(false);

}

*Source code* di bawah Konstruktor GUI\_BarangKeluar

int random;

public Connection conn;

String kode\_barangKeluar1,kode1;

public void itempilih(){ cmb\_barang.setSelectedItem(kode1);

}

public void batal(){ cmb\_barang.setSelectedItem("--Pilih--"); txt\_NamaBarang.setText(" "); txt\_deskripsi.setText(" "); txt\_harga.setText(" "); txt\_stock.setText(" "); txt\_Jumlah.setText(" "); Calender.setDateFormatString( " ");}

*Source code method* koneksi :

public void koneksi() throws SQLException { try {

conn=null; Class.forName("com.mysql.jdbc.Driver"); conn=

DriverManager.getConnection("jdbc:mysql://localhost/Invetor ySparepartMotor?user=root&password=");

}

catch (ClassNotFoundException ex){

Logger.getLogger(Inventory.Bab2\_konstructor.Bab3\_Inheritanc e.Bab4\_Enkapsulasi.Bab5\_OverloadingdanOverriding.Bab6\_Abstr act.bab7\_Polimorfisme.Bab8\_TryCatch.Bab9\_Interface.Bab10\_Da tabase.GUI\_BarangMasuk.class.getName()).log(Level.SEVERE,nu ll, ex);

}

catch (SQLException e) {

Logger.getLogger(Inventory.Bab2\_konstructor.Bab3\_Inheritanc e.Bab4\_Enkapsulasi.Bab5\_OverloadingdanOverriding.Bab6\_Abstr act.bab7\_Polimorfisme.Bab8\_TryCatch.Bab9\_Interface.Bab10\_Da tabase.GUI\_BarangMasuk.class.getName()).log(Level.SEVERE,nu ll, e);

}

catch (Exception es) {

Logger.getLogger(Inventory.Bab2\_konstructor.Bab3\_Inheritanc e.Bab4\_Enkapsulasi.Bab5\_OverloadingdanOverriding.Bab6\_Abstr act.bab7\_Polimorfisme.Bab8\_TryCatch.Bab9\_Interface.Bab10\_Da tabase.GUI\_BarangMasuk.class.getName()).log(Level.SEVERE,nu ll, es);

}

}

*Source code method* rand :

public int rand(){

Random rndm = new Random(); for(int i = 0; i<10 ; i++){

random = rndm.nextInt(100000);

}

return random;

}

*Source code method* tampil\_barang :

public void tampil\_barang(){ try {

koneksi();

String sql = "SELECT barang\_kode FROM

`tb\_barangMasuk`";

Statement stt = conn.createStatement(); ResultSet res = stt.executeQuery(sql); while(res.next()){

Object[] ob = new Object[3]; ob[0] = res.getString(1);

cmb\_barang.addItem((String) ob[0]);

}

res.close();

stt.close();

}catch (Exception e) { System.out.println(e.getMessage());

}

}

*Source code method* refresh:

public void refresh(){

new GUI\_BarangKeluar().setVisible(true); this.setVisible(false);

}

*Source code method* insert:

public void insert(){

int Quntity=0,total; int rdm = rand();

String KdBarang = (String) cmb\_barang.getSelectedItem();

SimpleDateFormat date = new SimpleDateFormat("yyyy-

MM-dd");

String tanggal = date.format(Calender.getDate()); int qty = Integer.parseInt(txt\_Jumlah.getText()); int harga = Integer.parseInt(txt\_harga.getText()); if(cmb\_satuan.getSelectedIndex() == 1){

Quntity = qty;

}else if(cmb\_satuan.getSelectedIndex() == 2){ Quntity = qty \* 12;

}else if(cmb\_satuan.getSelectedIndex() == 3){ Quntity = qty \* 24;

}

total = harga \* Quntity; try {

koneksi();

Statement statement = conn.createStatement(); statement.executeUpdate("INSERT INTO

tb\_barangKeluar (KodeBarang\_keluar, barangKode,tanggalKeluar,Harga,Quantity,TotalHarga)"

+"VALUES('"+rdm+"','"+KdBarang

+"','"+tanggal+"','"+harga+"','"+Quntity+"','"+total+"')"); update\_stock();

statement.close(); JOptionPane.showMessageDialog(null, "Berhasil

Memasukan Data Barang!");

} catch (Exception e) { JOptionPane.showMessageDialog(null, "Terjadi

Kesalahan Input!");

}

refresh();

}

*Source code method* tampil :

public void tampil() {

DefaultTableModel tabelhead = new DefaultTableModel();

tabelhead.addColumn("ID Keluar"); tabelhead.addColumn("KODE"); tabelhead.addColumn("TanggaL Keluar"); tabelhead.addColumn("Harga"); tabelhead.addColumn("QTY"); tabelhead.addColumn("Total");

try {

koneksi();

