Jobsheet 7 Interface

Name : Evan Diantha Fafian

Class : SIB 2G

Absent : 09

NIM : 2341760163

# Kompetensi

Setelah menyelesaikan lembar kerja ini mahasiswa diharapkan mampu:

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

# Pendahuluan

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

# 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 meng- implement interface tersebut

# 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.

# C:\Users\Zuraida\Pictures\int.pngNotasi 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

# 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

# Sintaks Interface

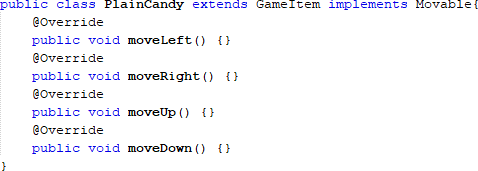
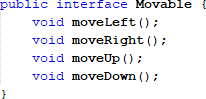
* Untuk mendeklarasikan suatu interface:

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:



# Implementasi Interface

Bila sebuah class mengimplementasikan suatu interface:

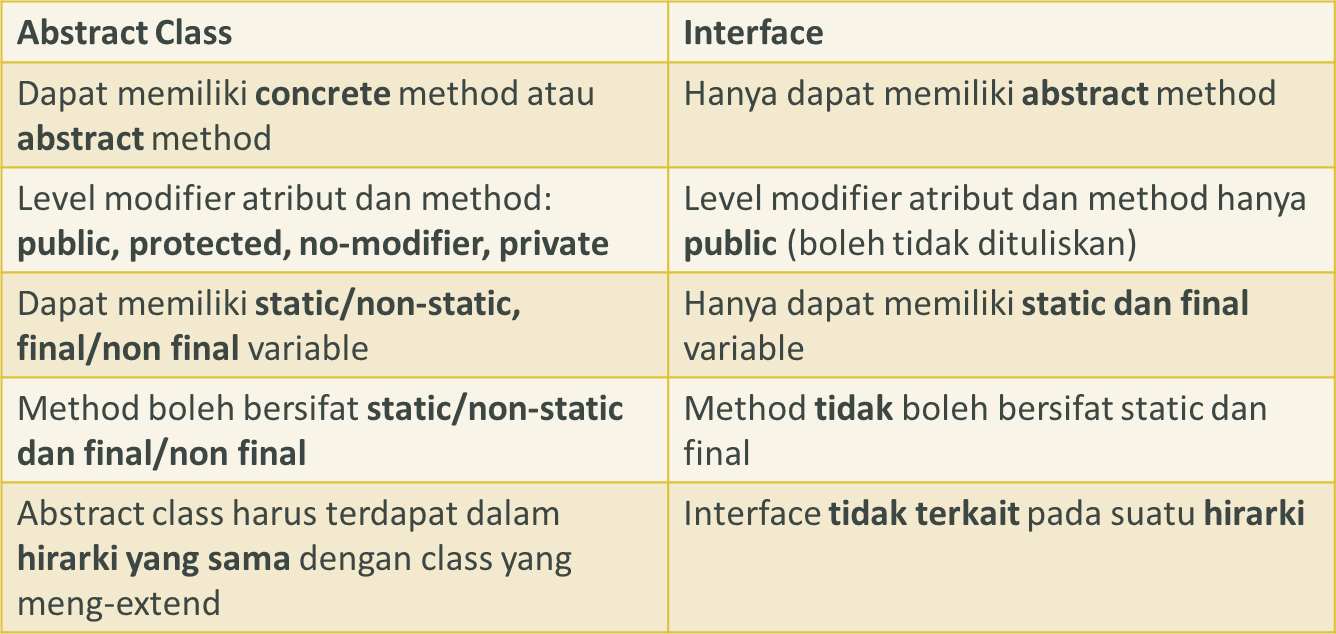
* **Seluruh konstanta** dari interface akan dimiliki oleh class tersebut
* **Seluruh method** pada interface harus diimplementasikan
* Bila class yang meng-implement interface **tidak mengimplementasikan semua method**, maka class tersebut harus dideklarasikan sebagai **abstract class**

