LAPORAN PENGERJAAN TUGAS PERTEMUAN 4

TEKNIK PEMOGRAMAN PRAKTIKUM

Disusun untuk memenuhi salah satu tugas pertemuan keempat mata kuliah

Teknik Pemograman

Oleh:

Fadilah Akbar (231524041)

Kelas :

Teknik Informatika D4 – 1B

PROGRAM STUDI SARJANA TERAPAN TEKNIK INFORMATIKA

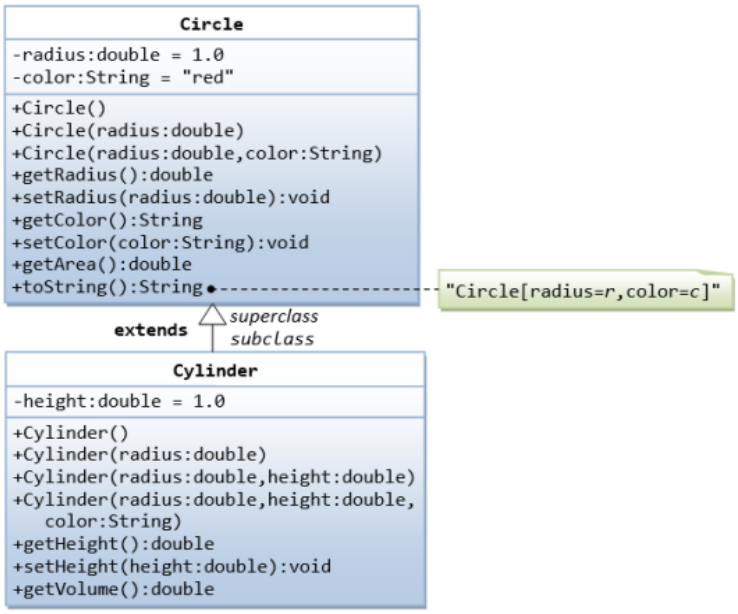
JURUSAN TEKNIK KOMPUTER DAN INFORMATIKA

POLITEKNIK NEGERI BANDUNG

2024

DAFTAR ISI

Exercise 1 : The Circle and Cylinder Classes



Task 1.1

Pada task 1.1 ini, saya diberikan perintah untuk

- Memodifikasi class Circle dengan menambahkan sebuah variable bernama color bertype string
- Membuat constructor Circle dengan arguments double radius dan String color.
- Membuat getter dan setter untuk attributes Color.

Berikut ini source code yang sudah saya modifikasi :

Menambahkan Sebuah Variable Bernama Color Bertype String

```
public class Circle { // Save as "Circle.java"
 // private instance variable, not accessible from outside this class
 protected double radius;
 private String color; //atribute baru dengan nama color bertipe string
```

Constructor Circle dengan arguments double radius dan String color.

```
 /*new contructor */
public Circle(double radius, String color){
this.radius = radius;
this.color = color;
}
```

Membuat Getter Dan Setter Untuk Attributes Color

```
// getter sertter color
    public String getColor() {
        return color;
    }

    public void setColor(String color) {
        this.color = color;
    }
```

Task 1.2

Pada task 1.2 ini diberikan perintah untuk melakukan override kepada method getArea dalam subclass Cylinder yang dimana method ini di inherit dari superclass Circle.

Jika method getArea dalam class Circle akan mengembalikan nilai dari luas lingkaran, maka pada method getArea dalam subclass Cylinder ini akan merubah behaviournya menjadi mengembalikan nilai “2*phi*radius*tinggi+2*luas-alas”.

```
@Override
public double getArea(){
 return 2 * Math.PI * getRadius() * height + 2 * super.getArea();
}       You, yesterday • first commit
```

Dari gambar di atas dalam pemanggilan ***getArea()*** yang dipanggil adalah dari class Circle sehingga menggunakan kata kunci ***“super”*** . bahwa return ini memanggil juga getter radius karena akan dipakai.

