

Jobsheet 7

Interface

A. Competence

After completing this worksheet students are expected to be able to:

1. Explain the purpose and objectives of using the interface;
2. Implement interfaces in program creation.

B. Introduction

An interface is a set of bodyless methods (abstract methods) that are related to each other.

1. Characteristics:

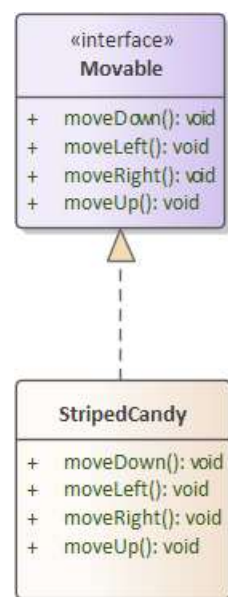
- a. Generally consists of abstract methods
- b. Always declared using the interface keyword.
- c. Implemented using the implements keyword
- d. Interface cannot be instantiated, it can only be instantiated through the class that implements the interface.

2. Usability:

It acts as a **contract/condition** that contains a **set of** interrelated **behaviors/methods** to fulfill a **capability**. In other words, interfaces provide guidance on what methods need to be implemented to fulfill a particular capability.

3. Interface Class Diagram Notation

- Interface names **are not** italicized
- Description <<interface>> above the interface name
- Method names may or may not be italicized
- Implements are denoted by a dotted arrow line



4. Interface Writing Rules

- The structure is almost the same as the class
- There are some rules in writing interfaces:
 - a. **Does not** have concrete methods (ordinary methods that are not abstract)
 - b. **Does not** have a constructor
 - c. Can have attributes, but can only be public, static, final

5. Interface Syntax

- To declare an interface:
`public interface <Interface Name>`
- To implement the interface:
`public class <NameClass> implements <NameInterface>`
- Interface names should be in the form of **adjectives** if they represent capabilities. Can also use **nouns**
- Example:

```
public interface Movable {  
    void moveLeft();  
    void moveRight();  
    void moveUp();  
    void moveDown();  
}
```

```
public class PlainCandy extends GameItem implements Movable{  
    @Override  
    public void moveLeft() {}  
    @Override  
    public void moveRight() {}  
    @Override  
    public void moveUp() {}  
    @Override  
    public void moveDown() {}  
}
```

6. Interface Implementation

When a class implements an interface:

- **All constants** of the interface will be owned by the class
- **All methods** in the interface must be implemented
- If a class that **implements** an interface **does not** implement **all methods**, then the class must be declared as an **abstract class**.

7. Multiple Interface

- A class can implement multiple interfaces
- If a class is a subclass and **implements** an interface, then the **extends keyword precedes implements**.
- Example:

```
public class PlainCandy extends GameItem implements Crushable, Movable
```

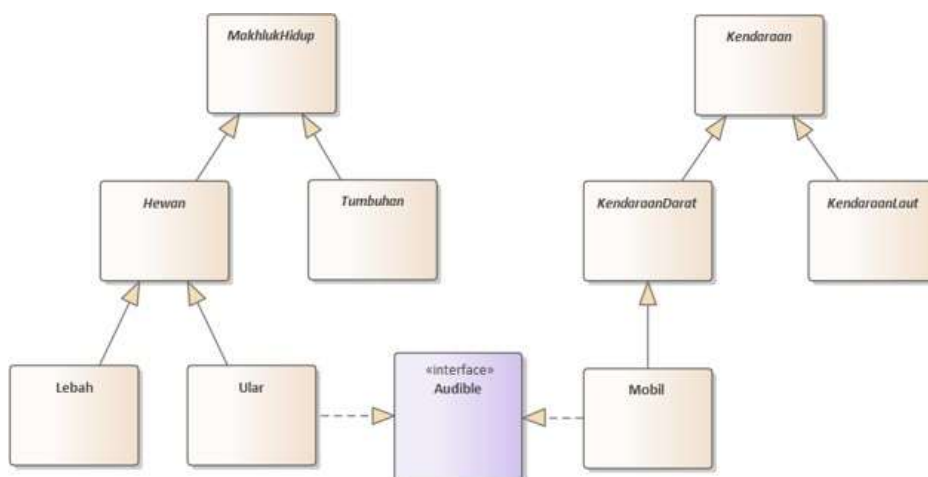
8. Difference between Abstract Class and Interface

Abstract Class	Interface
Dapat memiliki concrete method atau abstract method	Hanya dapat memiliki abstract method
Level modifier atribut dan method: public, protected, no-modifier, private	Level modifier atribut dan method hanya public (boleh tidak dituliskan)
Dapat memiliki static/non-static, final/non final variable	Hanya dapat memiliki static dan final variable
Method boleh bersifat static/non-static dan final/non final	Method tidak boleh bersifat static dan final
Abstract class harus terdapat dalam hirarki yang sama dengan class yang meng-extend	Interface tidak terkait pada suatu hirarki

9. Interface not bound to hierarchy

A class in java can only extend or subclass directly from **one** superclass. As a result, the class will be bound to a certain hierarchy. For example, the Bee class is a subclass of Animal while the Animal class itself is a subclass of Living Things. This restriction of 1 parent class directly causes the Bee class to be bound to the living creature hierarchy and cannot be related to other hierarchies.

