LAPORAN PRAKTIKUM
PEMROGRAMAN II
MODUL 5

POLIMORFISME

Oleh:

Afrian Pradipta Rizky NIM. 2410817210028

PROGRAM STUDI TEKNOLOGI INFORMASI
FAKULTAS TEKNIK
UNIVERSITAS LAMBUNG MANGKURAT
OKTOBER 2025

LEMBAR PENGESAHAN
LAPORAN PRAKTIKUM PEMROGRAMAN II
MODUL 5

Laporan Praktikum Pemrograman II Modul 5: Polimorfisme ini disusun sebagai syarat lulus mata kuliah Praktikum Pemrograman II. Laporan Praktikum ini dikerjakan oleh:

Nama Praktikan : Afrian Pradipta Rizky
NIM : 2410817210028

Menyetujui, Asisten Praktikum	Mengetahui, Dosen Penanggung Jawab Praktikum
Jovan Gilbert Natamasindah NIM. 2310817310002	Irham Maulani Abdul Gani, S.Kom., M.Kom. NIP. 199710312025061009

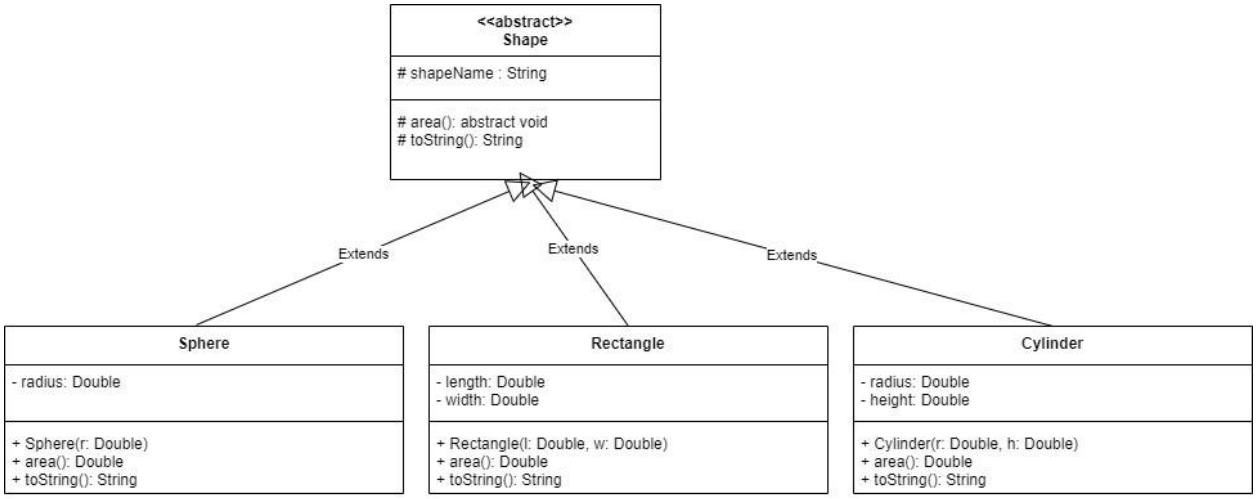
DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

SOAL 1

Pada praktikum kali ini anda akan diminta untuk membuat sebuah program yang dapat menghitung banyaknya liter cat yang digunakan untuk mewarnai bentuk ruang yang beragam. Buatlah sebuah hierarki kelas abstrak Shape dimana memiliki 3 kelas anak yaitu Sphere, Rectangle, dan Cylinder seperti ditunjukkan oleh diagram kelas berikut.



Gambar 1. Diagram Soal

Method area() digunakan untuk menghitung luas masing-masing objek. Berikut adalah formula yang digunakan untuk menghitung luas masing-masing bangun yang harus diimplementasikan.

