**LAPORAN PRAKTIKUM**

**TEKNIK PEMROGRAMAN**

**INTERAKSI CLASS**



DOSEN PEMBIMBING :

Zulkifli Arsyad, S. kom. M.T

IHSAN FAUZI

241524048

1B – D4

**LABORATORIUM TEKNIK INFORMATIKA**

**FAKULTAS TEKNIK**

**POLITEKNIK NEGERI BANDUNG**

**2024**

# DAFTAR ISI

[DAFTAR ISI i](#_Toc191195714)

[BAB I PENDAHULUAN 1](#_Toc191195715)

[BAB II SOURCE CODE 2](#_Toc191195716)

[A. Kelas IDCard 2](#_Toc191195717)

[B. Kelas MachineATM 2](#_Toc191195718)

[C. Kelas Main 4](#_Toc191195719)

[BAB III PEMBAHASAN 5](#_Toc191195720)

[BAB IV HASIL IMPLEMENTASI 7](#_Toc191195721)

[BAB V KESIMPULAN & LESSON LEARN 10](#_Toc191195722)

# BAB I PENDAHULUAN

Program ini merupakan simulasi sederhana dari sistem ATM yang menggunakan dua kelas utama: IDCard sebagai representasi kartu ATM dan MachineATM sebagai mesin ATM yang menangani transaksi. Program ini dibuat menggunakan bahasa pemrograman Java dengan menerapkan prinsip enkapsulasi dan relationship "use-a".

# BAB II SOURCE CODE

## Kelas IDCard

*public class* IDCard {  
 *private* String IDNumber;  
 *private* String Pin;  
 *private double* Balance;  
  
 *public* IDCard(String IDNumber, String Pin, *double* Balance) {  
 *this*.IDNumber = IDNumber;  
 *this*.Balance = Balance;  
 *this*.Pin = Pin;  
 }  
  
 *public double* getBalance() {  
 *return* Balance;  
 }  
  
 *public void* setBalance(*double* amount) {  
 *this*.Balance = amount;  
 }  
  
 *public* String getIDNumber() {  
 *return* IDNumber;  
 }  
  
 *public* String getPIN() {  
 *return* Pin;  
 }  
}

## Kelas MachineATM

*import* java.util.Scanner;  
  
*public class* MachineATM {  
 *private* IDCard ID;  
 *public void* insertCard(IDCard ID) {  
 *this*.ID = ID;  
 }  
  
 *public void* start() {  
 Scanner **scanner** = *new* Scanner(System.in);  
  
 *if* (ID == *null*) {  
 System.out.println("Tidak ada kartu yang dimasukkan!");  
 *return*;  
 }  
  
 System.out.println("Selamat datang di ATM!");  
  
 // Looping untuk cek IDNumber tanpa batas  
 *while* (*true*) {  
 System.out.print("Masukkan ID Number: ");  
 String **inputID** = **scanner**.next();  
  
 *if* (**inputID**.equals(ID.getIDNumber())) {  
 System.out.println("ID benar! Masukkan PIN.");  
 *break*; // Jika ID benar, lanjut ke validasi PIN  
 } *else* {  
 System.out.println("ID salah! Silakan coba lagi.");  
 }  
 }  
  
 // PIN hanya boleh dicoba 3 kali  
 *int* **attempts** = 0;  
 *final int* **MAX\_ATTEMPTS** = 3;  
 *while* (**attempts** < **MAX\_ATTEMPTS**) {  
 System.out.print("Masukkan PIN: ");  
 String **inputPIN** = **scanner**.next();  
  
 *if* (**inputPIN**.equals(ID.getPIN())) {  
 System.out.println("Login berhasil!");  
 showMenu(**scanner**);  
 *return*;  
 } *else* {  
 **attempts**++;  
 System.out.println("PIN salah! Percobaan ke-" + **attempts**);  
 }  
 }  
  
 System.out.println("Terlalu banyak percobaan salah. Kartu Anda diblokir.");  
 }  
  
 *private void* showMenu(Scanner scanner) {  
 *int* **choice** = -1;  
 *while* (**choice** != 4) {  
 System.out.println("\nATM Menu:");  
 System.out.println("1. Cek Saldo");  
 System.out.println("2. Deposit");  
 System.out.println("3. Tarik Tunai");  
 System.out.println("4. Keluar");  
 System.out.print("Pilih menu: ");  
 **choice** = scanner.nextInt();  
 scanner.nextLine();  
  
 *switch* (**choice**) {  
 *case* 1:  
 System.out.println("Saldo Anda: " + ID.getBalance());  
 *break*;  
 *case* 2:  
 System.out.print("Masukkan jumlah deposit: ");  
 *double* **depositAmount** = scanner.nextDouble();  
 deposit(**depositAmount**);  
 scanner.nextLine();  
 *break*;  
 *case* 3:  
 System.out.println("Jumlah uang yang bisa ditarik adalah : " + ID.getBalance());  
 System.out.print("Masukkan jumlah penarikan: ");  
 *double* **withdrawAmount** = scanner.nextDouble();  
 withdraw(**withdrawAmount**);  
 scanner.nextLine();  
 *break*;  
 *case* 4:  
 System.out.println("Terima kasih telah menggunakan ATM.");  
 System.*exit*(0);  
 *break*;  
 *default*:  
 System.out.println("Pilihan tidak valid.");  
 }  
 System.out.println("Tekan Enter untuk melanjutkan...");  
 scanner.nextLine();  
 }  
 }  
  
