

LAPORAN PROJECT
PEMROGRAMAN BERBASIS OBJEK

Penerapan Inheritance, Abstract Class, dan Interface
dalam OOP



disusun oleh:

Kelompok 11 : 17. Nauval Hilmi Hakim (2401082011)
23. Dimas Fachri Husaini (2401083004)

Kelas : Teknik Komputer 1 B

Dosen : Indri Rahmayuni, S.T., M.T

JURUSAN TEKNOLOGI INFORMASI
PROGRAM STUDI D3 TEKNIK KOMPUTER
POLITEKNIK NEGERI PADANG

2025

DAFTAR ISI

Daftar Isi	ii
Tugas 1: Program Dinamis Menghitung Luas & Keliling	1
A. Deskripsi Tugas	1
B. Diagram Class	1
C. Penjelasan Cara Kerja	2
1. Tampilan Menu,	2
2. Pengambilan Input,	3
3. Pembuatan Objek Dan Proses,	4
4. Pengulangan,	5
D. Kelebihannya	5
Tugas 2: Analisis Error Dan Implementasi Abstract Class	6
A. Deskripsi Tugas	6
B. Identifikasi Dan Penjelasan Error	6
C. Perbaikan	7
D. Hasil Output Perbaikan	9
Tugas 3: Penerapan Interface Pada Bangun Datar	10
A. Deskripsi Tugas	10
B. Diagram Class	10
C. Penjelasan Cara Kerja Program	11
1. Implementasi Interface	11
2. Penggunaan @Override	11
3. Cara Kerja Main Class	12
D. Perbandingan Abstract Class Dan Interface	13
Pembagian Tugas	15

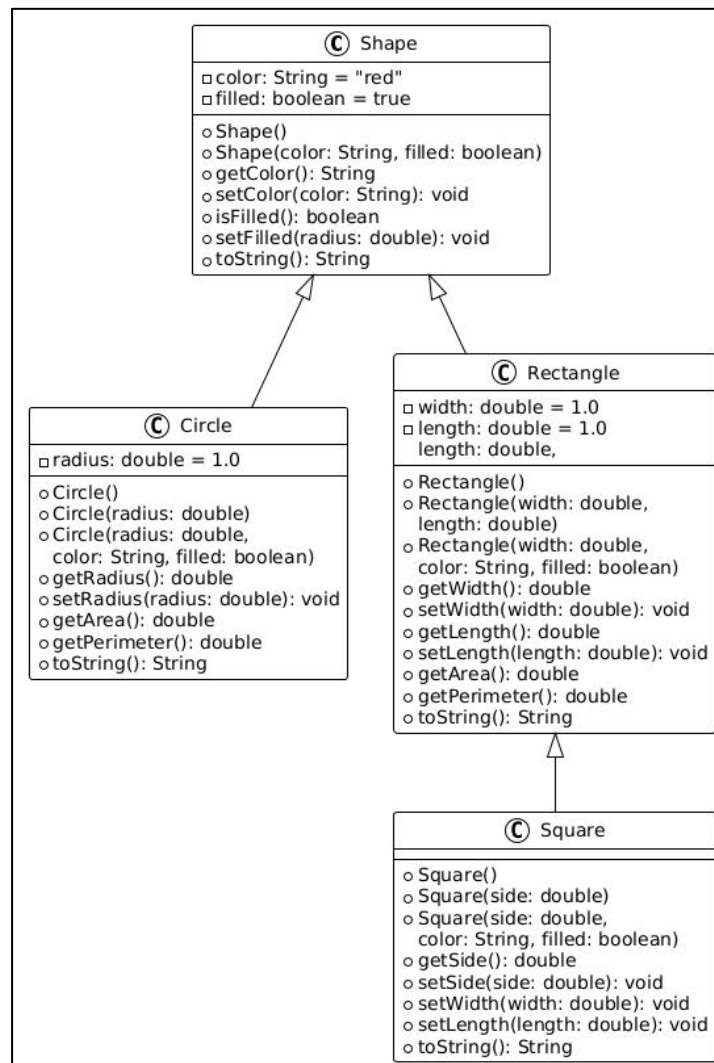
TUGAS 1

Program Dinamis Menghitung Luas & Keliling

A. DESKRIPSI TUGAS

Membuat program Java dengan **main class** dinamis yang mampu menerima input dari pengguna, menghitung luas dan keliling berbagai bangun datar (lingkaran, persegi, dan persegi panjang), serta menampilkan hasil perhitungan dengan output rapi.

