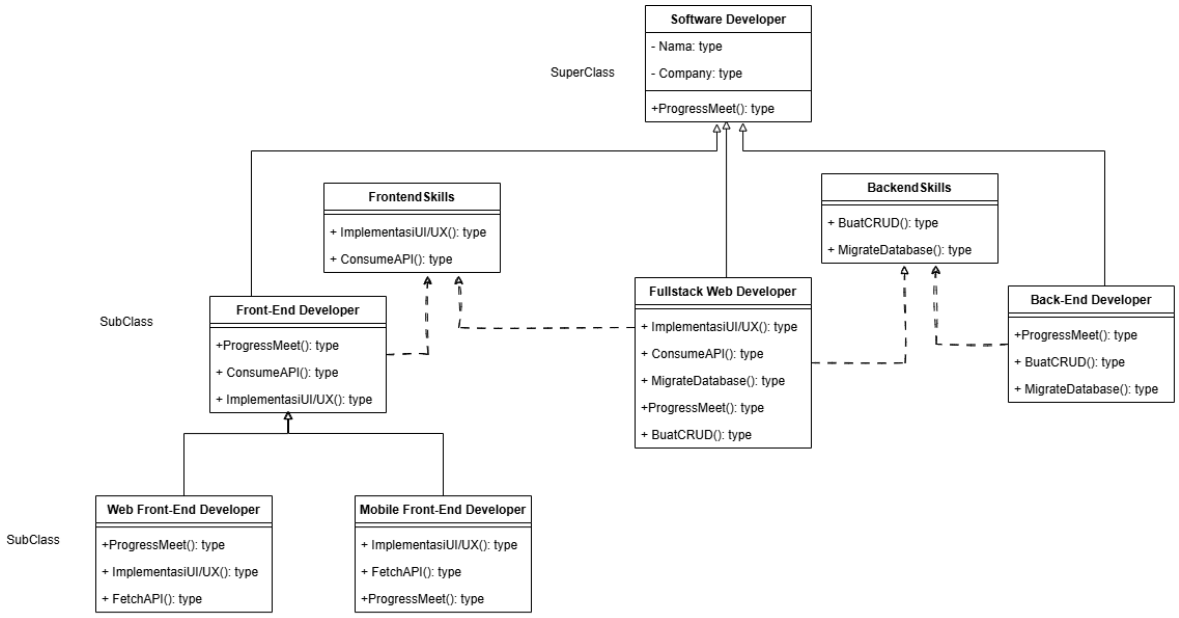
ABSTACT CLASS & INTERFACE & MULTIPLE INHERITANCE

Diajukan untuk memenuhi salat satu tugas praktikum Mata kuliah Pemrograman Berorientasi Objek

Disusun Oleh:
Daiva Raditya Pradipa (231511039)

Jurusan Teknik Komputer dan Informatika
Program Studi D-3 Teknik Informatika
Politeknik Negeri Bandung
2024

