

# RUNNING EXAMPLE SOLID

**Interface Segregation Principle (ISP)**

**Tangguh Chairunnisa**  
**21120122140103**



# SOLID

SOLID adalah lima prinsip desain yang membantu pengembang perangkat lunak membuat kode yang lebih mudah dipelihara, fleksibel, dan terstruktur dengan baik. Prinsip ini bertujuan untuk menghindari kode yang tidak terkelola dan kesalahan dalam desain perangkat lunak. Berikut adalah prinsip-prinsipnya:

1. Single Responsibility Principle (SRP)
2. Open/Closed Principle (OCP)
3. Liskov Substitution Principle (LSP)
4. Interface Segregation Principle (ISP)
5. Dependency Inversion Principle (DIP)

# INTERFACE SEGREGATION

## PRINCIPLE (ISP):

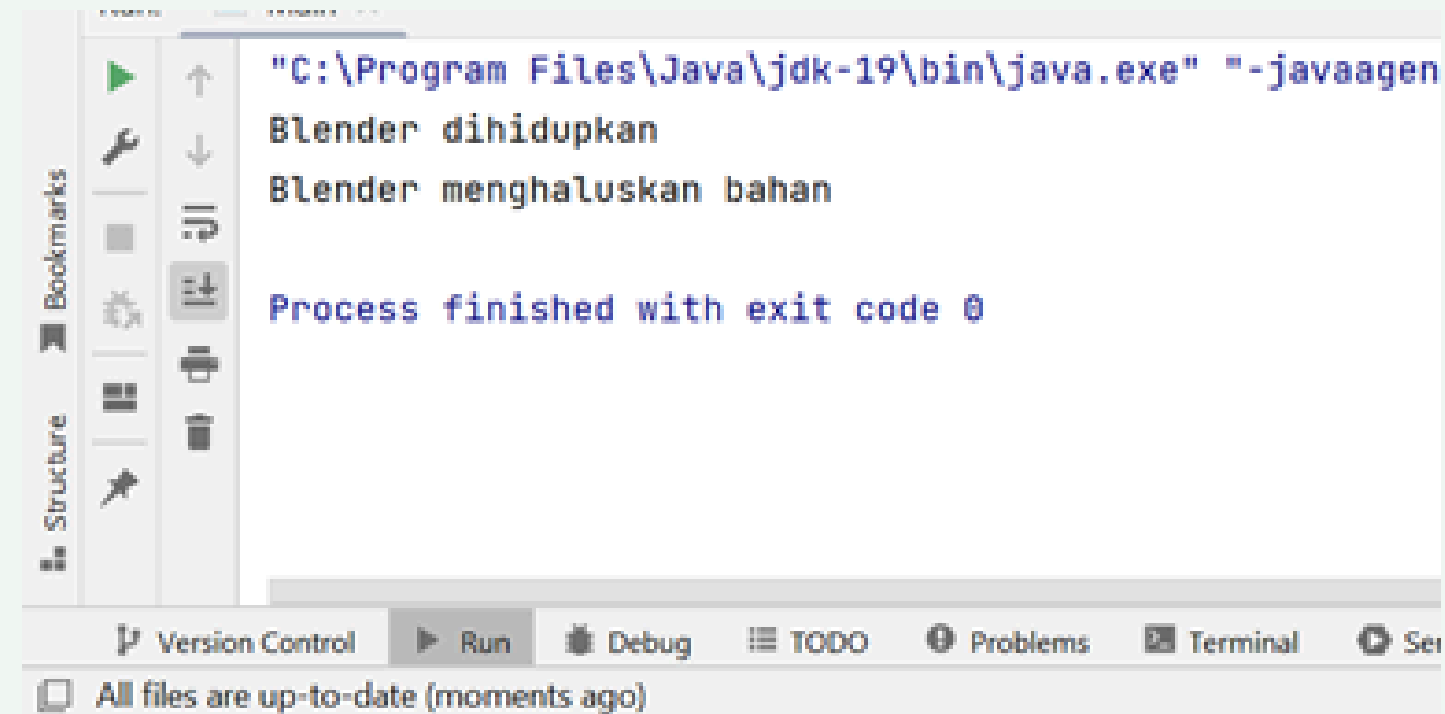
Prinsip ini menyatakan bahwa lebih baik memiliki antarmuka khusus (kecil) yang terpisah untuk setiap fitur daripada satu antarmuka besar dengan banyak fungsi. Dengan cara ini, kelas yang mengimplementasikan antarmuka hanya harus menerapkan fungsi yang mereka butuhkan.

