LAPORAN PENGERJAAN TUGAS PERTEMUAN 5

TEKNIK PEMOGRAMAN PRAKTIKUM

Disusun untuk memenuhi salah satu tugas pertemuan ke lima pada mata kuliah

Teknik Pemograman

Oleh:

Fadilah Akbar (231524041)

Kelas :

Teknik Informatika D4 – 1B

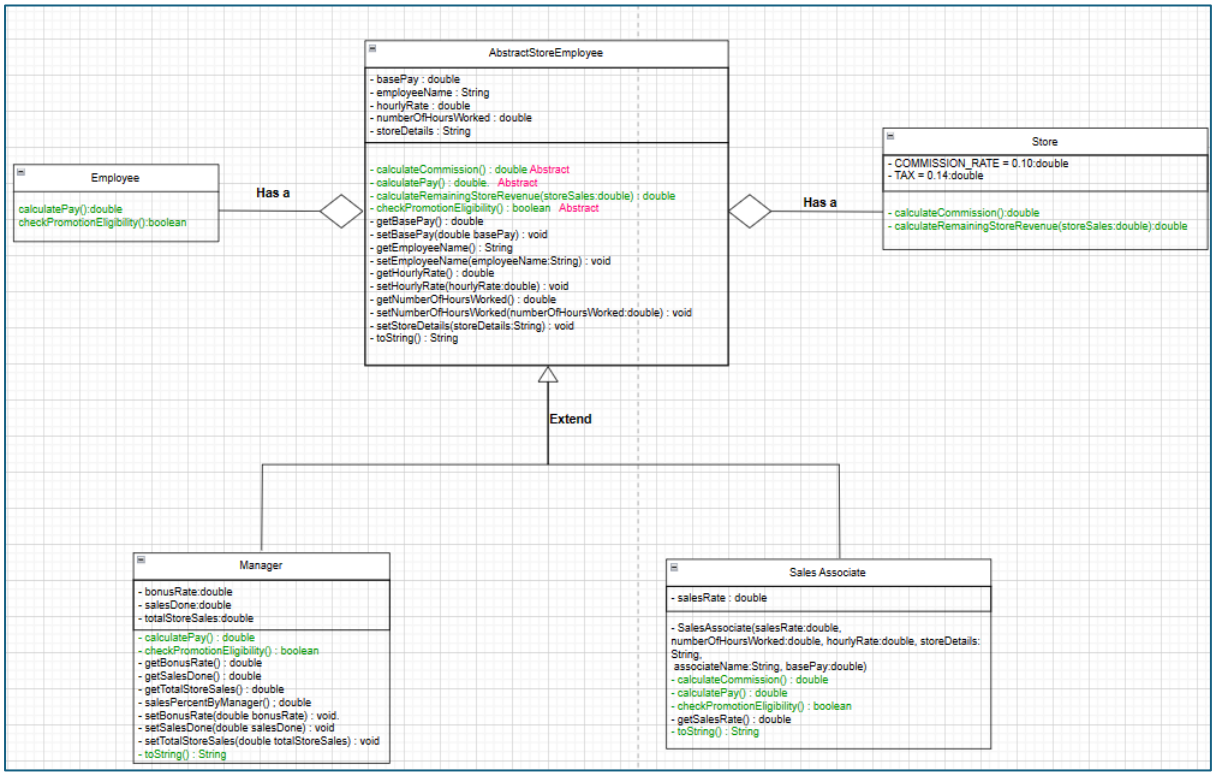
PROGRAM STUDI SARJANA TERAPAN TEKNIK INFORMATIKA

JURUSAN TEKNIK KOMPUTER DAN INFORMATIKA

POLITEKNIK NEGERI BANDUNG

2024

1. **Fahami Program Dan Gambarkan Diagram Kelas Dari Source Code Berikut**



Sesuai dengan gambar ilustrasi di atas terdapat 5 kelas, yang dimana ada 2 kelas sebagai interface yaitu ***kelas Employee*** dan ***kelas Store*** , selain itu terdapat kelas AbstractStoreEmployee yang mengimplement kedua kelas interface tersebut**,** karena class tersebut bersifat abstrak yang dimana nanti akan akan di extend oleh ***Kelas Manager*** dan ***Kelas SalesAssociate.*** Karena melakukan extend maka semua method, konstruktor dll pada superclass harus di adakan dalam subclass nya