Class Diagram

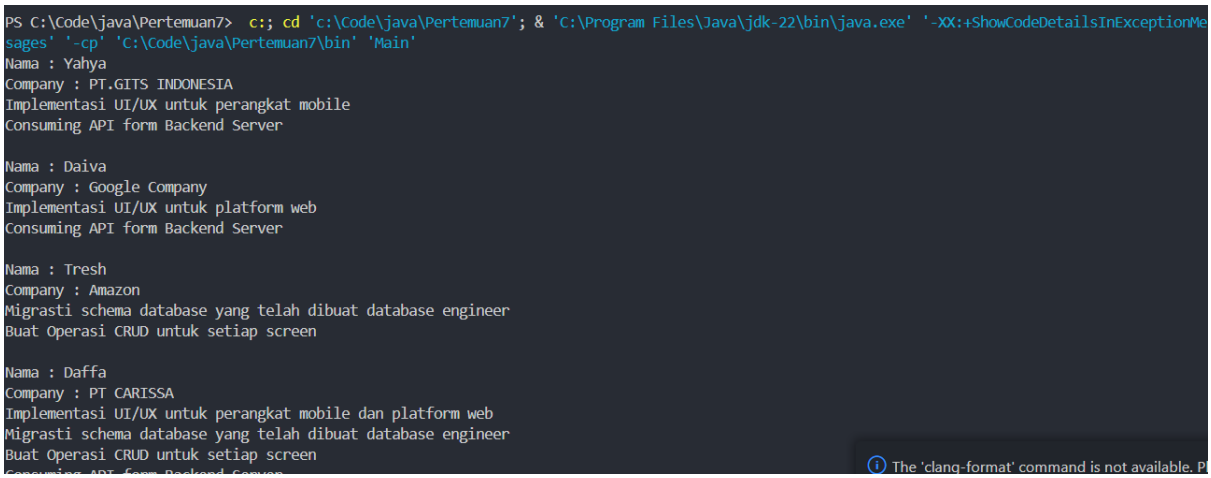


Penjelasan:

1. Software Developer merupakan abstract class yang berfungsi sebagai superclass untuk berbagai jenis developer. Kelas ini mendefinisikan perilaku umum yang dimiliki oleh semua developer.
2. Front End Developer adalah subclass dari Software Developer dan juga mengimplementasikan interface FrontendSkills. Kelas ini merepresentasikan seorang developer yang fokus pada pengembangan antarmuka pengguna (UI) dan pengalaman pengguna (UX) menggunakan teknologi front-end.
3. Mobile Front End Developer merupakan subclass dari Front End Developer. Kelas ini mengkhususkan dirinya dalam pengembangan front-end untuk aplikasi mobile, menggunakan teknologi dan framework yang berfokus pada platform mobile seperti React Native atau Flutter.
4. Web Front End Developer juga merupakan subclass dari Front End Developer. Kelas ini lebih fokus pada pengembangan front-end untuk aplikasi web, menggunakan teknologi seperti HTML, CSS, JavaScript, dan framework web front-end seperti React.js atau Angular.
5. Fullstack Developer merupakan subclass dari Software Developer, tetapi juga mengimplementasikan dua interface, yaitu FrontendSkills dan BackendSkills. Developer jenis ini memiliki kemampuan untuk mengembangkan baik bagian front-end (antarmuka pengguna) maupun back-end (logika server, database). Mereka dapat mengerjakan keseluruhan aplikasi dari antarmuka hingga server.
6. Back End Developer merupakan subclass dari Software Developer dan juga mengimplementasikan interface BackendSkills. Kelas ini merepresentasikan developer yang

fokus pada pengembangan sisi server, termasuk logika bisnis, database, API, dan integrasi dengan sistem lainnya.

Screenshot hasil run program



Screenshot code program

a. SoftwareDeveloper.java

```java
abstract class SoftwareDeveloper{
    private String name;
    private String company;

    public SoftwareDeveloper(String nama, String company){
        this.name = nama;
        this.company = company;
    }

    public String getName(){
        return this.name;
    }

    public String getCompany(){
        return this.company;
    }

    public void attendMeet(){
        System.out.println(x:"attend proyek progress meet");
    }
}
```

Class SoftwareDeveloper adalah sebuah abstract class yang berfungsi sebagai superclass bagi kelas-kelas turunannya. Kelas ini mendefinisikan atribut dan metode yang umum dimiliki oleh semua jenis developer, seperti nama (name) dan company (tempat dia bekerja). Atribut ini relevan untuk setiap tipe developer, dan method umum yang dimiliki setiap developer yaitu attendMeet(), dimana setiap developer pasti akan mengikuti meeting untuk menyampaikan progress pekerjaan mereka atau rencana pengembangan kedepannya. Ada pula getter dan setter yang digunakan untuk mengakses atribut nama dan company.