Pada proses override methode ***getArea()***, terdapat methode ***getVolume()*** tidak bekerja sama. Dikarenakan untuk mendapatkan sebuah volume tabung kita harus mengalikan luas alat berupa lingkaran dengan tinggi dari tabung tersebut.

Untuk memperbaiki methode ini karena harus mengembalikan nilai luas lingkaran yang digunakan dalam method ***getVolume(),*** harus menambahkan kata kunci ***“super”*** sebelum ***getArea()*** untuk mendapatkan nilai luas alas.

```
// A public method for computing the volume of cylinder
// use superclass method getArea() to get the base area
public double getVolume() {
return super.getArea()*height;
}       You, yesterday • first commit
```

Task 1.3

Diberikan perintah untuk melakukan override terhadap method toString yang sebelumnya ada pada superclass Circle. Berikut ini adalah source code sebelum dan sesudah dari method toString ini :

Sebelum

```
/** Return a self-descriptive string of this instance in the form of
Circle[radius=?,color=?] */
public String toString() {
return "Circle[radius=" + radius + " color=" + color + "]";
}
```

Sesudah

```
}
 return "Cylinder: subclass of " + super.toString() + " height=" + height;
public String toString() { // in Cylinder class
@Override
// Test Override a toString() method
```

Exercise 2 : Superclass Shape and its Subclasses Circle, Rectangle and Square

Task 2.1

[Task 2.1]

Write a superclass called `Shape` (as shown in the class diagram), which contains:

- Two instance variables `color` (`String`) and `filled` (`boolean`).
- Two constructors: a no-arg (no-argument) constructor that initializes the `color` to "green" and `filled` to `true`, and a constructor that initializes the `color` and `filled` to the given values.
- Getter and setter for all the instance variables. By convention, the getter for a `boolean` variable `xxx` is called `isXXX()` (instead of `getXxx()` for all the other types).
- A `toString()` method that returns "`A Shape with color of xxx and filled/Not filled`".

Write a test program to test all the methods defined in `Shape`.

Write two subclasses of `Shape` called `Circle` and `Rectangle`, as shown in the class diagram.

The `Circle` class contains:

- An instance variable `radius` (`double`).
- Three constructors as shown. The no-arg constructor initializes the radius to `1.0`.
- Getter and setter for the instance variable `radius`.
- Methods `getArea()` and `getPerimeter()`.
- Override the `toString()` method inherited, to return "`A Circle with radius=xxx, which is a subclass of yyy`", where `yyy` is the output of the `toString()` method from the superclass.

The `Rectangle` class contains:

- Two instance variables `width` (`double`) and `length` (`double`).
- Three constructors as shown. The no-arg constructor initializes the `width` and `length` to `1.0`.
- Getter and setter for all the instance variables.
- Methods `getArea()` and `getPerimeter()`.
- Override the `toString()` method inherited, to return "`A Rectangle with width=xxx and length=zzz, which is a subclass of yyy`", where `yyy` is the output of the `toString()` method from the superclass.

Write a class called `Square`, as a subclass of `Rectangle`. Convince yourself that `Square` can be modeled as a subclass of `Rectangle`. `Square` has no instance variable, but inherits the instance variables `width` and `length` from its superclass `Rectangle`.

- Provide the appropriate constructors (as shown in the class diagram). Hint:

```
public Square(double side) {
   super(side, side);  // Call superclass Rectangle(double, double)
}
```

- Override the `toString()` method to return "`A Square with side=xxx, which is a subclass of yyy`", where `yyy` is the output of the `toString()` method from the superclass.
- Do you need to override the `getArea()` and `getPerimeter()`? Try them out.
- Override the `setLength()` and `setWidth()` to change both the `width` and `length`, so as to maintain the square geometry.

pada task 2.1 ini diperintahkan membuat sebuah cource code dari class class ini dan isinya sesuai dengan spesifikasi dari setiap class seperti pada gambar di atas

