**Jobsheet 7**

**Interface**

**SAFRIZAL**

**RAHMAN\_19\_SIB\_2G**

[https://github.com/safrizalrahma n46/PBO\_SAFRIZ\_THEVIGIL ANTE](https://github.com/safrizalrahman46/PBO_SAFRIZ_THEVIGILANTE)

# A. Kompetensi

Setelah menyelesaikan lembar kerja ini mahasiswa diharapkan mampu:

1. Menjelaskan maksud dan tujuan penggunaan interface;
2. Menerapkan interface di dalam pembuatan program.

# B. Pendahuluan

*Interface* merupakan sekumpulan method tanpa body (abstract method) yang saling berkaitan

1. **Karakteristik:** 
   1. Umumnya terdiri dari abstract method
   2. Selalu dideklarasikan dengan menggunakan kata kunci interface.
   3. Diimplementasikan dengan menggunakan kata kunci implements
   4. Interface tidak dapat diinstansiasi, hanya dapat diinstasiasi melalui class yang mengimplement interface tersebut
2. **Kegunaan:**

Bertindak sebagai **kontrak/syarat** yang berisi **sekumpulan behavior/method** yang saling terkait untuk memenuhi suatu **kapabilitas**. Dengan kata lain, interface memberikan panduan mengenai method apa saja yang perlu diimplementasikan untuk memenuhi kapabilitas tertentu.

# 3. Notasi Class Diagram Interface

* Nama interface **tidak** dicetak miring
* Keterangan <<interface>> di atas nama interface
* Nama method boleh dicetak miring atau tidak
* Implements dilambangkan dengan garis panah putus-putus

# 4. Aturan Penulisan Interface

* Secara struktur hampir sama dengan class
* Ada beberapa aturan dalam penulisan interface:
  1. **Tidak** memiliki concrete method (method biasa yang bukan abstract)
  2. **Tidak** memiliki constructor
  3. Dapat memiliki atribut, tapi hanya dapat bersifat public, static, final

# 5. Sintaks Interface

* Untuk mendeklarasikan suatu interface:

public interface <NamaInterface>

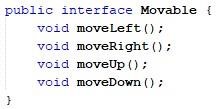
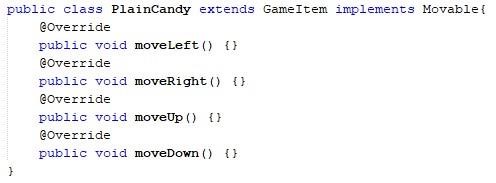
* Untuk mengimplementasikan interface:

public class <NamaClass> implements <NamaInterface>

* Nama interface sebaiknya dalam bentuk **adjective/kata sifat** jika merepresentasikan kapabilitas. Dapat juga menggunakan **kata benda**

•

Contoh:



# 6. Implementasi Interface

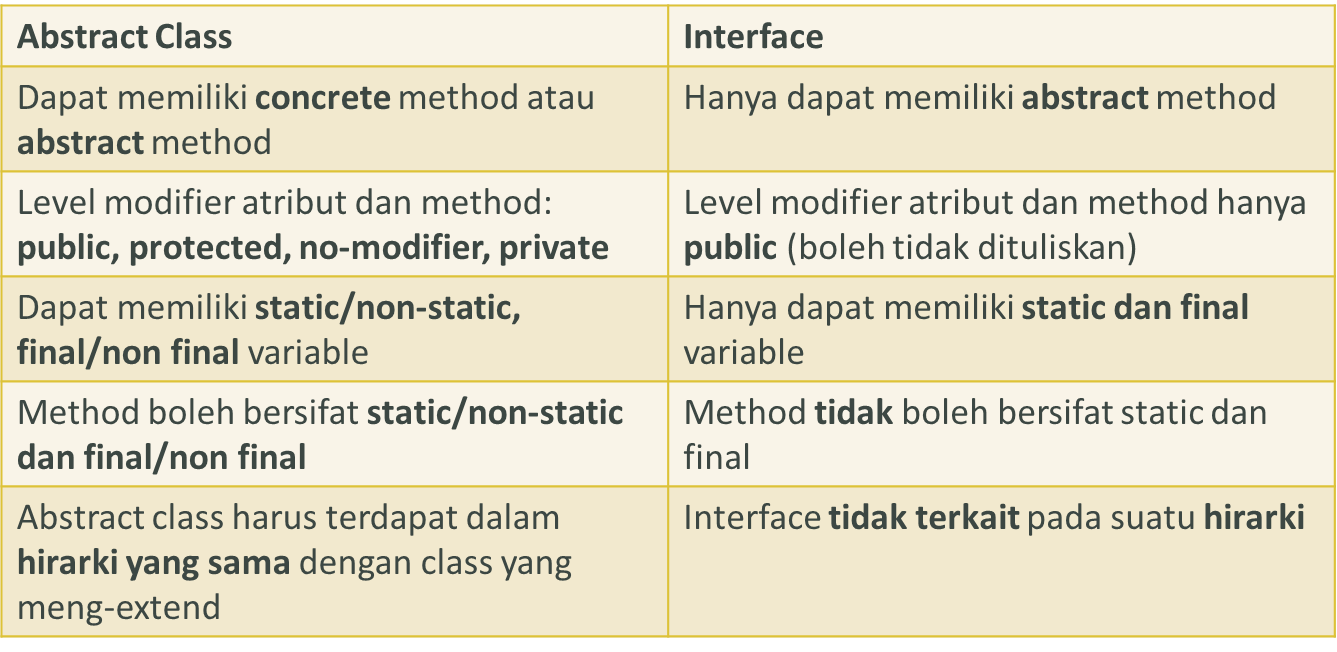
Bila sebuah class mengimplementasikan suatu interface:

