LAPORAN TUGAS PEMROGRAMAN DASAR

LECTURE 12 : INHERITANCE PADA OBJECT ORIENTED PROGRAMMING C++

Dosen Pengampu : Warsun Najib, S.T., M.Sc.

Disusun oleh :

1. **Muhammad Akbar Rabbani (23/515815/TK/56739)**
2. **Polikarpus Arya Pradhanika (23/512404/TK/56325)**
3. **Zaidan Harith (23/512629/TK/56334)**

DEPARTEMEN TEKNIK ELEKTRO DAN TEKNOLOGI INFORMASI

FAKULTAS TEKNIK UNIVERSITAS GADJAH MADA

YOGYAKARTA, NOVEMBER 2023

A. Algoritma Program

Program ini merupakan aplikasi dari berbagai materi, seperti dasar pemrograman C++ dan Object Oriented Programming (OOP) dalam C++. Konsep OOP yang umum dipakai pada program ini adalah konsep Inheritance pada kelas. Pada program ini digunakan tiga kelas, yaitu "bankAccount" sebagai parent class, serta "checkingAccount" dan "savingsAccount" yang merupakan kelas turunan (child class) dari kelas "bankAccount".

1. Penjelasan kelas
   a. Kelas "bankAccount"

      Kelas yang berfungsi sebagai *parent class* atau kelas utama. Kelas ini digunakan untuk membuat *instance* dan *method* dasar yang bisa digunakan di kelas ini dan seluruh kelas turunannya, seperti Account Number dan Balance.

   b. Kelas "checkingAccount"

      Kelas yang berfungsi sebagai *child class* atau kelas turunan dari "bankAccount". Kelas ini bekerja seperti sistem penarikan saldo dalam bank. *User* dapat melakukan penarikan saldo, pengecekan saldo, menentukan besar bunga, dan mengecek Service Charge.

   c. Kelas "savingsAccount"

      Kelas yang berfungsi sebagai *child class* atau kelas turunan dari "bankAccount". Kelas ini bekerja seperti sistem penyimpanan uang. *User* dapat menyimpan uang pada bank. *User* dapat mengecek saldo setelah uang tersebut di simpan di sistem bank tersebut

2. Penjelasan algoritma program

   Dalam algoritma OOP ini, program dibagi menjadi tiga file utama, yaitu "account.h", "account.cpp" dan "app.cpp". Masing-masing file memiliki tugas dan kegunaannya masing-masing.

   File "account.h" merupakan file *header* yang menjadi file dasar bagi seluruh program. Dalam file ini, semua kelas dibuat, termasuk kelas utama (*parent class*) dan kelas turunan (*child class*). Untuk masing-masing kelas, akan diinisialisasi *instance* (variabel yang ada di dalam kelas) dan *method* (*function* yang ada di dalam kelas) beserta dengan visibilitasnya.

a. Inklusi Header

Meng-*include*-kan *header* yang mencakup sebagian besar pustaka C++ standar "#include <bits/stdc++.h>" dan mendeklarasi penggunaan *namespace* C++ standar.

b. Mendefinisikan kelas "bankAccount"

Kelas ini didefinisikan dengan "class bankAccount{}". Di dalam kelas ini, *instance* akan berada dalam visibilitas *protected*. Beberapa *instance* yang terdapat dalam kelas ini adalah "int accountNumber" dan "double balance".

Selain *instance*, kelas ini juga terdapat beberapa *method* yang memiliki visibilitas *public* yang dapat melakukan fungsi tertentu. *Method* yang dimiliki oleh kelas ini adalah "void setAccountNumber()", "int retrieveAccountNumber(), "double retrieveBalance()", "voidBalance()", "void depositMoney()", "double withdraw()", dan "void printAccountInfo()".