Source Code Class Shape

```java
package Praktek.Exercise2;

public class Shape {
    // attribute class sahpe
    private String color = "red";
    private boolean filled = true;

    // constructor tanpa parameter
    public Shape(){
        color = "green";
        filled = true;
    }

    // constructor parameter color, filled
    public Shape(String color, boolean filled){
        this.color = color;
        this.filled = filled;
    }

    //getter setter atribute color
    public String getColor() {
        return color;
    }
    public void setColor(String color) {
        this.color = color;
    }

    //getter setter atribute filled
    public boolean isFilled(){
        return isFilled();
    }
    public void setFilled(boolean filled) {
        this.filled = filled;
    }

    //method toString
    public String toString(){
        return ("A Shape with color of" + this.color + (this.filled==true) + "and filled/Not filled");
    }
}
```

Source Code Class Circle

```java
package Praktek.Exercise2;
public class Circle extends Shape {
    //atribute circle
    private double radius;

    // constructor tanpa parameter
    public Circle(){
        super();
        radius = 1.0;
    }

    // constructor parameter rad/radius
    public Circle(double Rad){
        super();
        this.radius=Rad;
    }

    //getter setter radius
    public double getRadius() {
        return radius;
    }
    public void setRadius(double radius) {
        this.radius = radius;
    }

    // method getArea
    public double getArea(){
        return  Math.PI*getRadius()*getRadius();
    }

    // method getPerimeter
    public double getPerimeter(){
        return 2*Math.PI*getRadius();
    }

    // override methode toString
    @Override
    public String toString(){
        return ("A Circle with radius" + getRadius() + "which is a subclass " + super.toString());
    }
}
```

Source Code Class Rectangle

```java
package Praktek.Exercise2;

public class Rectangle extends Shape {
    //atribute rectangle
    private double width;
    private double length;

    //construktor tanpa parameter
    public Rectangle(){
        super();
        this.width = 1.0;
        this.length = 1.0;
    }

    // construktor parameter witdh, length
    public Rectangle(double width, double length){
        super();
        this.width = width;
        this.length = length;
    }

    // construktor parameter witdh, length, color, fill
    public Rectangle(double width, double length, String color, boolean Fill){
        super(color, Fill );
        this.width = width;
        this.length = length;
    }

    //Methode getArea
    public double getArea(){
        return getLength()*getWidth();
    }

    // Methode getParimeter
    public double getPerimeter(){
        return 2*(getLength()*getWidth());
    }

    //setter getter Atribute width & length
    public double getWidth() {
        return width;
    }
    public void setWidth(double width) {
        this.width = width;
    }

    //setter getter Atribute length
    public double getLength() {
        return length;
    }
    public void setLength(double length) {
        this.length = length;
    }

    @Override
    public String toString(){
        return ("\nA Rectangle with width " + getWidth() + " and " + length + " which is a subclass of " + super.toString() + "\n");
    }
```

Source Code Class Square

```java
package Praktek.Exercise2;

public class Square extends Rectangle{
    // Construktor tanpa parameter
    public Square(){
        super();
    }

    // Construktor parameter double side
    public Square(double side){
        super(side, side);
    }

    //Setter getter side
    public void setSide(double side){
        super.setLength(side);
        super.setWidth(side);
    }
    public double getSide(){
        return super.getLength();
    }

    //override methode getArea
    @Override
    public double getArea(){
        return getSide()*getSide();
    }

    //override methode getPerimeter
    @Override
    public double getPerimeter(){
        return getSide()*4;
    }

    //override methode toString
    @Override
    public String toString(){
        return ("A Square with side = " + getSide() +  " , which is a subclass of " + super.toString());
    }
```

Output dari setiap class

Output Shape

```
Shape[color = green, filled = true]
```

Output Circle

```
A circle with radius = 1.0, Which is a subclass of Shape[color = green, filled = true]
Luas = 3.141592653589793
Keliling = 6.283185307179586
```

Output rectangle

```
A Rectangle with width 1.0 and 1.0 which is a subclass of A Shape with color
ofgreentrueand filled/Not filled

luas = 1.0 keliling = 2.0
```

Output Square

```
A Square with side = 1.0 , which is a subclass of
Keliling = 4.0
Luas = 1.0
```

Exercise 3 : Multiple Inheritance

Task 3.1

Pada task 3.1 ini, diperintahkan untuk menyalin 4 class berupa class Employee, EmployeeTest, Manager, dan ManagerTest, selain diperintahkan juga untuk membuat sebuah class abstract bernama Sortable yang akan menjadi superclass untuk class Employee.