* **Seluruh konstanta** dari interface akan dimiliki oleh class tersebut
* **Seluruh method** pada interface harus diimplementasikan
* Bila class yang meng-implement interface **tidak mengimplementasikan semua method**, maka class tersebut harus dideklarasikan sebagai **abstract class**

# 7. Multiple Interface

* Suatu class dapat meng-implement multiple interface
* Bila suatu class merupakan subclass dan meng-implement interface, maka **keyword extends mendahului implements** ▪ Contoh: public class PlainCandy extends GameItem implements Crushable, Movable

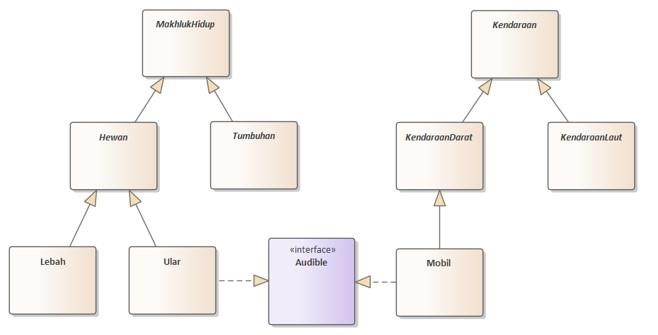
# 8. Perbedaan Abstract Class dan Interface



# 9. Interface tidak terikat pada hirarki

Suatu class di java hanya dapat meng-extend atau menjadi subclass secara langsung dari **satu** superclass saja. Akibatnya class tersebut akan terikat pada suatu hirarki tertentu. Misalnya class Lebah merupakan subclass Hewan sedangkan class Hewan sendiri merupakan subclass MakhlukHidup. Pembatasan 1 parent class secara langsung ini menyebabkan class Lebah terikat pada hirarki makhluk hidup dan tidak bisa terkait dengan hirarki lainnya.

Sementara itu interface tidak terikat pada suatu hirarki. Interface dibuat “secara lepas” tanpa bergantung pada hirarki. Misalkan terdapat interface Audible, interface tersebut dapat diimplementasikan di class apapun dari hirarki manapun. Misal class Ular bisa berbunyi, class ini dapat mengimplementasikan interface Audible. Begitu juga dengan class Mobil dari hirarki kendaraan dapat pula mengimplementasikan interface Audible.



# 10. Penggunaan Abstract Class vs Interface

Abstract class dapat memiliki atribut (instance variable), yaitu suatu variable yang dimiliki oleh objek tertentu. Atribut dan method ini (jika access level modifier-nya sesuai) akan diwariskan terhadap subclass nya. Oleh karena itu, jika suatu class memiliki **common properties (dan method)** maka sebaiknya dibuat abstract class sebagai generalisasi. Misal

ada beberapa class PlainCandy, StripedCandy, RainbowChocoCandy, Wall dll yang merupakan jenis item dalam game dengan atribut yang sama, misalnya positionX, positionY, dan iconName, sebaiknya kita buat abstract class Hewan

sebagai generalisasi dari class-class tersebut.