String sql = "SELECT \* FROM tb\_barangKeluar"; 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), });

}

tabel\_data.setModel(tabelhead);

} catch (Exception e) { JOptionPane.showMessageDialog(null, "BELUM

TERKONEKSI");

}

}

*Source code method* update\_stock :

public void update\_stock(){

int Quntity=0,jml\_stock; String kode = (String)

cmb\_barang.getSelectedItem();

int qty = Integer.parseInt(txt\_Jumlah.getText()); int stock =Integer.parseInt(txt\_stock.getText()); if(cmb\_satuan.getSelectedIndex() == 1){

Quntity = qty;

}else if(cmb\_satuan.getSelectedIndex() == 2){ Quntity = qty \* 12;

}else if(cmb\_satuan.getSelectedIndex() == 3){ Quntity = qty \* 24;

}

jml\_stock = stock - Quntity; try {

Statement statement = conn.createStatement(); statement.executeUpdate("UPDATE tb\_barangMasuk

SET quantity\_barang='"+jml\_stock+"'WHERE barang\_kode='"+kode+"'");

statement.close(); conn.close();

} catch (Exception e) { System.out.println("Error : "+e);

}

refresh();

}

*Source code method* search\_barang :

public void search\_barang(){ try{

try (Statement statement = conn.createStatement())

{

String sql="SELECT \* FROM tb\_barang WHERE

`Kode\_barang` LIKE '%"+cmb\_barang.getSelectedItem()+"%'"; ResultSet rs=statement.executeQuery(sql);

//menampilkan data dari sql query

if(rs.next()) // .next() = scanner method

{

txt\_NamaBarang.setText(rs.getString(3)); txt\_deskripsi.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* search\_barangMasuk :

public void search\_barangMasuk(){ try{

try (Statement statement = conn.createStatement())

{

String sql="SELECT \* FROM tb\_barangMasuk WHERE

`barang\_kode` LIKE '%"+cmb\_barang.getSelectedItem()+"%'"; ResultSet rs=statement.executeQuery(sql);

//menampilkan data dari sql query

if(rs.next()) // .next() = scanner method

{

txt\_harga.setText(rs.getString(5)); txt\_stock.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* 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\_barangKeluar WHERE KodeBarang\_keluar='"+kode\_barangKeluar1+"'";

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();

