LAPORAN TUGAS BESAR
PEMROGRAMAN BERBASIS OBJEK
PROGRAM BANK

Disusun Oleh :

NAMA : SILVI NURCAHYANINGSIH

NIM : 32602200007

PROGRAM STUDI TEKNIK INFORMATIKA

FAKULTAS TEKNOLOGI INDUSTRI

UNIVERSITAS ISLAM SULTAN AGUNG

SEMARANG

2024

DAFTAR ISI

DAFTAR GAMBAR

BAB I
PENDAHULUAN

1.1 Latar Belakang

Perkembangan teknologi informasi dan kebutuhan masyarakat *modern menu*ntut adanya inovasi dalam sektor perbankan. Saat ini, perbankan menjadi salah satu sektor yang sangat terpengaruh oleh kemajuan teknologi. Penggunaan pemrograman berbasis objek dalam pengembangan sistem perbankan menjadi solusi efektif untuk meningkatkan kinerja, efisiensi, dan pengelolaan data.

Dalam era di mana teknologi informasi menjadi pilar utama kemajuan, sektor perbankan turut mengalami transformasi besar-besaran. Pemahaman konsep pemrograman berbasis objek menjadi kunci penting dalam mengembangkan sistem perbankan yang *responsif*, efisien, dan mudah dielola.

1.2 Tujuan

1. Untuk media pembelajaran mengenai penerapan konsep – konsep pada bahasa pemrograman *Java*
2. Untuk me*menu*hi tugas besar Pemrograman Berorientasi Objek pada Jurusan Teknik Informatika semester 3

1.3 Rumusan Masalah

1. Bagaimana penerapan bahasa pemrograman *java* pada program BANK?

1.4 Manfaat

1. Untuk mengetahui konsep pengaplikasian jumlah penarikan saldo serta saldo akhir.
2. Mempermudah *custumer* untuk mengetahui jumlah saldo akhir setelah melakukan proses penarikan dan mendapatkan bonus tambahan.

BAB II
STRUKTUR PROGRAM

2.1 Akun (Class *Interface*)

1. Kode program

```
package TUBES_BANK;
public interface akun {
String getAccountNumber();
String getAccountHolderName();
double getBalance();
void displayAccountInfo();
void withdraw(double amount);
void deposit(double amount);
}
```

2. Penjelasan Kode program

   Kelas akun merupakan kelas *interface* yang mendfefinisikan *method* yang harus diimplementasikan di setiap kelas yang mengimplementasi kelas akun ini. *Method* yang harus diimplementasikan antara lain: (Jeni, 2014)

   a. *getAccountNumber* untuk mengambil nomor rekening pengguna
   b. *getAccountHolderName* untuk mengambil nama asli pemilik akun
   c. *getBalance* untuk mengambil saldo.
   d. *displayAccountInfo* untuk mengambil informasi mengenai akun.
   e. *withdraw* untuk menarik saldo
   f. Deposit untuk deposito

2.2 Class Akun Bank

1. Kode program

```
package TUBES_BANK;
public class akun_bank implements akun{
private String accountNumber;
private String accountHolderName;
private double balance;

// Constructor
public     akun_bank(String     accountNumber,     String
```

```
accountHolderName, double initialBalance) {
this.accountNumber = accountNumber;
this.accountHolderName = accountHolderName;
this.balance = initialBalance;
}

// Getter and Setter
public String getAccountNumber() {
return accountNumber;
}
public String getAccountHolderName() {
return accountHolderName;
}

public double getBalance() {
return balance;
}

protected void setBalance(double balance) {
this.balance = balance;
}

// Interface methods
public void displayAccountInfo() {
System.out.println("Nomor       Rekening:       "       +
getAccountNumber());
System.out.println("Nama        Pemilik:        "        +
getAccountHolderName());
System.out.println("Saldo: " + getBalance());
}

public void withdraw(double amount) {
if (amount > 0 && amount <= getBalance()) {
setBalance(getBalance() - amount);
System.out.println("Penarikan berhasil. Saldo sekarang: "
+ getBalance());
} else {
System.out.println("Jumlah penarikan tidak valid atau saldo
```

```
tidak mencukupi.");
}
}

public void deposit(double amount) {
if (amount > 0) {
setBalance(getBalance() + amount);
System.out.println("Setor tunai berhasil. Saldo sekarang:
" + getBalance());
} else {
System.out.println("Jumlah setor tunai tidak valid.");
}
}
}
```

