Praktikum inheritance

Nama : Agi Hanan Munawar

NIM : 20210040035

Kelas : TI21A

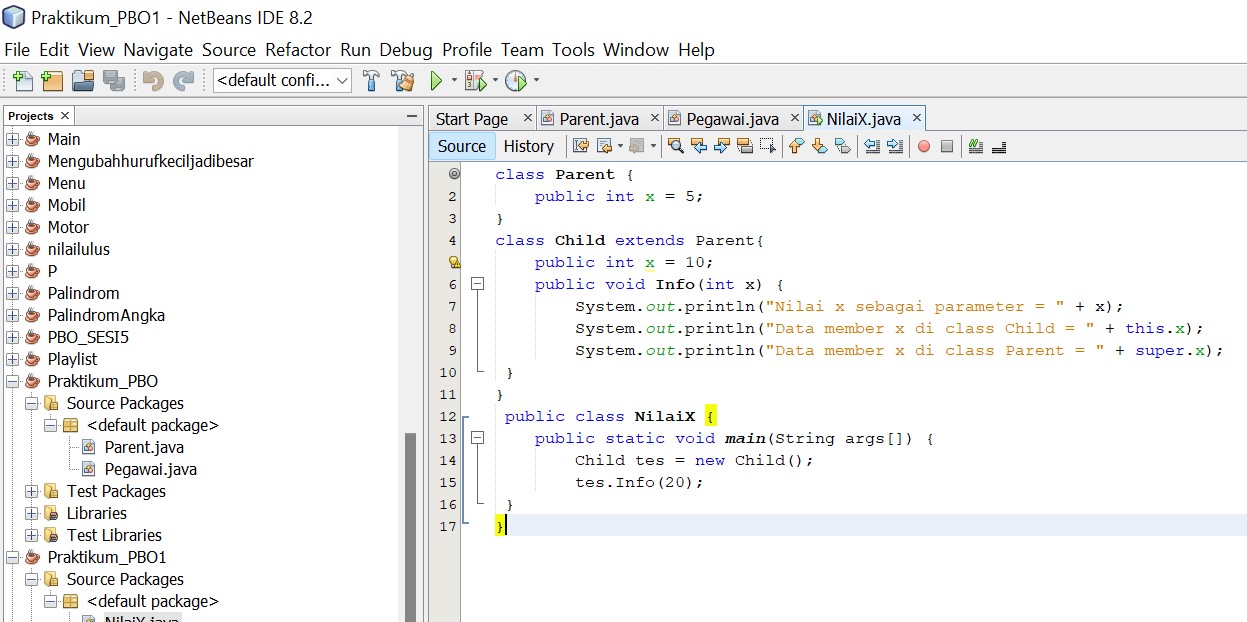
1. Kode Program upload ke github dengan nama repository praktikum-inheritance
2. Berikan analisa setiap percobaan dalam bentuk File teks pdf dan upload juga ke github praktikum-inheritance

JAWAB:

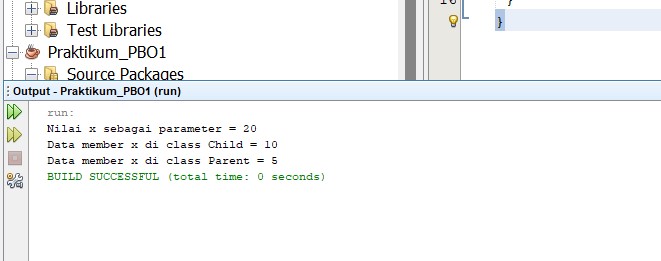
# Percobaan berikut ini menunjukkan penggunaan kata kunci “super”.

* + PERCOBAAN 1

INPUT



OUTPUT

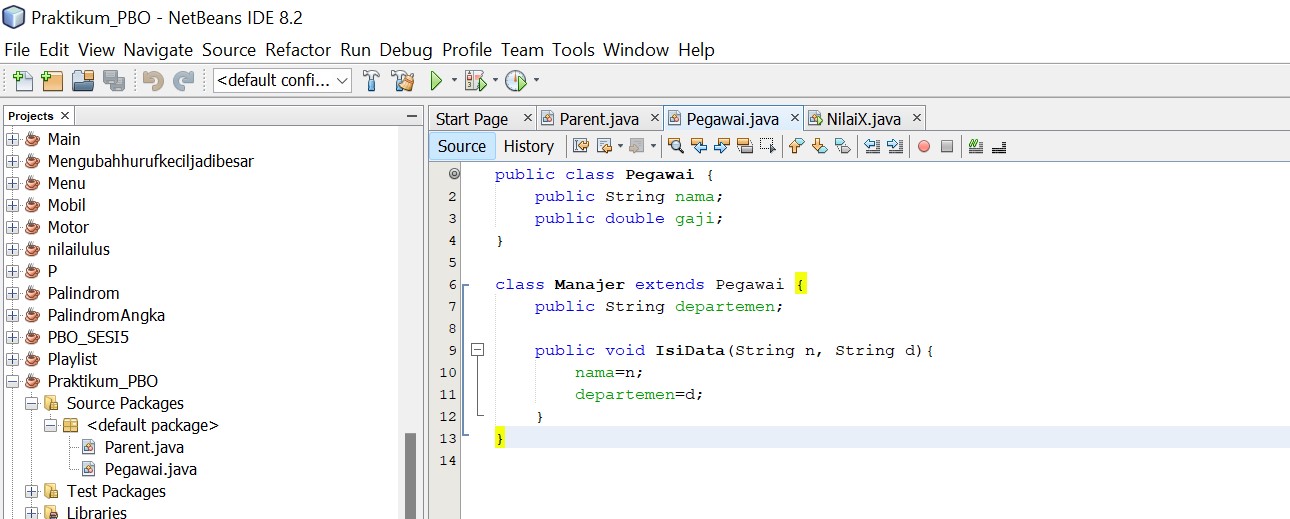


Pada percobaan satu ini Class Parent sebagai Class utama yang Mempunyai Atribut Integer bernilai 5/ int

= 5, Child dikatakan sebagai sub Class Dan di dalam Class Child terdapat Parameter 20, Dimana semuanya itu ditentukan dari tes.info. dan dalam data member dari Class parent bernilai 5 Karena “Super” mengambil nilai integer dari Class parent

* + PERCOBAAN 2

penggunaan kontrol akses terhadap atribut parent class. Mengapa terjadi error, dan bagaimana solusinya?