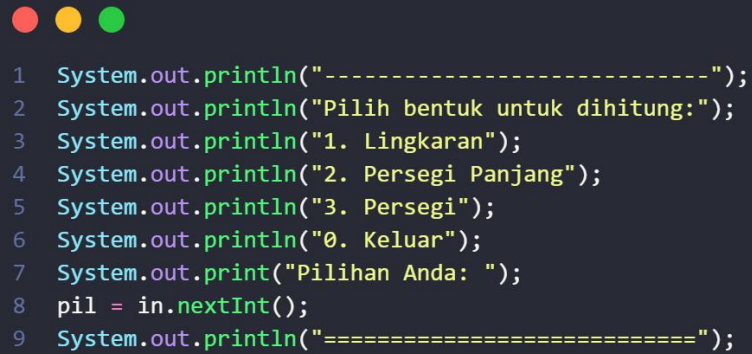
B. DIAGRAM CLASS



C. PENJELASAN CARA KERJA

1. Tampilan menu,

- Program meminta memilih Lingkaran, Persegi Panjang atau Persegi.
- User memilih bentuk (1–3) atau keluar (0).



```
1 System.out.println("-----");
2 System.out.println("Pilih bentuk untuk dihitung:");
3 System.out.println("1. Lingkaran");
4 System.out.println("2. Persegi Panjang");
5 System.out.println("3. Persegi");
6 System.out.println("0. Keluar");
7 System.out.print("Pilihan Anda: ");
8 pil = in.nextInt();
9 System.out.println("=====");
```

2. Pengambilan input,

- Setelah user memilih bentuk:
 - Untuk Lingkaran , user diminta nilai jari-jari.
 - Untuk Persegi Panjang, user diminta panjang dan lebar.
 - Untuk Persegi, user diminta panjang sisi.
- Dan untuk semua bentuk ditanyakan warna dan terisi atau tidaknya.

```
1  case 1: // Lingkaran
2      System.out.print("Masukkan jari-jari: ");
3      double radius = in.nextDouble();
4      // [Input warna dan terisi atau tidak]
5      // [Pembuatan objek dan proses]
6      break;
7
8  case 2: // Persegi Panjang
9      System.out.print("Masukkan panjang: ");
10     double length = in.nextDouble();
11     System.out.print("Masukkan lebar: ");
12     double width = in.nextDouble();
13     // [Input warna dan terisi atau tidak]
14     // [Pembuatan objek dan proses]
15     break;
16
17 case 3: // Persegi
18     System.out.print("Masukkan panjang sisi: ");
19     double side = in.nextDouble();
20     // [Input warna dan terisi atau tidak]
21     // [Pembuatan objek dan proses]
22     break;
```

3. Pembuatan Objek dan Proses,

- Program membuat objek dari kelas sesuai bentuk yang dipilih.
- Objek menghitung luas dan keliling melalui method yang diwariskan.
- Hasil ditampung dan ditampilkan oleh method toString di tiap - tiap kelas.

```
1  switch (pil) {
2      case 0: // Keluar
3          System.out.println("Thanks! Program ended");
4          break;
5
6      case 1: // Lingkaran
7          // [Pengambilan input]
8          Circle circle = new Circle(radius);
9          System.out.println(circle);
10         System.out.println("Area\t : " + circle.getArea());
11         System.out.println("Perimeter: " + circle.getPerimeter());
12         break;
13
14     case 2: // Persegi Panjang
15         // [Pengambilan input]
16         Rectangle rectangle = new Rectangle(length, width);
17         System.out.println(rectangle);
18         System.out.println("Area\t : " + rectangle.getArea());
19         System.out.println("Perimeter: " + rectangle.getPerimeter());
20         break;
21
22     case 3: // Persegi
23         // [Pengambilan input]
24         Square square = new Square(side);
25         System.out.println(square);
26         System.out.println("Area\t : " + square.getArea());
27         System.out.println("Perimeter: " + square.getPerimeter());
28         break;
29
30     default:
31         System.out.println("Invalid, please choose a valid option");
32         break;
```