2. Penjelasan Kode program

   Kelas akun_bank adalah implementasi dari kelas *interface* akun. Ini berarti kelas ini harus mendefinisikan semua *method* yang dideklarasikan dalam *interface* akun. Berikut adalah penjelasan tentang bagian-bagian dari kelas akun_bank:(J.E.N.I, 2023)

   a. *Fields*: Kelas ini memiliki tiga *field private*: *accountNumber*, *accountHolderName*, dan *balance*. Ini adalah data yang disimpan oleh objek akun_bank.

   b. *Constructor*: *Constructor* ini digunakan untuk membuat objek AkunBank baru. Ini menerima tiga parameter: *accountNumber*, *accountHolderName*, dan *initialBalance*, yang digunakan untuk menginisialisasi *field – field* di atas.

   c. *Getter* and *Setter* : *Method* getAccountNumber, getAccountHolderName, dan getBalance adalah *getter* yang mengembalikan nilai dari *field – field* yang bersesuaian. *Method* setBalance adalah *setter* yang digunakan untuk mengubah nilai dari *field balance*. *Method* juga *setBalance* dideklarasikan sebagai *protected*. Ini berarti *method* ini hanya dapat diakses dari dalam kelas AkunBank dan kelas turunannya saja.

d. *displayAccountInfo*: *Method* ini mencetak informasi tentang akun yang berisi Nomor Rekening, Nama, dan saldo pemilik akun.
e. *withdraw*: *Method* ini digunakan untuk menarik uang dari akun. Terdapat validasi jika jumlah yang ingin ditarik valid dan cukup saldo, maka saldo akan dikurangi dan pesan sukses akan ditampilkan. Jika tidak, pesan *error* akan ditampilkan.
f. deposit: *Method* ini digunakan untuk menyetor uang ke akun. Jika jumlah yang ingin disetor valid, maka saldo akan ditambah dan pesan sukses akan dicetak. Jika tidak, pesan error akan dicetak.

2.3 Class Akun Bank *Premium*

1. Kode program

```
package TUBES_BANK;
public class Akun_Bank_Premium extends akun_bank {
public AkunBankPremium(String accountNumber, String
accountHolderName, double initialBalance) {
super(accountNumber, accountHolderName, initialBalance);
}

@Override
public void withdraw(double amount) {
// Akun premium dapat melakukan penarikan bahkan jika saldo
tidak mencukupi (sampai negatif)
setBalance(getBalance() - amount);
System.out.println("Penarikan berhasil. Saldo sekarang: "
+ getBalance());
}

@Override
public void deposit(double amount) {
// Akun premium dapat tambahan bonus 1% dari jumlah setoran
double bonus = amount * 0.01;
setBalance(getBalance() + amount + bonus);
System.out.println("Setoran berhasil. Bonus 1% telah
ditambahkan. Saldo sekarang: " + getBalance());
}
```

```
}
```

2. Penjelasan Kode program

   Kelas AkunBank*Premium* adalah kelas warisan dari AkunBank. Ini berarti AkunBank*Premium* mewarisi semua properti dan metode dari AkunBank, tetapi juga dapat menambahkan atau mengubah beberapa dari mereka. Berikut adalah penjelasan tentang bagian-bagian dari kelas AkunBank*Premium*:

   a. *Constructor*: *Constructor* ini digunakan untuk membuat objek AkunBank*Premium* baru. *Method constructor* ini menerima tiga parameter: accountNumber, accountHolderName, dan initialBalance, yang digunakan untuk menginisialisasi dari kelas induk. *Constructor* ini memanggil *constructor* dari induk (AkunBank) dengan syntax super.

   b. withdraw: *Method* ini di-override dari kelas induk. Dalam AkunBank*Premium,* pengguna dapat menarik uang bahkan jika saldo tidak mencukupi (saldo dapat menjadi negatif). Ini berbeda dari perilaku withdraw (tarik saldo) di AkunBank, dimana pengguna hanya dapat menarik uang jika saldo mencukupi.

   c. deposit: *Method* ini juga di-override dari kelas induknya. Dalam AkunBank*Premium,* pengguna mendapatkan bonus setoran sebesar 1% dari jumlah yang disetor. Ini berbeda dari perilaku deposit di AkunBank, di mana pengguna hanya menambahkan jumlah yang disetor ke saldo.

