Tugas Proyek Makassar, 20 Juni 2020

PEMROGRAMAN BERORIENTASI OBJEK
(MENGHITUNG GAJI BERSIH PEGAWAI CV.XXXI)

Nama : Siti Nursahida Imlan

Stambuk : 13020180104

Kelas : B2

PROGRAM STUDI TEKNIK INFORMATIKA

FAKULTAS ILMU KOMPUTER

UNIVERSITAS MUSLIM INDONESIA

MAKASSAR

2020

KATA PENGANTAR

Assalamualaikum Warahmatullahi Wabarakatuh.

Puji syukur kami ucapkan atas kehadirat Allah SWT, karena dengan rahmat dan karunia-Nya, kami masih diberi kesempatan untuk menyelesaikan laporan ini tentang konsep *Menghitung Gaji Bersih Pegawai* sebagai salah satu tugas Proyek Pemrograman Berorientasi Objek.

Tidak lupa kami ucapkan terima kasih kepada bapak/ibu dosen serta berbagai pihak yang telah memberikan dukungan serta memberikan petunjuk dalam menyelesaikan laporan ini. Kami menyadari bahwa dalam penulisan laporan ini masih banyak kekurangan, oleh sebab itu kami sangat mengharapkan kritik dan saran yang membangun.

Semoga dengan selesainya laporan ini dapat bermanfaat bagi pembaca dan teman-teman.

Makassar, 20 Juni 2020

Siti Nursahida Imlan

BAB I

PENDAHULUAN

1.1 Latar Belakang

Perkembangan ilmu pengetahuan dan teknologi saat ini sangatlah pesat. Kini banyak aplikasi pemrograman yang telah diciptakan dan berkembang untuk mempermudah berbagai pekerjaan manusia. Berbagai model aplikasi mulai dari yang sifatnya edukasi, informatif, hiburan, dan banyak lainnya yang dapat kita nikmati saat ini. Terutama aplikasi yang berbasic perhitungan. Dalam hal ini penulis mencoba membuat model sederhana untuk berbagai metoda perhitungan gaji pegawai dengan menggunakan Application Programming Interface (API).

API merupakan kumpulan dari pustaka class atau komponen atau library, yang sudah disediakan oleh Sistem Operasi, dimana berisi kumpulan perintah yang membentuk sebuah komponen yang sangat mempermudah para programmer dalam membangun aplikasi. Tentunya model aplikasi apapun dapat kita bangun menjadi aplikasi yang sangat variatif dan dapat digunakan dengan mudah.

Pada tugas kali ini penulis mencoba membangun aplikasi perhitungan sederhana untuk menghitung gaji seorang pegawai

1.2 Rumusan Masalah

Sesuai dengan latar belakang yang telah dijelaskan, maka rumusan masalah yang akan dikaji dalam laporan ini adalah bagaimana membangun sebuah aplikasi pemrograman berorientasi objek yang efektif dalam pengolahan gaji karyawan dengan komputerisasi. Dengan begitu bagian keuangan tidak harus melakukan perhitungan gaji karyawan dengan cara manual.

BAB II
LANDASAN TEORI

2.1 Konsep Pemrograman Berorientasi Objek

Desain berorientasi object adalah sebuah teknik yang memfokuskan desain pada object dan class berdasarkan pada skenario dunia nyata. Pemrograman berorientasi objek diciptakan untuk mempermudah pengembangan program dengan cara mengikuti model yang telah ada dalam kehidupan nyata. Dalam paradigma ini, sesuai dengan model kehidupan nyata, segala bagian (entiti) dari suatu permasalahan adalah objek. Hal ini menegaskan keadaan (state), behaviour dan interaksi dari objek. Objek-objek ini kemudian juga dapat berupa gabungan dari beberapa objek yang lebih kecil. Sebagai contoh, tengoklah sebuah mobil. Mobil adalah sebuah objek dalam kehidupan nyata. Namun mobil sendiri terbentuk dari beberapa objek yang lebih kecil seperti roda ban, mesin, jok, dll. Mobil sebagai objek yang merupakan gabungan dari objek yang lebih kecil dibentuk dengan membentuk hubungan antara objek-objek penyusunnya. Begitu juga dengan sebuah program. Objek besar dapat dibentuk dengan menggabungkan beberapa objek-objek dalam bahasa pemrograman. Objek-objek tersebut berkomunikasi dengan saling mengirim pesan kepada objek lain. Selain itu juga menyediakan manfaat akan kebebasan pengembangan, meningkatkan kualitas, mempermudah pemeliharaan, mempertinggi kemampuan dalam modifikasi dan meningkatkan penggunaan kembali software. Konsep-konsep pemrograman berorientasi objek dalam Java secara umum sama dengan yang digunakan oleh bahasa-bahasa lain. Jadi kebanyakan konsep yang kita bahas juga terdapat dalam bahasa selain Java. Namun, terkadang terdapat perbedaan-perbedaan kecil antara penerapan konsep-konsep tersebut dalam masing-masing bahasa. Perbedaan-perbedaan ini juga akan dijelaskan seiring