# DEMO: PROBLEM

```
3 interface PeralatanRumahTangga {  
    2 usages 1 implementation  
4     void hidupkan();  
    1 usage 1 implementation  
5     void matikan();  
    no usages 1 implementation  
6     void panaskan();  
    no usages 1 implementation  
7     void bekukan();  
    2 usages 1 implementation  
8     void haluskan();  
9 }  
10  
11 // Blender dipaksa untuk mengimplementasikan metode yang tidak perlu  
    2 usages  
12 class Blender implements PeralatanRumahTangga {  
    2 usages  
13     @Override  
14     public void hidupkan() {  
15         System.out.println("Blender dihidupkan");  
16     }  
17  
    1 usage  
18     @Override  
19     public void matikan() {  
20         System.out.println("Blender dimatikan");  
21     }
```

```
18     @Override  
19     public void matikan() {  
20         System.out.println("Blender dimatikan");  
21     }  
22  
    2 usages  
23     @Override  
24     public void haluskan() {  
25         System.out.println("Blender menghaluskan bahan");  
26     }  
27  
    no usages  
28     @Override  
29     public void panaskan() {  
30         // Tidak diperlukan, tapi harus diimplementasikan  
31     }  
32  
    no usages  
33     @Override  
34     public void bekukan() {  
35         // Tidak diperlukan, tapi harus diimplementasikan  
36     }  
37 }
```

# DEMO: OUTPUT PROBLEM



```
"C:\Program Files\Java\jdk-19\bin\java.exe" "-javaagen
Blender dihidupkan
Blender menghaluskan bahan

Process finished with exit code 0
```

The screenshot shows an IDE interface with a terminal window. The terminal output is as follows:

```
"C:\Program Files\Java\jdk-19\bin\java.exe" "-javaagen
Blender dihidupkan
Blender menghaluskan bahan

Process finished with exit code 0
```

The IDE interface includes a sidebar with 'Bookmarks' and 'Structure' panels, and a bottom toolbar with 'Version Control', 'Run', 'Debug', 'TODO', 'Problems', 'Terminal', and 'Set' buttons. A status bar at the bottom indicates 'All files are up-to-date (moments ago)'.

Pada contoh ini, sebuah antarmuka besar digunakan untuk berbagai peralatan rumah tangga. Kelas Blender dipaksa mengimplementasikan metode yang tidak relevan seperti panaskan dan bekukan. Blender dipaksa untuk mengimplementasikan metode panaskan dan bekukan, padahal metode tersebut tidak relevan dengan fungsi Blender