2.4 Class Main

1. Kode program

```
package TUBES_BANK;
import java.util.Scanner;
public class Main {
public static void main(String[] args) {
Scanner scanner = new Scanner(System.in);
AkunBank akunBank = new AkunBankPremium("11111111", "Silvi
Nurcahyaningsih", 100000);
```

```
while (true) {
System.out.println("\n=== Menu ===");
System.out.println("1. Tampilkan Info Akun");
System.out.println("2. Tarik Saldo");
System.out.println("3. Setor Tunai");
System.out.println("4. Keluar");
System.out.print("Pilih fitur: ");
int pilihan = scanner.nextInt();

switch (pilihan) {
case 1:
akunBank.displayAccountInfo();
break;
case 2:
System.out.print("Masukkan jumlah penarikan: ");
double penarikan = scanner.nextDouble();
akunBank.withdraw(penarikan);
break;
case 3:
System.out.print("Masukkan jumlah setoran: ");
double setoran = scanner.nextDouble();
akunBank.deposit(setoran);
break;
case 4:
System.out.println("Terima kasih telah menggunakan
aplikasi kami.");
System.exit(0);
return;
default:
System.out.println("Pilihan tidak valid. Silakan coba
lagi.");
}
}
}
}
```

2. Penjelasan Kode program

   Kelas Main merupakan kelas yang digunakan untuk menjalankan program dengan fungsionalitas yang telah dibuat. Di dalam kelas Main terdapat fungsi `main()` yang merupakan *method* utama yang dijalankan pertama kali pada saat menjalankan program *Java*. (Library, n.d.)

   a. Di awal program akan import *Scanner* yang digunakan untuk menerima *input* dari user.
   b. Kemudian program akan membuat objek AkunBank dengan membuat objek AkunBank*Premium* serta parameter yang diperlukan oleh objeknya.
   c. Selanjutnya akan masuk ke loop While yang akan menampilkan *menu* fitur.
   d. Pengguna akan memilih fitur yang akan digunakan:
      1) Jika pengguna memilih 1, menampilkan informasi akun.
      2) Jika pengguna memilih 2, melakukan penarikan dan meminta pengguna untuk memasukkan nominal yang akan ditarik
      3) Jika pengguna memilih 3, Melakukan setoran dan meminta pengguna untuk memasukkan nominal yang akan disetor.
      4) Jika pengguna memilih 4, menampilkan pesan penutupan dan menghentikan program.
      5) Jika pilihan pengguna tidak valid, menampilkan pesan error.

Program ini menggunakan beberapa konsep dasar dari Pemrograman Berorientasi Objek (OOP):(Avestro et al., 2007)

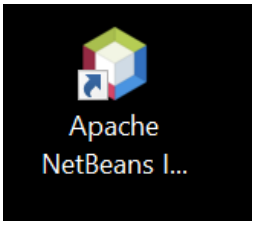
1. *Encapsulation*: *Encapsulation* adalah konsep menyembunyikan detail internal dari suatu objek dan hanya mengekspos apa yang diperlukan. Dalam program ini, detail internal dari kelas AkunBank dan AkunBank*Premium* (seperti accountNumber, accountHolderName, dan *balance*) dienkapsulasi dan hanya dapat diakses melalui metode yang disediakan (seperti displayAccountInfo, withdraw, dan deposit).