# 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

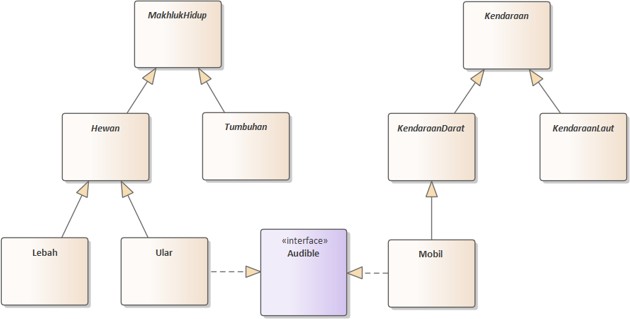
# Perbedaan Abstract Class dan Interface



* 1. **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.



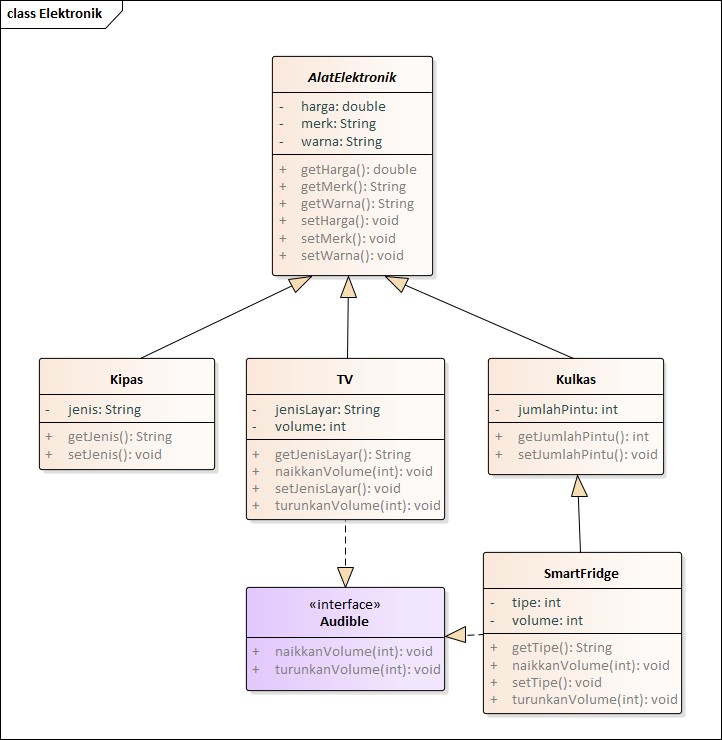
# Penggunaan Abstract Class vs Interface

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.