4. Pengulangan,

- Setelah hasil ditampilkan, akan muncul menu untuk user mengulang atau tidak (opsi 0).

```
1  while (ulang) {
2
3      // [Inti main]
4
5      if (ulang && pil != 0) {
6          System.out.println("-----");
7          System.out.println("Do you want to try another shape?");
8          String jawab = in.next().toLowerCase();
9          ulang = jawab.equals("yes") || jawab.equals("y");
10         System.out.println();
11     }
12 }
```

D. KELEBIHANNYA

1. Dinamis & User-Friendly

- Pemilihan bentuk dan input nilai disesuaikan dari input langsung user, sehingga tidak statis dan terasa lebih nyata dan fleksibel.

2. Menggunakan Konsep Inheritance

- Kelas-kelas bentuk kemungkinan diturunkan dari kelas dasar (misalnya BangunDatar) sehingga menunjukkan pewarisan (inheritance) yang baik dan mempercepat pengembangan.

3. Terstruktur & Modular

- Masing-masing bentuk geometri dibuat dalam kelasnya sendiri.
- Tugas perhitungan diserahkan pada kelas masing-masing, menjadikan kode lebih bersih dan mudah dikelola.

TUGAS 2

Analisis Error dan Implementasi Abstract Class

A. DESKRIPSI TUGAS

Di tugas ini, kita melanjutkan dari tugas sebelumnya tapi dengan tambahan konsep abstract class. Kelas `Shape` diubah jadi abstract, lalu ditambah method abstract `getArea()` dan `getPerimeter()` yang harus di-*override* di kelas turunannya.

Kita juga diminta menganalisis kode main yang sudah disiapkan: mencari error-nya, menjelaskan kenapa bisa error, dan memperbaikinya. Setelah itu, kita buat versi main class yang sudah diperbaiki agar bisa jalan lancar dan hasil output-nya benar.

B. IDENTIFIKASI DAN PENJELASAN ERROR

1. Bagian Error: Penjelasan = Kompiler hanya melihat tipe referensi (`Shape`) dan tidak mengetahui bahwa objek di dalamnya adalah `Circle` saat kompilasi.

```
17 Shape s1 = new Circle(5.5, "RED", false); // Upcast Circle to Shape
18 System.out.println(s1); // which version?
19 System.out.println(s1.getArea()); // which version?
20 System.out.println(s1.getPerimeter()); // which version?
21 System.out.println(s1.getColor());
22 System.out.println(s1.isFilled());
23 System.out.println(s1.getRadius());
```

2. Bagian Error: Penjelasan = Kelas `Shape` merupakan abstract class kita tidak dapat membuat instance atau objek langsung dari kelas abstrak.

```
32 Shape s2 = new Shape();
34
```

3. Bagian Error: Penjelasan = Sama seperti Error 1. Tipe referensi `s3` adalah `Shape`, sedangkan `Shape` tidak memiliki method `getLength()`.

```
35 Shape s3 = new Rectangle(1.0, 2.0, "RED", false); // Upcast
36 System.out.println(s3);
37 System.out.println(s3.getArea());
38 System.out.println(s3.getPerimeter());
39 System.out.println(s3.getColor());
40 System.out.println(s3.getLength());
```


4. Bagian Error: Penjelasan = Apabila konstruktor Square adalah new Square(double side, String color, boolean filled) maka baris ini akan error karena kurang 2 argumen. Dan tipe referensi s4 adalah Shape, dan Shape tidak memiliki metode getSide().

```
Shape s4 = new Square(6.6); // Upcast
49 System.out.println(s4);
50 System.out.println(s4.getArea());
51 System.out.println(s4.getColor());
System.out.println(s4.getSide());
```

5. Bagian Error: Penjelasan = Kelas Rectangle tidak memiliki metode getSide().

```
54 // Take note that we downcast Shape s4 to Rectangle,
55 // which is a superclass of Square, instead of Square
56 Rectangle r2 = (Rectangle)s4;
57 System.out.println(r2);
58 System.out.println(r2.getArea());
59 System.out.println(r2.getColor());
60 System.out.println(r2.getLength());
System.out.println(r2.getSide());
```