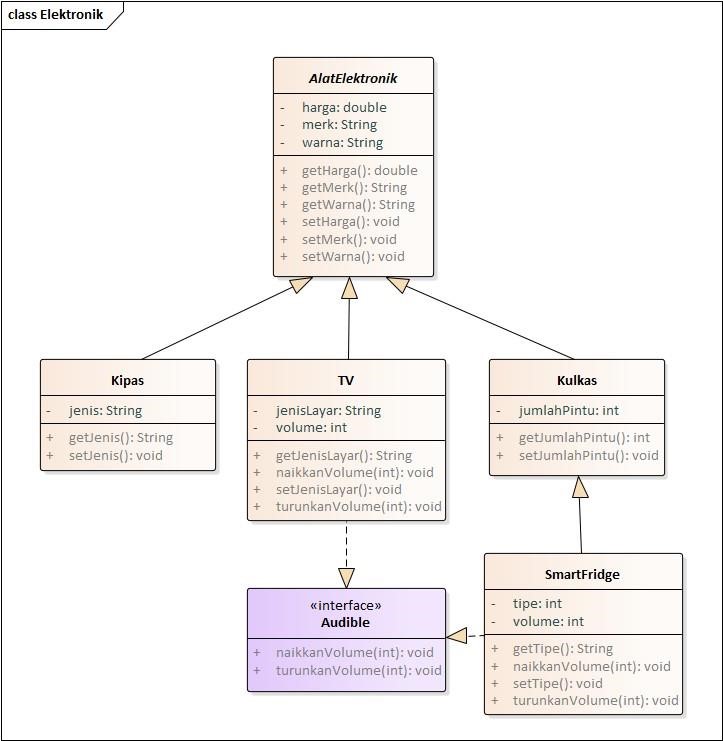
Sementara itu, jika beberapa class memiliki **common behavior** (perilaku atau kapabilitas yang sama) kita bisa menggunakan interface untuk memberikan panduan mengenai method apa saja yang perlu diimplementasikan untuk memenuhi kapabilitas tertentu. Misalnya jika suatu class memiliki kapabilitas untuk dapat berpindah atau

Movable, seharusnya dia memiliki method moveLeft(), moveRight(),

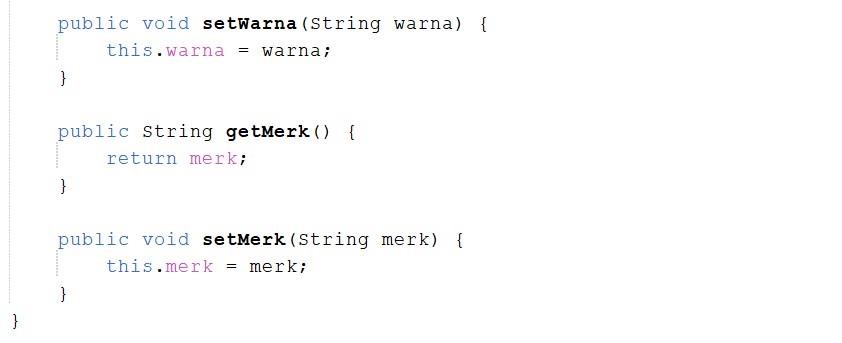
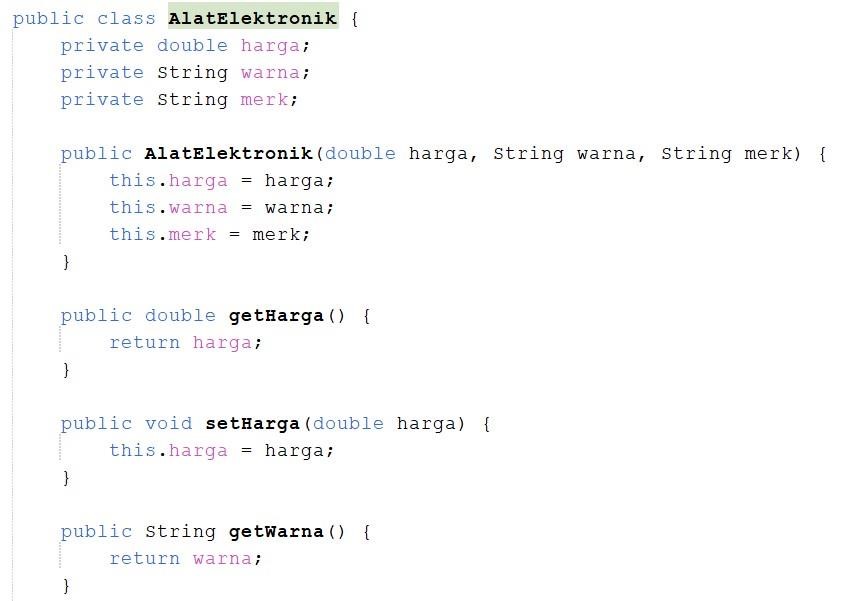
moveDown, moveUp. Sekumpulan method dalam interface ini akan menjadi panduan atau pedoman, bahwa jika selanjutnya ada pengembangan atau penambahan game item lain dan item tersebut dapat bergerak juga maka method-method tersebut harus diimplementasikan dalam class nya.