Class ini juga memiliki constructor namun bukan digunakan untuk menginstansikan class ini. Hal ini dikarenakan class ini merupakan class abstract yang tidak bisa diinstansikan. Constructor ini akan digunakan oleh subclassnya yang bukan merupakan abstract class untuk mengasign atirbut nama dan company.

b. FrontEndDeveloper.java

```
abstract class FrontEndDeveloper extends SoftwareDeveloper implements FrontendSkills{

    public FrontEndDeveloper(String nama, String company){
        super(nama,company);
    }

    public abstract void implementasiUIUX();

    @Override
    public void ConsumeAPI(){
        System.out.println(x:"Consuming API form Backend Server");
    }

}
```

Class FrontEndDeveloper merupakan abstract class yang menjadi subclass dari SoftwareDeveloper dan mengimplementasikan interface FrontendSkills. Dalam class ini, terdapat abstract method implementasiUIUX() yang harus diimplementasikan oleh subclass yang lebih spesifik, seperti MobileFrontEndDeveloper dan WebFrontEndDeveloper. Selain itu, method ConsumeAPI() dari interface FrontendSkills telah diimplementasikan dalam FrontEndDeveloper dikarenakan setiap subclass dari class ini akan memiliki behavior yang sama yaitu consume API dari server backend.

c. MobileFrontEndDeveloper.java

```
class MobileFrontEndDeveloper extends FrontEndDeveloper{

    public MobileFrontEndDeveloper(String nama, String company){
        super(nama,company);
    }

    @Override
    public void implementasiUIUX(){
        System.out.println(x:"Implementasi UI/UX untuk perangkat mobile");
    }

}
```

Class MobileFrontEndDeveloper.java merupakan subclass dari class FrontEndDeveloper. Pada class ini, method implementasiUIUX() yang merupakan abstract method pada class FrontEndDeveloper akan diimplementasikan pada class ini dengan memprint string pada terminal dengan kalimat "Implementasi UI/UX untuk perangkat mobile". Pada class ini juga mendefiiniskan constructor yang akan digunakan untuk mengisi atirbut name dan company yang ada pada class SoftwareDeveloper yang ditandai dengan pemanggilan method super dan parameter berupa atribut yang diperlukan. Karena class ini bukan merupakan abstract class maka class ini bisa diinstansikan dengan mengisi parameter yang diperlukan pada main file.

d. WebFrontEndDeveloper.java

```
public class WebFrontEndDeveloper extends FrontEndDeveloper {

    public WebFrontEndDeveloper(String nama, String company){
        super(nama,company);
    }

    @Override
    public void implementasiUIUX(){
        System.out.println(x:"Implementasi UI/UX untuk platform web");
    }

}
```

Class WebFrontEndDeveloper memiliki konsep yang identic dengan class MobileFrontEndDeveloeper, hanya saja pada class ini method implementasiUIUX() di override dengan value yang berbeda dengan class MobileFrontEndDeveloper. Pada class ini juga bukan merupakan abstract class sehingga bisa diinstansikan juga di main file dengan meberikan parameter constructor yang dibutuhkan.

e. BackEndDeveloper.java

```
public class BackEndDeveloper extends SoftwareDeveloper implements BackEndSkills {

    public BackEndDeveloper(String name, String company){
        super(name, company);
    }

    @Override
    public void BuatCRUD() {
        System.out.println(x:"Buat Operasi CRUD untuk setiap screen");
    }

    @Override
    public void MigrateDatabase() {
        System.out.println(x:"Migrasti schema database yang telah dibuat database engineer");
    }

}
```