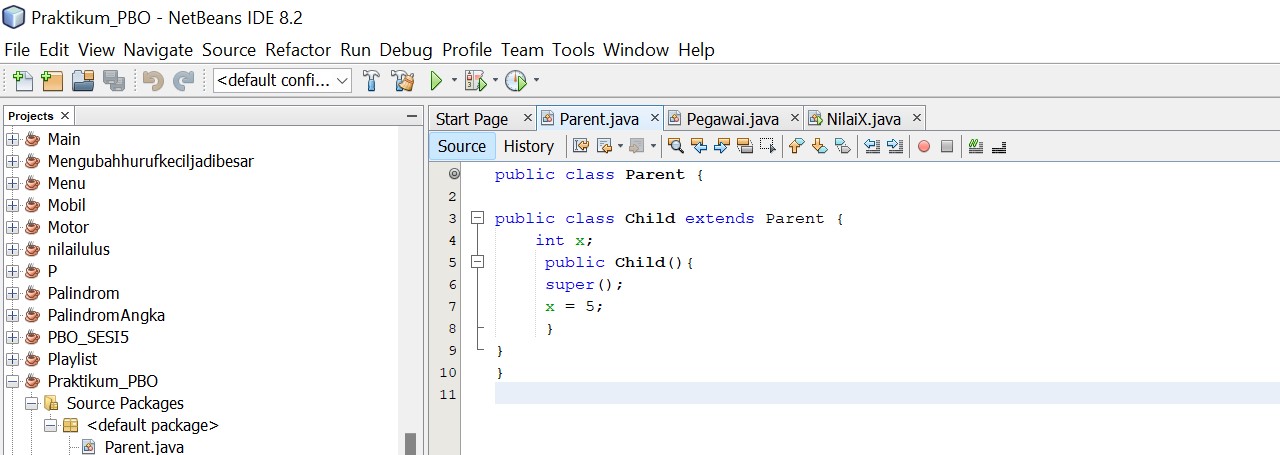


Dalam penggunaan kontrol akses terhadap atribut parent class. Solusi yang saya lakukan adalah Mengubah Private String nama Menjadi Public String nama, kemudian Menghapus public di class manajer extends pegawai agar si void isidata tidak error dan kontrol akses terhadap atribu parent class tidak mengalami error

* + PERCOBAAN 3

Percobaan berikut ini menunjukkan penggunaan konstruktor yang tidak diwariskan.

Mengapa terjadi error, dan bagaimana solusinya?

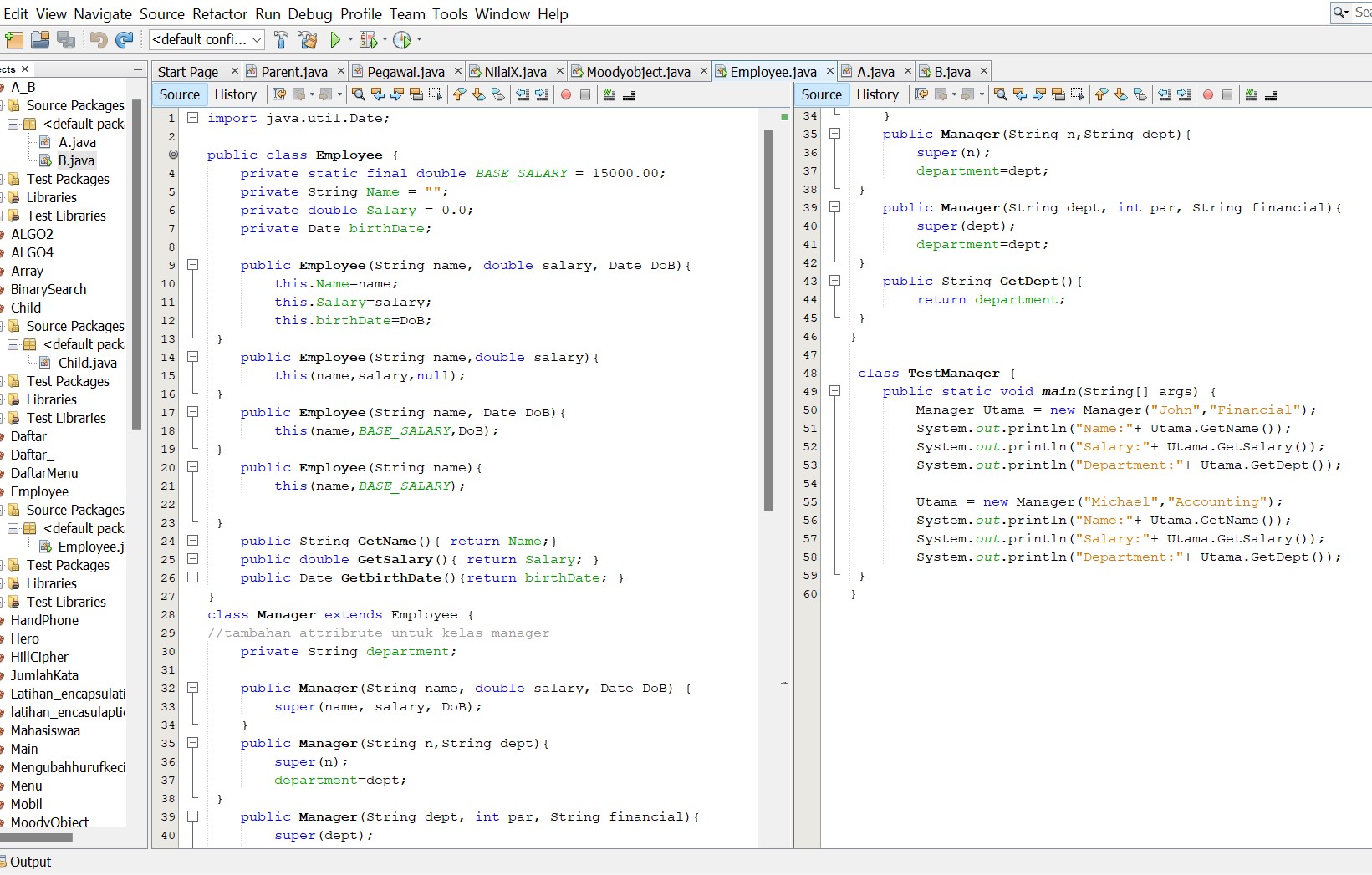


Sebelum subclass menjalankan konstruktornya sendiri, subclass akan menjalankan constructor superclass terlebih dahulu. Hal ini terjadi karena secara implisit pada constructor subclass ditambahkan pemanggilan super() yang bertujuan memanggil constructor superclass oleh kompiler.