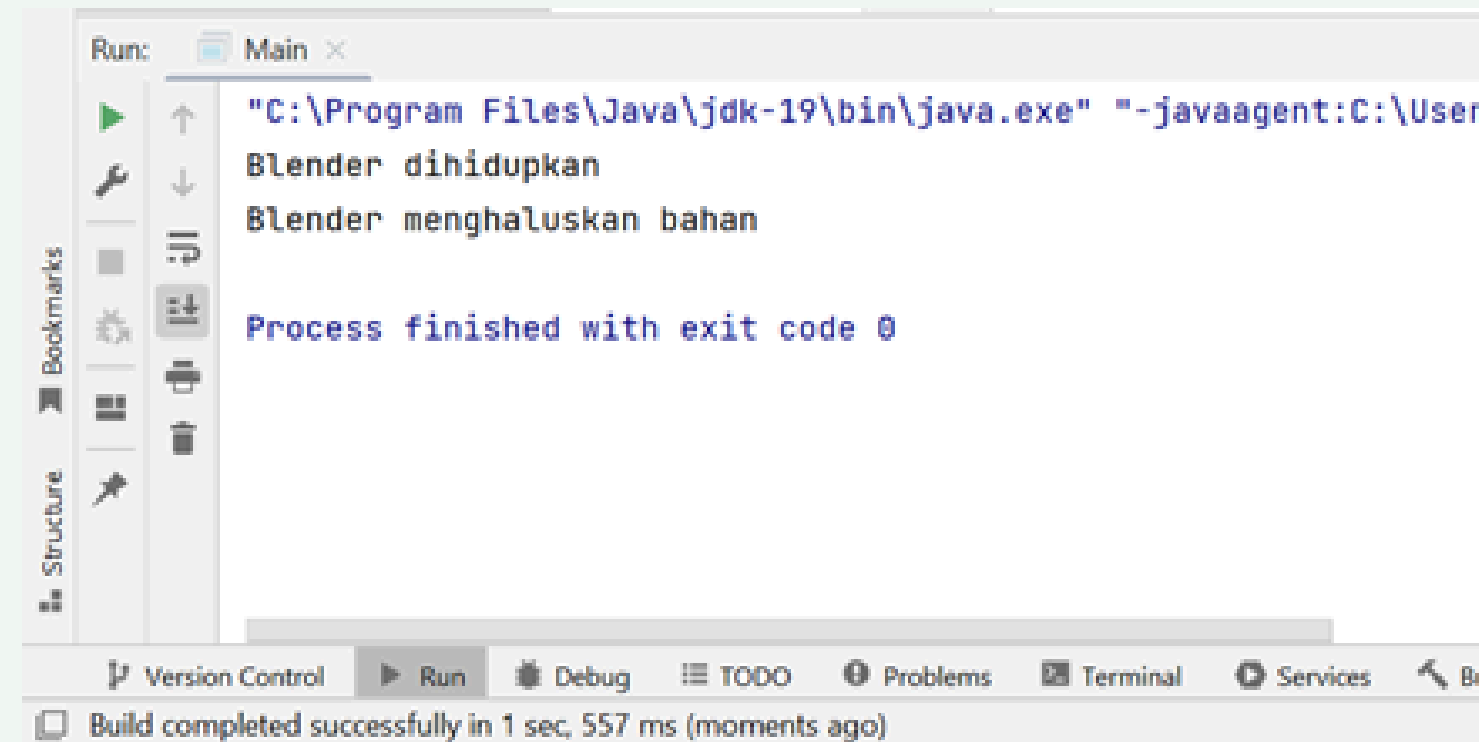


# DEMO: SOLVER

```
3  ↓ interface Mesin {  
    1 usage 1 implementation  
4  ↓   void hidupkan();  
    1 usage 1 implementation  
5  ↓   void matikan();  
6  ↓ }  
7  
    1 usage 1 implementation  
8  ↓ interface BlenderFungsi {  
    1 usage 1 implementation  
9  ↓   void haluskan();  
10 ↓ }  
11  
    no usages  
12 ↓ interface KulkasFungsi {  
    no usages  
13   void bekukan();  
14 ↓ }  
15  
    no usages  
16 ↓ interface KomporFungsi {  
    no usages  
17   void panaskan();  
18 ↓ }  
19  
20 // Blender hanya mengimplementasikan metode yang relevan
```

```
20 // Blender hanya mengimplementasikan metode yang relevan  
21 // 2 usages  
21 ↓ class Blender implements Mesin, BlenderFungsi {  
    1 usage  
22   @Override  
23   public void hidupkan() {  
24     System.out.println("Blender dihidupkan");  
25   }  
26  
    1 usage  
27   @Override  
28   public void matikan() {  
29     System.out.println("Blender dimatikan");  
30   }  
31  
    1 usage  
32   @Override  
33   public void haluskan() {  
34     System.out.println("Blender menghaluskan bahan");  
35   }  
36 }
```

# DEMO: OUTPUT SOLVER



Di sini, antarmuka dipecah menjadi beberapa antarmuka kecil sesuai kebutuhan. Blender hanya mengimplementasikan metode yang relevan. Antarmuka dipecah menjadi BlenderFungsi, KulkasFungsi, dan KomporFungsi. Blender hanya mengimplementasikan antarmuka Mesin dan BlenderFungsi, tanpa harus terlibat dengan metode yang tidak dibutuhkannya seperti panaskan dan bekukan.

THANK  
YOU

