

TUGAS PRAKTIKUM 3 INHERITANCE

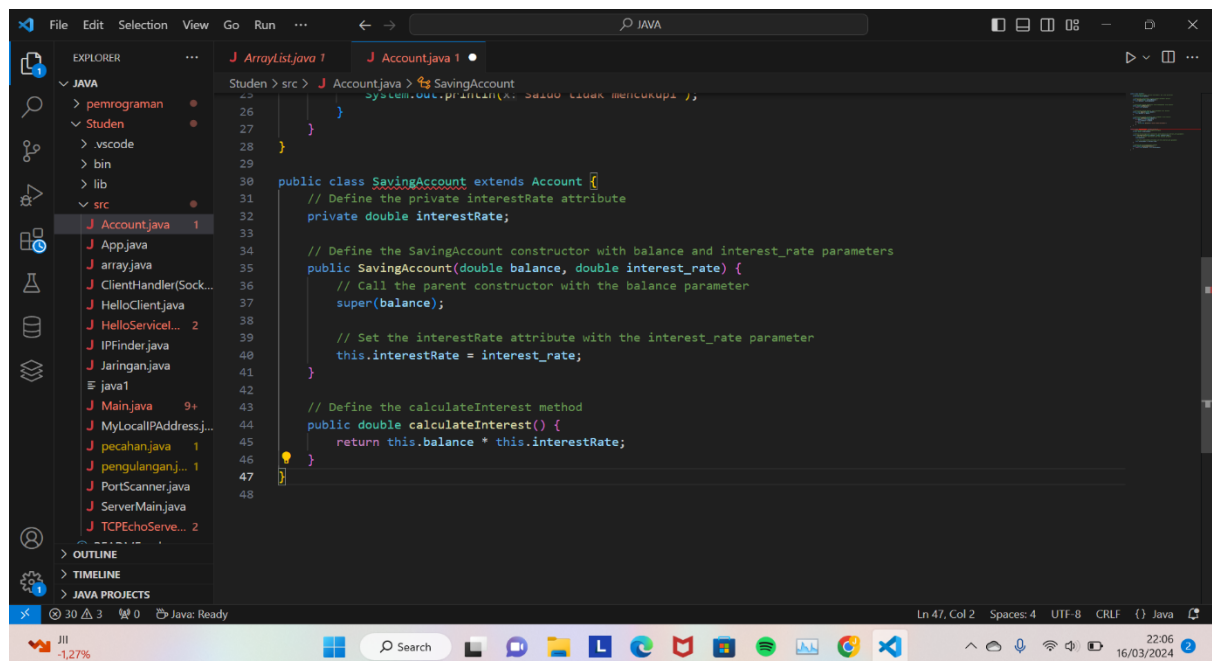


NAMA:JUNIAR AKHSAN

KELAS:TI22C

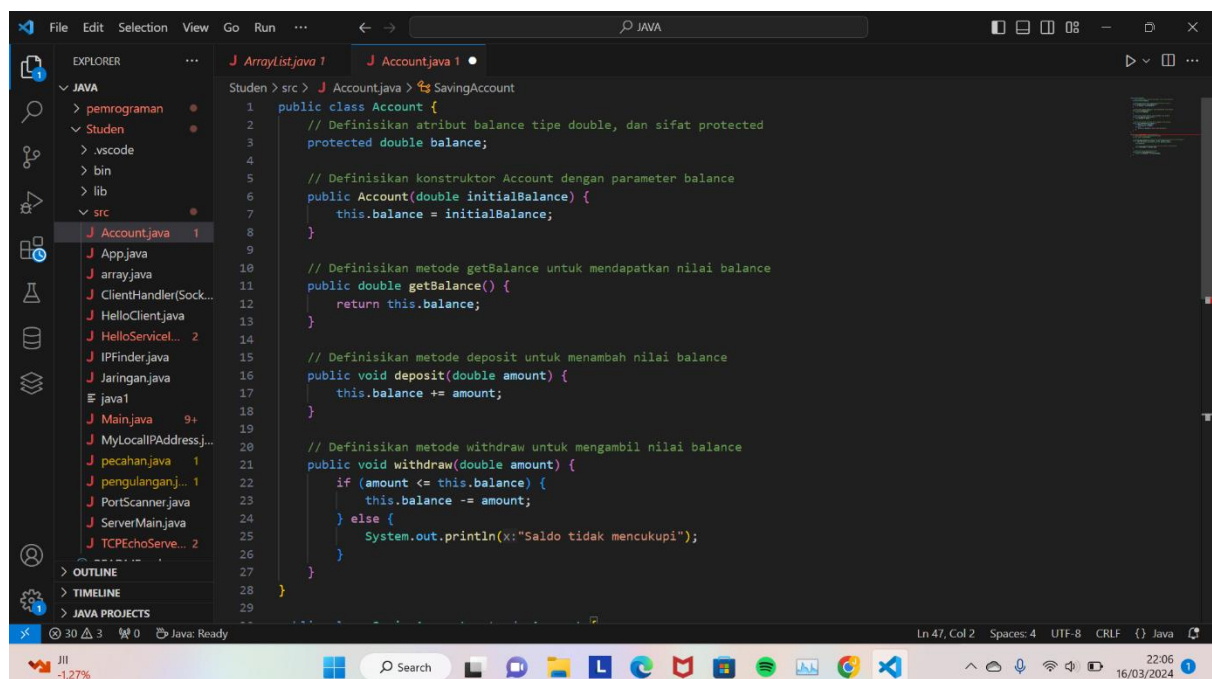
NIM:20220040085

1. ANALISA PERCOBAAN



The screenshot shows the VS Code editor with the 'SavingAccount' class implemented. The Explorer panel on the left shows the project structure with 'Account.java' selected. The main editor displays the following code:

```
26 }
27 }
28 }
29 }
30
31 public class SavingAccount extends Account {
32     // Define the private interestRate attribute
33     private double interestRate;
34
35     // Define the SavingAccount constructor with balance and interest_rate parameters
36     public SavingAccount(double balance, double interest_rate) {
37         // Call the parent constructor with the balance parameter
38         super(balance);
39
40         // Set the interestRate attribute with the interest_rate parameter
41         this.interestRate = interest_rate;
42     }
43
44     // Define the calculateInterest method
45     public double calculateInterest() {
46         return this.balance * this.interestRate;
47     }
48 }
```



The screenshot shows the VS Code editor with the 'Account' class implemented. The Explorer panel on the left shows the project structure with 'Account.java' selected. The main editor displays the following code:

```
1 public class Account {
2     // Definisikan atribut balance tipe double, dan sifat protected
3     protected double balance;
4
5     // Definisikan konstruktor Account dengan parameter balance
6     public Account(double initialBalance) {
7         this.balance = initialBalance;
8     }
9
10    // Definisikan metode getBalance untuk mendapatkan nilai balance
11    public double getBalance() {
12        return this.balance;
13    }
14
15    // Definisikan metode deposit untuk menambah nilai balance
16    public void deposit(double amount) {
17        this.balance += amount;
18    }
19
20    // Definisikan metode withdraw untuk mengambil nilai balance
21    public void withdraw(double amount) {
22        if (amount <= this.balance) {
23            this.balance -= amount;
24        } else {
25            System.out.println(x:"Saldo tidak mencukupi");
26        }
27    }
28 }
29 }
```

Berikut adalah analisa setiap percobaan dalam kodingan Account class di atas:

Atribut balance: Atribut ini bertipe double dan memiliki sifat protected. Hal ini berarti bahwa atribut ini dapat diakses oleh kelas yang sama dan kelas turunannya. Atribut ini digunakan untuk menyimpan saldo akun.

Konstruktor Account: Konstruktor ini memiliki parameter initialBalance yang digunakan untuk memberi nilai awal atribut balance. Konstruktor ini dipanggil saat membuat objek baru dari kelas Account.

Metode getBalance: Metode ini digunakan untuk mendapatkan nilai dari atribut balance. Metode ini hanya membalikkan nilai atribut balance dan tidak melakukan apa-apa pengolahan lain.