Selain *method* yang disebutkan tersebut, kelas ini memiliki *constructor method* yang berfungsi untuk mendeklarasikan sebuah objek dari kelas ini. *Constructor method* yang dimiliki kelas ini adalah bankAccount() yang memiliki dua parameter, yaitu "accountNumber" dan "balance".

c. Mendefinisikan kelas "checkingAccount"

Kelas ini didefinisikan dengan "class checkingAccount{}". Di dalam kelas ini, *instance* akan berada dalam visibilitas *protected*. Beberapa *instance* yang terdapat dalam kelas ini sudah diwakili oleh *parent class*. Namun terdapat beberapa tamabahan *instance* yang berlaku pada kelas ini yaitu "double interest", "double minimumBalance", dan "double serviceCharge".

Selain *instance*, kelas ini juga terdapat beberapa *method* yang memiliki visibilitas *public* yang dapat melakukan fungsi tertentu. *Method* yang dimiliki oleh kelas ini adalah "void setInterestRate()", "double retrieveInterestRate(), "void setMinimumBalance()", "double retrieveMinimumBalance()", "void setServiceCharge()", "double retrieveServiceCharge()", "double postInterest()", "bool balanceVerify()", "void writecheck()" dan "double withdraw()".

Selain *method* yang disebutkan tersebut, kelas ini memiliki *constructor method* yang berfungsi untuk mendeklarasikan sebuah objek dari kelas ini. *Constructor method* yang dimiliki kelas ini adalah checkingAccount() yang memiliki lima parameter, yaitu "accountNumber", "balance", "interest", "minimumBalance", "serviceCharge".

d. Mendefinisikan kelas “savingsAccount”

Kelas ini didefinisikan dengan “class savingsAccount{}”. Di dalam kelas ini, *instance* akan berada dalam visibilitas *protected*. Beberapa *instance* yang terdapat dalam kelas ini sudah diwakili oleh *parent class*. Namun terdapat beberapa tamabahan *instance* yang berlaku pada kelas ini yaitu “double interest”, “double minimumBalance”, dan “double serviceCharge”.

Selain *instance*, kelas ini juga terdapat beberapa *method* yang memiliki visibilitas *public* yang dapat melakukan fungsi tertentu. *Method* yang dimiliki oleh kelas ini adalah “double retrieveInterestRate(), “void deposit()”, “double withdraw()”, “void setInterestRate()”, dan “double postInterest()”.

Selain *method* yang disebutkan tersebut, kelas ini memiliki *constructor method* yang berfungsi untuk mendeklarasikan sebuah objek dari kelas ini. *Constructor method* yang dimiliki kelas ini adalah checkingAccount() yang memiliki lima parameter, yaitu “accountNumber”, “balance”, “interest”, “minimumBalance”, “serviceCharge”.

Selanjutnya, file “account.cpp” merupakan file yang berisi seluruh *statement* yang ada di dalam *function* yang telah dideklarasikan sebelumnya di file *header* “account.h”.

a. Implementasi fungsi anggota kelas “bankAccount”

1) Method “setAccountNumber()”

Method ini akan menerima satu parameter, yaitu “_accountNumber”. Method akan menugaskan *instance* accountNumber sebagai “_accountNumber”.

2) Method “retrieveAccountNumber()”

Method tidak memiliki parameter. Method akan menerima isi dari *instance* accountNumber.

3) Method “retrieveBalance()”

Method tidak memiliki parameter. Method akan mengembalikan isi dari *instance* balance.

4) Method “depositMoney()”

Method ini akan menerima satu parameter, yaitu “amount”. Method akan menugaskan *instance* balance sebagai penjumlahan antara balance sebelumnya dengan nilai dari amount.

5) Method “withdraw()”

Method ini akan menerima satu parameter, yaitu “amount”. Method akan mengembalikan jumlah uang yang telah ditarik dengan nominal sebesar amount.

6) Method “printAccountInfo()”

Method tidak memiliki parameter. Method akan menampilkan informasi kepada *user* berupa accountNumber dan balance.

b. Implementasi fungsi anggota kelas “checkingAccount”

1) Method “setInterestRate()”

Method ini akan menerima satu parameter, yaitu “_interest”. Method akan menugaskan *instance* interest sebagai “_interest”.