}

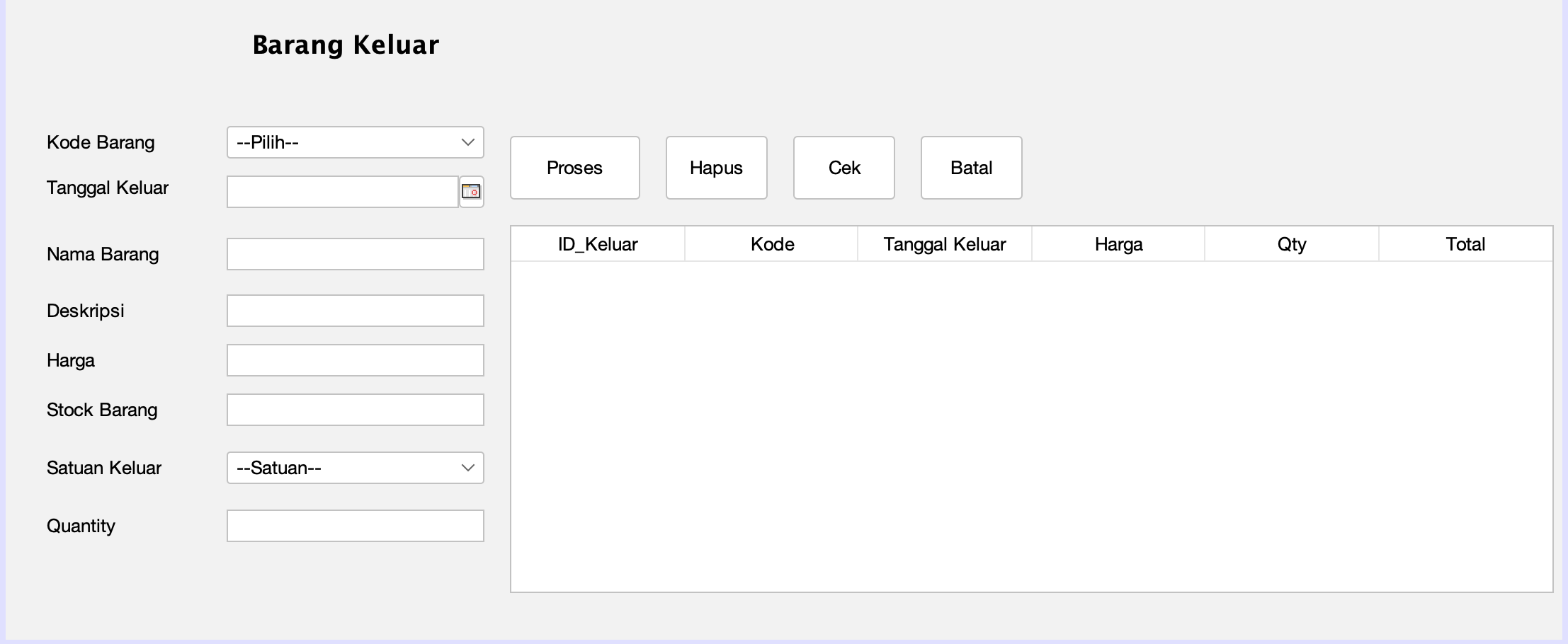
}

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



### Gambar 6.43 Menambahkan library Connector

### Desain Form



### Gambar 6.44 Desain Form Barang keluar Tabel 6.7 Properti GUI\_BarangKeluar

|  |  |  |  |
| --- | --- | --- | --- |
| **No** | **Objek** | **Properti** | **Nilai** |
| 1 | Jlabel1 | Text | Barang Keluar |
| 2 | Jlabel2 | Text | Kode Keluar |
| 3 | Jlabel3 | Text | Tanggal keluar |
| 4 | Jlabel4 | Text | Nama Barang |
| 5 | Jlabel5 | Text | Deskripsi |
| 6 | Jlabel6 | Text | Harga |
| 7 | Jlabel7 | Text | Stock Barang |
| 8 | Jlabel8 | Text | Satuan Keluar |

|  |  |  |  |
| --- | --- | --- | --- |
| 9 | Jlabel9 | Text | Quantity |
| 10 | Combobox1 | Name | cmb\_barang |
| Model | --Pilih-- |
| 11 | Combobox2 | Name | cmb\_satuan |
| Model | --Satuan--  ,Unit,Kardus,kodi |
| 12 | JDateChooser | Name | Calender |
| 13 | JTextField1 | Name | txt\_NamaBarang |
| Text | “ “ |
| 14 | JTextField2 | Name | txt\_deskripsi |
| Text | “ “ |
| 15 | JTextField3 | Name | txt\_harga |
| Text | “ “ |
| 16 | JTextField4 | Name | txt\_stock |
| Text | “ “ |
| 17 | JTextField5 | Name | txt\_Jumlah |
| Text | “ “ |
| 18 | Jbutton1 | Name | btn\_Proses |
| Text | Proses |
| 19 | Jbutton2 | Name | btn\_delete |
| Text | Hapus |
| 20 | Jbutton3 | Name | btn\_cek |
| Text | Cek |
| 21 | Jbutton4 | Name | btn\_batal |
| Text | Batal |
| 22 | Jtable | Name | tabel\_data |
| Text | ID\_Keluar, Kode, Tanggal Keluar, Harga,  Qty,Total |

*Source code* pada comboBox Kode Barang*:*

search\_barang(); search\_barangMasuk();

*Source code* pada tombol Proses*:*

insert();

*Source code* pada tombol Hapus*:*

delete();

*Source code* pada tombol cek :

try{

Calender.getCalendar(); txt\_Jumlah.getText();

if(cmb\_barang.getSelectedIndex()==0) throw new

Exception();

if(cmb\_satuan.getSelectedIndex()==0) throw new

Exception();

}catch(Exception e){ JOptionPane.showConfirmDialog(rootPane, "Input

Anda Salah","WARNING",2);

}

*Source code* pada tombol batal*:*

batal();