# C. PERCOBAAN

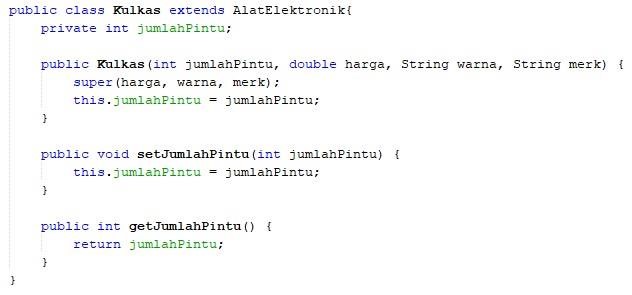
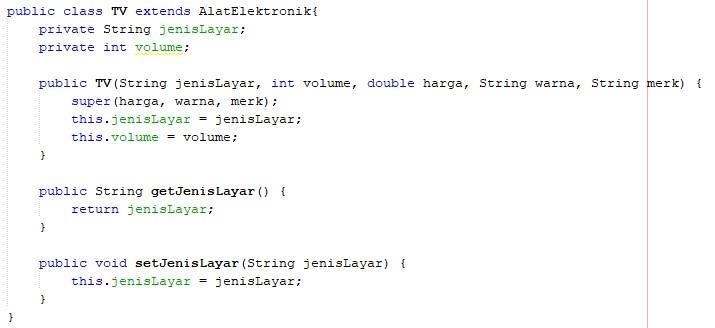
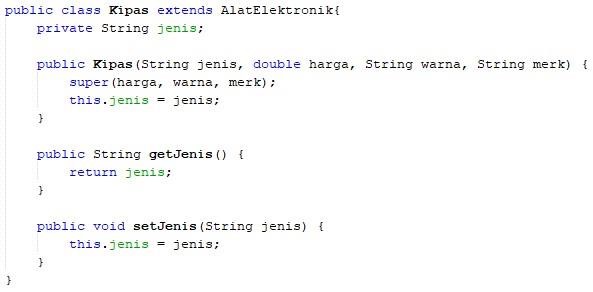
Implementasikan class diagram berikut ke dalam kode program.



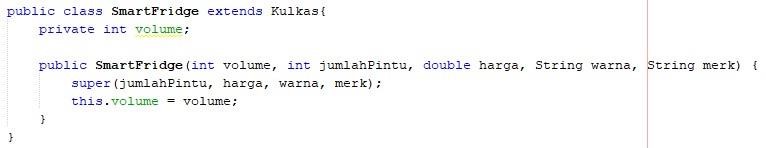
1. Buat project baru dengan nama InterfaceLatihan (boleh disesuaikan)
2. Pada sebuah package, buatlah abstract class AlatElektronik



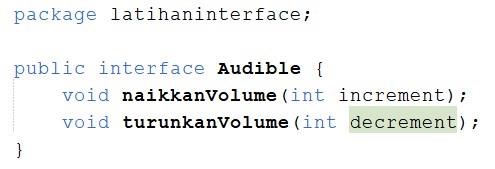
1. Selanjutnya buatlah subclass dari AlatElektronik, yaitu Kipas, TV, dan Kulkas sebagai berikut.