2) Method “retrieveInterestRate()”

Method tidak memiliki parameter. Method akan menerima isi dari *instance* interest.

3) Method “setMinimumBalance()”

Method ini akan menerima satu parameter, yaitu “_minimumBalance”. Method akan menugaskan *instance* minimumBalance sebagai “_minimumBalance”.

4) Method “retrieveMinimumBalance()”

Method tidak memiliki parameter. Method akan menerima isi dari *instance* minimumBalance.

5) Method “setServiceCharge()”

Method ini akan menerima satu parameter, yaitu “_serviceCharge”. Method akan menugaskan *instance* serviceCharge sebagai “_serviceCharge”.

6) Method “retrieveServiceCharge()”

Method tidak memiliki parameter. Method akan menerima isi dari *instance* serviceCharge.

7) Method “postInterest()”

Method ini tidak memiliki parameter. Method akan mengembalikan nominal dari bunga yang ada di *instance* interest.

8) Method “balanceVerify()”

Method ini tidak memiliki parameter. Method akan mengecek apakah nilai balance kurang dari minimumBalance. Jika nilai balance kurang dari nilai minimumBalance, method ini akan mengembalikan nilai boolean True dan begitupun sebaliknya.

9) Method “writeCheck():

Method ini tidak memiliki parameter. method ini akan menampilkan informasi kepada *user* berupa nilai dan nominal bunga serta nominal Service Charge apabila ada.

10) Method “withdraw()”

Method ini akan menerima satu parameter, yaitu “amount”. Method akan mengurangi nilai *instance* balance sebesar “amount”. Apabila nilai balance lebi besar dari “amount”, maka *instance* balance akan ditugaskan dengan nilai 0.

c. Implementasi fungsi anggota kelas “savingsAccount”

1) Method “retrieveInterestRate()”

Method tidak memiliki parameter. Method akan menerima isi dari *instance* interest.

2) Method “deposit():

Method ini akan menerima satu parameter berupa nominal uang yang akan disimpan sebesar “amount”. Method akan menambahkan nilai *instance* balance dengan “amount”.

3) Method “withdraw()”

Method ini akan menerima satu parameter, yaitu “amount”. Method akan mengurangi nilai *instance* balance sebesar “amount”. Apabila nilai balance lebi besar dari “amount”, maka *instance* balance akan ditugaskan dengan nilai 0.

4) Method “setInterestRate()”

Method ini akan menerima satu parameter, yaitu “amount”. Method akan menugaskan *instance* interest sebagai “amount”.

5) Method “postInterest()”

Method ini tidak memiliki parameter. Method akan mengembalikan nominal dari bunga yang ada di instance interest.

Lalu, file “app.cpp” adalah file yang akan dijalankan oleh *user*. Seluruh deklarasi *object* dari kelas-kelas yang dideklarasikan di file *header* “account.h” akan dibuat di file ini. *User* juga dapat mengakses seluruh isi kelas di file ini.

a. Fungsi

Dibuat sebuah fungsi untuk mendeklarasikan objek yang ada di dalam program ini, yaitu objek pertama dengan fungsi “testA()” dan objek kedua dengan fungsi “testB()”. Objek pertama dibuat sekaligus ditampilkan pada suatu fungsi yaitu fungsi “testA()”. Objek kedua dibuat sekaligus ditampilkan pada suatu fungsi yaitu “testB()”. Pada fungsi ini, akan dideklarasikan variabel–variabel yang telah dibuat di setiap kelasnya.

b. Tampilan Output

Menampilakan informasi “bankAccount”, “checkingAccount”, dan “savingsAccount” yang melibatkan variabel pada tiap kelasnya. Pada program ini akan menampilan keluaran atau Print Account Information, seperti Set Account Number, Retrieve Account Number, Retrieve Balance, Deposit dan Withdraw Money.