2. *Inheritance* (Pewarisan): *Inheritance* adalah konsep di mana kelas dapat mewarisi properti dan metode dari kelas lain. Dalam program ini, AkunBank*Premium* adalah subclass dari AkunBank, yang berarti AkunBank*Premium* mewarisi semua properti dan metode dari AkunBank.
3. *Polymorphism* (Polimorfisme): *Polymorphism* adalah konsep di mana suatu objek dapat mengambil banyak bentuk. Dalam program ini, objek akunBank adalah instance dari AkunBank*Premium,* tetapi diperlakukan sebagai AkunBank. Ini berarti bahwa metode yang dipanggil pada akunBank akan menjadi metode yang didefinisikan dalam AkunBank*Premium* jika ada, atau metode dari AkunBank jika tidak.
4. *Interface*: *Interface* merupakan sebuah ‘blueprint’ yang digunakan untuk mendefinisikan kontrak atau perilaku yang harus diimplementasikan oleh kelas yang mengimplementasikan *interface* tersebut (kelas turunannya). Dalam program tersebut, kelas *interface* diterapkan pada kelas Akun. Kelas-kelas turunan dari kelas Akun harus mengimplementasikan *method-method* yang ada pada kelas *Interface* Akun
5. *Getter* and *Setter*: Getter dan setter adalah konsep yang digunakan dalam OOP untuk mengendalikan akses ke variabel objek. Dalam program ini, *getter* dan *setter* digunakan untuk variabel *balance*:
    a. *Getter*: Metode get*Balance* adalah getter untuk variabel *balance*. Ini mengembalikan nilai dari variabel *balance*. *Getter* digunakan untuk membaca nilai dari variabel.
    b. *Setter*: Metode set*Balance* adalah setter untuk variabel *balance*. Ini mengatur nilai dari variabel *balance* ke nilai yang diberikan sebagai parameter. *Setter* digunakan untuk mengubah nilai dari variabel.

BAB III
IMPLEMENTASI PROGRAM

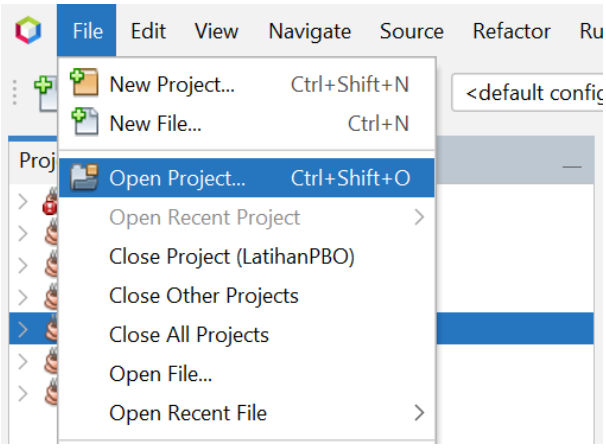
3.1 Menjalankan Program

1. Untuk menjalankan program, pertama klik ikon aplikasi natbeans pada layar desktop.



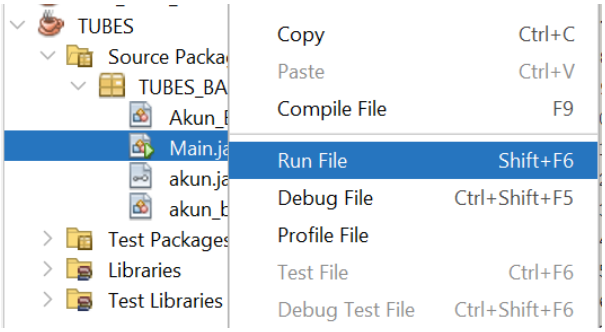
Gambar 3. 1 Aplikasi NeatBeans

2. Setelah itu arahkan ke kiri pada bagian *file*, lalu klik *open project*, klik *project* yang akan dijankan yaitu “TUBES”

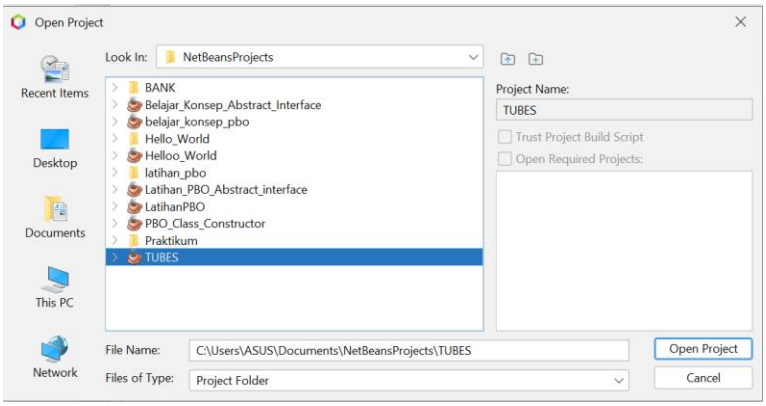


Gambar 3. 2 *Open Projects*

3. Klik kanan pada class pemanggil yaitu main.java, pilih run *file* atau shift + f6, untuk mengetahui hasil *output* dari program