penjelasan masing-masing konsep. Kelas merupakan inti dari pemograman java. Java itu sendiri merupakan pemograman yang mendukung dan mengimplementasikan konsep pemograman berorientasi objek sepenuhnya.

2.2 Objek

Baik dalam dunia nyata atau dalam sebuah program, sebuah objek memiliki dua karakteristik, yaitu state dan behaviour. State adalah keadaan dari sebuah objek, seperti mobil memiliki state warna, model, tahun pembuatan, kondisi, dll. Sedang behaviour adalah kelakuan dari objek tersebut, seperti mobil dapat melaju, membelok, membunyikan klakson, dll. Objek menyimpan statenya dalam satu atau lebih variabel, dan mengimplementasikan behaviournya dengan metode. Dengan penjelasan di atas, dapat disimpulkan bahwa objek adalah bagian software yang dibentuk dengan variabel-variabel dan metode-metode yang berhubungan dengan variabel tersebut. Dengan karakteristik tersebut, kita dapat memodelkan berbagai objek dalam kehidupan nyata ke dalam objek-objek dalam sebuah program. Lebih lanjut kita dapat memodelkan objek-objek abstrak ke dalam sebuah program. Contoh umum untuk konsep abstrak seperti ini adalah objek Event, yaitu objek untuk mewakili peristiwa klik atau tombol ditekan. Sebuah objek yang dibentuk dari sebuah kelas biasa disebut instans dalam terminologi OOP. Artinya objek tersebut adalah wujud nyata dari sebuah kelas. Variabel dan metode dari instans ini disebut variabel instans dan metode instans. Setiap instans menyimpan variabelnya sendiri-sendiri, jadi nilai variabel untuk tiap instans bisa berbeda.

2.3 Kelas

Kelas adalah semacam cetakan, atau template, untuk membuat objek. Ibaratkan sebuah rancangan rumah yang digunakan untuk membangun ratusan rumah. Rumah yang dibangun tersebut adalah objek dari kelas rancangan rumah. Hal ini dapat dilakukan karena semua objek rumah yang dibangun memiliki karakteristik yang sama, sehingga dapat dibuatkan semacam

blueprintnya. Tetapi objek-objek yang dibangun tetap akan memiliki bentuk fisik tertentu sendiri-sendiri, seperti variabel dalam sebuah program, atau pintu sebuah objek rumah. Dengan penjelasan ini, kelas dapat kita definisikan kembali menjadi sebuah blueprint, atau prototipe, yang mendefinisikan variabel dan metode yang sama untuk semua objek sejenis. Sebagai contoh, misalkan kita ingin membuat kelas Rumah, maka kita harus membuat sebuah kelas yang mendefinisikan semua variabel yang dimiliki objek dari kelas tersebut. Selain itu, kelas tersebut juga harus mendeklarasikan metode-metode yang dimiliki objek dari kelas dan juga membuat implementasi dari metode tersebut. Dengan adanya kelas ini, kita dapat membuat sebanyak apapun objek-objek rumah yang sejenis, yaitu jenis yang didefinisikan oleh kelas Rumah. Setiap objek Rumah diciptakan, sistem akan mengalokasikan sejumlah memori untuk objek tersebut dan variabel-variabelnya. Dengan begitu setiap objek akan memiliki salinan masing-masing untuk setiap variabel instans. Variabel kelas sebenarnya sama dengan variabel instans. Bedanya adalah, setiap objek berbagi satu dan hanya satu variabel kelas, tapi masing-masing memiliki salinan dari variable instans. Misalkan kelas Rumah yang kita buat hanya akan mendukung 2 lantai, dan setiap objek Rumah terdiri atas 2 lantai. Maka informasi ini cukup disimpan satu kali, karena nilainya tidak berbeda untuk semua objek. Lebih jauh, bila ada satu objek yang mengubah nilai dari variabel kelas, maka semua objek sejenis lainnya akan mengikuti perubahan itu. Di samping variabel, terdapat juga metode kelas. Metode jenis ini dapat langsung dipanggil melalui kelas dan bukan dari instans kelas tersebut.