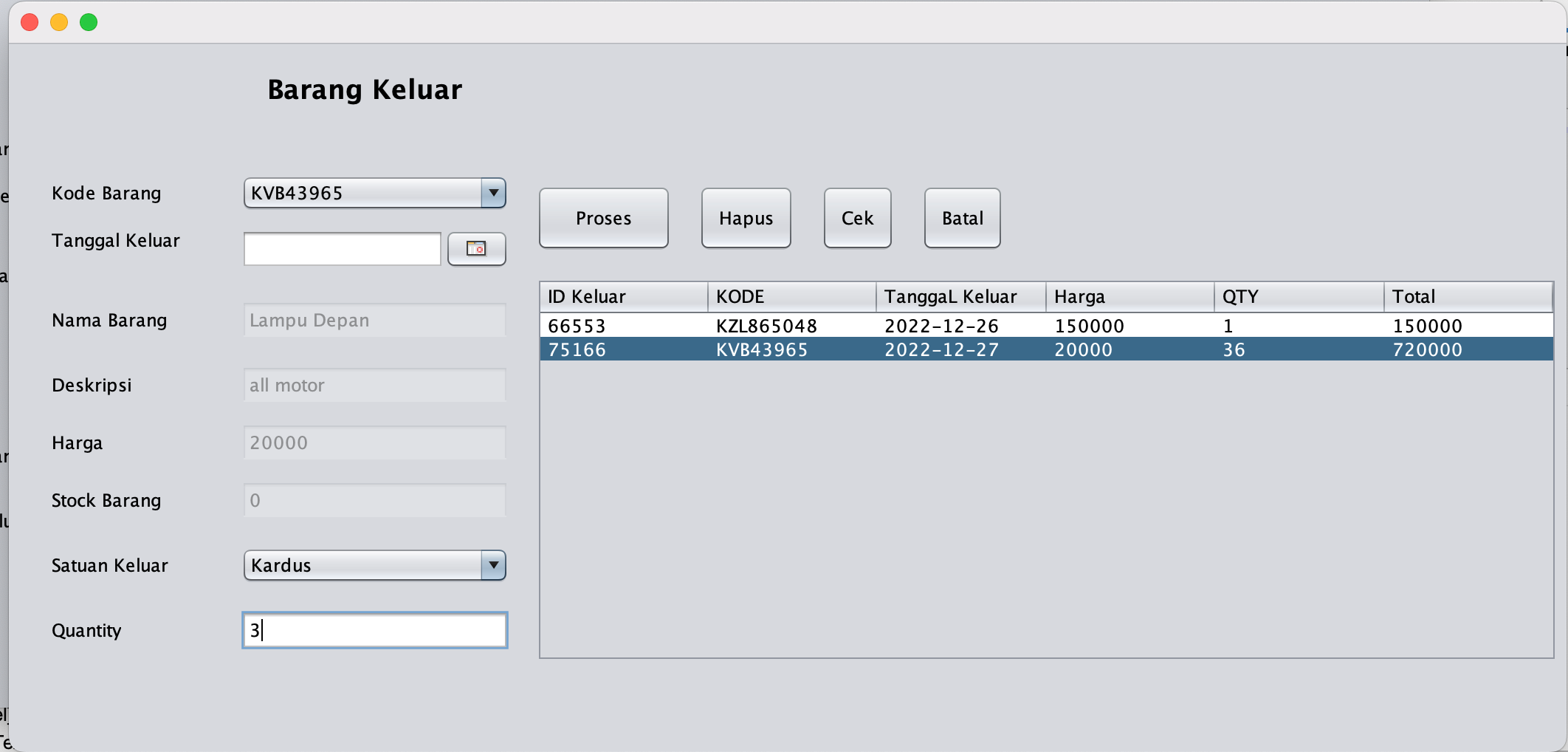
*Source code* pada tabel*:*

int tabel = tabel\_data.getSelectedRow(); kode\_barangKeluar1 = tabel\_data.getValueAt(tabel,

0).toString();

kode1 = tabel\_data.getValueAt(tabel, 1).toString(); itempilih();

### Tampilan hasil :



### Gambar 6.45 Hasil Tampilan Barang Keluar

### Analisa :

### Pada soure code diatas merupakan program untuk barang keluar, dimana untuk user memilih Kode barang yang sudah ada di combobox kemudian bebrapa text field akan menampilkan databarang dari kode yang sudah di pilih. User tinggal memasukkan tanggal barang tersebut keluar dan jumlah barang yang dikeluarkan. Untuk button proses digunaan untuk memperoses barang yang akan di keluarkan yang nantinya akan tercatat dalam database. Tombol hapus digunakan untuk menghapus data yang mana diambil dari kode barang keluar yang diinputkan secara random oleh program. Kemudian untuk tombol cek digunakan untuk mencek error di dalam text field, tombol batal digunakan utntuk meremove semua isi yang ada di combobox dan text field.