2. **Lengkapi Input Dan Output File Text Pada Kelas Storedriver.Java**

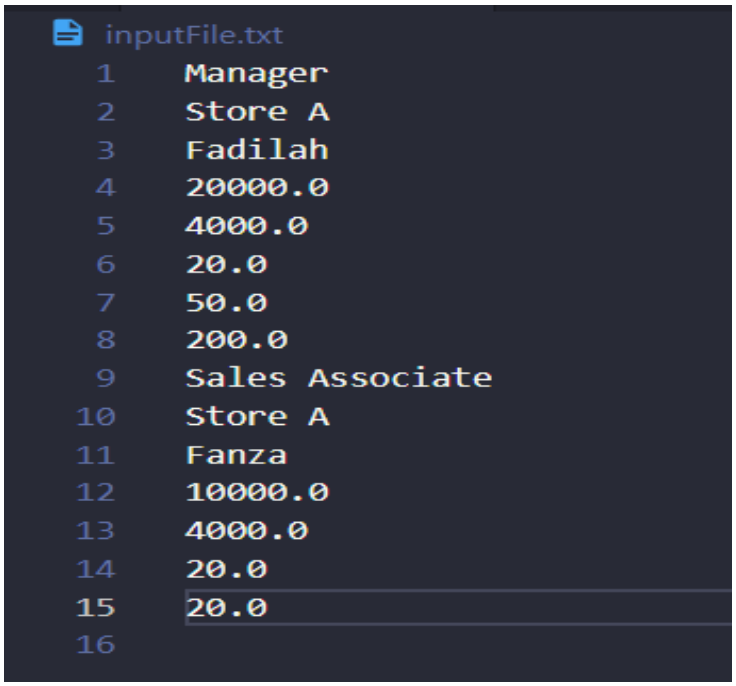
Pada awalnya saat menjalankan program pertama kali terdapat sebuah error, karena tidak terdapat ***inputFile.txt*** yang dipanggil untuk membaca sebuah inputan, dan kurangnya ***outputFile.txt*** yang akan menampilkan isi dari ***inputFile.txt***

```
**************************************************
Number of employees working as MANAGER are: 0
**************************************************
**************************************************
Number of employees working as SALES ASSOCIATES are: 0
**************************************************
```

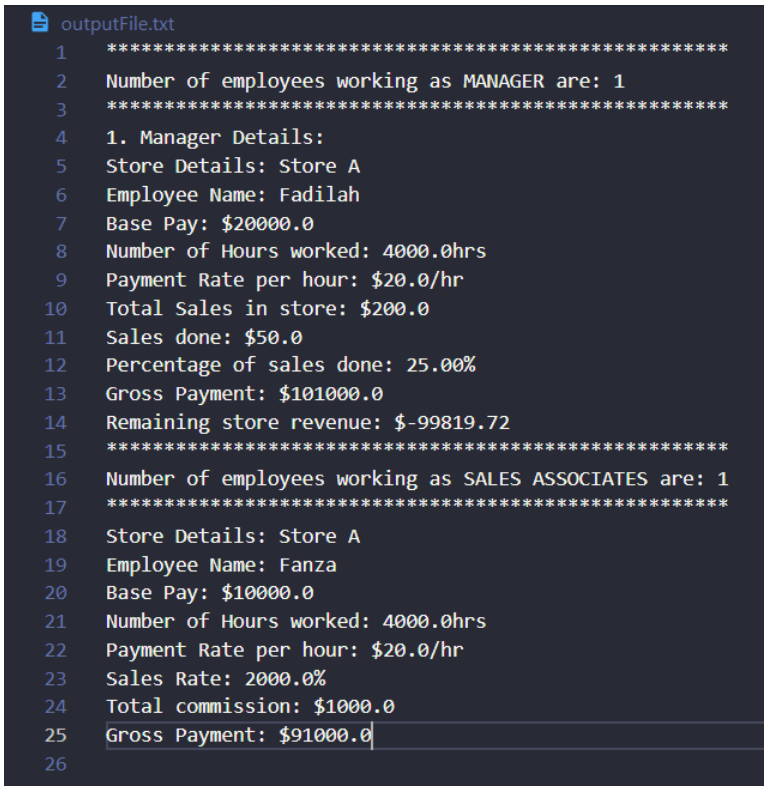
Setelah menambahkan file baru yaitu inputFile.txt dan outputFile.txt pada saat dijalankan kembali tidak terdapat error dan menampilkan seperti pada gambar di atas. Karena isi inputFile.txt belum terisi apa apa maka akan menampilkan seperti gambar di atas.