2.4 Method

Method merupakan suatu operasi berupa fungsi-fungsi yang dapat dikerjakan oleh suatu object. Method didefinisikan pada class akan dipanggil melalui object. Dengan kata lain method juga dikenal sebagai fungsi atau prosedur. Pemanggilan method dalam suatu kelas dilakukan dengan menuliskan objek

pemiliknya dan diikuti oleh operator titik (.) beserta nama method yang akan di eksekusi.

2.5 Pewarisan Sifat Objek (Inheritance)

Pewarisan adalah salah satu ciri pemograman berorientasi objek, yang menyatakan suatu kelas dapat diturunkan lagi menjadi kelas-kelas baru yang lainnya sehingga dapat membentuk sebuah hiraki. Kelas yang merupakan kelas turunan ini bisa disebut dengan kelas anak (subclass) dan kelas yang menjadi dasar penurunan disebut kelas orang tua (superclass).

a. Dasar Pewarisan (Superclass dan Subclass) Pemograman java mengizinkan kita untuk mendefinisikan suatu kelas yang bersifat generic. Selanjutnya kelas tersebut diturunkan lagi menjadi kelas baru yang bersifat lebih spesifik. Adapun kelas baru yang hasil turunan disebut subclass. Pada proses penurunan kelas ini kelas turunan akan mewarisi sifat-sifat yang terdapat pada kelas induknya, selanjutnya, kelas turunan tersebut dapat memiliki sifat-sifat spesifik yang sebelumnya tidak dimiliki oleh kelas induk. Sebagai contoh, yakni terdapat kelas binatang, kelas tersebut selanjutnya akan diturunkan lagi menjadi kelas-kelas baru yang lebih spesifik tanpa meninggalkan sifat-sifat dari kelas binatang, menjadi kelas: Herbivora (binatang pemakan tumbuhan) dan Karnivora (binatang pemakan tumbuhan). Kelas turunan yang lain akan diturunkan lagi menjadi kelas turunan yang lainnya seperti kelas herbivora bisa diturunkan lagi menjadi kelas Kambing dan Kerbau, sedangkan untuk kelas Karnivora dapat diturunkan lagi menjadi Anjing, Harimau, Singa dsb.( ) kelas binatang adalah superclass dari kelas herbivoran dan karnivora itu sendiri berperan sebagai subclass. kemudian dari superclass tersebut terdapat subclass nya yakni pada superclass Herbivora dapat menjadi Subclass dari kelas Kambinga dan Kerbau. Begitupun pada superclass Karnivora dapat menjadi subclass dari kelas Harimau dan Kucing. Pada pemograman java menyediakan kata kunci extends yang

digunakan untuk proses penurunan terhadap suatu kelas. Bentuk umum dari penggunaan kata kunci tersebut adalah sebagai berikut: Class nama-subclass extends nama-superclass{ // badan kelas } Ketika kita sedang membentuk sebuah hiraki kelas constructor yang akan dipanggil pertama kali dalam setiap pembentukan objek adalah constructor dari kelas induk paling dasar. Sebagai contoh kita memiliki induk dengan nama A dan diturunkan lagi menjadi kelas C, pada situasi saat ini setiap kita membentuk objek dari kelas C, maka urutan yang dipanggil adalah contructor dari kelas A disusul dengan contructor dari kelas B, baru kemudian contructor dari kelas c. Pada saat kita membentuk atau melakukan instansiasi terhadap suatu objek turunan, maka sebelumnya java juga akan membentuk objek dari kelas induk-induk nya.

b. Pengertian Polimorfisme Polimorfisme adalah kemampuan suatu objek untuk mengungkap banyak hal melalui satu cara yang sama. Polimorfisme merupakan salah satu hal esensial dalam pemograman berorientasi objek karena alasan beriku: yakni polimorfisme mengizinkan kelas induk untuk mendefinisikan sebuah method general (bersifat umum) untuk semua kelas turunannya. Polimorfisme dapat berupa overloading atau overriding. Overloading merupakan bentuk polimorfisme yaitu beberapa metode yang dapat memiliki nama yang sama dengan isi dan parameter yang berbeda didalam sebuah kelas. Biasanya eksekusi program akan langsung mengacu pada metode yang dipanggil sesuai dengan parameter
c. Overriding merupakan bentuk polimorfisme yaitu berupa metode pada kelas orang tua yang dapat di tulis ulang pada kode kelas anak dalam pewarisan (inherentance) dengan memiliki nama yang sama dan memiliki isi ataupun parameter yang sama atau berbeda.