1. Buatlah class SmartFridge yang merupakan sublass dari class Kulkas



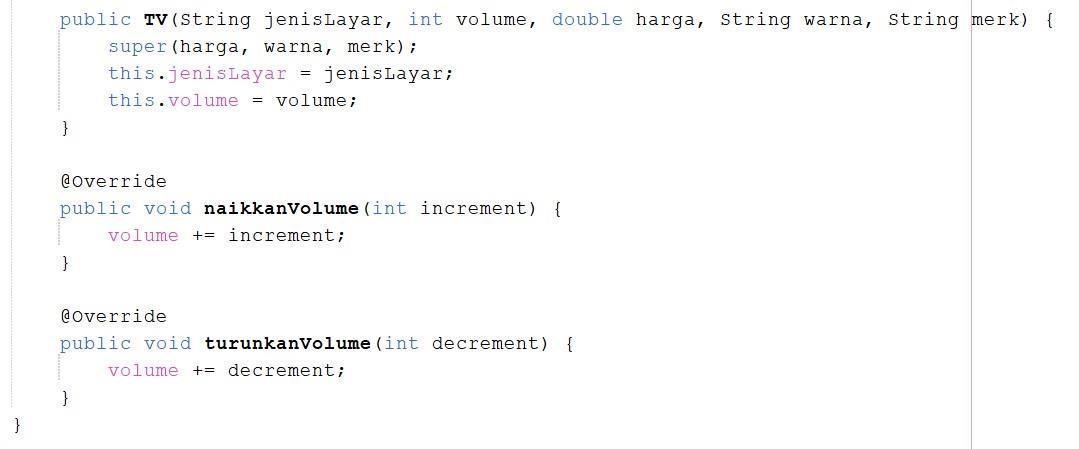
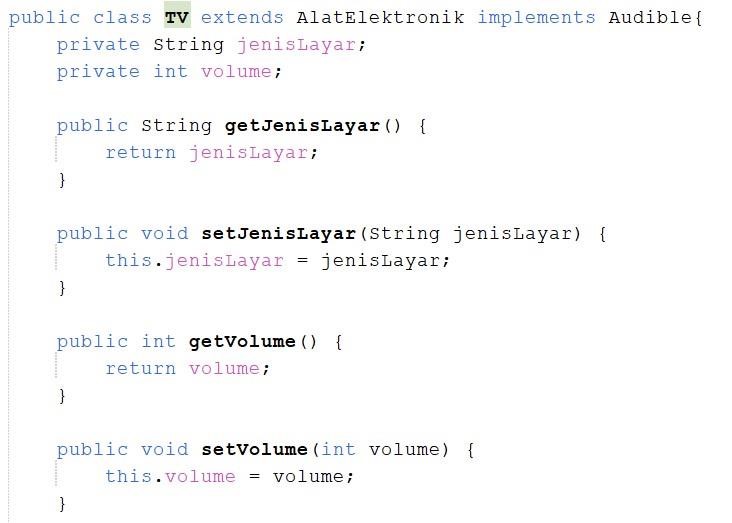
1. Beberapa dari alat elektronik dapat mengeluarkan suara. Kapabilitas ini kita buat ke dalam kode program dengan interface Audible dengan method naikkanVolume() dan turunkanVolume() sebagai berikut