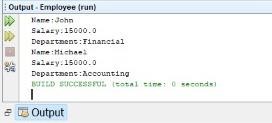
* + PERCOBAAN 4

Percobaan berikut ini menunjukkan penggunaan kelas Employee dan subkelas Manager yang merupakan turunannya. Kelas TestManager digunakan untuk menguji kelas Manager.

INPUT

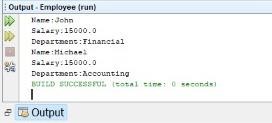
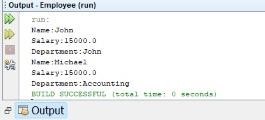


OUTPUT



Dari percobaan diatas public class testmanager tidak perlu memakai public lagi karena tipe dari class testmanager sudah tipe public. Untuk menunjukan output nama,gaji, dan departemen dimana sebelum departemen john hasilnya john yaitu salah, maka 500000 di dalam kodingan nya tersebut di hapus sehingga departemen john menjadi benar dan ini menandakan bahwa kelas testmanager yang digunakan untuk menguji kelas manager berjalan dengan baik.

Ini kodingan output sebelum 500000 dihapus dan sesudah

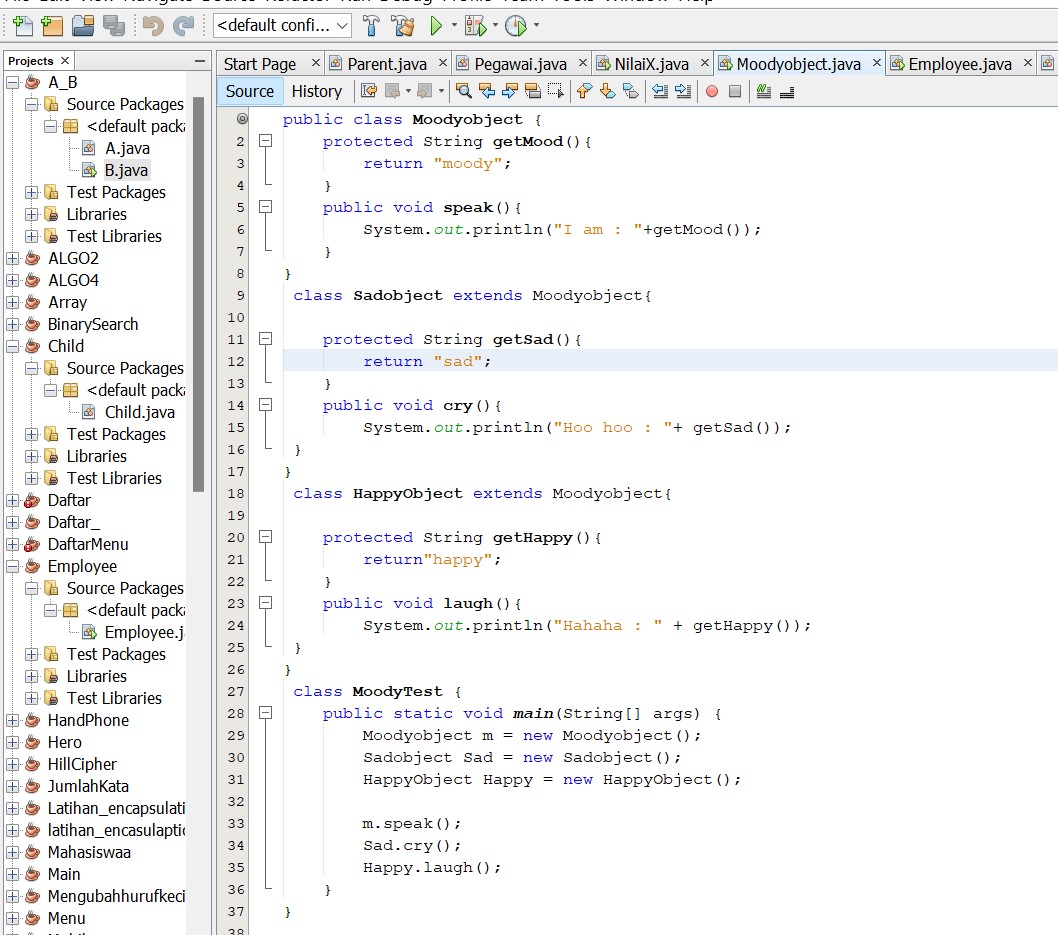


* PERCOBAAN 5

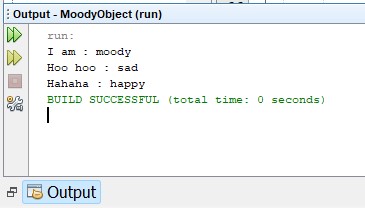
Percobaan berikut ini menunjukkan penggunaan kelas MoodyObject dengan subkelas HappyObject dan SadObject. Kelas MoodyTest digunakan untuk menguji kelas dan subkelas.

* SadObject berisi :
* sad, method untuk menampilkan pesan, tipe public - HappyObject berisi :
* laugh, method untuk menampilkan pesan, tipe public - MoodyObject berisi :
* getMood, memberi nilai mood sekarang, tipe public, return type string o speak, menampilkan mood, tipe public.