Untuk lebih jelasnya, berikut di bawah adalah rincian setiap classnya :

- **Class Sortable**
  Pada class sortable ini memiliki 2 buah method namun tidak memiliki atribute, method yang pertama dengan nama ***compare(Sortable b)*** yang berisikan parameter b bertipe nama class nya, method ini bersifat abstrak yang dimana dalam implementasi harus dibuat behavior nya di subclass. Method kedua dengan nama ***shell_sort(Sortable[] a)*** method ini statis menerima sebuah array dari objek ***"Sortable"*** yang akan diurutkan menggunakan algoritma Shell sort.

  Algoritma shell sort yang diimplemtasikan menggunakan sebuah gap(jarak) yang dibagi dua (dalam loop 'gap /= 2') sampai gap mencapai 0. Dalam setiap proses dilakukan pertukaran element element array yang berjarak satu sama lainya, serta dalam setiap pertukaran dilakukan perbandingan menggunakan method ***compare()*** yang di implementasikan oleh objek objek Sortable.

- **Class Employee**
  Class ini merupakan turunan dari ***Class Sortable*** yang dimana harus menerapkan metode ***compare()*** sesuai dengan kebutuhan dalam pengurutan objek Employee, class ini beberapa atribut seperti ***name*** bertipe string, ***salary*** bertipe double, ***hireday*** bertipe integer, ***hiremonth*** bertipe integer, dan ***hireyear*** bertipe integer yang merepresentasikan data seorang karyawan. Selain itu terdapat konstruktor yang digunakan untuk menginisialisasi data karyawan.

  Selain itu, terdapat beberapa method diantaranya, method ***print()*** untuk mencetak informasi data karyawan, method ***raiseSalary()*** untuk menaikan gaji karyawan berdasarkan persentase yang diberikan, selanjutnya method ***hireYear()*** untuk mendapatkan tahun perekrutan karyawan, dan method ***compare()*** yang merupakan implementasi dari class Sortable untuk membandingkan karyawan berdasarkan gaji karyawan.

```java
package Praktek.Exercise3;
abstract class Sortable{
    public abstract int compare(Sortable b);
    public static void shell_sort(Sortable[] a){
         int n = a.length;

        for (int gap = n/2; gap > 0; gap /= 2) {
            for (int i = gap; i < n; i += 1) {
                Sortable temp = a[i];

                int j;
               for (j = i; j >= gap && a[j - gap].compare(temp) < 0; j -= gap){
                    a[j] = a[j - gap];
                    a[j] = temp;
                }
            }
        }
    }
}

class Employee extends Sortable{
    //atribute
    private String name;
    private double salary;
    private int hireday;
    private int hiremonth;
    private int hireyear;

    // constructor
    public Employee(String n, double s, int day, int month, int year){
        name = n;
        salary = s;
        hireday = day;
        hiremonth = month;
        hireyear = year;
    }

    public void print(){
        System.out.println(name + " " + salary + " " + hireYear());
    }

    public void raiseSalary(double byPercent){
        salary *= 1 + byPercent / 100;
    }

    public int hireYear(){
        return hireyear;
    }

    @Override //dari class Sortable
    public int compare(Sortable b){
        Employee eb = (Employee) b;
        if (salary<eb.salary) return -1;
        if (salary>eb.salary) return +1;
        return 0;
    }
}
```

- **Class EmployeeTest**
  Pada class ini sesuai dengan namanya yaitu untuk menguji implementasi dari class Employee dan pengurutan ***shell_sort()*** yang telah didefinisikan dalam class Sortable.