C. PERBAIKAN

1. Perbaikan: Untuk mengakses getRadius(), kita perlu melakukan downcasting s1 ke Circle terlebih dahulu, seperti yang dilakukan pada baris berikutnya Circle c1 = (Circle)s1;

```
18 Shape s1 = new Circle(5.5, "RED", false);
19 System.out.println(s1);
20 System.out.println(s1.getArea());
21 System.out.println(s1.getPerimeter());
22 System.out.println(s1.getColor());
23 System.out.println(s1.isFilled());
24
25 Circle c1 = (Circle)s1;
```

2. Perbaikan: Anda harus membuat instance dari salah satu subclass konkret dari Shape, seperti Circle, Rectangle, atau Square.

```
33 Shape s2 = new Circle(1.0, "GREEN", true);
34 System.out.println(s2);
35 System.out.println(s2.getArea());
36 System.out.println(s2.getColor());
```

3. Perbaikan: kita perlu melakukan downcasting s3 ke Rectangle untuk dapat mengakses method getLength().

```
38 Shape s3 = new Rectangle(1.0, 2.0, "RED", false);
39 System.out.println(s3);
40 System.out.println(s3.getArea());
41 System.out.println(s3.getPerimeter());
42 System.out.println(s3.getColor());
43 System.out.println(s3.isFilled());
44
45 Rectangle r1 = (Rectangle)s3;
```

4. Perbaikan: Maka seharusnya perbaikannya seperti:

```
Shape s4 = new Square(6.6, "BLUE", true);
```

Dan melakukan Downcasting s4 ke Square (atau Rectangle terlebih dahulu, lalu Square) untuk mengakses getSide().

```
53 Shape s4 = new Square(6.6, "BLUE", true);
54 System.out.println(s4);
55 System.out.println(s4.getArea());
56 System.out.println(s4.getPerimeter());
57 System.out.println(s4.getColor());
58 System.out.println(s4.isFilled());
59
60 Rectangle r2 = (Rectangle)s4;
```

5. Perbaikan: melakukan downcasting r2 ke Square untuk mengakses getSide().

```
60 Rectangle r2 = (Rectangle)s4;
61 System.out.println(r2);
62 System.out.println(r2.getArea());
63 System.out.println(r2.getPerimeter());
64 System.out.println(r2.getColor());
65 System.out.println(r2.isFilled());
66 System.out.println(r2.getLength());
67
68 Square sq1 = (Square)r2;
```

D. HASIL OUTPUT PERBAIKAN

```
Output - NetBeans (run) X
run:
Circle[Shape[color=RED, filled=false], radius=5.5]
95.03317777109123
34.55751918948772
RED
false
Circle[Shape[color=RED, filled=false], radius=5.5]
95.03317777109123
34.55751918948772
RED
false
5.5
Circle[Shape[color=GREEN, filled=true], radius=1.0]
3.141592653589793
GREEN
Rectangle[Shape[color=RED, filled=false], width=1.0, length=2.0]
2.0
6.0
RED
false
Rectangle[Shape[color=RED, filled=false], width=1.0, length=2.0]
2.0
6.0
RED
false
2.0
Square[Rectangle[Shape[color=BLUE, filled=true], width=6.6, length=6.6]]
43.559999999999995
26.4
BLUE
true
Square[Rectangle[Shape[color=BLUE, filled=true], width=6.6, length=6.6]]
43.559999999999995
26.4
BLUE
true
6.6
Square[Rectangle[Shape[color=BLUE, filled=true], width=6.6, length=6.6]]
43.559999999999995
```