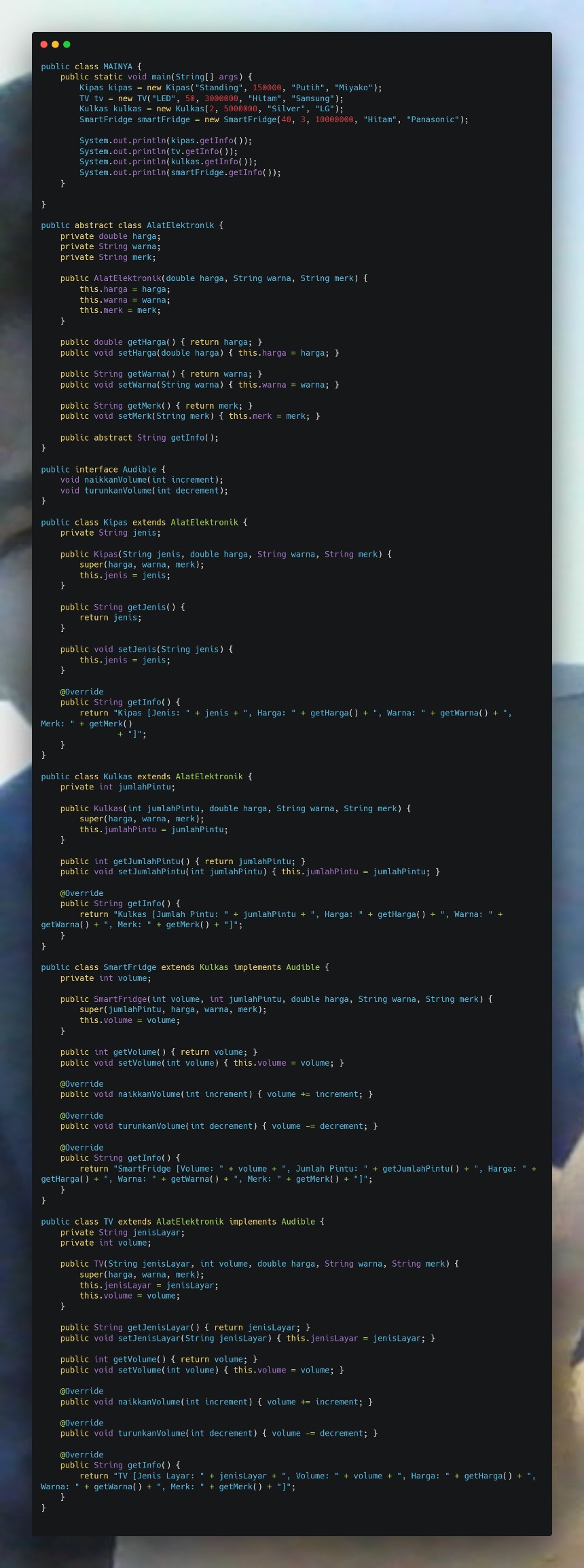


1. Ubah class TV untuk meng-impelement interface Audible

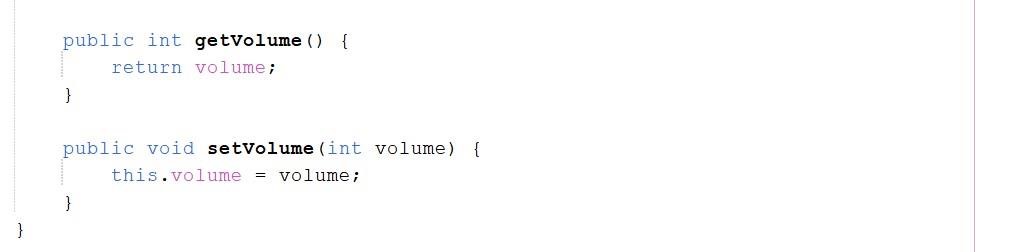
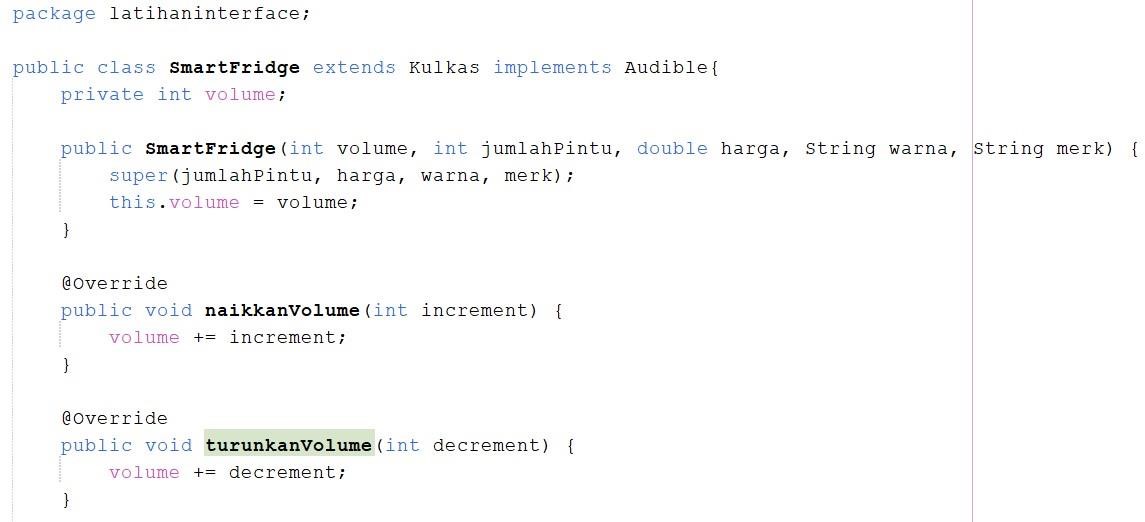