 *public void* deposit(*double* amount) {  
 *if* (amount > 0) {  
 *double* **newBalance** = ID.getBalance() + amount;  
 ID.setBalance(**newBalance**);  
 System.out.println("Deposit berhasil: " + amount);  
 System.out.println("saldo anda sekarang : " + ID.getBalance());  
 } *else* {  
 System.out.println("Jumlah deposit tidak valid!");  
 }  
 }  
  
 *public void* withdraw(*double* amount) {  
 *if* (amount > 0 && amount <= ID.getBalance()) {  
 *double* **newBalance** = ID.getBalance() - amount;  
 ID.setBalance(**newBalance**);  
 System.out.println("Penarikan berhasil: " + amount);  
 System.out.println("Saldo anda sekarang : " + ID.getBalance());  
 } *else* {  
 System.out.println("Saldo tidak cukup atau jumlah tidak valid.");  
 }  
 }  
}

## Kelas Main

*public class* Main {  
 *public static void* main(String[] args) {  
 IDCard **myCard** = *new* IDCard("123456789", "1234", 100000);  
 MachineATM **myATM** = *new* MachineATM();  
  
 **myATM**.insertCard(**myCard**); // 🔵 Masukkan kartu  
 **myATM**.start(); // 🔵 Mulai ATM  
 }  
}

# BAB III PEMBAHASAN

+----------------+ +----------------+

| IDCard | | MachineATM |

+----------------+ +----------------+

| - IDNumber |<>---| - ID : IDCard |

| - Pin | +----------------+

| - Balance | | + start() |

+------------------+ +----------------+

| + getBalance() |

| + getIDNumber()|

| + getPIN() |

| + deposit() |

| + withdraw() |

+----------------+

a. **Interaksi Kelas**

* **MachineATM** memanggil metode dari **IDCard**:  
  ID.getBalance(); // Mengambil saldo  
  ID.setBalance(newBalance); // Mengatur saldo
* Di dalam kelas **MachineATM**, terdapat metode **insertCard(IDCard ID)** yang menerima objek **IDCard** sebagai parameter. Ini menunjukkan bahwa **MachineATM** berinteraksi dengan **IDCard** untuk melakukan validasi ID dan PIN sebelum mengizinkan transaksi.
* Setelah validasi, **MachineATM** memanggil metode **getBalance()**, **deposit()**, dan **withdraw()** dari **IDCard** untuk menangani transaksi keuangan. Contohnya, dalam metode **showMenu**, **ID.getBalance()** digunakan untuk menampilkan saldo, dan metode **deposit()** dan **withdraw()** memanggil metode **setBalance()** dari **IDCard**.

b. **Implementasi Relationship Dependence (Use-A)**

* Atribut **IDCard** dalam **MachineATM**:  
  private IDCard ID;
* MachineATM memiliki hubungan "use-a" dengan IDCard karena bergantung pada IDCard untuk mendapatkan informasi pengguna dan melakukan transaksi.
* Konstruktor MachineATM(IDCard ID) tidak ada dalam kode yang diberikan, tetapi jika ada, itu akan menunjukkan bahwa MachineATM memerlukan objek IDCard sebagai parameter untuk berfungsi. Dalam kode yang ada, hubungan ini terlihat pada metode insertCard(IDCard ID) yang mengatur objek IDCard yang dimasukkan ke dalam MachineATM.

c. **Enkapsulasi pada Atribut**

* Atribut dalam IDCard (IDNumber, Pin, dan Balance) dibuat private agar tidak bisa diakses langsung dari luar kelas.

private String IDNumber;  
private String Pin;  
private double Balance;

* Akses terhadap atribut tersebut hanya dapat dilakukan melalui metode getter dan setter. Misalnya, metode **getBalance()**, **getIDNumber()**, **getPIN()**, dan **setBalance(double amount)** digunakan untuk mengakses dan memodifikasi nilai atribut tersebut, menjaga keamanan dan integritas data.

# BAB IV HASIL IMPLEMENTASI

input



output



input



output

A screenshot of a computer screen

AI-generated content may be incorrect.

Input 1

A screenshot of a computer screen

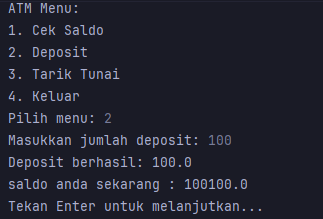
AI-generated content may be incorrect.

output

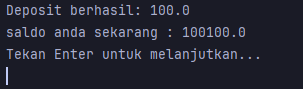
A screenshot of a computer

AI-generated content may be incorrect.

Input deposit 100



output



Input tarik uang 5000

A screen shot of a computer

AI-generated content may be incorrect.

output



Input 4/exitprogram

A screenshot of a computer screen

AI-generated content may be incorrect.

output

A screen shot of a computer

AI-generated content may be incorrect.

# BAB V KESIMPULAN & LESSON LEARN

* 1. Kesimpulan

1. Program ini menunjukkan bagaimana ATM berinteraksi dengan kartu ATM (IDCard).
2. ATM melakukan validasi ID dan PIN sebelum mengizinkan transaksi.
3. Enkapsulasi digunakan untuk melindungi atribut IDCard agar tidak bisa diakses langsung.
4. Relationship "use-a" diterapkan di mana MachineATM bergantung pada IDCard.
5. Program berjalan sesuai dengan skenario transaksi ATM pada umumnya.
6. Implementasi ini memberikan gambaran sederhana tentang pemrograman berbasis objek di Java dengan prinsip OOP seperti enkapsulasi dan dependensi.
   1. Lesson Learn
7. Menggunakan enkapsulasi meningkatkan keamanan data.
8. Dependensi antara kelas memungkinkan pemisahan tugas yang lebih baik.
9. Validasi input penting untuk mencegah kesalahan penggunaan.
10. OOP membantu memahami hubungan antar kelas dengan lebih baik.