TUGAS 3

Penerapan Interface pada Bangun Datar

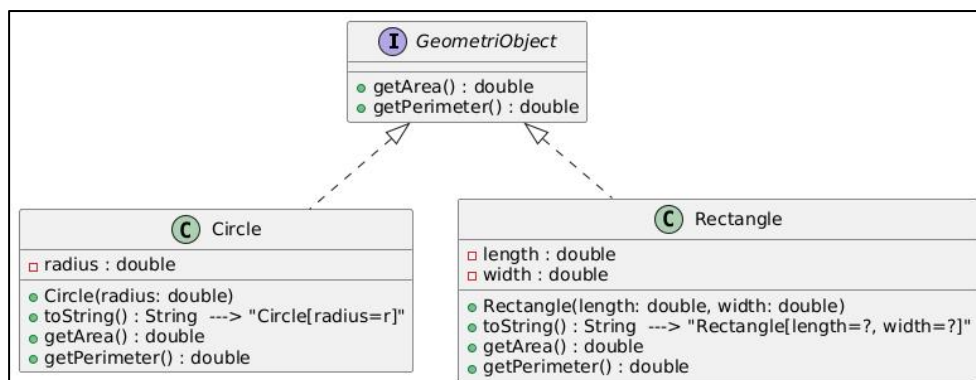
A. DESKRIPSI TUGAS

Di tugas ini, kita diminta buat sesuatu yang mirip kayak tugas sebelumnya, tapi kali ini pakai interface, bukan abstract class. Kita harus bikin interface bernama `GeometriObject` yang isinya dua method: `getArea()` dan `getPerimeter()`.

Nah, interface itu nanti di-implement sama dua kelas: `Circle` dan `Rectangle`. Jadi dua kelas itu wajib punya versi mereka sendiri dari `getArea()` dan `getPerimeter()`, dan semuanya harus dikasih anotasi `@Override` biar jelas.

Terakhir, kita juga harus bikin main class buat nguji apakah semuanya jalan atau enggak. Intinya, kita bikin dan ngetes lagi kayak sebelumnya, tapi pakai pendekatan interface. Di akhir tugas juga disuruh ngebandingin, sih — gimana bedanya waktu kita pakai abstract class di tugas 2, dan interface di tugas 3 ini.

B. DIAGRAM CLASS



C. PENJELASAN CARA KERJA PROGRAM

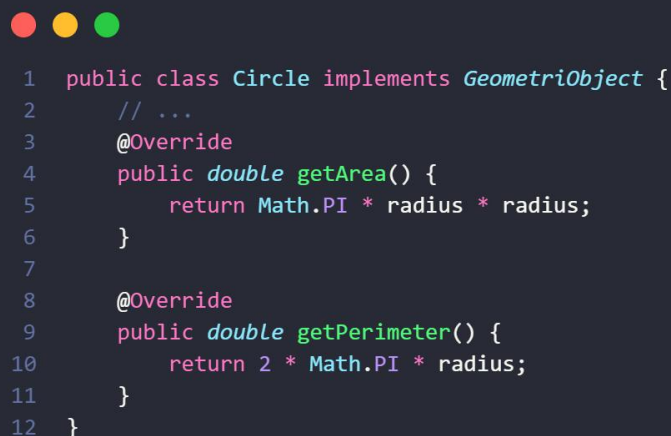
1. IMPLEMENTASI INTERFACE

Interface `GeometriObject` berfungsi sebagai aturan dasar yang harus diikuti oleh setiap kelas yang menggunakannya. Dalam hal ini, interface tersebut memiliki dua metode:

- `getArea()` : digunakan untuk menghitung luas suatu objek geometri.
- `getPerimeter()` : digunakan untuk menghitung keliling suatu objek geometri.

Kelas `Circle` dan `Rectangle` mengimplementasikan interface ini. Artinya, mereka harus menyediakan sendiri cara kerja dari kedua metode tersebut sesuai bentuk masing - masing.

Contoh nya seperti ini:



```
1 public class Circle implements GeometriObject {
2     // ...
3     @Override
4     public double getArea() {
5         return Math.PI * radius * radius;
6     }
7
8     @Override
9     public double getPerimeter() {
10        return 2 * Math.PI * radius;
11    }
12 }
```

2. PENGGUNAAN @OVERRIDE

Anotasi `@Override` digunakan untuk menandai bahwa sebuah metode adalah versi yang dimodifikasi (override) dari metode yang sudah ada di interface atau kelas induk.