Meanwhile, interfaces are not bound to a hierarchy. Interfaces are created "off-the-shelf" without relying on a hierarchy. Suppose there is an Audible interface, it can be implemented in any class from any hierarchy. For example, if class Snake can make noise, this class can implement the Audible interface. Similarly, the Car class from the vehicle hierarchy can also implement the Audible interface.



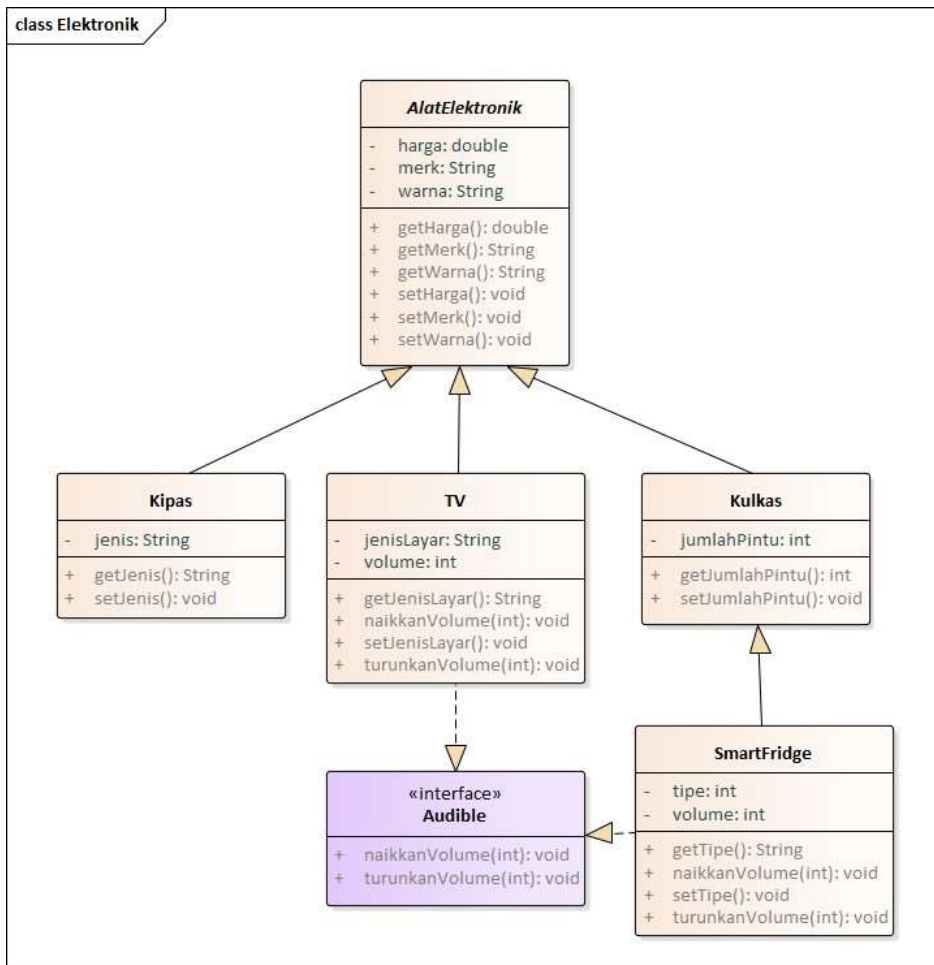
10. Use of Abstract Class vs Interface

Abstract classes can have attributes (instance variables), which are variables that belong to a particular object. These attributes and methods (if the access level modifier is appropriate) will be inherited by its subclasses. Therefore, if a class has **common properties (and methods)** then it is better to create an abstract class as a generalization. For example, there are several classes PlainCandy, StripedCandy, RainbowChocoCandy, Wall etc. which are types of items in the game with the same attributes, such as positionX, positionY, and iconName, we should create an abstract class Animal as a generalization of these classes.

Meanwhile, if several classes have a **common** behavior or capability, we can use interfaces to provide guidance on what methods need to be implemented to fulfill a particular capability. For example, if a class has the capability to be movable, it should have moveLeft(), moveRight(), moveDown, moveUp methods. A set of methods in this interface will be a guide or guideline, that if there is further development or addition of other game items and the item can move as well then these methods must be implemented in the class.

C. TRIAL

Implement the following class diagram into program code.