Sphere: $4 \times \pi \times radius2$

Rectangle: $length \times width$

Cylinder: $\pi \times radius2 \times height$

Method toString() digunakan untuk mengembalikan nilai String dari nama bangun

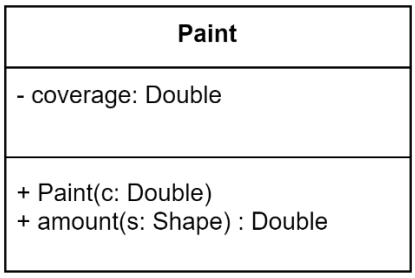
Berikut adalah ilustrasi dari kelas Sphere.java. Implementasikan kelas lainnya untuk Shape, Rectangle dan Cyclinder

Tabel 1. Contoh Ilustrasi Sphere

Contoh Ilustrasi Sphere.java

```
public class Sphere extends Shape
{
private double radius; //radius in feet
//---------------------------------- //
Constructor: Sets up the sphere. //----
-----------------------------   public
Sphere(double r)
{
super("Sphere"); radius
= r;
}
//---------------------------------------- //
Returns the surface area of the sphere. //----
-----------------------------------   public
double area()
{
return 4*Math.PI*(radius*radius);
}
//---------------------------------- //
Returns the sphere as a String. //------
----------------------------     public
String toString()
{
return super.toString() + " of radius " + radius;
}
}
```

Selanjutnya, Buatlah kelas Paint.java seperti ditunjukkan diagram kelas berikut.



Gambar 2. Diagram Soal Paint

Method amount digunakan untuk menghitung banyaknya liter cat yang digunakan dengan persamaan berikut:

$$amount\ of\ paint = \frac{area\ of\ shape}{coverage}$$

Lengkapi kode dibawah supaya menghasilkan keluaran yang diinginkan

Tabel 2. Contoh Ilustrasi Paint

Paint.java

```
public class Paint
{
private double coverage; //number of square feet per gallon
//----------------------------------------- //
Constructor: Sets up the paint object. //-----
-----------------------------------    public
Paint(double c)
{
coverage = c;
}
//-------------------------------------------------
// Returns the amount of paint (number of gallons)
// needed to paint the shape given as the parameter. //-----
---------------------------------------------- public double
amount(Shape s)
{
System.out.println ("Computing amount for " + s); return
0;
}
}
```

Terakhir, Buatlah kelas main bernama PaintThings.java. Tambahkan beberapa hal berikut agar program berjalan sesuai yang diinginkan.

1. Instansiasi 3 bentuk objek:
    a. objek bernama deck berbentuk persegi panjang dengan ukuran Panjang 20cm dan lebar 30cm.
    b. objek bernama bigBall berbentuk bola dengan ukuran radius 15cm.
    c. objek bernama tank berbentuk silinder dengan ukuran radius 10cm dan tinggi 30cm.
2. Panggil fungsi yang tepat agar dapat menghitung jumlah cat yang diperlukan.

Petunjuk untuk kelas main PaintThings.java

Tabel 3. Contoh Ilustrasi PaintThings

```
import java.text.DecimalFormat; public class PaintThings
{
//----------------------------------------
// Creates some shapes and a Paint object // and prints the amount of paint
needed // to paint each shape.
//---------------------------------------- public static void main (String[]
args)
{
final double COVERAGE = 350;
Paint paint = new Paint(COVERAGE);
Rectangle deck;
Sphere bigBall; Cylinder tank;
double deckAmt, ballAmt, tankAmt;


// Instantiate the three shapes to paint


// Compute the amount of paint needed for each shape


// Print the amount of paint for each.
DecimalFormat fmt = new DecimalFormat("0.#");
System.out.println ("\nNumber of gallons of paint needed...");
System.out.println ("Deck " + fmt.format(deckAmt));
System.out.println ("Big Ball " + fmt.format(ballAmt));
System.out.println ("Tank " + fmt.format(tankAmt));
}
}
```

1. Jalankan program dan perhatikan hasil untuk ketiga bentuk yang berbeda, screenshot hasil yang didapatkan dan lampirkan di dalam source code.