B. Compile dan Running Command

1. Build command: g++ -o app app.cpp account.cpp
2. Run command: app

C. Source Code

1. File “account.h”

```cpp
#ifndef __ACCOUNT_H__
#define __ACCOUNT_H__

#include <bits/stdc++.h>
using namespace std;

class bankAccount{
    protected:
        int accountNumber;
        double balance;

    public:
        bankAccount(int _accountNumber, double _balance);
        void setAccountNumber(int _accountNumber);
        int retrieveAccountNumber();
        double retrieveBalance();
        void setBalance(double amount);
        void depositMoney(double amount);
        double withdraw(double amount);
        void printAccountInfo();
};

class checkingAccount : public bankAccount{
    protected:
        double interest, minimumBalance, serviceCharge;

    public:
        checkingAccount(int _accountNumber, double _balance, double _interest, double _minimumBalance, double _serviceCharge);
        void setInterestRate(double _interest);
        double retrieveInterestRate();
        void setMinimumBalance(double _minimumBalance);
        double retrieveMinimumBalance();
        void setServiceCharge(double _serviceCharge);
        double retrieveServiceCharge();
        double postInterest();
        bool balanceVerify();
        void writeCheck();
        double withdraw(double amount);
};

class savingsAccount : public bankAccount{
    protected:
        double interest, minimumBalance, serviceCharge;

    public:
        savingsAccount(int _accountNumber, double _balance, double _interest, double _minimumBalance, double _serviceCharge);
        double retrieveInterestRate();
        void deposit(double amount);
        double withdraw(double amount);
        void setInterestRate(double amount);
        double postInterest();
};

#endif
```

2. File “account.cpp”

```cpp
#include "account.h"
#include <bits/stdc++.h>

using namespace std;

// BANK ACCOUNT CLASS
bankAccount::bankAccount(int _accountNumber, double _balance){
    accountNumber = _accountNumber;
    balance = _balance;
}

void bankAccount::setAccountNumber(int _accountNumber){
    accountNumber = _accountNumber;
}

int bankAccount::retrieveAccountNumber(){
    return accountNumber;
}

double bankAccount::retrieveBalance(){
    return balance;
}

void bankAccount::depositMoney(double amount){
    balance += amount;
}

double bankAccount::withdraw(double amount){
    balance = max((double)0, balance - amount);
    return min(balance, amount);
}

void bankAccount::printAccountInfo(){
    cout << "account number: " << accountNumber << '\n';
    cout << "balance: " << balance << '\n';
}


// CHECKING ACCOUNT

checkingAccount::checkingAccount(int _accountNumber, double _balance, double _interest, double _minimumBalance, double _serviceCharge)
    : bankAccount::bankAccount(_accountNumber, _balance){
    interest = _interest;
    minimumBalance = _minimumBalance;
    serviceCharge = _serviceCharge;
}

void checkingAccount::setInterestRate(double _interest){
    interest = _interest;
}

double checkingAccount::retrieveInterestRate(){
    return interest;
}

void checkingAccount::setMinimumBalance(double _minimumBalance){
    minimumBalance = _minimumBalance;
}

double checkingAccount::retrieveMinimumBalance(){
    return minimumBalance;
}

void checkingAccount::setServiceCharge(double _serviceCharge){
    serviceCharge = _serviceCharge;
}

double checkingAccount::retrieveServiceCharge(){
    return (balance < minimumBalance) ? serviceCharge : 0;
}

double checkingAccount::postInterest(){
    return (balance+balance*(interest/100));
}

bool checkingAccount::balanceVerify(){
    return ((balance<minimumBalance) ? true : false);
}

void checkingAccount::writeCheck(){
    cout << "CHECK INFORMATION" << endl;
    cout << "=========================================" << endl;
    cout << left << setw(25) << "Interest" << " : " << interest << "%" << endl;
    cout << left << setw(25) << "Post Interest" << " : " << postInterest() << endl;

    cout << left << setw(25) << "Service Charge" << " : ";
    if(balanceVerify()) cout << (minimumBalance-balance) << endl;
    else cout << 0 << endl;
}

double checkingAccount::withdraw(double amount){
    if(balance < amount){
        balance = 0;
        return balance;
    }
    else{
        balance -= amount;
        return amount;
    }
}

// SAVING ACCOUNT

savingsAccount::savingsAccount(int _accountNumber, double _balance, double _interest, double _minimumBalance, double _serviceCharge) :
bankAccount(_accountNumber, _balance){
    interest = _interest;
    minimumBalance = _minimumBalance;
    serviceCharge = _serviceCharge;
}

double savingsAccount::retrieveInterestRate(){
    return interest;
}

void savingsAccount::deposit(double amount){
    balance += amount;
}

double savingsAccount::withdraw(double amount){
    if(balance < amount){
        balance = 0;
        return balance;
    }
    else{
        balance -= amount;
        return amount;
    }
}

void savingsAccount::setInterestRate(double amount){
    interest = amount;
}

double savingsAccount::postInterest(){
    return balance + balance * (interest) / 100;
}