3. Perbaiki Agar Output Dapat Dijalankan Dengan Baik

Agar file dapat berjalan maka harus berisi dari ***inputFile.txt*** untuk inputnya yang disesuaikan dengan scan yang dilakukan berikut adalah contoh isi file yang berisikan 2 data Employee, ***1 Manager*** dan ***1 Sales Associate***. Maka ketika program dijalankan, program akan membaca input data dan akan menuliskan hasil pembacaanya file ***inputFile.txt*** ke file ***outputFile.txt***



Berikut ini adalah hasil dari menjalankan program yang telah berisi pada file ***inputFile.txt*** dan menampilkan ***outputFile.txt*** seperti pada gambae dibawah



4. Tambahkan 1 Class Yang Merupakan Sub Class Dari Abstractstoreemployee

Dalam kasus ini saya membuat sebuah class baru bernama “ShiftManager” yang dimana class ini memiliki 1 atribut yaitu “numberOfShifts” bertipe double. Yang bertujuan untuk menyimpan jumlah shift yang dikerjakan oleh seorang Shift Manager. Setiap shift manager dapat memiliki jumlah shift yang berbeda-beda tergantung pada kebutuhan toko atau bisnis tempat mereka bekerja.

Dengan memiliki informasi tentang jumlah shift yang dikerjakan, manajer shift dapat mengelola jadwal kerja dan melakukan perhitungan gaji berdasarkan jumlah shift yang dikerjakan.

```java
public class ShiftManager extends AbstractStoreEmployee {

    private double numberOfShifts;

    public ShiftManager(double numberOfHoursWorked, double hourlyRate, String storeDetails,
     double basePay, String employeeName, double numberOfShifts) {
        super(numberOfHoursWorked, hourlyRate, storeDetails, basePay, employeeName);
        this.numberOfShifts = numberOfShifts;
    }

    @Override
    public double calculatePay() {
        return getBasePay() + getHourlyRate() * getNumberOfHoursWorked();
    }

    @Override
    public boolean checkPromotionEligibility() {
        return false;
    }

    // setter numberOfShifts
    public void setNumberOfShifts(int numberOfShifts) {
        this.numberOfShifts = numberOfShifts;
    }

    // numberOfShifts
    public double getNumberOfShifts() {
        return numberOfShifts;
    }

    @Override
    public String toString() {
        return super.toString() + "Shift Manager: " + getNumberOfShifts();
    }

```

5. **Buatlah Method Abstract Di Interface Employee , Store Yang Di Implementasikan Pada Kelas Employee, Salesassociate, Manager Dan Kelas Yang Ditambahkan Pada Poin No 4**

Dalam soal 5 ini adalah membuat method abstrak pada class interface Employe dan Store yang akan di implementasikan pada superclass ***AbstractStoreEmployee***, kemudian di 3 subclass, yaitu ***Manager, SalesAssociate,*** dan ***ShiftManager***.

Untuk di interface Employee saya membuat method baru bernama ***calculateBonusPay(); bertipe double***, Method ini ketika di override di abstract class dibuat lagi menjadi abstract method, sehingga perlu untuk dibuat bodynya di masing masing subclass pada method ini betujuan untuk memberikan bonus gaji tambahan untuk setiap karyawan di masing masing divisi namun dengan peryaratan yang berbeda beda.

Pada class Manager

```
//Overide method abstrak interfaces employe
@Override
public double calculateBonusPay() {
    double bonusPay = 0.0;
    if (salesDone > 20) {
        bonusPay = 0.10 * super.getBasePay(); // Bonus is 10% of the base pay
    }
    return bonusPay;
}
```

Pada class SalesAssociate

```
// method abstrak interface employee
@Override
public double calculateBonusPay() {
    if (salesRate > 1.0) {
        return super.getBasePay() * 0.05;
    } else {
        return 0.0;
    }
}
```

Pada class ShiftManager

```
//Overide method abstrak interfaces employee
@Override
public double calculateBonusPay() {
    if (getNumberOfHoursWorked() > 30) {
        return getBasePay() * 0.10;
    } else {
        return 0.0;
    }
}
```

Selanjutnya untuk dibagian interface di Store saya membuat method baru bernama ***isFired(): bertipe boolean*** dimana method ini akan mengembalikan nilai boolean untuk menentukan apakah karyawan di setiap divisi ini akan di pecat atau tidak dengan syarat ketentuan tertentu

Pada class Manager

```
@Override
public boolean isFired(){
    if (calculateRemainingStoreRevenue(totalStoreSales)<0) {
        return true;
    }
    else{
        return false;
    }
}
```

Pada class SalesAssociate

```
@Override
public boolean isFired(){
    if (getSalesRate()<10) {
        return true;
    }
    else{
        return false;
    }
}
```

Pada class ShiftManager

```
//Overide method abstrak interfaces Store
@Override
public boolean isFired(){
    if (getNumberOfHoursWorked()<30) {
        return true;
    }
    else{
        return false;
    }
}
```