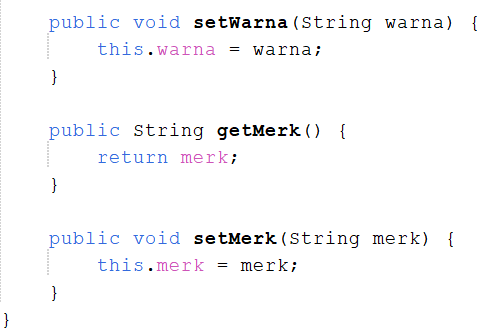
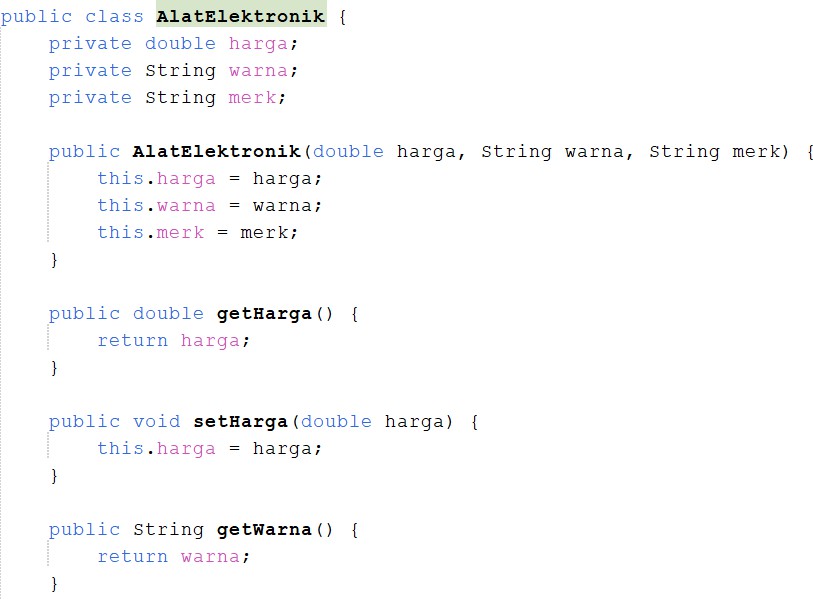
Sementara itu, jika beberapa class memiliki **common behavior** (perilaku atau kapabilitas yang sama) kita bisa menggunakan interface untuk memberikan panduan mengenai method apa saja yang perlu diimplementasikan untuk memenuhi kapabilitas tertentu. Misalnya jika suatu class memiliki kapabilitas untuk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

# PERCOBAAN

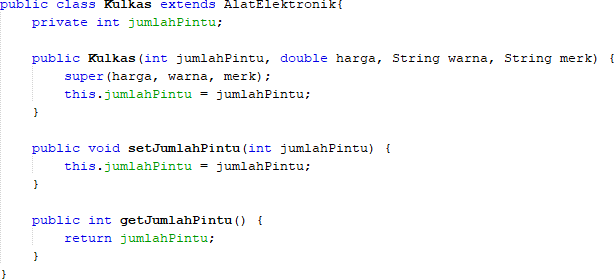
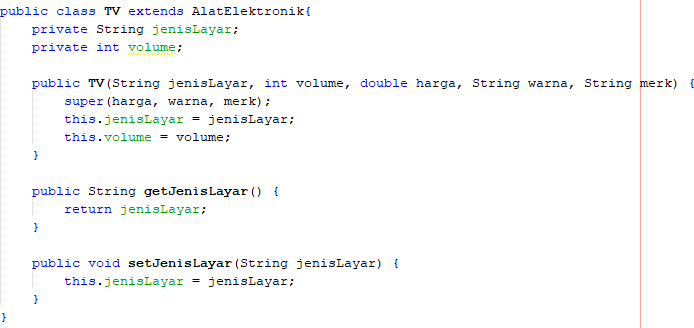
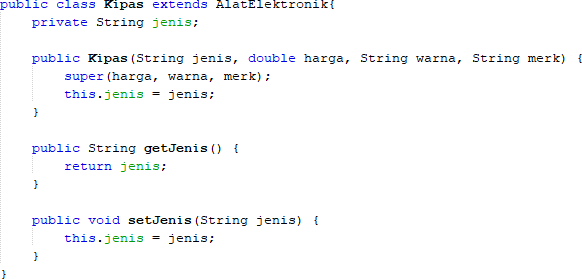
Implementasikan class diagram berikut ke dalam kode program.



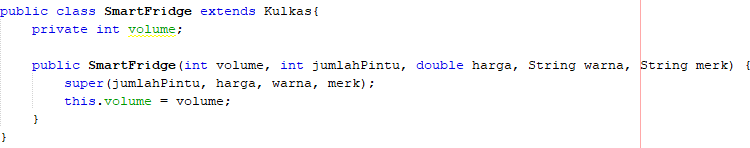
* 1. Buat project baru dengan nama InterfaceLatihan (boleh disesuaikan)
  2. Pada sebuah package, buatlah abstract class AlatElektronik



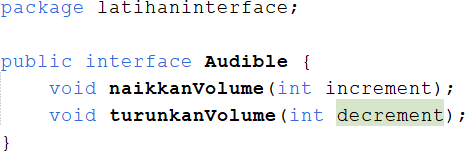
* 1. Selanjutnya buatlah subclass dari AlatElektronik, yaitu Kipas, TV, dan Kulkas sebagai berikut.