INPUT

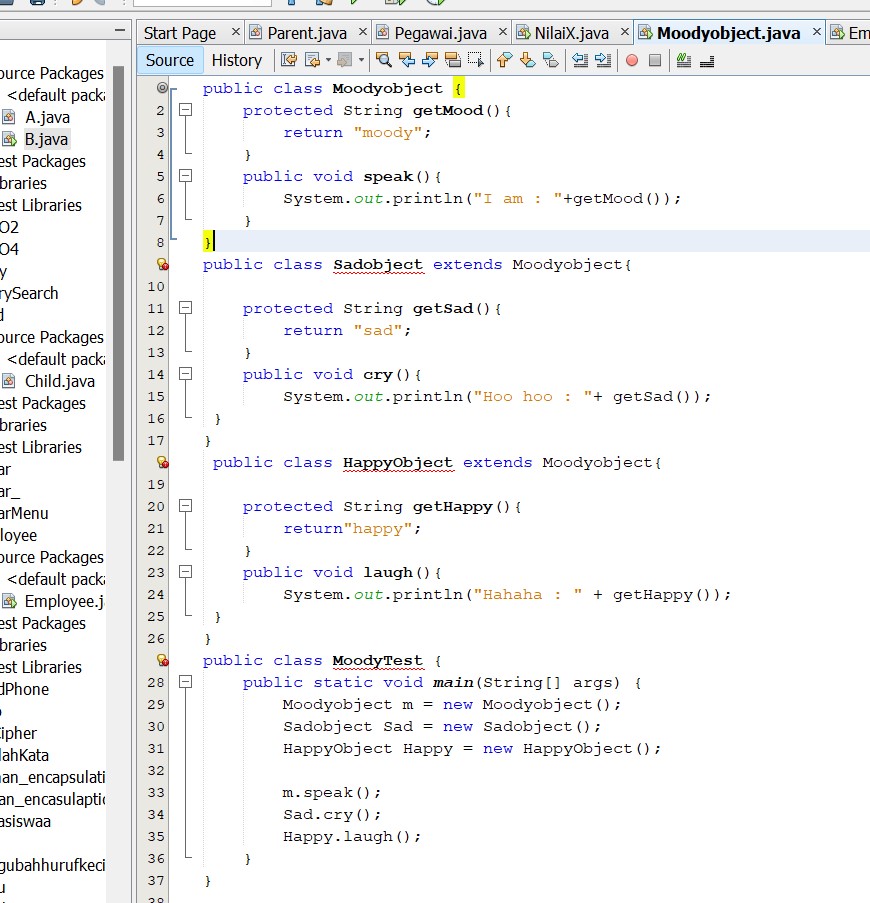


OUTPUT



Percobaan diatas menunjukan penampilan pesan mood, dimana kelas ini diuju dengan moody test yang digunakan untuk menguji kelas dan subkelas. Mengapa public class di sub kelas dan di kelas moodytest dihapus publicnya, karena sudah merupakan tipe public jadi kita tidak perlu menuliskan lagi public di dalam subkelas dan kelas moodytest, hanya moodyobject. Jika ditambahkan public lagi pada sub kelas maka akan terjadi error dan tidak bisa di run.

Seperti ini:

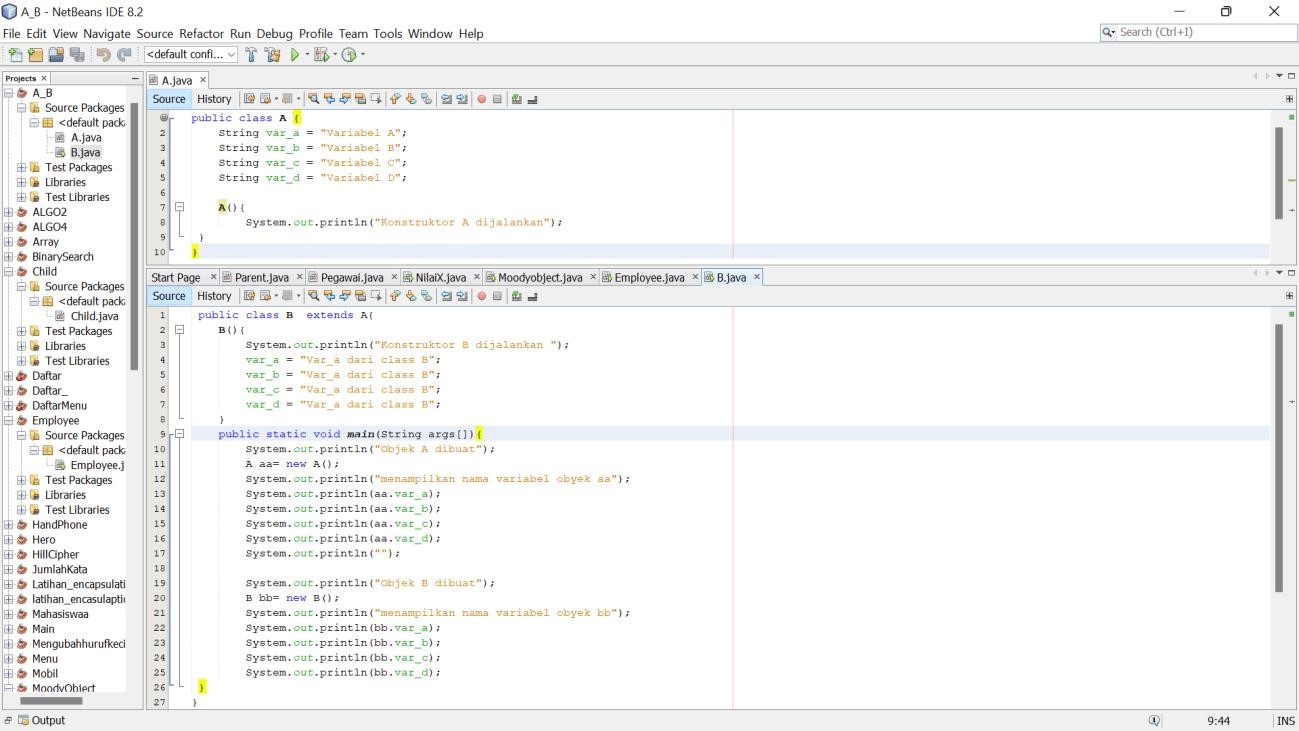


* PERCOBAAN 6

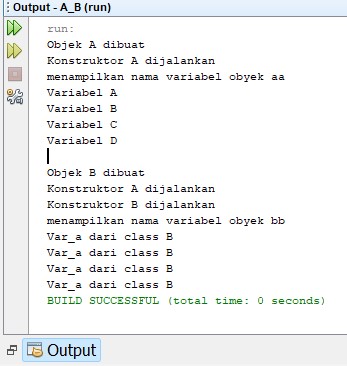
Percobaan berikut ini menunjukkan penggunaan kelas A dan dengan subkelas B. Simpan kedua kelas ini dalam 2 file yang berbeda (A.java dan B.java) dan dalam satu package. Perhatikan proses pemanggilan konstruktor dan pemanggilan variabel.