  Inisialisasi terdapat tiga objek Employee dibuat dan disampai kedalam array ***'staff',*** setiap objek mewakili seorang karyawan yang berisikan data nama, gaji, dan tanggal perekrutan yang berbeda. Method ***'shell_sort()'*** dari class Sortable dipanggil untuk mengurutkan array staff yang berisikan objek objek Emlpoyee, objek oebjek ini akan diurutkan berdasarkan gaji mereka dari yang terbesar hingga yang terkecil sesuai dengan implementasi pada method ***compare()***

Setelah itu gaji karyawan dinaikan sebesar5% menggunakan method ***raiseSalary(),*** dan informasi karyawan akan dicetak menggunakan method ***print()*** yang menampilkan nama, gaji, dan tahun perekrutan karyawan setelah penaikan gaji. Berikut adalah source code dan output dari program EmployeeTest

```java
package Praktek.Exercise3;

public class EmployeeTest {
    public static void main (String[] args){
        Employee[] staff = new Employee[3];
        staff[0] = new Employee("Antonio Rossi", 2000000, 1, 10, 1989);
        staff[1] = new Employee("Maria Bianchi", 2500000, 1, 12, 1991);
        staff[2] = new Employee("Isabel Vidal", 3000000, 1, 11, 1993);
        Sortable.shell_sort(staff);
         for (int i = 0; i < 3; i++) {
            staff[i].raiseSalary(5);
            staff[i].print();
        }
    }
}
```

```
Isabel Vidal 3150000.0 1993
Maria Bianchi 2625000.0 1991
Antonio Rossi 2100000.0 1989
```

- **Class Manager**
  Pada class ini untuk merepresentasikan seorang manager dalam sebuah prusahaan, class ini merupakan turunan dari class Employee yang berari akan mewarasi sifat dari kelas tersebut, namun dapat memiliki perilahu tambahan yang khusus seperti mendapatkan bonus berdasarkan tahun layanan dan memiliki sekertaris

  Pada programnya terdapat sebuah konstraktor yang digunakan untuk membuat objek Manager dengan parameter n bertipe string, s bertipe double, atribut d, m , dan y bertipe integer. Yang diteruskan kelas Employee menggunakan kata kunci **‘super’.**

  Selanjutnya terdapat method ***raiseSalary(double byPercent)*** yang merupakan override dari class Emloyee, serta terdapat method ***getSecretaryName()*** untuk mendapatkan nama sekertaris, sekertaris ini direpsentasikan oleh atribut ‘***secretartName’.***

```java
package Praktek.Exercise3;

import java.util.Calendar;
import java.util.GregorianCalendar;

class Manager extends Employee{
    public Manager (String n, double s, int d, int m, int y){
        super(n, s, d, m, y);
        secretaryName = "";
    }

    public void raiseSalary(double byPercent){
        // add 1/2% bonus for every year of service
        GregorianCalendar todaysDate = new GregorianCalendar();
        int currentYear = todaysDate.get(Calendar.YEAR);
        double bonus = 0.5 * (currentYear - hireYear());
        super.raiseSalary(byPercent + bonus);
    }

    public String getSecretaryName(){
        return secretaryName;
    }

    private String secretaryName;
}
```

- **Class ManagerTest**
  Class ini adalah sebuah class yang digunakan untuk menguji/ implementasi hasil dari kelas Employee dan Manager seperti halnya class EmployeTest. Berikut adalah sourcecode dan hasil programnya.

```java
package Praktek.Exercise3;

public class ManagerTest {
    public static void main (String[] args){
        Employee[] staff = new Employee[3];
        staff[0] = new Employee("Antonio Rossi", 2000000, 1, 10, 1989);
        staff[1] = new Manager("Maria Bianchi", 2500000, 1, 12, 1991);
        staff[2] = new Employee("Isabel Vidal", 3000000, 1, 11, 1993);
        int i;
        Sortable.shell_sort(staff);
        for (i = 0; i < 3; i++) staff[i].raiseSalary(5);
        for (i = 0; i < 3; i++) staff[i].print();
    }
}
```

```
Isabel Vidal 3150000.0 1993
Maria Bianchi 3037500.0 1991
Antonio Rossi 2100000.0 1989
```