* 1. Buatlah class SmartFridge yang merupakan sublass dari class Kulkas



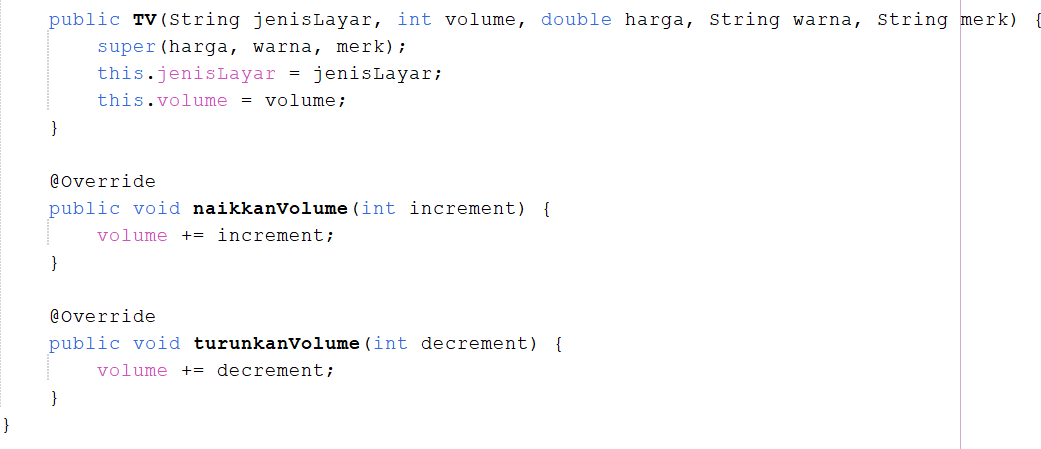
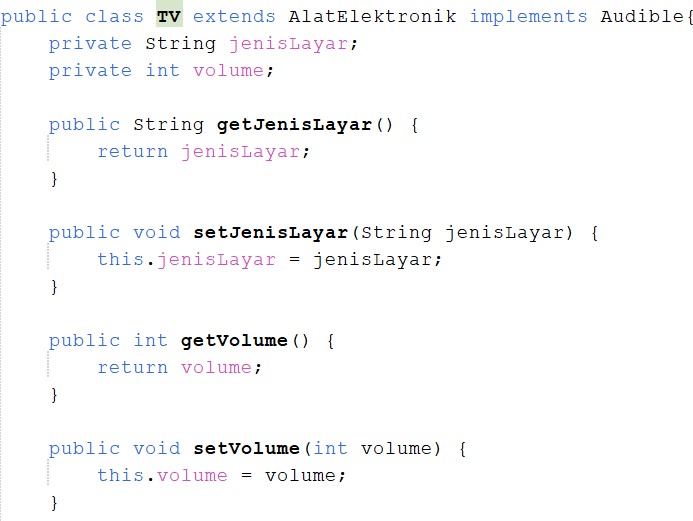
* 1. Beberapa dari alat elektronik dapat mengeluarkan suara. Kapabilitas ini kita buat ke dalam kode program dengan interface Audible dengan method naikkanVolume() dan turunkanVolume() sebagai berikut