Metode deposit: Metode ini digunakan untuk menambahkan jumlah amount ke atribut balance. Hal ini dilakukan dengan menggunakan operator +=.

Metode withdraw: Metode ini digunakan untuk mengurangi jumlah amount dari atribut balance. Namun, sebelum melakukan pengurangan, metode ini akan memeriksa apakah jumlah amount lebih kecil atau sama dengan nilai saat ini dari atribut balance. Jika amount lebih besar dari nilai saat ini dari atribut balance, maka akan ditampilkan pesan "Saldo tidak mencukupi"

Dalam kelas ini, atribut interestRate bertipe double dan memiliki sifat private. Konstruktor SavingAccount mengambil dua parameter, yaitu balance dan interest_rate. Parameter balance dipassing ke konstruktor Account dengan menggunakan super(balance), sehingga nilai awal dari atribut balance diinisialisasi. Nilai dari parameter interest_rate kemudian diset ke atribut interestRate.

2.

```
1 public class CheckingAccount extends Account {
2     private double overdraftProtection;
3
4     public CheckingAccount(double balance, double protect) {
5         super(balance + protect);
6         this.overdraftProtection = protect;
7     }
8
9     public CheckingAccount(double balance) {
10         this(balance, -1.0);
11     }
12
13     @Override
14     public boolean withdraw(double amount) {
15         double overdraftNeeded = amount - this.balance;
16         if (overdraftNeeded > 0) {
17             if (this.overdraftProtection < 0 || this.overdraftProtection < overdraftNeeded) {
18                 return false;
19             } else {
20                 this.balance = 0.0;
21                 this.overdraftProtection -= overdraftNeeded;
22                 return true;
23             }
24         } else {
25             if (this.balance < amount) {
26                 return false;
27             } else {
28                 this.balance -= amount;
29                 return true;
30             }
31         }
32     }
33 }
34
```

Kelas `CheckingAccount` dalam Java digunakan untuk merepresentasikan akun giro dalam sebuah program. Kelas ini biasanya mewarisi kelas `Account` yang lebih umum, yang menyediakan fungsionalitas dasar untuk akun bank.

Atribut Kelas `CheckingAccount`:

- `saldo`: Saldo akun giro.
- `biayaTransaksi`: Biaya per transaksi.
- `batasTransaksiHarian`: Batas maksimum transaksi per hari.

Metode Kelas `CheckingAccount`:

- `deposit(double amount)`: Menambahkan dana ke akun.
- `withdraw(double amount)`: Menarik dana dari akun.
- `transfer(Account toAccount, double amount)`: Mentransfer dana ke akun lain.
- `checkBalance()`: Mengecek saldo akun.
- `getMonthlyFee()`: Menghitung biaya bulanan.

Kelas `CheckingAccount` dalam Java menyediakan fungsionalitas untuk mensimulasikan akun giro dalam sebuah program. Kelas ini memungkinkan pengembang untuk melakukan operasi seperti deposit, penarikan, transfer, dan pengecekan saldo.

