



# Bab 6

## Abstract

# Definisi Abstract Class



Abstract Class adalah sebuah class yang tidak bisa di-instansiasi (tidak bisa dibuat menjadi objek) , berperan sebagai 'kerangka dasar' bagi class turunannya. Di dalam sebuah abstract class setidaknya memiliki satu atau lebih method abstrak. Fungsi dari class abstract ini adalah untuk mempertahankan hirarky dari parent class ke kelas turunan dari induknya. Abstract class digunakan di dalam inheritance (pewarisan class) untuk 'memaksakan' implementasi method yang sama bagi seluruh class yang diturunkan dari abstract class. Hal tersebut memungkinkan untuk menyediakan struktur dasar dan peraturan yang harus dipatuhi oleh subclass-subclass yang akan di buat.

Syntax penulisan :



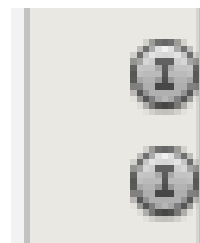
```
public abstract class BangunDatar {
```

# Definisi Abstract Method



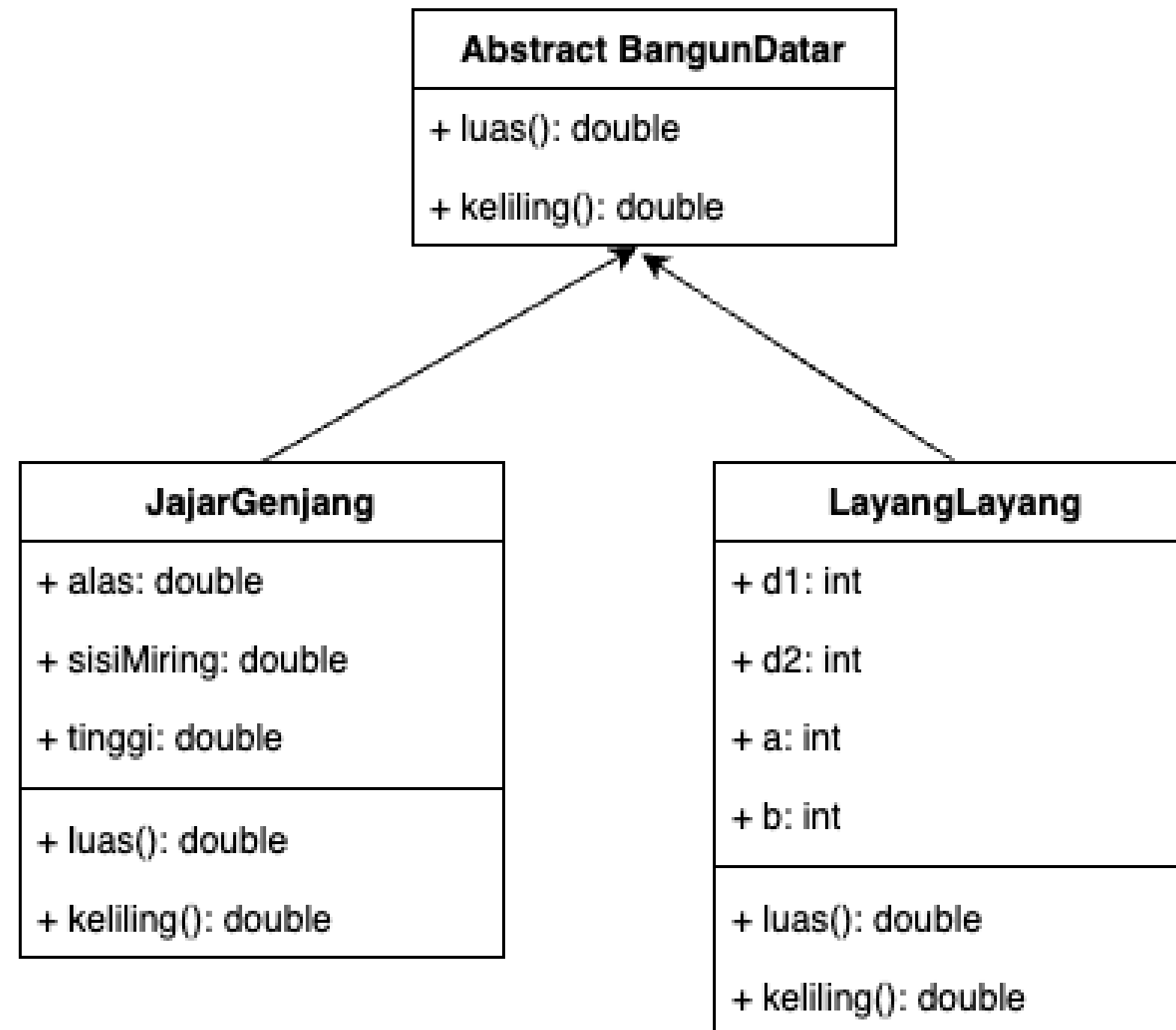
Abstract Method adalah sebuah 'method dasar' yang harus direpresentasikan ulang di dalam class anak (child class). Abstract method ditulis tanpa isi dari method, melainkan hanya 'signature'-nya saja (ciri khas). Signature dari sebuah method adalah bagian method yang terdiri dari nama method dan parameternya (jika ada).