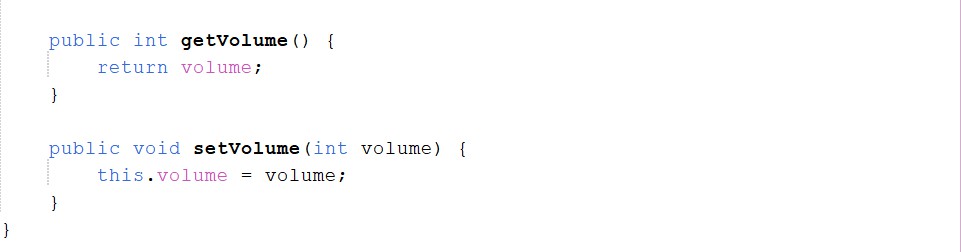
* 1. Ubah class TV untuk meng-impelement interface Audible



* 1. Implementasi abstract method pada interface Audible pada class TV

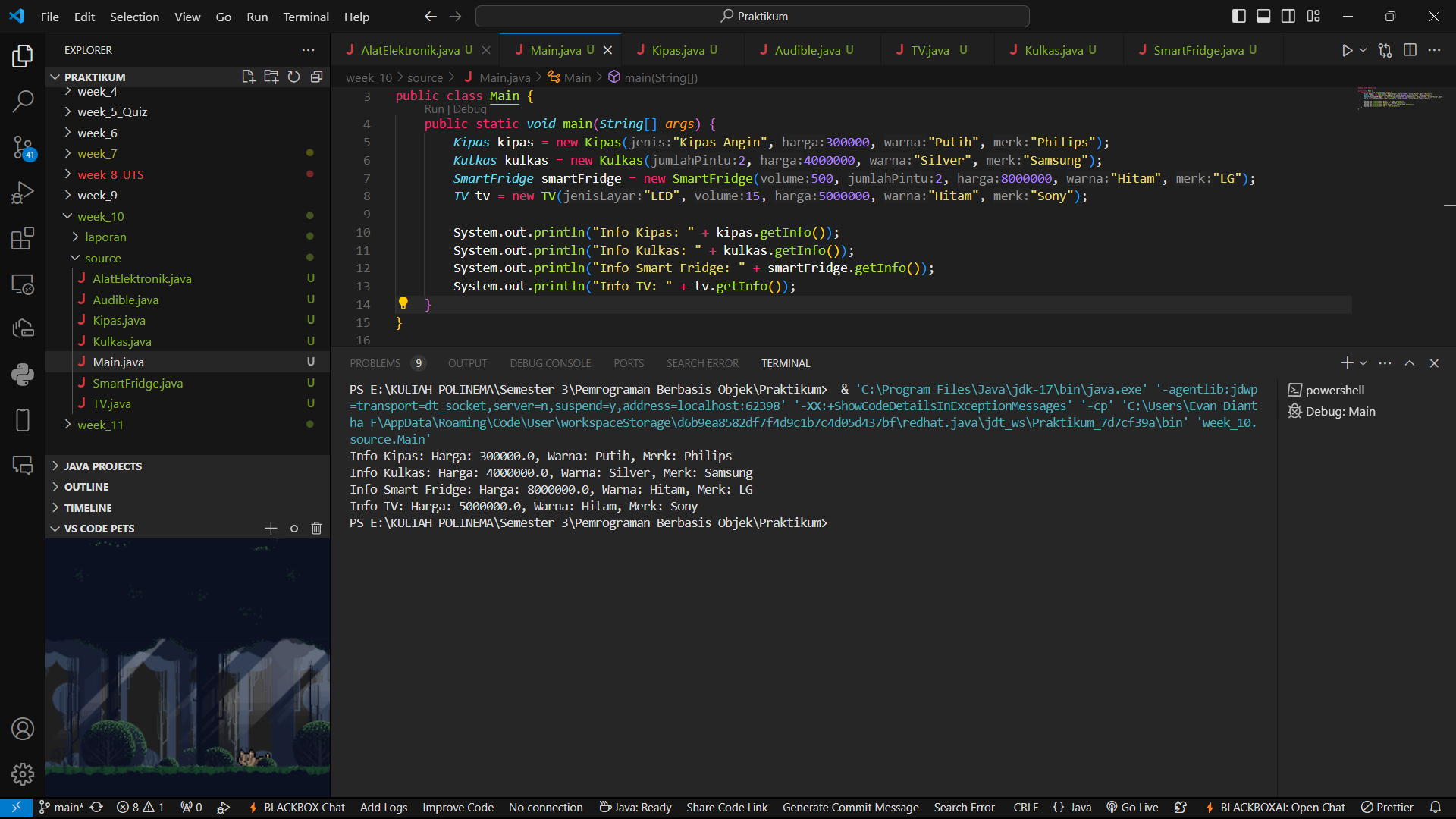


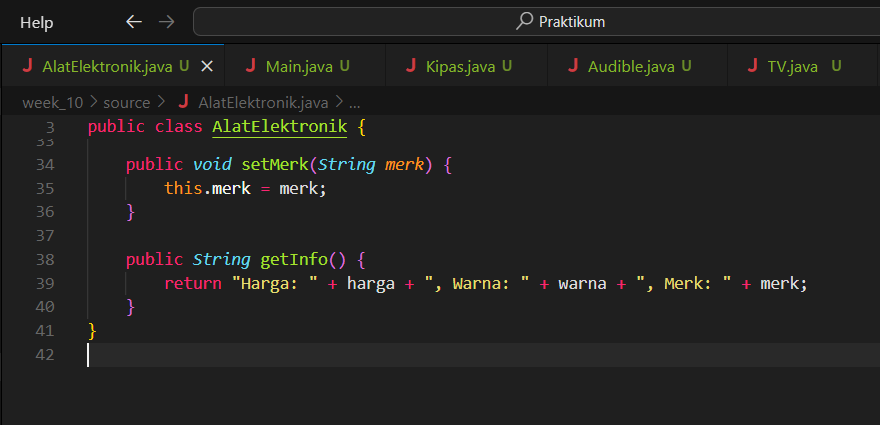
* 1. Lakukan hal yang sama pada class SmartFridge



# PERTANYAAN 2

* 1. Mengapa terjadi error pada langkah 5?
* Karena seluruh method yang ada pada interface Audible belum diterapkan pada kelas TV.
  1. Mengapa Audible tidak dapat dibuat sebagai class?
* Memang, tujuan dari Audible sebagai interface adalah untuk memberikan kemampuan yang dapat ditambahkan ke berbagai kelas tanpa terikat pada struktur tertentu.
  1. Mengapa method dalam interface Audible tidak memiliki access level modifier?
* Karena sudah dikontrakkan untuk diakses oleh class yang mengimplementasikannya, tingkat modifikasi akses untuk metode dalam interface tidak dapat diubah menjadi private atau protected.
  1. Method naikkanVolume() dan turunkanVolume() memilki implementasi yang sama pada TV dan SmartFridge(), mengapa tidak langsung diimplementasikan pada interface Audible()?
* untuk memastikan bahwa interface hanya beroperasi sebagai kontrak dan cetak biru untuk kelas yang menggunakannya.
  1. Method naikkanVolume() dan turunkanVolume() memilki implementasi yang sama pada TV dan SmartFridge(), mengapa tidak langsung diimplementasikan pada class AlatElektronik?
* Tidak semua alat elektronik dapat mengeluarkan suara atau mengubah volume.
  1. Semua yang Audible seharusnya memiliki nilai volume, mengapa atribut volume tidak dideklarasikan dalam interface Audible()?
* Variable interface adalah public, static, dan final, yang berarti mereka hanya dapat menyimpan konstan.
  1. Apa fungsi dari interface?
* Kontrak antarmuka menetapkan metode yang harus digunakan pada class yang sama.
  1. Buat method getInfo() untuk setiap class. Instansiasi objek dari setiap concrete class pada main class, kemudian tampilkan infonya.





**GITHUB**

<https://github.com/rankadian/PEMROGRAMAN-BERBASIS-OBJEK.git>