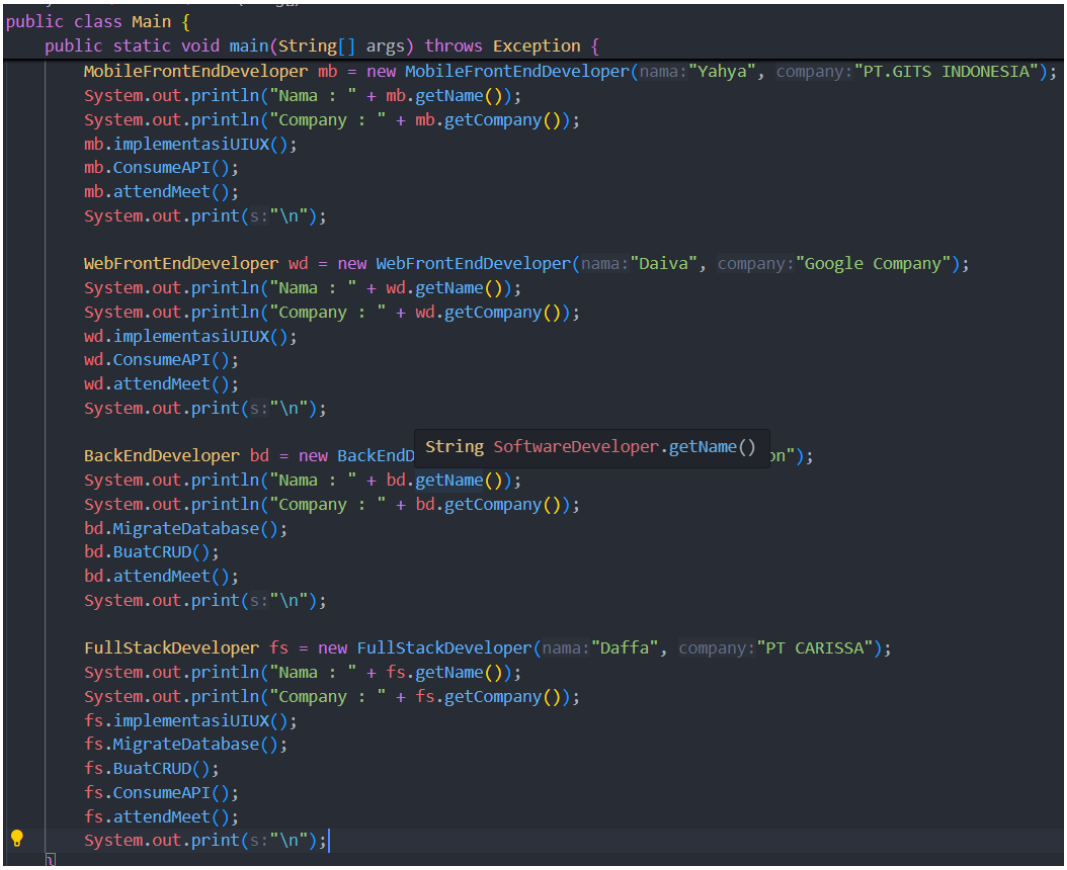
Class BackEndDeveloper merupakan subclass dari class SoftwareDeveloper. Class ini bukan merupakan class abstract sehingga bisa dinstasnsikan pada main file dengan mengirimkan parameter constructor yang dibutuhkan. Class ini juga mengimplementasikan abstract method yang ada pada interface BackendSkills dengan memberikan value pada setiap method dengan terlebih dahulu di anotasikan dengan syntax @override untuk menandakan dia akan mengganti isi atau menggimplmentasikan isi dari method pada interface BackendSkills.

f. FullstackDeveloper.java

```java
public class FullStackDeveloper extends SoftwareDeveloper implements FrontendSkills, BackEndSkills {

    public FullStackDeveloper(String nama, String company) {
        super(nama, company);
    }

    @Override
    public void BuatCRUD() {
        System.out.println(x:"Buat Operasi CRUD untuk setiap screen");
    }

    @Override
    public void MigrateDatabase() {
        System.out.println(x:"Migrasti schema database yang telah dibuat database engineer");
    }

    @Override
    public void implementasiUIUX(){
        System.out.println(x:"Implementasi UI/UX untuk perangkat mobile dan platform web");
    }

    @Override
    public void ConsumeAPI() {
        System.out.println(x:"Consuming API form Backend Server");
    }


}
```

Class FullstackDeveloper merupakan subclass dari class SoftwareDeveloper yang mengimplmentasikan dua interface yaitu FrontendSkills dan BackendSkills. Class ini akan mengimplemntasikan method method dari kedua interface tersebut dengan ditandai anotasi @Override pada setiap method yang akan diimplementasikan pada class ini. Class ini juga bukan merupakan abtract class dan dapat diinstansikan pada main file.

g. Main.java



Pada main file ini akan menginstansiasi semua subclass yang bukan abstract class dan mencoba memanggil semua method yang tersedia di setiap instansiasi tersebut. Selain itu, kita juga akan memanggil method getName() dan getCompany() dari superclass SoftwareDeveloper untuk mendapatkan nilai atribut **nama** dan **company** dari setiap objek yang diinstansiasi.