2. Simpan coding anda dengan nama package: **soal1**
3. Pastikan terdapat screenshoot pada repositori github

A. Source Code

Tabel 4. Source Code Abstract Shape

```
1  package Soal1;
2
3  abstract public class Shape {
4      protected String shapename;
5
6      public Shape(){
7          this.shapename = shapename;
8      }
9
10     abstract protected double area();
11
12     protected String toString(String shapename){
13         return shapename;
14     }
15 }
```

Tabel 5. Source Code Sphere

```
1  package Soal1;
2
```

```
3  public class Sphere extends Shape {
4      private double radius;
5
6      public Sphere(double radius){
7          this.radius = radius;
8      }
9      public double area(){
10         return 4 * Math.PI * (radius*radius);
11     }
12
13     @Override
14     public String toString() {
15         return super.toString("Sphere") + " of radius " + radius;
16     }
17 }
```

Tabel 6. Source Code Rectangle

```
1  package Soal1;
2
3  public class Rectangle extends Shape {
4      private double length;
5      private double width;
6
7      public Rectangle(double length, double widht){
```

```
8           this.length = length;
9           this.width = widht;
10      }
11
12      public double area(){
13          return length * width;
14      }
15
16      @Override
17      public String toString() {
18          return super.toString("Rectangle") + " of length " + length
    + " and width " + width;
19      }
20  }
```

Tabel 7. Source Code Cylinder

```
1   package Soal1;
2
3   public class Cylinder extends Shape{
4       private double radius;
5       private double height;
6
7       public Cylinder(double radius, double height){
8           this.height = height;
```

```
9           this.radius = radius;
10      }
11
12      public double area(){
13          return 2 * Math.PI * (radius*radius) + (2 * Math.PI * radius
    * height);
14      }
15
16      @Override
17      public String toString() {
18          return super.toString("Cylinder") + " of radius " + radius
    + " and height " + height;
19      }
20  }
```

Tabel 8. Source Code paint

```
1   package Soal1;
2
3   public class Paint {
4       private double coverage;
5
6       public Paint(double coverage){
7           this.coverage = coverage;
8       }
```

```
9
10      public double amount(Shape s){
11          System.out.println("Computing amount of: "+ s);
12          return s.area() / coverage;
13      }
14  }
```

Tabel 9. Source Code PaintThings

```
1   package Soal1;
2   import java.text.DecimalFormat;
3
4   public class PaintThings {
5       public static void main (String [] args){
6           final double COVERAGE = 350;
7           Paint paint = new Paint(COVERAGE);
8           Rectangle deck;
9           Sphere bigBall;
10          Cylinder tank;
11          double deckAmt, ballAmt, tankAmt;
12
13          deck = new Rectangle(20,30);
14          bigBall = new Sphere(15);
15          tank = new Cylinder(10, 30);
16
```

```
17          deckAmt = paint.amount(deck);
18          ballAmt = paint.amount(bigBall);
19          tankAmt = paint.amount(tank);
20
21          DecimalFormat fmt = new DecimalFormat("0.#");
22          System.out.println ("\nNumber of gallons of paint
    needed...");
23          System.out.println ("Deck " + fmt.format(deckAmt));
24          System.out.println ("Big Ball " + fmt.format(ballAmt));
25          System.out.println ("Tank " + fmt.format(tankAmt));
26      }
27  }
```

B. Output Program

```
"C:\Program Files\Java\jdk-25\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2025.
Computing amount of: Rectangle of length 20.0 and width 30.0
Computing amount of: Sphere of radius 15.0
Computing amount of: Cylinder of radius 10.0 and height 30.0

Number of gallons of paint needed...
Deck 1,7
Big Ball 8,1
Tank 7,2

Process finished with exit code 0
```

Gambar 3. Screenshot Hasil Jawaban Soal 1

C. Pembahasan

a. File abstract shape

Ini adalah *class* **abstrak** yang berfungsi sebagai "kontrak" atau *blueprint* dasar untuk semua bentuk (shape).