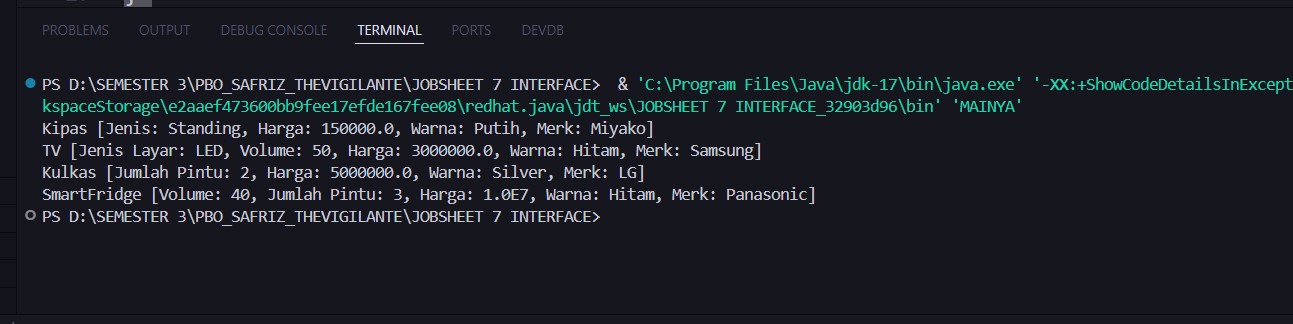
1. Implementasi abstract method pada interface Audible pada class TV





1. Lakukan hal yang sama pada class SmartFridge





# D. PERTANYAAN 2

1. Mengapa terjadi error pada langkah 5?

Error pada langkah 5 mungkin terjadi jika Anda mencoba menginstansiasi interface

**Audible** atau jika kelas yang mengimplementasikan **Audible** (misalnya, **TV** atau

**SmartFridge**) tidak menyediakan implementasi untuk semua metode di dalamnya. Di Java, interface hanya mendeklarasikan metode tanpa implementasi, sehingga setiap kelas konkret yang mengimplementasikannya harus mengimplementasikan semua metode tersebut.

1. Mengapa Audible tidak dapat dibuat sebagai class?

**Audible** tidak bisa dijadikan sebagai class karena interface ini berfungsi sebagai kontrak untuk perilaku tertentu (kemampuan untuk mengatur volume). Jika **Audible** dijadikan class, maka kelas lain yang ingin menggunakan fitur ini harus mewarisi dari **Audible**, yang akan membatasi fleksibilitas karena Java hanya mendukung pewarisan tunggal. Sebagai interface, **Audible** bisa diimplementasikan oleh banyak kelas tanpa mengganggu hierarki pewarisan utama.

1. Mengapa method dalam interface Audible tidak memiliki access level modifier?

Dalam interface, semua metode secara default bersifat public dan abstract. Ini karena interface bertujuan untuk mendefinisikan kontrak publik bagi kelas yang mengimplementasikannya, dan metode tersebut akan diakses oleh kelas lain yang mengimplementasikannya. Oleh karena itu, access level modifier seperti public tidak perlu ditulis secara eksplisit.

1. Method naikkanVolume() dan turunkanVolume() memilki implementasi yang sama pada TV dan SmartFridge(), mengapa tidak langsung diimplementasikan pada interface Audible()?

Sebelum Java 8, interface tidak bisa memiliki implementasi metode, sehingga semua metode dalam interface harus diimplementasikan oleh kelas yang menggunakannya. Sejak Java 8, interface bisa memiliki implementasi default dengan keyword default. Namun, biasanya interface hanya digunakan untuk mendeklarasikan metode tanpa implementasi untuk memisahkan kontrak perilaku dari implementasi spesifik.

1. Method naikkanVolume() dan turunkanVolume() memilki implementasi yang sama pada

TV dan SmartFridge(), mengapa tidak langsung diimplementasikan pada class AlatElektronik?

Tidak semua **AlatElektronik** memiliki fitur volume (misalnya, **Kipas** tidak memiliki volume). Oleh karena itu, metode naikkanVolume() dan turunkanVolume() tidak relevan untuk semua turunan **AlatElektronik**. Metode ini hanya diimplementasikan pada kelas yang membutuhkan fitur volume, seperti **TV** dan **SmartFridge**.

1. Semua yang Audible seharusnya memiliki nilai volume, mengapa atribut volume tidak dideklarasikan dalam interface Audible()?