1. Create a new project with the name InterfaceExercise (can be customized)
2. In a package, create an abstract class ToolsElectronics

```
public class AlatElektronik {  
    private double harga;  
    private String warna;  
    private String merk;  
  
    public AlatElektronik(double harga, String warna, String merk) {  
        this.harga = harga;  
        this.warna = warna;  
        this.merk = merk;  
    }  
  
    public double getHarga() {  
        return harga;  
    }  
  
    public void setHarga(double harga) {  
        this.harga = harga;  
    }  
  
    public String getWarna() {  
        return warna;  
    }  
  
    public void setWarna(String warna) {  
        this.warna = warna;  
    }  
  
    public String getMerk() {  
        return merk;  
    }  
  
    public void setMerk(String merk) {  
        this.merk = merk;  
    }  
}
```

```
AlatElektronik11.java x Kipas11.java U TV11.java U Kulkas11.java U
InterfaceLatihan > AlatElektronik11.java > AlatElektronik11 > AlatElektronik11(double, String, String)
1 package InterfaceLatihan;
2
3 public abstract class AlatElektronik11 {
4     private double harga;
5     private String warna;
6     private String merk;
7
8     public AlatElektronik11(double harga, String warna, String merk) {
9         this.harga = harga;
10        this.warna = warna;
11        this.merk = merk;
12    }
13
14    public double getHarga() {
15        return harga;
16    }
17
18    public void setHarga(double harga) {
19        this.harga = harga;
20    }
21
22    public String getWarna() {
23        return warna;
24    }
25
26    public void setWarna(String warna) {
27        this.warna = warna;
28    }
29
30    public String getMerk() {
31        return merk;
32    }
33
34    public void setMerk(String merk) {
35        this.merk = merk;
36    }
37 }
38
```

3. Next, create subclasses of Electronic Devices, namely Fan, TV, and Refrigerator as follows.

```
public class Kipas extends AlatElektronik{
    private String jenis;

    public Kipas(String jenis, double harga, String warna, String merk) {
        super(harga, warna, merk);
        this.jenis = jenis;
    }

    public String getJenis() {
        return jenis;
    }

    public void setJenis(String jenis) {
        this.jenis = jenis;
    }
}

public class TV extends AlatElektronik{
    private String jenisLayar;
    private int volume;

    public TV(String jenisLayar, int volume, double harga, String warna, String merk) {
        super(harga, warna, merk);
        this.jenisLayar = jenisLayar;
        this.volume = volume;
    }

    public String getJenisLayar() {
        return jenisLayar;
    }

    public void setJenisLayar(String jenisLayar) {
        this.jenisLayar = jenisLayar;
    }
}

public class Kulkas extends AlatElektronik{
    private int jumlahPintu;

    public Kulkas(int jumlahPintu, double harga, String warna, String merk) {
        super(harga, warna, merk);
        this.jumlahPintu = jumlahPintu;
    }

    public void setJumlahPintu(int jumlahPintu) {
        this.jumlahPintu = jumlahPintu;
    }

    public int getJumlahPintu() {
        return jumlahPintu;
    }
}
```



```
AlatElektronik11.java U Kipas11.java U X TV11.java U Kulkas11.java U
InterfaceLatihan > J Kipas11.java > ...
1 package InterfaceLatihan;
2
3 public class Kipas11 extends AlatElektronik11{
4     private String jenis;
5
6     public Kipas11 (String jenis, double harga, String warna, String merk)
7         super(harga, warna, merk);
8         this.jenis = jenis;
9     }
10
11     public String getJenis() {
12         return jenis;
13     }
14
15     public void setJenis(String jenis) {
16         this.jenis = jenis;
17     }
18 }
19
20
```

```
TV11.java t, U X Kulkas11.java U SmartFridge11.java U Audible11.java U
InterfaceLatihan > J TV11.java > TV11
1 package InterfaceLatihan;
2
3 public class TV11 extends AlatElektronik11 {
4     private String jenisLayar;
5     private int volume;
6
7     public TV11(String jenisLayar, int volume, double harga, String warna,
8         super(harga, warna, merk);
9         this.jenisLayar = jenisLayar;
10        this.volume = volume;
11    }
12
13    public String getJenisLayar() {
14        return jenisLayar;
15    }
16
17    public void setJenisLayar(String jenisLayar) {
18        this.jenisLayar = jenisLayar;
19    }
20 }
```

```
AlatElektronik11.java U Kipas11.java U TV11.java U Kulkas11.java U X SmartFridge11.java U
InterfaceLatihan > J Kulkas11.java > Kulkas11
1 package InterfaceLatihan;
2
3 public class Kulkas11 extends AlatElektronik11 {
4     private int jumlahPintu;
5
6     public Kulkas11(int jumlahPintu, double harga, String warna, String merk) {
7         super(harga, warna, merk);
8         this.jumlahPintu = jumlahPintu;
9     }
10
11     public int getJumlahPintu() {
12         return jumlahPintu;
13     }
14
15     public void setJumlahPintu(int jumlahPintu) {
16         this.jumlahPintu = jumlahPintu;
17     }
18 }
```

4. Create a SmartFridge class that is a subclass of the Refrigerator class

```
public class SmartFridge extends Kulkas{
    private int volume;

    public SmartFridge(int volume, int jumlahPintu, double harga, String warna, String merk) {
        super(jumlahPintu, harga, warna, merk);
        this.volume = volume;
    }
}
```



5. Some of the electronic devices can emit sound. We build this capability into the program code with the Audible interface with the raiseVolume() and lowerVolume() methods as follows

```
package latihaninterface;

public interface Audible {
    void naikkanVolume(int increment);
    void turunkanVolume(int decrement);
}
```



6. Modify the TV class to implement the Audible interface

```
public class TV extends AlatElektronik implements Audible{  
    private String jenisLayar;  
    private int volume;  
}
```