```

3. File “app.cpp”

```cpp
#include "account.h"
#include <bits/stdc++.h>
using namespace std;

void testA(){
    cout << "Simulating checkingAccount\n";
    cout << "=================================\n\n";
    checkingAccount A(100, 100, 100, 100, 100);
    A.printAccountInfo();
    A.setInterestRate(5);
    A.printAccountInfo();

    cout << endl;
    cout << "Interest Rate " << A.retrieveInterestRate() << endl;
    A.setMinimumBalance(50);
    cout << "Minimum Balance " << A.retrieveMinimumBalance() << endl;
    A.setServiceCharge(60);
    cout << "Service charge " << A.retrieveServiceCharge() << endl;
    cout << "Post Interest " << A.postInterest() << endl;
    cout << "Is balance verified? " << (A.balanceVerify() ? "Yes" : "No") << endl << endl;

    A.writeCheck();
    cout << endl;
    A.printAccountInfo();
    A.withdraw(75);
    A.printAccountInfo();

    cout << endl;
    cout << "Interest Rate " << A.retrieveInterestRate() << endl;
    A.setMinimumBalance(50);
    cout << "Minimum Balance " << A.retrieveMinimumBalance() << endl;
    A.setServiceCharge(60);
    cout << "Service charge " << A.retrieveServiceCharge() << endl;
    cout << "Post Interest " << A.postInterest() << endl << endl;
    cout << "Is balance verified? " << (A.balanceVerify() ? "Yes" : "No") << endl;
    cout << "\nDone\n\n";
}

void testB(){
    cout << "Simulating savingsAccount\n";
    cout << "=================================\n\n";
    savingsAccount B(100, 100, 100, 100, 100);
    B.printAccountInfo();
    B.setInterestRate(5);
    cout << endl;
    cout << "Interest Rate " << B.retrieveInterestRate() << endl;
    cout << "Post Interest " << B.postInterest() << endl << endl;
    B.withdraw(200);
    cout << "After withdrawal:\n";
    B.printAccountInfo();
    cout << endl;
    B.deposit(75);
    cout << "After deposit:\n";
    B.printAccountInfo();
    cout << "Post Interest " << B.postInterest() << endl;
    cout << "\nDone\n\n";
}

int main(){

    testA();
    testB();

    return 0;
}
```

D. Tangkapan Layar Hasil Running Program

```
Simulating checkingAccount
====================================

account number: 100
balance: 100
account number: 100
balance: 100

Interest Rate 5
Minimum Balance 50
Service charge 0
Post Interest 105
Is balance verified? No

CHECK INFORMATION
=======================================
======
Interest                  : 5%
Post Interest             : 105
Service Charge            : 0

account number: 100
balance: 100
account number: 100
balance: 25

Interest Rate 5
Minimum Balance 50
Service charge 60
Post Interest 26.25

Is balance verified? Yes

Done

Simulating savingsAccount
====================================

account number: 100
balance: 100

Interest Rate 5
Post Interest 105
After withdrawal:
account number: 100
balance: 0

After deposit:
account number: 100
balance: 75
Post Interest 78.75

Done
```