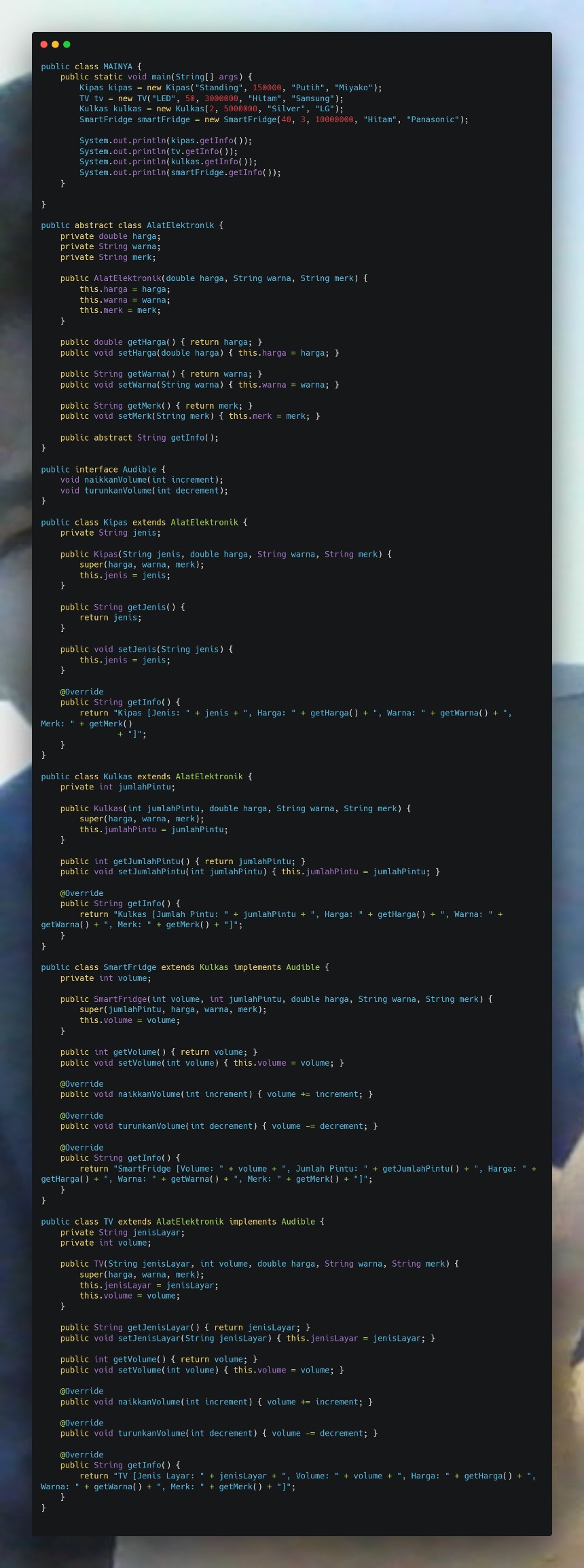
Interface tidak dapat mendeklarasikan atribut instance karena interface bertujuan untuk mendefinisikan kontrak perilaku, bukan untuk menyimpan state atau data. Setiap kelas yang mengimplementasikan interface harus mendefinisikan atribut volume secara terpisah jika diperlukan. Dengan cara ini, setiap kelas bisa mengatur bagaimana volume digunakan atau disimpan tanpa bergantung pada implementasi yang ditetapkan oleh interface.

1. Apa fungsi dari interface?

Interface berfungsi untuk mendefinisikan kontrak atau perilaku yang harus dimiliki oleh kelas-kelas yang mengimplementasikannya. Ini memungkinkan pemrogram untuk mengatur kelas-kelas yang tidak memiliki hubungan pewarisan langsung agar bisa berbagi perilaku yang sama. Dengan menggunakan interface, kita bisa mencapai polimorfisme, di mana berbagai kelas yang mengimplementasikan interface dapat digunakan secara interchangeably berdasarkan perilaku yang didefinisikan di dalam interface tersebut.

1. Buat method getInfo() untuk setiap class. Instansiasi objek dari setiap concrete class pada main class, kemudian tampilkan infonya.

Berikut adalah contoh implementasi method getInfo() di setiap class, beserta kode untuk menginstansiasi dan menampilkan informasi objek di main class:



Di contoh yang diberikan, setiap kelas memiliki metode getInfo() yang berfungsi untuk menampilkan informasi spesifik mengenai objek tersebut. Kelas utama (main class) berperan untuk membuat instance dari setiap kelas konkret dan menampilkan informasi tersebut dengan memanggil metode getInfo().