INPUT

Program A dan Program B dalam satu package A\_B



OUTPUT

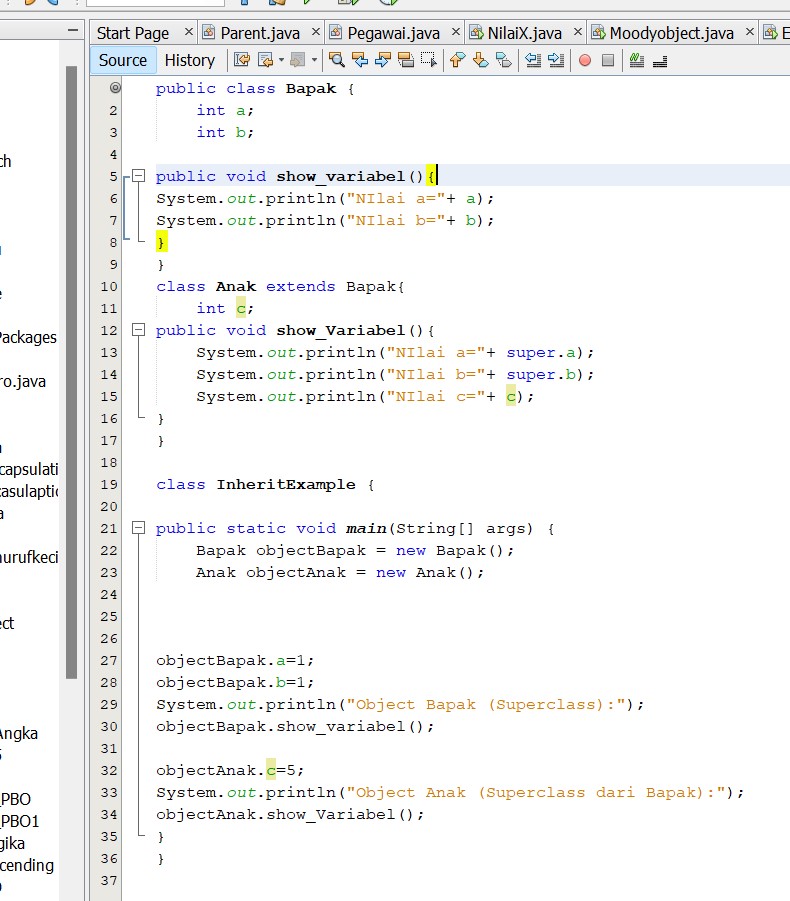


Percobaan diatas menggunakan kelas A dan sub kelas B dimana dalam 2 file yang berbeda di dalam satu package yang harus menghasilkan/ memanggil konstruktor dan memanggil variabel, kita hanya perlu menambahkan public disamping class karena berbeda file programnya kemudian menambahkan var\_c dan var\_d untuk pemanggilan konstruktor dan variabelnya.

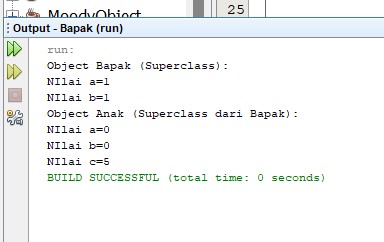
* PERCOBAAN 7

Percobaan berikut ini menunjukkan penggunaan Inheritance dan Overriding method pada kelas Bapak dan subkelas Anak. Terjadi override pada method show\_variabel.Perhatikan perubahan nilai pada variabel a, b, dan c. Kemudian lakukan modifikasi pada method show\_variabel() pada class Anak. Gunakan super untuk menampilkan nilai a dan b (memanfaatkan method yang sudah ada pada superclass).

INPUT



OUTPUT

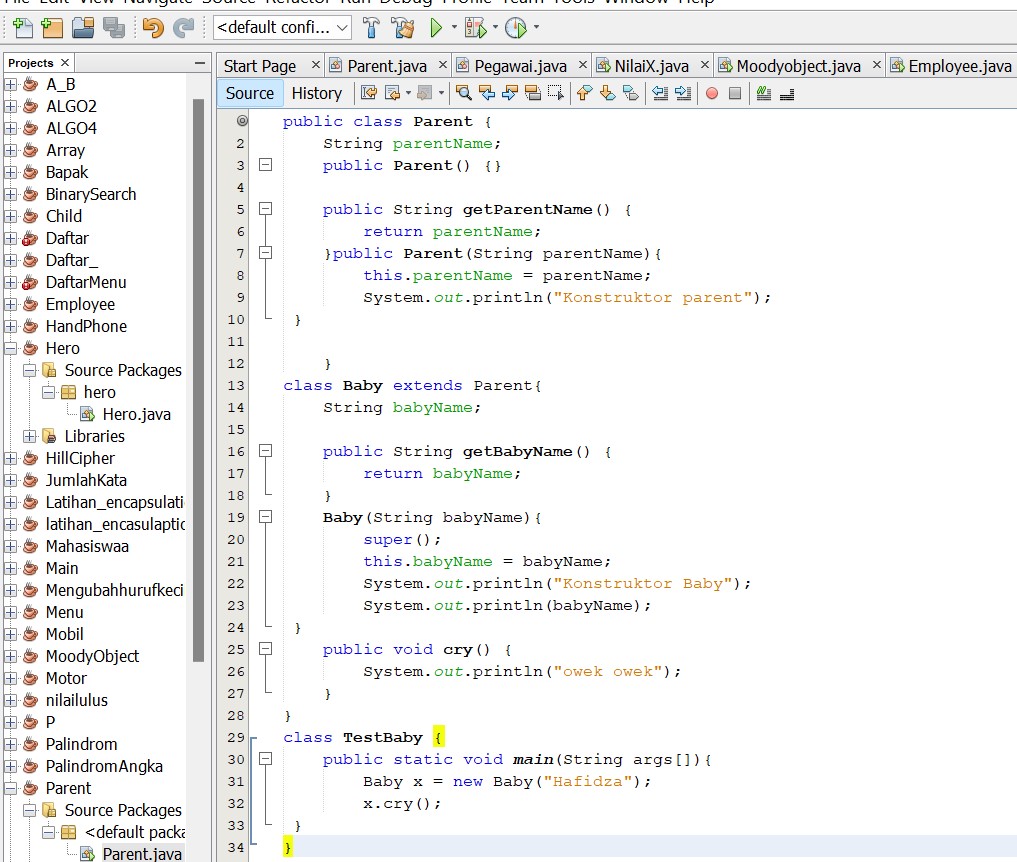


Percobaan diatas menampilkan bilangan a dan b juga perubahan variabel a,b,c yang telah di modifikasi di class anak. Mengapa class anak dan inheritExample tidak diberi public disamping class, karena satu program dengan kelas bapak sehingga jika satu program diberikan public pada subclass akan terjadi error kecuali di berbeda program.