Syntax penulisan :



```
public abstract double luas();  
public abstract double keliling();
```

# Class Diagram



# Implementasi Abstract



```
public abstract class BangunDatar {  
    abstract double luas();  
    abstract double keliling();  
}
```

---

# Implementasi Abstract



```
11 public class JajajarGenjang extends BangunDatar{
12     double a,t,b;
13     public JajajarGenjang(){
14         this.a = 8;
15         this.t = 3;
16         this.b = 9;
17     }
18     @Override
19     public double luas() {
20         return (a*t);
21     }
22
23     @Override
24     public double keliling() {
25         return (2*(a+b));
26     }
27 }
```

# Implementasi Abstract



```
11 public class LayangLayang extends BangunDatar {
12
13     int d1, d2, a, b;
14
15     public LayangLayang() {
16         this.d1 = 5;
17         this.d2 = 6;
18         this.a = 12;
19         this.b = 35;
20     }
21
22     @Override
23     public double luas() {
24         return (0.5 * d1 * d2);
25     }
26
27     @Override
28     public double keliling() {
29         return (2 * (a + b));
30     }
31 }
```

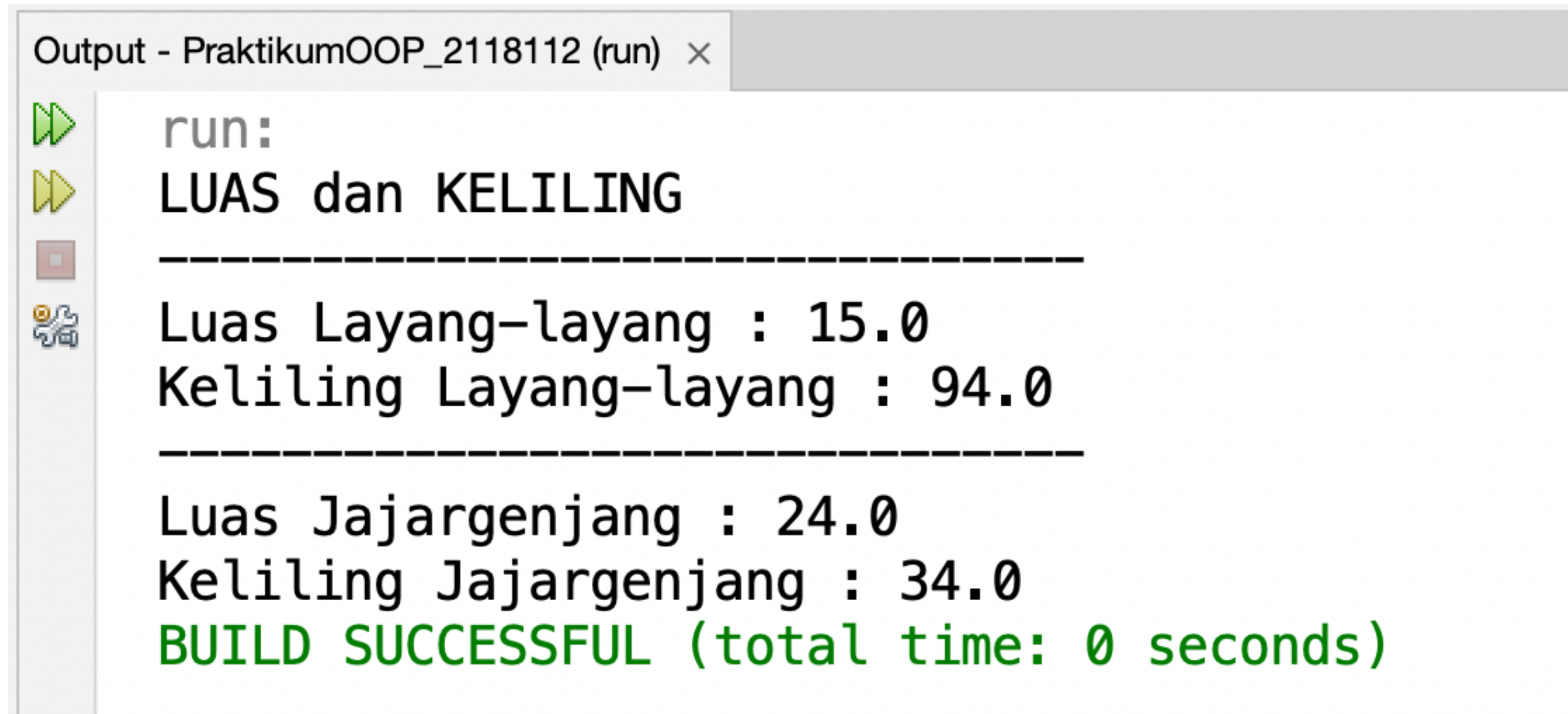
# Implementasi Abstract



```
11 public class Main {  
12  
13     public static void main(String[] args) {  
14         System.out.println( x: "LUAS dan KELILING");  
15         LayangLayang x = new LayangLayang();  