6. Tambahkan Mothod Abstrac Pada Abstract Class Abstractstoreemployee Dan Amati Perbedaan Abstract Di Interface Dan Di Class Abstractstoreemployee

Pada soal 6 ini saya membuat method abstrak pada class AbstractStoreEmployee dan kemudian diimplementasikan kedalam sub classnya yaitu class Manager, SalesAssociate dan ShiftManager. Saya membuat method abstract baru dengan nama ***DivisioinSalalry()*** bertipe double dimana method ini akan menghitung total gaji karyawan di setiap classnya. Berikut adalah pembuatan method DivisionSalary() di berbagai sub classnya.

Pada class Manager

```
@Override
public double divisionSalary() {
    double managerSalary = calculatePay() + calculateBonusPay();
    return managerSalary;
}
```

Pada class SalesAssociate

```
@Override
public double divisionSalary() {
    double salesSalary = calculatePay() + calculateBonusPay();
    return salesSalary;
}
```

Pada class ShiftManager

```
@Override
public double divisionSalary() {
    double shiftSalary = calculatePay() + calculateBonusPay();
    return shiftSalary;
}
```

Yang menjadi perbedaan antara method abstract di interface dan dari class abstract adalah pada method abstract kita dapat mengimplementasikan secara langsung di superclassnya dan dalam subclass nya dapat digunakan secala langsung, namun jika membuat method abstract baru di class abstract maka kita harus meng implementasikan body atau behavior di masing masing subclassnya

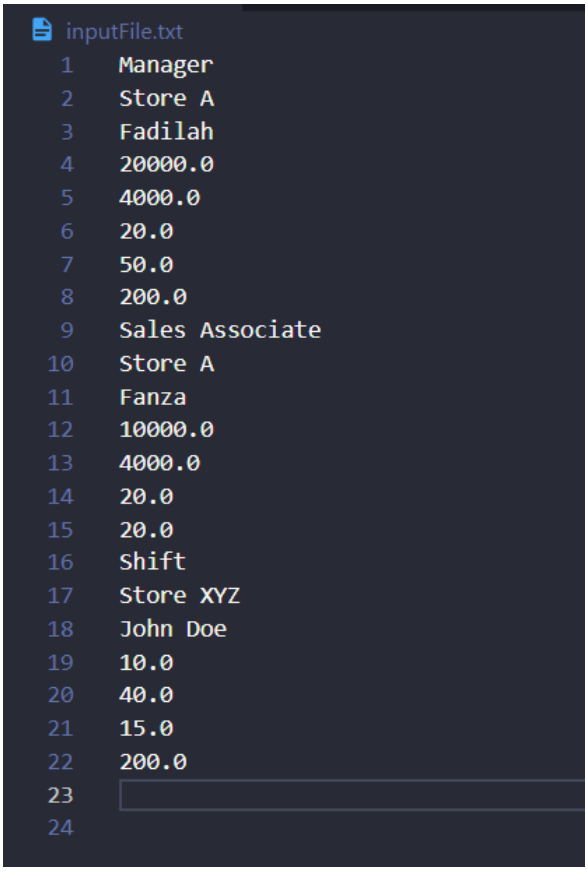
Secara singkat, perbedaan utama adalah bahwa antarmuka menyediakan kontrak tanpa implementasi, sedangkan kelas abstrak dapat memberikan implementasi parsial atau lengkap dari metode yang dideklarasikan di dalamnya. Selain itu, kelas abstrak dapat memiliki variabel instance dan metode non-abstrak, sementara antarmuka hanya mendukung metode abstrak dan tidak memiliki variabel instance.

Selain itu saya juga membuat class baru dengan nama ***ShiftManager*** dan akan menampilkan ke layar dari print ke file outputFile.txt dan juga sudah terdapat penambahan method baru yang dibuat pada superclassnya. Berikut full source code di class ***ShiftManager***, contoh ***inputFile***, ***OutputFile***