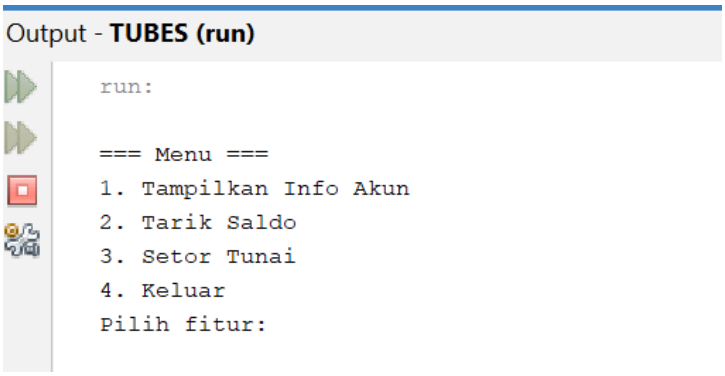


Gambar 3. 3 *Open Project*



Gambar 3. 4 Neatbeans Project

4. Program akan muncul dengan tampilan seperti dibawah ini



Gambar 3. 5 Tampilan *Output*

3.2 Mengoperasikan Program

1. *Input*kan angka sesuai dengan *menu*
   a. 1 untuk Tampilkan Info Akun

```
run:

=== Menu ===
1. Tampilkan Info Akun
2. Tarik Saldo
3. Setor Tunai
4. Keluar
Pilih fitur: 1
Nomor Rekening: 11111111
Nama Pemilik: Silvi Nurcahyaningsih
Saldo: 100000.0
```

Gambar 3. 6 *Output Menu* 1 Menampilkan Info Akun

b. 2 untuk Tarik Saldo

```
=== Menu ===
1. Tampilkan Info Akun
2. Tarik Saldo
3. Setor Tunai
4. Keluar
Pilih fitur: 2
Masukkan jumlah penarikan: 12000
Penarikan berhasil. Saldo sekarang: 88000.0
```

Gambar 3. 7 *Output Menu* 2 Tarik Saldo

c. 3 untuk Setor Tunai

```
=== Menu ===
1. Tampilkan Info Akun
2. Tarik Saldo
3. Setor Tunai
4. Keluar
Pilih fitur: 3
Masukkan jumlah setoran: 2000000
Setoran berhasil. Bonus 1% telah ditambahkan. Saldo sekarang: 2108000.0
```

Gambar 3. 8 *Output Menu* 3 Setor Tunai

d. 4 untuk Keluar

```
=== Menu ===
1. Tampilkan Info Akun
2. Tarik Saldo
3. Setor Tunai
4. Keluar
Pilih fitur: 4
Terima kasih telah menggunakan aplikasi kami.
BUILD SUCCESSFUL (total time: 2 minutes 49 seconds)
```

Gambar 3. 9 *Output Menu* 4 Keluar

BAB IV
PENUTUP

4.1 Kesimpulan

Dalam merampungkan tugas besar pemrograman berbasis objek mengenai program bank sederhana, kami dapat menyimpulkan bahwa proyek ini memberikan kontribusi positif terhadap pemahaman dan keterampilan peserta dalam menerapkan konsep pemrograman berbasis objek dalam konteks dunia perbankan.

Melalui tugas besar ini, peserta telah berhasil membuktikan kemampuan mereka dalam menggabungkan konsep OOP dengan kebutuhan sederhana dunia perbankan, memberikan dasar yang kuat untuk pertumbuhan mereka sebagai pengembang perangkat lunak.

DAFTAR PUSTAKA

Avestro, J., Hutcherson, R., Thompson, M., Thamura, F., Sari, D. M., & Rizzatama, N. S. (2007). *Pengenalan Pemrograman 2*.

J.E.N.I. (2023). BAB 11 Pewarisan , Polimorfisme , dan Interface. *J.E.N.I*, 1–16.

Jeni. (2014). *Bekerja dengan Java Class Library*. 1–21.

Library, J. C. (n.d.). *BAB 10 Membuat Class Sendiri □□□□ Mendefinisikan Class Anda*. 1–24.