Pada baris [3], merupakan definisi `abstract public class Shape`. Kata kunci `abstract` berarti kelas ini tidak dapat dibuat objeknya secara langsung (Anda tidak bisa `new Shape()`). Ia hanya dimaksudkan untuk di-*inherit* (diwariskan) oleh kelas lain.

Pada baris [4], merupakan deklarasi *field* `shapename` dengan *modifier* `protected`, yang berarti variabel ini bisa diakses oleh kelas `Shape` itu sendiri dan semua kelas turunannya (seperti `Sphere`, `Rectangle`, dll.).

Pada baris [6] hingga [8], merupakan sebuah konstruktor. Tujuannya adalah untuk menginisialisasi *field* `shapename`. *(Catatan: Ada sedikit bug di kode ini, `shapename` di sisi kanan seharusnya menjadi parameter, namun kita jelaskan tujuannya)*.

Pada baris [10], merupakan definisi `abstract protected double area()`. Ini adalah **metode abstrak**. Ia tidak memiliki isi (tidak ada `{}`). Ini adalah sebuah "kontrak" yang *memaksa* setiap kelas turunan (seperti `Sphere` atau `Rectangle`) untuk *wajib* membuat dan mendefinisikan versi `area()` mereka sendiri.

Pada baris [12] hingga [14], merupakan sebuah *method helper* `toString`. *Method* ini akan dipanggil oleh kelas turunan untuk membantu membangun deskripsi teks dari objek tersebut.

b. File Sphere

Ini adalah *class* **konkret** yang mewarisi (extends) `Shape`.

Pada baris [3], merupakan definisi `public class Sphere extends Shape`. Ini menyatakan bahwa `Sphere` **adalah sebuah** `Shape` (`Sphere` *is a* `Shape`). `Sphere` mewarisi semua *field* dan *method* (termasuk kewajiban) dari `Shape`.

Pada baris [4], merupakan deklarasi *field* `private double radius`, yang khusus dimiliki oleh `Sphere`.

Pada baris [6] hingga [8], merupakan konstruktor `Sphere` yang menerima `radius` dan menginisialisasi *field* `radius` milik objek.

Pada baris [9] hingga [11], merupakan **implementasi** dari *method* `area()` yang diwajibkan oleh kelas `Shape`. *Method* ini menyediakan logika perhitungan luas permukaan spesifik untuk `Sphere`.

Pada baris [13] hingga [16], merupakan *method* `toString()` yang di-*override*. *Method* ini digunakan untuk memberikan representasi teks yang deskriptif dari objek `Sphere`. Ia memanggil `super.toString("Sphere")` (dari kelas `Shape`) dan menambahkannya dengan informasi radius.

c. File Rectangle

Ini adalah *class* **konkret** lain yang mewarisi (extends) `Shape`.

Pada baris [3], merupakan definisi `public class Rectangle extends Shape`. Ini menyatakan bahwa `Rectangle` **adalah sebuah** `Shape`.

Pada baris [4] hingga [5], merupakan deklarasi *field* `private length` dan `width`, yang spesifik untuk `Rectangle`.

Pada baris [7] hingga [10], merupakan konstruktor `Rectangle` yang menerima `length` dan `width` untuk menginisialisasi *field* objek.

Pada baris [12] hingga [14], merupakan **implementasi** dari *method* `area()` yang diwajibkan oleh `Shape`. *Method* ini menyediakan logika perhitungan luas spesifik untuk `Rectangle`.

Pada baris [16] hingga [19], merupakan *method* `toString()` yang di-*override* untuk memberikan deskripsi teks yang spesifik untuk `Rectangle`.

d. File Cylinder

Ini adalah *class* **konkret** ketiga yang mewarisi (extends) `Shape`.