BAB III

IMPLEMENTASI & PEMBAHASAN

3.1 Implementasi

Sebelum membuat frame, kita harus membuat kelas terlebih dahulu.

```
private void cmbjkActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    if(cmbjk.getSelectedItem().equals("Pilih Jenis Kelamin"))
    {}
    else if(cmbjk.getSelectedItem().equals("Pria"))
    {}
    else if(cmbjk.getSelectedItem().equals("Wanita"))
    {}
}
```

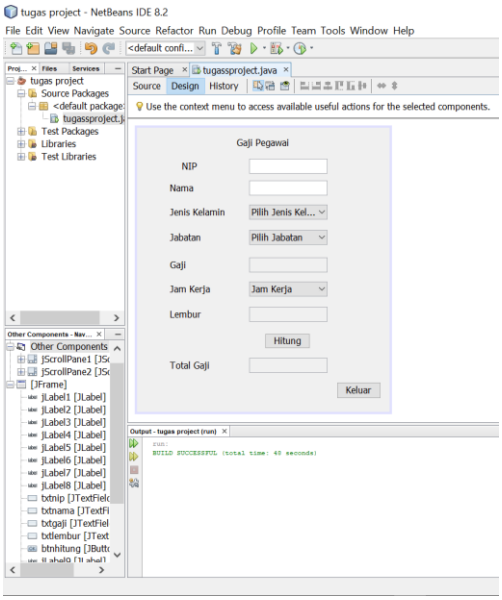
```
private void cmbjActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    if(cmbj.getSelectedItem().equals("Pilih Jabatan"))
    {
        txtgaji.setText(" ");
    }
    else if(cmbj.getSelectedItem().equals("Manager"))
    {
        txtgaji.setText("3000000");
    }
    else if(cmbj.getSelectedItem().equals("Customer Services"))
    {
        txtgaji.setText("2500000");
    }
    else if(cmbj.getSelectedItem().equals("Security"))
    {
        txtgaji.setText("1000000");
    }
}
```

```
private void cmbjamkerActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    if(cmbjamker.getSelectedItem().equals("Jam Kerja"))
    {
        txtlembur.setText(" ");
    }
    else if(cmbjamker.getSelectedItem().equals("1 jam"))
    {
        txtlembur.setText("200000");
    }
    else if(cmbjamker.getSelectedItem().equals("2 jam"))
    {
        txtlembur.setText("250000");
    }
    else if(cmbjamker.getSelectedItem().equals("3 jam"))
    {
        txtlembur.setText("300000");
    }
    else if(cmbjamker.getSelectedItem().equals("4 jam"))
    {
        txtlembur.setText("350000");
    }
    else if(cmbjamker.getSelectedItem().equals("5 jam"))
    {
```

```
private void btnhitungActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    int x = Integer.parseInt(txtgaji.getText());
    int y = Integer.parseInt(txtlembur.getText());
    int z = x+y;
    txttotal.setText(String.valueOf(z));
}
```

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

Selanjutnya membuat desain untuk frame Pegawai, fungsi yang dibutuhkan adalah 8 classjLabel, 5 jTextField, 3 jComboBox dan 2 komponen jButton.



Desain framenya :

3.2 Pembahasan

Bagian ini membahas tiap-tiap hasil implementasi untuk setiap fitur / fungsi atau komponen.

BAB IV

PENUTUP

4.1 Kesimpulan

Pada praktikum Pemrograman Berorientasi Objek ini saya menggunakan Netbeans 8.2 sebagai editor. Dari hasil pembelajaran, saya bisa memahami dan mengerti cara merancang tampilan form dari sebuah aplikasi dan mengerti fungsi dari source code masing-masing form dan menghubungkannya kedatabase.

4.2 Saran

Semoga teori dan praktikum Pemrograman Berorientasi Objek dapat ditingkatkan lagi. Agar mahasiswa dapat memahami sebuah aplikasi atau sistem informasi yang kemudian bisa diterapkan dalam kehidupan sebagai kebutuhan akan informasi.

DAFTAR PUSTAKA

https://www.academia.edu/11981246/Laporan_Praktikum_Java

https://ahmadtaufiq135.blogspot.com/2016/02/membuat-perhitungan-gaji-karyawan.html

https://www.academia.edu/8023993/APLIKASI_PERHITUNGAN_GAJI_PEGAWAI_DENGAN_JAVA_Disusun_untuk_memenuhi_tugas_matakuliah_Pemrograman_JavaIII