7. Implementation of abstract method on Audible interface in TV class

```
public class TV extends AlatElektronik implements Audible{  
    private String jenisLayar;  
    private int volume;  
  
    public String getJenisLayar() {  
        return jenisLayar;  
    }  
  
    public void setJenisLayar(String jenisLayar) {  
        this.jenisLayar = jenisLayar;  
    }  
  
    public int getVolume() {  
        return volume;  
    }  
  
    public void setVolume(int volume) {  
        this.volume = volume;  
    }  
  
    public TV(String jenisLayar, int volume, double harga, String warna, String merk) {  
        super(harga, warna, merk);  
        this.jenisLayar = jenisLayar;  
        this.volume = volume;  
    }  
  
    @Override  
    public void naikkanVolume(int increment) {  
        volume += increment;  
    }  
  
    @Override  
    public void turunkanVolume(int decrement) {  
        volume -= decrement;  
    }  
}
```

```
→ Jobsheet9 interface

J TV11.java U x J Kulkas11.java U J SmartFridge11.java U J Audible11.java U

InterfaceLatihan > J TV11.java > _

30 public class TV11 extends AlatElektronik11 implements Audible11 {
31     private String jenisLayar;
32     private int volume;
33
34     public String getJenisLayar() {
35         return jenisLayar;
36     }
37
38     public void setJenisLayar(String jenisLayar) {
39         this.jenisLayar = jenisLayar;
40     }
41
42     public int getVolume() {
43         return volume;
44     }
45
46     public void setVolume(int volume) {
47         this.volume = volume;
48     }
49
50     public TV11(String jenisLayar, int volume, double harga, String warna,
51         super(harga, warna, merk);
52         this.jenisLayar = jenisLayar;
53         this.volume = volume;
54     }
55
56     @Override
57     public void naikkanVolume(int increment) {
58         this.volume += increment;
59     }
60
61     @Override
62     public void turunkanVolume(int decrement) {
63         volume -= decrement;
64     }
65 }
```

8. Do the same with the SmartFridge class

```
package latihaninterface;

public class SmartFridge extends Kulkas implements Audible {
    private int volume;

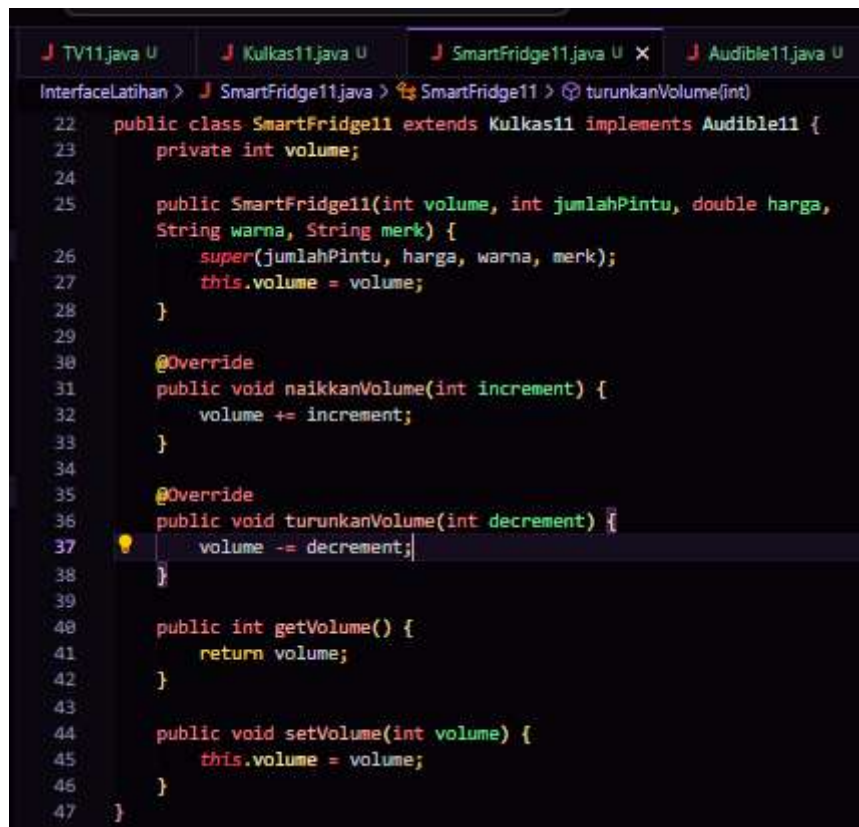
    public SmartFridge(int volume, int jumlahPintu, double harga, String warna, String merk) {
        super(jumlahPintu, harga, warna, merk);
        this.volume = volume;
    }

    @Override
    public void naikkanVolume(int increment) {
        volume += increment;
    }

    @Override
    public void turunkanVolume(int decrement) {
        volume -= decrement;
    }

    public int getVolume() {
        return volume;
    }

    public void setVolume(int volume) {
        this.volume = volume;
    }
}
```



```
TV11.java U  Kulkas11.java U  SmartFridge11.java U X  Audible11.java U
InterfaceLatihan > SmartFridge11.java > SmartFridge11 > turunkanVolume(int)
22  public class SmartFridge11 extends Kulkas11 implements Audible11 {
23      private int volume;
24
25      public SmartFridge11(int volume, int jumlahPintu, double harga,
26          String warna, String merk) {
27          super(jumlahPintu, harga, warna, merk);
28          this.volume = volume;
29      }
30
31      @Override
32      public void naikkanVolume(int increment) {
33          volume += increment;
34      }
35
36      @Override
37      public void turunkanVolume(int decrement) {
38          volume -= decrement;
39      }
40
41      public int getVolume() {
42          return volume;
43      }
44
45      public void setVolume(int volume) {
46          this.volume = volume;
47      }
48  }
```

D. QUESTION 2

1. Why is there an error in step 5?

because `Audible` is declared as an *interface*, not a *class*. In an interface, the methods are just declarations with no implementation.