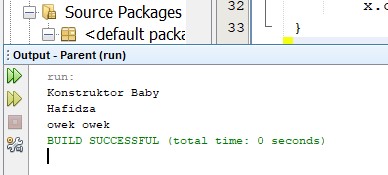
* PERCOBAAN 8

Percobaan berikut ini menunjukkan penggunaan overriding method pada kelas Parent dan subkelas Baby, saat dilakukan pemanggilan konstruktor superclass dengan menggunakan super.

INPUT



OUTPUT



Percobaan diatas pemanggilan konstruktor dengan menggunakan super, dimana memanggil konstruktor Baby dengan nama hafidza dan menangis dengan kata owek owek berhasil dijalankan.

* PERCOBAAN 9

Pembuatan kelas Account dan subkelas SavingAccount, CheckingAccount

**Buat kelas Account** sesuai dengan diagram UML untuk kelas Account sebelumnya, dengan definisi :

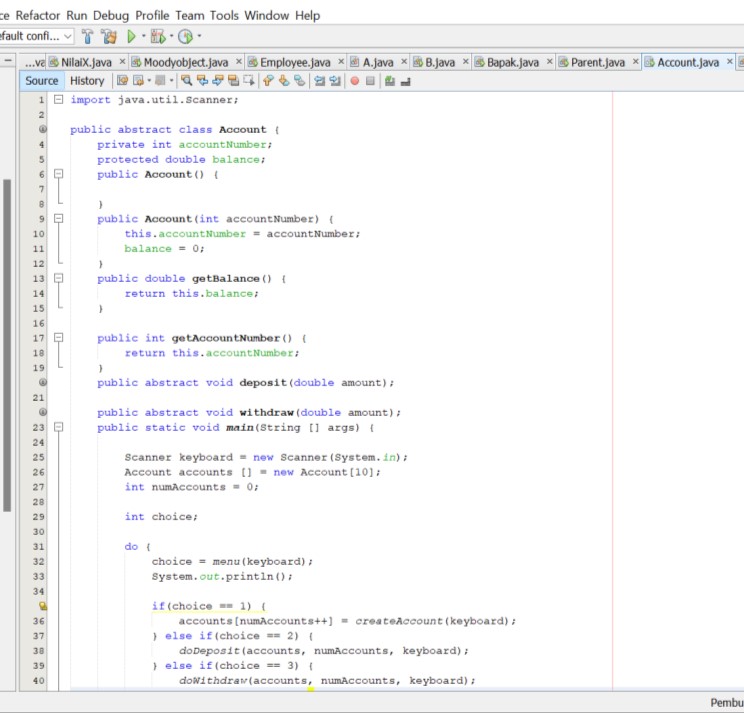
◦ Atribut balance tipe double, dan sifat protected

◦ Constructor Account untuk memberi nilai awal balance

◦ Method getBalance untuk mendapatkan nilai balance

◦ Method deposit untuk menambah nilai balance

◦ Method withdraw untuk mengambil nilai balance



|  |
| --- |
| import java.util.Scanner;    public abstract class Account { private int accountNumber; protected double balance; public Account() {  }  public Account(int accountNumber) { this.accountNumber = accountNumber; balance = 0;  } |

public double getBalance() { return this.balance;

|  |
| --- |
| } public int getAccountNumber() { return this.accountNumber;  } public abstract void deposit(double amount);  public abstract void withdraw(double amount); public static void main(String [] args) {    Scanner keyboard = new Scanner(System.in); Account accounts [] = new Account[10]; int numAccounts = 0;  int choice;  do {  choice = menu(keyboard);  System.out.println();  if(choice == 1) { accounts[numAccounts++] = createAccount(keyboard);  } else if(choice == 2) { doDeposit(accounts, numAccounts, keyboard);  } else if(choice == 3) { doWithdraw(accounts, numAccounts, keyboard);  } else if(choice == 4) { applyInterest(accounts, numAccounts, keyboard);  } else {  System.out.println("GoodBye!");  }  System.out.println();  } while(choice != 5); } |

public static int accountMenu(Scanner keyboard) {

|  |
| --- |
| System.out.println("Select Account Type");  System.out.println("1. Checking Account");  System.out.println("2. Savings Account");  int choice; do {  System.out.print("Enter choice: "); choice = keyboard.nextInt();  }while(choice < 1 || choice > 2);  return choice;  } public static int searchAccount(Account accounts [], int count, int accountNumber) {  for(int i=0; i<count; i++) { if(accounts[i].getAccountNumber() == accountNumber) { return i;  }  }  return -1;  } public static void doDeposit(Account accounts [], int count,  Scanner keyboard) {    System.out.print("\nPlease enter account number: "); int accountNumber = keyboard.nextInt();      int index = searchAccount(accounts, count, accountNumber);    if(index >= 0) { |

System.out.print("Please enter Deposit Amount: ");

|  |
| --- |
| double amount = keyboard.nextDouble();  accounts[index].deposit(amount);  } else {  System.out.println("No account exist with  AccountNumber: " + accountNumber);  }  }  public static void doWithdraw(Account accounts [], int count, Scanner keyboard) {    System.out.print("\nPlease enter account number: "); int accountNumber = keyboard.nextInt();  int index = searchAccount(accounts, count, accountNumber);  if(index >= 0) {    System.out.print("Please enter Withdraw Amount: "); double amount = keyboard.nextDouble(); accounts[index].withdraw(amount);  } else {  System.out.println("No account exist with  AccountNumber: " + accountNumber);  }  } public static void applyInterest(Account accounts [], int count, Scanner keyboard) {    System.out.print("\nPlease enter account number: "); int accountNumber = keyboard.nextInt(); |

|  |
| --- |
| int index = searchAccount(accounts, count, accountNumber);  if(index >= 0) {  if(accounts[index] instanceof SavingAccount) {  ((SavingAccount)accounts[index]).applyInterest();  }  } else {  System.out.println("No account exist with  AccountNumber: " + accountNumber);  }  }  public static Account createAccount(Scanner keyboard) {  Account account = null; int choice = accountMenu(keyboard);  int accountNumber;  System.out.print("Enter Account Number: "); accountNumber = keyboard.nextInt();  if(choice == 1) { // chekcing account  System.out.print("Enter Transaction Fee: "); double fee = keyboard.nextDouble();  account = new CheckingAccount(accountNumber, fee);  } else { // Savings account    System.out.print("Please enter Interest Rate: "); double ir = keyboard.nextDouble(); account = new SavingAccount(accountNumber, ir);  }  return account; |