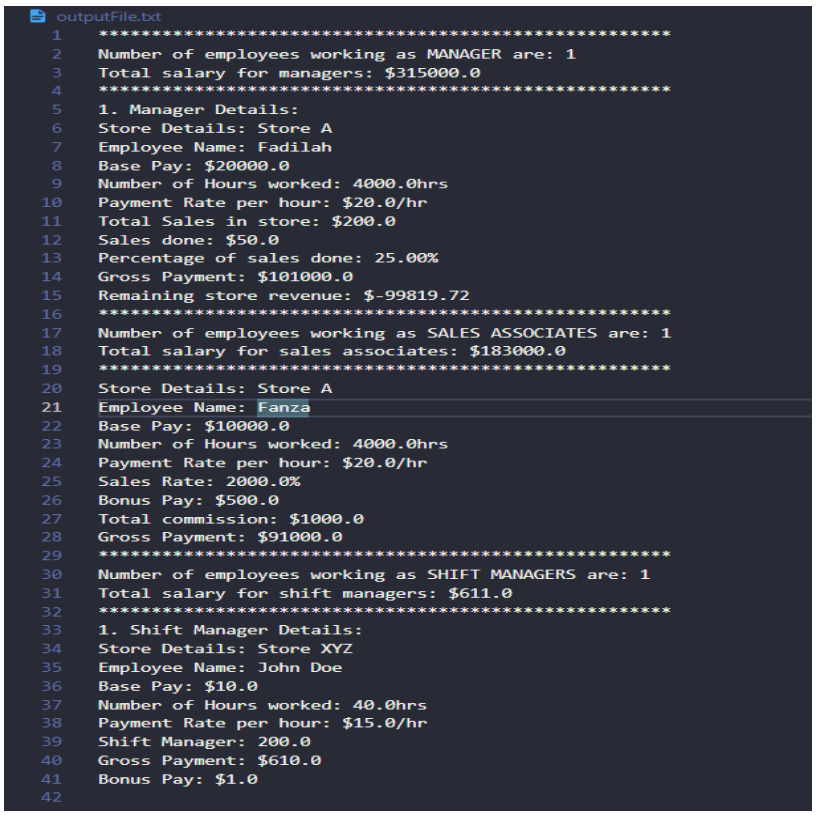
Souce Code Class ShiftManager

```java
public class ShiftManager extends AbstractStoreEmployee {

    private double numberOfShifts;

    public ShiftManager(double numberOfHoursWorked, double hourlyRate, String storeDetails,
     double basePay, String employeeName, double numberOfShifts) {
        super(numberOfHoursWorked, hourlyRate, storeDetails, basePay, employeeName);
        this.numberOfShifts = numberOfShifts;
    }

    @Override
    public double calculatePay() {
        return getBasePay() + getHourlyRate() * getNumberOfHoursWorked();
    }

    @Override
    public boolean checkPromotionEligibility() {
        return false;
    }

    // setter numberOfShifts
    public void setNumberOfShifts(int numberOfShifts) {
        this.numberOfShifts = numberOfShifts;
    }

    // numberOfShifts
    public double getNumberOfShifts() {
        return numberOfShifts;
    }

    @Override
    public String toString() {
        return super.toString() + "Shift Manager: " + getNumberOfShifts();
    }

    //method baru
     @Override
    public double divisionSalary() {
        double shiftSalary = calculatePay() + calculateBonusPay();
        return shiftSalary;
    }

    //Overide method abstrak interfaces Store
    @Override
    public boolean isFired(){
        if (getNumberOfHoursWorked()<30) {
            return true;
        }
        else{
            return false;
        }
    }

    //Overide method abstrak interfaces employee
    @Override
    public double calculateBonusPay() {
        if (getNumberOfHoursWorked() > 30) {
            return getBasePay() * 0.10;
        } else {
            return 0.0;
        }
    }
}
```

inputFile.txt



outputFile.txt



Terminal

```
*******************************************************
Number of employees working as MANAGER are: 1
Total salary for managers: $315000.0
*******************************************************
1. Manager Details:
Store Details: Store A
Employee Name: Fadilah
Base Pay: $20000.0
Number of Hours worked: 4000.0hrs
Payment Rate per hour: $20.0/hr
Total Sales in store: $200.0
Sales done: $50.0
Percentage of sales done: 25.00%
Gross Payment: $101000.0
Remaining store revenue: $-99819.72
Is Fadilah eligible for promotion? Yes, he is

*******************************************************
Number of employees working as SALES ASSOCIATES are: 1
Total salary for sales associates: $183000.0
*******************************************************
1. Sales Associate Details:
Store Details: Store A
Employee Name: Fanza
Base Pay: $10000.0
Number of Hours worked: 4000.0hrs
Payment Rate per hour: $20.0/hr
Sales Rate: 2000.0%
Total commission: $1000.0
Gross Payment: $91000.0
Bonus Pay: $500.0
Is Fanza eligible for promotion? Yes, he/she is eligible

*******************************************************
Number of employees working as SHIFT MANAGERS are: 1
Total salary for shift managers: $611.0
*******************************************************
1. Shift Manager Details:
Store Details: Store XYZ
Employee Name: John Doe
```

Final Structur