## Kesimpulan

1. Isi dari *interface* adalah *method abstract*, artinya *method* hanya dideklarasikan dan tanpa badan *method*. Deklarasi *method* pada *interface* serupa dengan *method* pada *class abstract.*
2. *Implements* bertujuan untuk menyamakan struktur dari *class* yang mengimplementasi *interface,* sedangkan *exteds* bertujuan untuk memperluas isi sebuah *class.*

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

**BAB VII**

**Kesimpulan**

### Dengan adanya kegiatan praktikum *OOP*. Penulis dapat menyimpulkan sebagai berikut :

### OOP (*Object Oriented Programming*) adalah suatu metode pemrograman yang berorientasi kepada objek yang bertujuan mempermudah pengembangan program dengan cara mengikuti model yang telah ada di kehidupan sehari-hari

### Konsktruktor adalah method yang pertama kali dijalankan pada saat sebuah objek pertama kali diciptakan, yang dimana syarat penulisan method konstruktor ini harus sama dengan nama class yang di naungi. Biasanya method ini digunakan untuk inisialisasi atau mempersiapkan data untuk objek

### *Inheritance* atau Pewarisan/Penurunan adalah konsep pemrograman dimana sebuah class dapat ‘menurunkan’ properti dan method yang dimilikinya kepada class lain

### Enkapsulasi merupakan proses pemaketan objek beserta *method*nya untuk menyembunyikan rincian implementasi dari pemakai/objek lainnya

### *Method Overloading* adalah sebuah kemampuan yang membolehkan sebuah class mempunyai 2 atau lebih method dengan nama yang sama, namun parameter yang berbeda. *Overriding* method adalah kemampuan dari subclass 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

### *Abstract Class* adalah sebuah *class* yang tidak bisa di-instansiasi (tidak bisa dibuat menjadi objek) dan berperan sebagai ‘kerangka dasar’ bagi class turunannya

1. *Poly* artinya banyak, *morfisme* artinya bentuk. *Polimorfisme* dalam OOP adalah sebuah prinsip di mana class dapat memiliki banyak “bentuk
2. *Exception Handling* merupakan mekanisme yang paling diperlukan dalam menangani *error* yang terjadi pada saat *runtime* (program berjalan) atau yang lebih dikenal dengan sebutan *runtime error*
3. *Interface* merupakan *prototype* atau template untuk sebuah *class*, deskripsi ini hampir serupa dengan *class* abstrak.

### 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.

**DAFTAR PUSTAKA**

Laboratorium Pemrograman dan RPL. 2021. *Modul Praktikum Algoritma Dan Pemrogramman.* Laboratorium Pemrograman Insitut Teknologi Nasional Malang.

### Danuri, Muhamad. "Object Oriented Programming (Oop) Pembangun Program Aplikasi Berbasis Windows." Jurnal Ilmiah Infokam 5.1 (2013).

### Alamsyah, Akbar. Dasar OOP Java (Inhertance). Abengz Media, 2020.

### Adnan, Iskandar Alisyahbana. Polimorfisme Sekuens D-Loop DNA Mitokondria Pada Populasi Kajang, Mandar, Dan Dayak Benuaq. Diss. IPB (Bogor Agricultural University).

### Jamal, Ade, et al. "Aplikasi Berorientasi Objek untuk Perhitungan Bagian Waris Berdasarkan Hukum Islam." Jurnal Al-Azhar Indonesia Seri Sains dan Teknologi 2.4 (2015): 234-247.

### Andaru, Andry. "Pengertian database secara umum." OSF Prepr 1.1 (2018): 6. Sibarani, Niko Sumanda, Ghifari Munawar, and Bambang Wisnuadhi. "Analisis

### Performa Aplikasi Android Pada Bahasa Pemrograman Java dan Kotlin." Prosiding Industrial Research Workshop and National Seminar. Vol. 9. 201

## LABORATORIUM PEMROGRAMAN KOMPUTER INSTITUT TEKNOLOGI NASIONAL

### Kampus II : Jl. Raya Karanglo Km. 2 Malang

**LEMBAR ASISTENSI PRAKTIKUM *OBJECT ORIENTED PROGRAMMING***

**SEMESTER GANJIL TAHUN AKADEMIK 2021/2022**

## Nama : Firman Frezy Pradana

## NIM 2118112

## Kelompok 25

### FOTO

### 3x4

(Menggunakan Jas Almamater)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **No.** | **Tanggal** | **Asistensi** | | | | **Paraf** |
| **Konsep** | | **Program** | **Hasil Akhir** |
| 1 |  | **Instruktur** | Konsep Dasar *OOP* Tipe Data, Konstruktor Enkapsulasi, Inheritance Abstrak,*Interface* Polimorfisme, Exception GUI Database |  |  |  |
| 2 |  | **Dosen** |  |  |  |  |
| **Batas Akhir:** | | | | | | |

## Asisten,

## (Silva Anjelina Tambunan)

## 1918023

### Malang, Januari 2023

## Dosen, Pembimbing,

## (Ahmad Fahrudi Setiawan, S.Kom, M.T.) NIP.P.1031500497