2. Why can't `Audible` be created as a class?

because it functions as a contract that other classes must follow, such as the `raiseVolume` and `lowerVolume` methods. If a class is created, `Audible` can no longer function as a common interface.

3. Why don't the methods in the `Audible` interface have access level modifiers?

because all methods in an interface are automatically considered `public`. So, even if you don't write `public`, the method can still be accessed by classes that implement the interface.

4. The `raiseVolume()` and `lowerVolume()` methods have the same implementation in `TV` and `SmartFridge()`, why are they not directly implemented in the `Audible()` interface?

because interfaces only declare methods without implementing them. The classes that implement this interface are responsible for writing their own ways of changing the volume.

5. The `raiseVolume()` and `lowerVolume()` methods have the same implementation in `TV` and `SmartFridge()`, why aren't they directly implemented in the `Electronic Devices` class?

because the `Electronic Devices` class is abstract and not all electronic devices have changeable volumes.

6. Everything `Audible` should have a volume value, why is the volume attribute not declared in the `Audible()` interface?

because interfaces only declare methods, not attributes or variables.

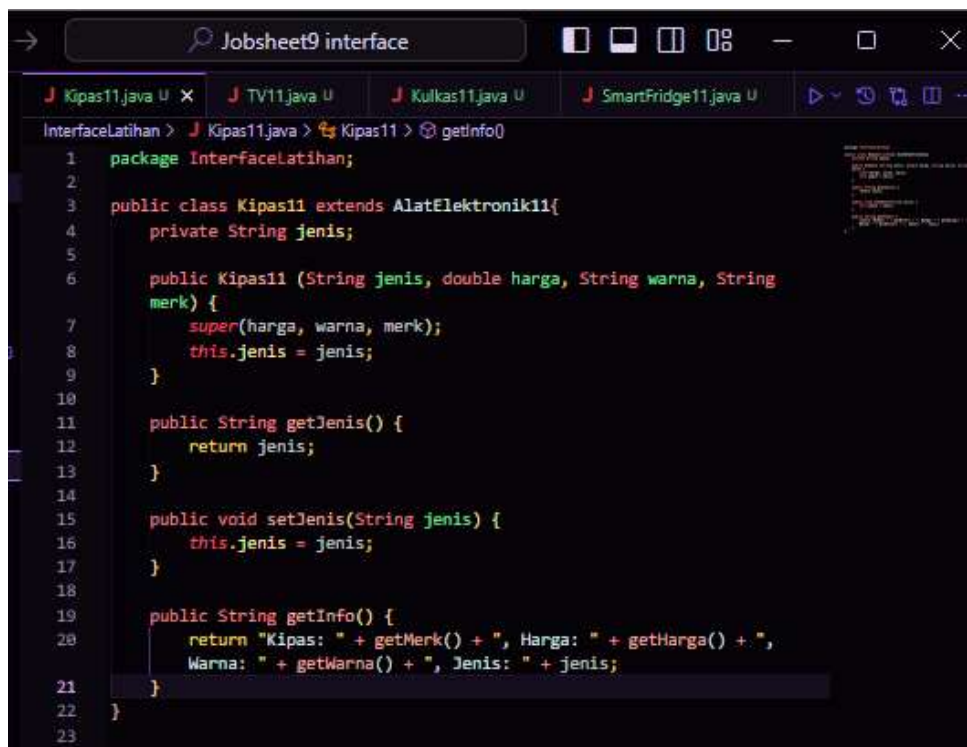
7. What is the function of the interface?

to define the general behavior that must be followed by the classes that implement it.

8. Create a `getInfo()` method for each class. Instantiate the object of each concrete class in the main class, then display the information.