}

|  |
| --- |
| public static int menu(Scanner keyboard) {  System.out.println("Bank Account Menu");  System.out.println("1. Create New Account");  System.out.println("2. Deposit Funds");  System.out.println("3. Withdraw Funds");  System.out.println("4. Apply Interest");  System.out.println("5. Quit");  int choice;    do {  System.out.print("Enter choice: "); choice = keyboard.nextInt();  } while(choice < 1 || choice > 5);  return choice;  }  } |

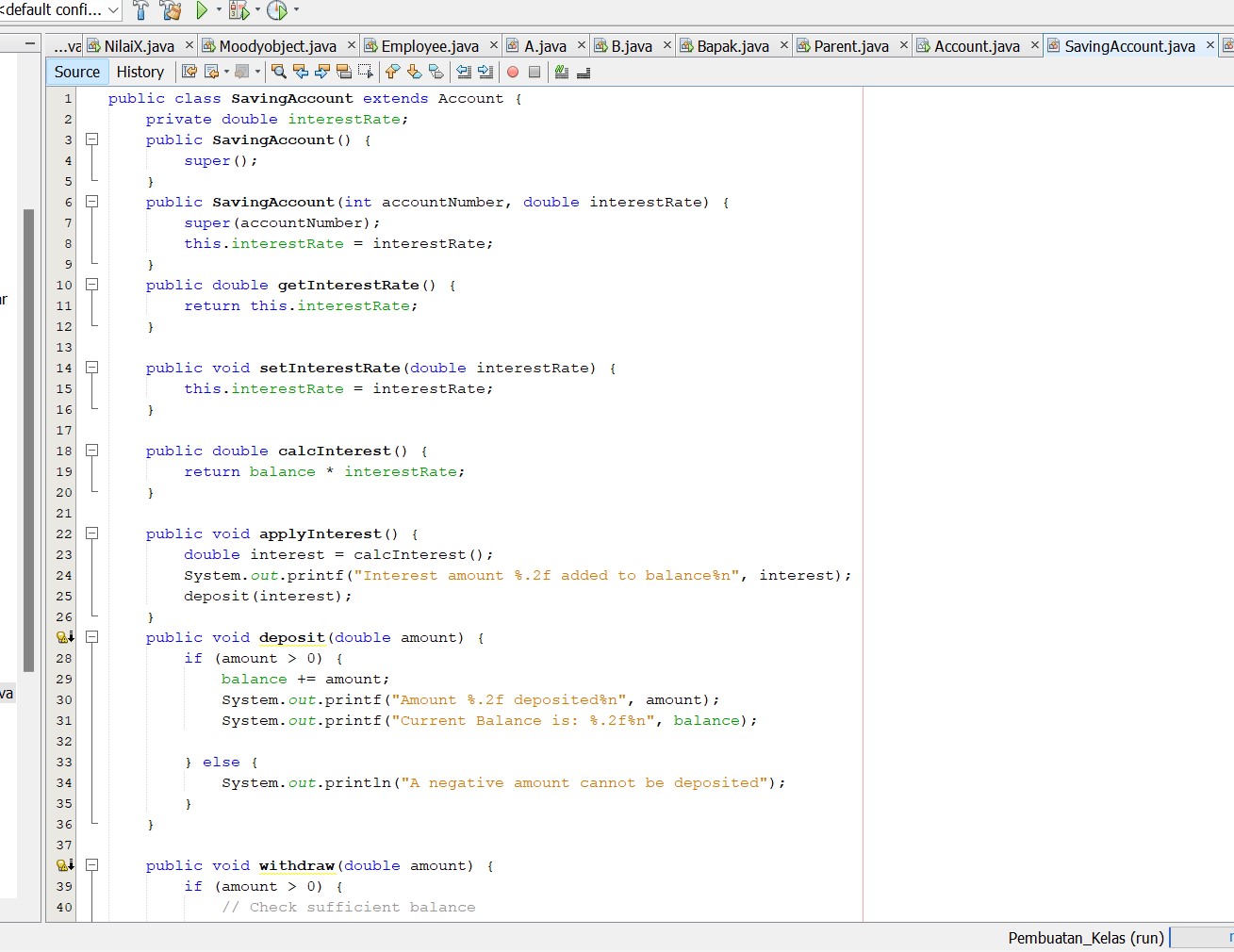
**Buat subkelas SavingAccount** sesuai dengan diagram UML sebelumnya dengan definisi :

◦ Kelas SavingAccount merupakan turunan kelas Account, gunakan keyword extends.

◦ Atribut interestRate, tipe double, sifat private

◦ Constructor SavingAccount dengan parameter balance dan interest\_rate.

Constructor ini harus passing parameter balance ke parent constructor dengan menggunakan super(balance) dan mengeset nilai variabel interestRate dengan nilai interest\_rate.



|  |
| --- |
| public class SavingAccount extends Account { private double interestRate; public SavingAccount() { super();  } public SavingAccount(int accountNumber, double interestRate)  { super(accountNumber);  this.interestRate = interestRate;  } public double getInterestRate() { return this.interestRate; |

|  |
| --- |
| } public void setInterestRate(double interestRate) { this.interestRate = interestRate;  } public double calcInterest() { return balance \* interestRate;  } public void applyInterest() {  double interest = calcInterest();  System.out.printf("Interest amount %.2f added to balance%n", interest); deposit(interest);  } public void deposit(double amount) { if (amount > 0) { balance += amount;  System.out.printf("Amount %.2f deposited%n", amount); System.out.printf("Current Balance is: %.2f%n", balance);    } else {  System.out.println("A negative amount cannot be deposited");  }  }  public void withdraw(double amount) { if (amount > 0) {  // Check sufficient balance if ((amount) <= balance) {  System.out.printf("Amount of %.2f withdrawn from Account%n", amount); balance -= amount; |

System.out.printf("Current Balance is: %.2f%n", balance); }

} else {

System.out.println("Negative amount cannot be withdrawn!");

}

}

}

**Buat kelas CheckingAccount** sesuai dengan diagram UML sebelumnya dengan definisi : ◦ Kelas CheckingAccount merupakan turunan kelas Account, gunakan keyword extends.