Manfaat dari penggunaan anotasi ini antara lain:

- Membantu pengecekan: Jika ternyata metode yang ditulis tidak cocok atau salah eja, program akan menampilkan kesalahan saat dikompilasi.
- Meningkatkan keterbacaan kode: Dengan anotasi ini, pembaca kode akan lebih mudah memahami bahwa metode tersebut merupakan turunan atau pengganti dari metode yang sudah ada sebelumnya.

Contoh penggunaan `@Override` dalam kelas `Circle` dan `Rectangle` adalah saat mengimplementasikan `getArea()`, `getPerimeter()`, dan `toString()`.

```
1 @Override
2 public double getArea() {
3     // Implementasi untuk menghitung luas
4 }
```

3. CARA KERJA MAIN CLASS

Kelas main berfungsi sebagai titik awal program dijalankan. Di dalam metode `main`, ada beberapa langkah utama yang dilakukan:

a. *Membuat Objek*: Objek dari kelas `Circle` dan `Rectangle` dibuat dengan memberikan nilai-nilai yang dibutuhkan, seperti jari-jari untuk lingkaran atau panjang dan lebar untuk persegi panjang.

```
1 GeometriObject circle = new Circle(5);
2 GeometriObject rectangle = new Rectangle(4, 6);
```

b. *Menampilkan Informasi Objek*: Metode `toString()` yang sudah dioverride digunakan untuk menampilkan informasi tentang objek tersebut dalam bentuk teks.

```
1 System.out.println(circle); // Output: Circle[radius=5.0]
2 System.out.println(rectangle); // Output: Rectangle[length=4.0, width=6.0]
```

c. *Menghitung luas dan keliling*: Metode `getArea()` dan `getPerimeter()` dipanggil untuk menghitung nilai luas dan keliling, kemudian hasilnya ditampilkan.

```
1 System.out.println("Area: " + circle.getArea());
2 System.out.println("Perimeter: " + circle.getPerimeter());
3 System.out.println("Area: " + rectangle.getArea());
4 System.out.println("Perimeter: " + rectangle.getPerimeter());
```

D. PERBANDINGAN ABSTRACT CLASS DAN INTERFACE

Apa Itu Abstract Class dan Interface?

- Abstract class itu kayak cetakan dasar sebuah kelas yang nggak bisa langsung dipakai, tapi bisa punya isi (method) dan juga tempat kosong (abstract method) yang harus diisi oleh anak-anaknya (kelas turunannya).
- Interface itu lebih kayak daftar aturan atau kontrak: "kalau kamu mau implement saya, kamu wajib punya method ini dan itu". Isinya cuma method kosong, yang baru nanti diisi di kelas yang pakai.

Soal Isi Method

- Di abstract class, kita boleh punya campuran: ada method yang udah jadi, dan ada juga yang masih abstract (belum diisi).
- Di interface, dulu (sebelum Java 8) semua method harus kosong. Tapi sejak Java 8, kita boleh punya method default juga, meskipun itu nggak umum banget.

Bisa Turun dari Berapa?

- Abstract class cuma bisa diturunin satu. Jadi satu kelas cuma bisa extend satu abstract class.
- Tapi kalau interface, bebas! Satu kelas bisa implement banyak interface sekaligus.

Kapan Dipakai?

- Pakai abstract class kalau kamu mau bikin struktur pewarisan yang kuat — misalnya semua bentuk geometri itu “adalah” Shape.
- Pakai interface kalau kamu cuma mau bilang: “kelas ini harus bisa melakukan hal tertentu”, tanpa peduli dia anak dari siapa.

Akses dan Aturan

- Di abstract class, kamu bisa atur akses method dan atributnya pakai public, protected, atau private.
- Di interface, semua method itu otomatis public dan abstract. Dan kalau kamu bikin variabel, dia otomatis public static final alias konstanta.

PEMBAGIAN TUGAS

Dalam pengerjaan proyek ini, kami membagi tugas sebagai berikut:

- Nauval Hilmi Hakim bertanggung jawab mengerjakan Tugas 1 dan Tugas 3, termasuk penulisan laporannya.
- Dimas Fachri Husaini mengerjakan Tugas 2 beserta laporannya.

Setelah semua bagian selesai, penyusunan laporan akhir dilakukan bersama-sama oleh kami berdua, agar hasilnya utuh dan rapi sebagai satu kesatuan.