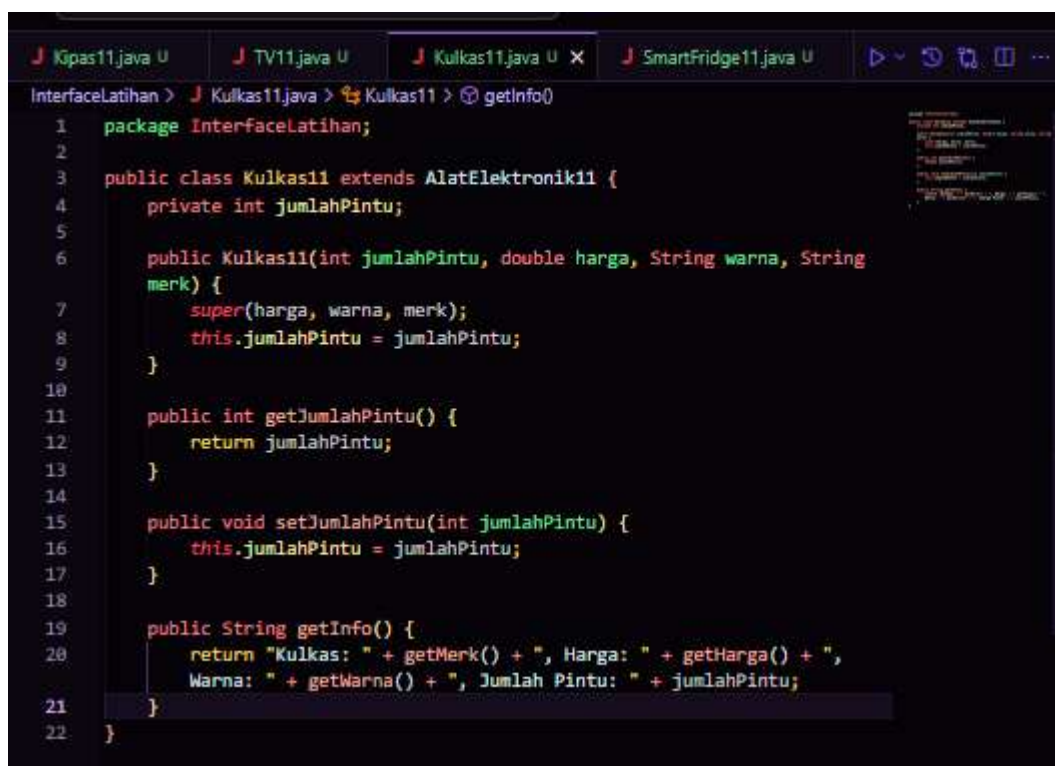
The `getInfo()` method exists in 4 concrete classes (`Fan11`, `Refrigerator11`, `TV11`, and `SmartFridge11`) because they are classes that can be created objects.

a. Fan Class11



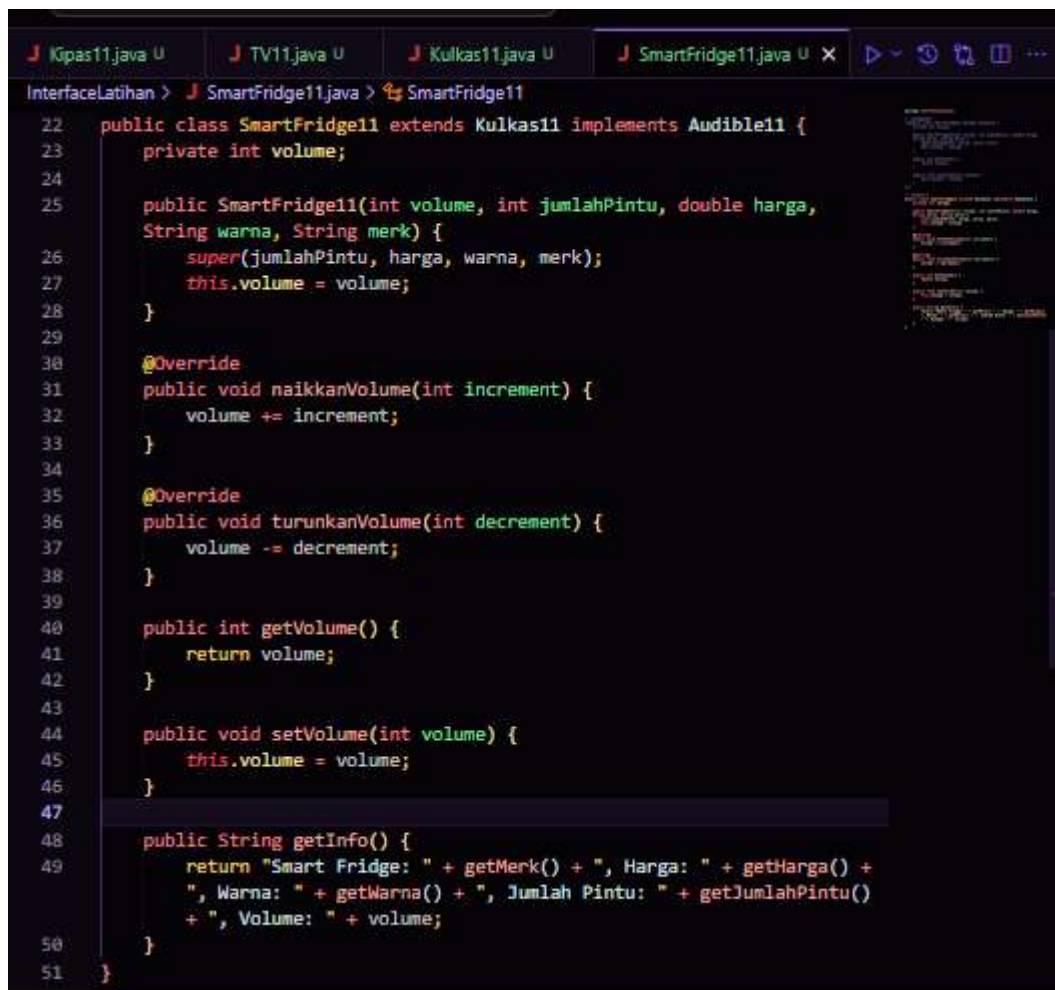
```
Jobsheet9 interface
Kipas11.java U TV11.java U Kulkas11.java U SmartFridge11.java U
InterfaceLatihan > Kipas11.java > Kipas11 > getInfo()
1 package InterfaceLatihan;
2
3 public class Kipas11 extends AlatElektronik11{
4     private String jenis;
5
6     public Kipas11 (String jenis, double harga, String warna, String
7         merk) {
8         super(harga, warna, merk);
9         this.jenis = jenis;
10    }
11
12    public String getJenis() {
13        return jenis;
14    }
15
16    public void setJenis(String jenis) {
17        this.jenis = jenis;
18    }
19
20    public String getInfo() {
21        return "Kipas: " + getMerk() + ", Harga: " + getHarga() + ",
22        Warna: " + getWarna() + ", Jenis: " + jenis;
23    }
24 }
```