Pada baris [3], merupakan definisi `public class Cylinder extends Shape`. Ini menyatakan bahwa `Cylinder` **adalah sebuah** `Shape`.

Pada baris [4] hingga [5], merupakan deklarasi *field* `private radius` dan `height`, yang spesifik untuk `Cylinder`.

Pada baris [7] hingga [10], merupakan konstruktor `Cylinder` yang menerima `radius` dan `height` untuk menginisialisasi *field* objek.

Pada baris [12] hingga [14], merupakan **implementasi** dari *method* `area()` yang diwajibkan oleh `Shape`. *Method* ini menyediakan logika perhitungan luas permukaan spesifik untuk `Cylinder`.

Pada baris [16] hingga [19], merupakan *method* `toString()` yang di-*override* untuk memberikan deskripsi teks yang spesifik untuk `Cylinder`.

e. File Paint

Ini adalah kelas utilitas yang *menggunakan* objek-objek `Shape`. Di sinilah **Polimorfisme** benar-benar ditunjukkan.

Pada baris [4], merupakan deklarasi *field* `private double coverage`, yang menyimpan data seberapa banyak area yang bisa dicat oleh satu unit cat.

Pada baris [6] hingga [8], merupakan konstruktor `Paint` yang menginisialisasi *field* `coverage`.

Pada baris [10] hingga [13], merupakan *method* `amount`. Ini adalah inti dari program.

Baris 10: *Method* ini menerima parameter `Shape s`. Ini sangat penting. Karena `Sphere`, `Rectangle`, dan `Cylinder` *semuanya* adalah turunan `Shape`, *method* ini dapat menerima *salah satu* dari objek tersebut.

Baris 11: Mencetak objek `s`. Secara otomatis Java akan memanggil *method* `.toString()` yang sesuai (milik `Sphere`, `Rectangle`, atau `Cylinder`).

Baris 12: Memanggil `s.area()`. Java secara "ajaib" (polimorfisme) tahu objek `s` itu sebenarnya apa. Jika `s` adalah `Sphere`, ia akan memanggil `area()` milik `Sphere`. Jika `s` adalah `Rectangle`, ia akan memanggil `area()` milik `Rectangle`. *Method* `amount` ini tidak perlu tahu detailnya, ia hanya perlu tahu bahwa `s` adalah `Shape` dan pasti memiliki *method* `area()`.

f. File PaintThings

Ini adalah *class* utama yang berisi *method* `main`, yang berfungsi sebagai titik awal eksekusi program.

Pada baris [5], merupakan *method* `main`, titik awal program.

Pada baris [6], merupakan deklarasi `final double COVERAGE`. Ini adalah konstanta untuk daya sebar cat.

Pada baris [7], merupakan pembuatan objek `Paint` dengan menggunakan konstanta `COVERAGE`.

Pada baris [8] hingga [14], merupakan deklarasi dan inisialisasi (pembuatan objek) dari tiga bentuk yang berbeda: `Rectangle`, `Sphere`, dan `Cylinder`.

Pada baris [16] hingga [18], merupakan bagian di mana **Polimorfisme** dieksekusi. *Method* `paint.amount()` yang sama dipanggil tiga kali dengan tiga tipe objek yang berbeda (`deck`, `bigBall`, `tank`). *Method* `amount` (di file `Paint.java`) dapat menangani semuanya dengan benar karena mereka semua adalah turunan dari `Shape`.

Pada baris [20], merupakan pembuatan objek `DecimalFormat` untuk merapikan hasil *output* desimal.

Pada baris [21] hingga [25], merupakan pencetakan hasil akhir yang sudah diformat, yang menunjukkan jumlah cat yang dibutuhkan untuk setiap bentuk.

D. Github

TAUTAN GIT

https://github.com/Dooughnut/Pemrograman-2.git