◦ Atribut overdraftProtection, tipe double, sifat private

◦ Terdapat public constructor dengan dua parameter: balance and protect. Constructor ini harus passing parameter balance ke parent constructor dengan menggunakan super(balance) dan mengeset nilai variabel overdraftProtection dengan nilai protect.

◦ Constructor dengan satu parameter yaitu balance. Constructor ini harus passing parameter balance ke lokal constructor dengan menggunakan this. Perhatikan bahwa constructor lain yang ada adalah constructor dengan dua parameter. Maka buat nilai protect default adalah -1.0 yang berarti bahwa pada account tidak terdapat overdraftProtection.

◦ Saldo = balance + overdraftProtection

◦ overdraftProtection = Saldo minimal, yaitu saldo yang diharapkan tidak boleh diambil pada suatu rekening, kecuali bila konsumen ingin menutup rekening. ◦ Class CheckingAccount harus mengoverride method withdraw. Method withdraw harus melakukan cek terhadap saldo (balance) apakah jumlahnya cukup bila terjadi pengambilan sejumlah uang (amount). Cek yang dilakukan adalah sebagai berikut :

◦ Jika balance – amount => 0.0 maka proses pengambilan diperbolehkan dan mengembalikan nilai true. Dan selanjutnya set balance = balance – amount; ◦ Jika balance – amount < 0.0 maka lakukan cek sebagai berikut:

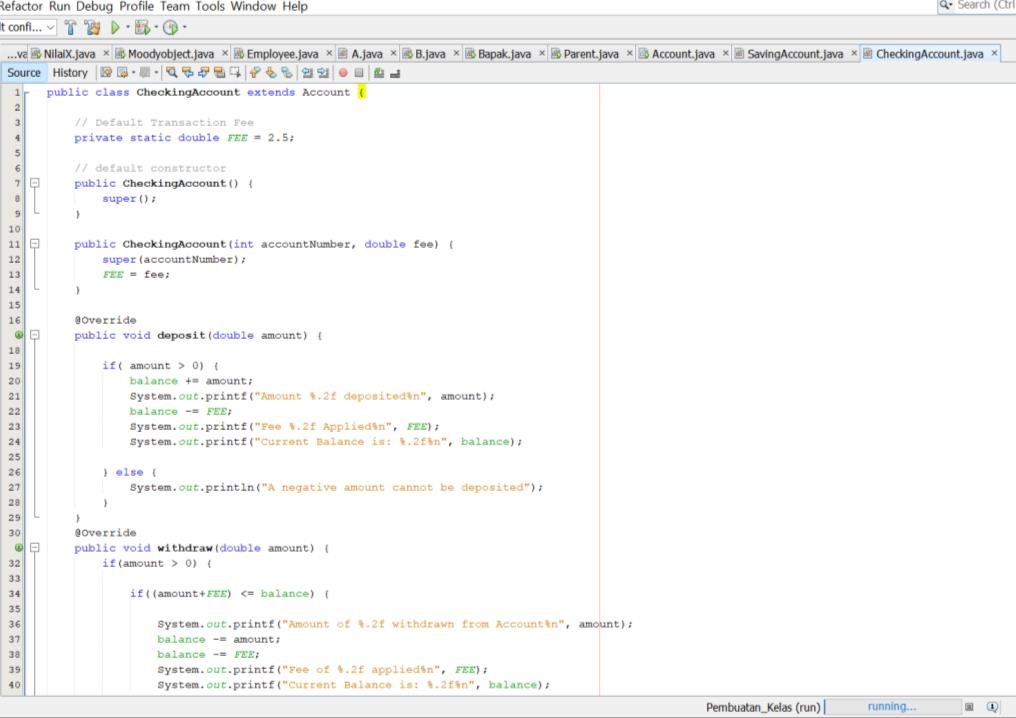
◦ Jika tidak ada overdraftProtection (nilai = -1.0) atau overdraftProtection < overdraftNeeded (amount-balance) maka gagalkan proses pengambilan uang dengan mengembalikan nilai false.

◦ Jika terdapat overdraftProtection atau overdraftProtection > overdraftNeeded (amount-balance) maka proses pengambilan uang berhasil dengan mengembalikan nilai true. Dan selanjutnya set balance = 0.0; overdraftProtection = overdraftProtection – overdraftNeeded;

◦ Constructor dengan satu parameter yaitu balance. Constructor ini harus passing parameter balance ke lokal constructor dengan menggunakan this. Perhatikan bahwa constructor lain yang ada adalah constructor dengan dua parameter. Maka buat nilai protect default adalah -1.0 yang berarti bahwa pada account tidak terdapat overdraftProtection.

◦ Saldo = balance + overdraftProtection, overdraftProtection adalah saldo minimal, yaitu saldo yang diharapkan tidak boleh diambil pada suatu rekening, kecuali bila konsumen ingin menutup rekening.

◦ Overdraft Protection (Proteksi Cerukan) yaitu fasilitas kredit kepada nasabah penyimpan dana untuk menutupi cerukan; fasilitas kredit bank tersebut memungkinkan nasabah untuk menarik cek yang melebihi dana tersedia pada saldo akunnya sehingga kelebihan penarikan dana tersebut dikenakan bunga harian; apabila kelebihan penarikan dana ditutup dengan fasilitas kreditnya, kelebihan penarikan itu tidak dikenakan bunga harian.



|  |
| --- |
| public class CheckingAccount extends Account {    // Default Transaction Fee private static double FEE = 2.5;    // default constructor public CheckingAccount() { super();  } public CheckingAccount(int accountNumber, double fee) { super(accountNumber);  FEE = fee;  }    @Override  public void deposit(double amount) {  if( amount > 0) { balance += amount; |
| System.out.printf("Amount %.2f deposited%n", amount); balance -= FEE;  System.out.printf("Fee %.2f Applied%n", FEE); System.out.printf("Current Balance is: %.2f%n", balance);    } else {  System.out.println("A negative amount cannot be deposited");  }  }  @Override public void withdraw(double amount) { if(amount > 0) {  if((amount+FEE) <= balance) {    System.out.printf("Amount of %.2f withdrawn from Account%n", amount); balance -= amount; balance -= FEE;  System.out.printf("Fee of %.2f applied%n", FEE); System.out.printf("Current Balance is: %.2f%n", balance); }  } else {  System.out.println("Negative amount cannot be withdrawn!");  }  }  } |

OUTPUT