b. Refrigerator Class11



```
Kipas11.java U TV11.java U Kulkas11.java U X SmartFridge11.java U
InterfaceLatihan > Kulkas11.java > Kulkas11 > getInfo()
1 package InterfaceLatihan;
2
3 public class Kulkas11 extends AlatElektronik11 {
4     private int jumlahPintu;
5
6     public Kulkas11(int jumlahPintu, double harga, String warna, String
7         merk) {
8         super(harga, warna, merk);
9         this.jumlahPintu = jumlahPintu;
10    }
11
12    public int getJumlahPintu() {
13        return jumlahPintu;
14    }
15
16    public void setJumlahPintu(int jumlahPintu) {
17        this.jumlahPintu = jumlahPintu;
18    }
19
20    public String getInfo() {
21        return "Kulkas: " + getMerk() + ", Harga: " + getHarga() + ",
22        Warna: " + getWarna() + ", Jumlah Pintu: " + jumlahPintu;
23    }
24 }
```

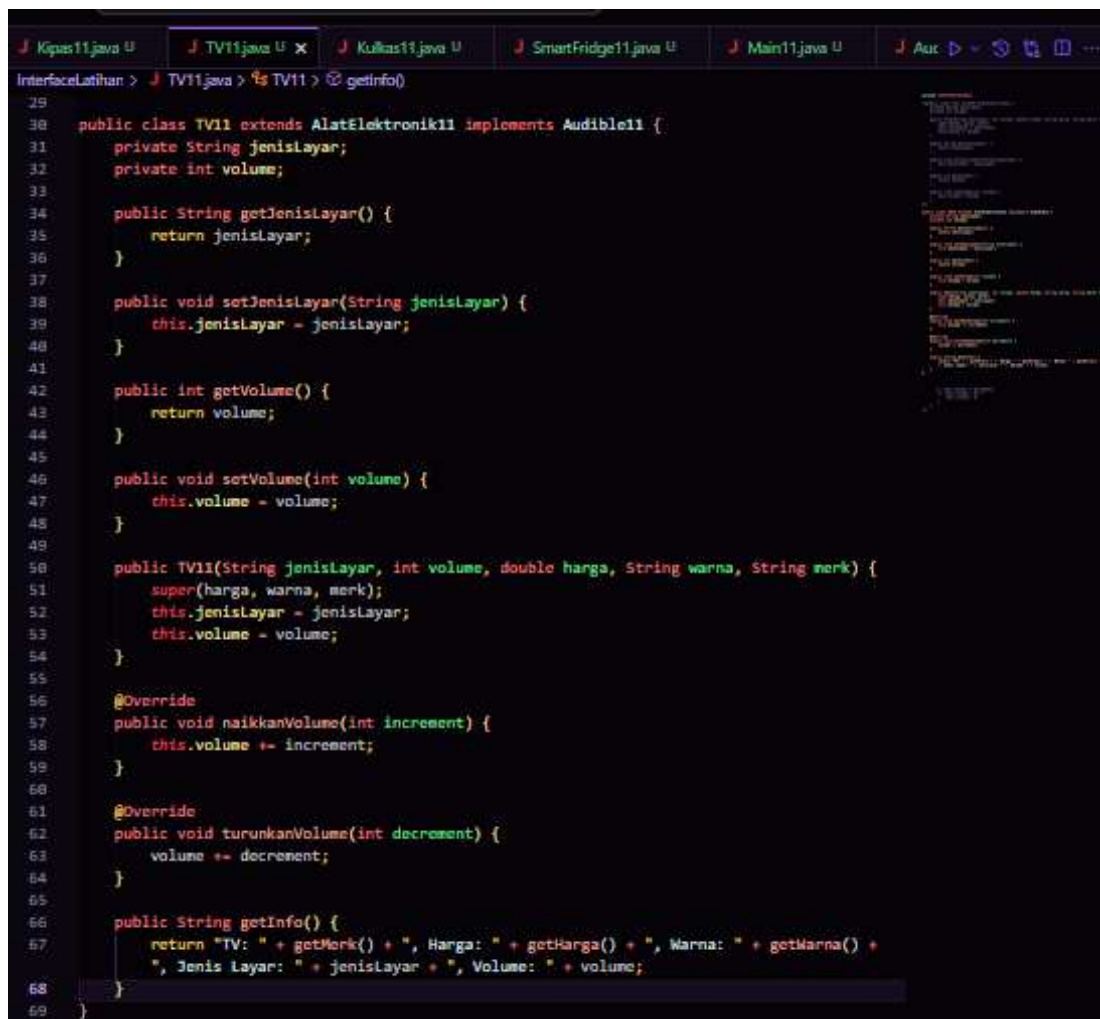
c. SmartFridge11 class



The screenshot shows an IDE with four tabs: Kipas11.java, TV11.java, Kulkas11.java, and SmartFridge11.java. The SmartFridge11.java tab is active, displaying the following Java code:

```
InterfaceLatihan > J SmartFridge11.java > SmartFridge11
22 public class SmartFridge11 extends Kulkas11 implements Audible11 {
23     private int volume;
24
25     public SmartFridge11(int volume, int jumlahPintu, double harga,
26         String warna, String merk) {
27         super(jumlahPintu, harga, warna, merk);
28         this.volume = volume;
29     }
30
31     @Override
32     public void naikanVolume(int increment) {
33         volume += increment;
34     }
35
36     @Override
37     public void turunkanVolume(int decrement) {
38         volume -= decrement;
39     }
40
41     public int getVolume() {
42         return volume;
43     }
44
45     public void setVolume(int volume) {
46         this.volume = volume;
47     }
48
49     public String getInfo() {
50         return "Smart Fridge: " + getMerk() + ", Harga: " + getHarga() +
51             ", Warna: " + getWarna() + ", Jumlah Pintu: " + getJumlahPintu()
52             + ", Volume: " + volume;
53     }
54 }
```


d. TV11 class



```
29
30 public class TV11 extends AlatElektronik11 implements Audible11 {
31     private String jenisLayar;
32     private int volume;
33
34     public String getJenisLayar() {
35         return jenisLayar;
36     }
37
38     public void setJenisLayar(String jenisLayar) {
39         this.jenisLayar = jenisLayar;
40     }
41
42     public int getVolume() {
43         return volume;