16         System.out.println( x: "-----");  
17         System.out.println("Luas Layang-layang : " + x.luas());  
18         System.out.println("Keliling Layang-layang : " + x.keliling());  
19         System.out.println( x: "-----");  
20         JajajarGenjang y = new JajajarGenjang();  
21         System.out.println("Luas Jajargenjang : " + y.luas());  
22         System.out.println("Keliling Jajargenjang : " + y.keliling());  
23     }  
24 }
```



# Hasil Running



The screenshot shows an IDE output window titled "Output - PraktikumOOP\_2118112 (run)". On the left side of the window, there is a vertical toolbar with four icons: a green double arrow (run), a yellow double arrow (debug), a red square (stop), and a gear icon (settings). The output text is as follows:

```
run:
LUAS dan KELILING
-----
Luas Layang-layang : 15.0
Keliling Layang-layang : 94.0
-----
Luas Jajargenjang : 24.0
Keliling Jajargenjang : 34.0
BUILD SUCCESSFUL (total time: 0 seconds)
```

# challenge

1. Buatlah konsep sederhana dari Abstract dengan menggunakan objek nyata "KENDARAAN"
  - a. Implementasikan hal tersebut menjadi class abstract
  - b. buatlah pula Method Abstract dari class tersebut
  - c. Kemudian buatlah implementasi konsep abstract tersebut pada subclass yang anda buat.



***"Open NetBeans, and let's write code Java"***