44     }
45
46     public void setVolume(int volume) {
47         this.volume = volume;
48     }
49
50     public TV11(String jenisLayar, int volume, double harga, String warna, String merk) {
51         super(harga, warna, merk);
52         this.jenisLayar = jenisLayar;
53         this.volume = volume;
54     }
55
56     @Override
57     public void naikanVolume(int increment) {
58         this.volume += increment;
59     }
60
61     @Override
62     public void turunkanVolume(int decrement) {
63         volume -= decrement;
64     }
65
66     public String getInfo() {
67         return "TV: " + getMerk() + ", Harga: " + getHarga() + ", Warna: " + getWarna() +
68             ", Jenis Layar: " + jenisLayar + ", Volume: " + volume;
69     }
70 }
```

e. Main11 class

```

1 package InterfaceLatihan;
2
3 public class Main11 {
4     public static void main(String[] args) {
5
6         System.out.println();
7         System.out.println
8         (x:"-----");
9
10        Kipas11 kipas = new Kipas11(jenis:"Kipas Angin Berdiri",
11        harga:450000, warna:"Pink", merk:"Panasonic");
12        Kulkas11 kulkas = new Kulkas11(jumlahPintu:2, harga:75000000,
13        warna:"Red", merk:"Samsung");
14        TV11 tv = new TV11(jenisLayar:"LED", volume:15, harga:55000000,
15        warna:"Blue", merk:"Sony");
16        SmartFridge11 smartFridge = new SmartFridge11(volume:300,
17        jumlahPintu:3, harga:100000000, warna:"Hitam", merk:"LG");
18        Kulkas11 kulkasBaru = new Kulkas11(jumlahPintu:4, harga:35000000,
19        warna:"Putih", merk:"Sharp");
20
21        System.out.println(kipas.getInfo());
22        System.out.println(kulkas.getInfo());
23        System.out.println(tv.getInfo());
24        System.out.println(smartFridge.getInfo());
25        System.out.println(kulkasBaru.getInfo());
26
27        System.out.println
28        (x:"-----");
29        System.out.println();
30    }
31}

```

f. Output

```

User\workspaceStorage\7daa3b0ffc8a317f2a8c391d57c9515b\redhat.java\jdt_ws\Jobsheet9
n' 'InterfaceLatihan.Main11'

-----
Kipas: Panasonic, Harga: 450000.0, Warna: Pink, Jenis: Kipas Angin Berdiri
Kulkas: Samsung, Harga: 7.5E7, Warna: Red, Jumlah Pintu: 2
TV: Sony, Harga: 5.5E7, Warna: Blue, Jenis Layar: LED, Volume: 15
Smart Fridge: LG, Harga: 1.0E7, Warna: Hitam, Jumlah Pintu: 3, Volume: 300
Kulkas: Sharp, Harga: 3500000.0, Warna: Putih, Jumlah Pintu: 4
-----

PS D:\SEMESTER 3\PS0\Java\Jobsheet9 interface>

```