LAPORAN PRAKTIKUM
PERTEMUAN-6

Diajukan untuk memenuhi salat satu tugas praktikum Mata kuliah Pemrograman Berorientasi Objek

Disusun Oleh:
Daiva Raditya Pradipa (231511039)
Jurusan Teknik Komputer dan Informatika

Program Studi D-3 Teknik Informatika
Politeknik Negeri Bandung
2024

1. Task 1.1

```java
*/
public class Circle { // Save as "Circle.java"
    // private instance variable, not accessible from outside this class
    private double radius;
    private String color;

    // Constructors (overloaded)
    /** Constructs a Circle instance with default value for radius and color
    public Circle() { // 1st (default) constructor
        radius = 1.0;
        color = "red";
    }

    /** Constructs a Circle instance with the given radius and default color
    public Circle(double r) { // 2nd constructor
        radius = r;
        color = "red";
    }

    public Circle(double radius, String color) { // 2nd constructor
        this.radius = radius;
        this.color = color;
    }

    public String getColor() {
        return this.color;
    }

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

Hasil akhir program

```
src > TestCylinder.java > TestCylinder > main(String[])
1   public class TestCylinder { // save as "TestCylinder.java"
        Run | Debug
2       public static void main (String[] args) {
3
4           Circle c = new Circle(radius:2.0,color:"blue");
5           System.out.println(c.getColor());
6           c.setColor(color:"purple");
7           System.out.println(c.getColor());
8
PROBLEMS 2   DEBUG CONSOLE   PORTS   TERMINAL
PS C:\Code\java\Pertemuan6\pertemuan6>  & 'C:\Program Files\Java\jdk-2
blue
purple
PS C:\Code\java\Pertemuan6\pertemuan6>
```

Penjelasan:

Pada task 1.1, kita diminta untuk menambahkan atribut baru bernama color dengan tipe data String, yang berfungsi untuk menyimpan warna dari objek class SHape. Selain itu, kita diminta membuat constructor tambahan dengan parameter tambahan untuk color,

sehingga saat membuat objek, kita bisa langsung menentukan warnanya. Kemudian, kita harus menambahkan metode getter dan setter untuk atribut color. Getter akan digunakan untuk mengambil nilai color, sedangkan setter berfungsi untuk mengubah atau menetapkan nilai baru pada atribut color

Masalah:-

Solusi:-

Teman yang membantu: -

2. Task 1.2

```java
public class Cylinder extends Circle { // Save as "Cylinder.java"

    // Constructor with default color, but given radius, height
    public Cylinder(double radius, double height) {
        super(radius); // call superclass constructor Circle(r)
        this.height = height;
    }

    @Override
    public double getArea() {
        return 2*Math.PI*super.getRadius()*height + 2*super.getArea();
    }
```

Hasil akhir program

```
PROBLEMS 2   DEBUG CONSOLE   PORTS   TERMINAL

PS C:\Code\java\Pertemuan6\pertemuan6> ^C
PS C:\Code\java\Pertemuan6\pertemuan6>
PS C:\Code\java\Pertemuan6\pertemuan6>  c:; cd 'c:\Code\java\Pertemuan6\pertemuan6'; & 'C:\Progra
\pertemuan6\bin' 'TestCylinder'
Cylinder: radius=1.0 height=1.0 base area=12.566370614359172 volume=3.141592653589793
Cylinder: radius=1.0 height=10.0 base area=69.11503837897544 volume=31.41592653589793
Cylinder: radius=2.0 height=10.0 base area=150.79644737231007 volume=125.66370614359172
PS C:\Code\java\Pertemuan6\pertemuan6>
```

Penjelasan:
Pada task 1.2, kita diminta untuk mengubah implementasi dari method getArea yang ada pada superclass Circle di dalam subclass Cylinder. Dalam subclass Cylinder, rumus yang digunakan untuk menghitung luas permukaan (surface area) adalah Surface Area=2π×radius×height+2×base area. Untuk menerapkan perubahan ini, kita bisa menggunakan konsep method overriding, di mana kita menulis ulang method getArea di subclass Cylinder agar sesuai dengan rumus tersebut dengan terlebih dahulu menuliskan syntax @Override untuk mengindikasikan bahwa method tersebut berasal dari superclass yang akan didefinisikan ulah dan diubah implementasinya. Dengan demikian, saat objek Cylinder memanggil getArea, hasilnya akan dihitung berdasarkan rumus luas permukaan tabung, bukan luas lingkaran seperti di superclass Circle.

Masalah:-

Solusi:-

Teman yang membantu: -

3. Task 1.3

```java
public class Cylinder extends Circle { // Save as "Cylinder.java"
    }

    @Override
    public String toString() { // in Cylinder class
        return "Cylinder: subclass of " + super.toString() // use Circle's toString()
        + " height=" + height;
    }
}
```

Hasil akhir program

```
PS C:\Code\java\Pertemuan6\pertemuan6>  c:; cd 'c:\Code\java\Pertemuan6\pertemuan6'; &
\pertemuan6\bin' 'TestCylinder'
Cylinder: subclass of Circle[radius=1.0 color=red] height=1.0
PS C:\Code\java\Pertemuan6\pertemuan6>
```

Penjelasan:
Pada task 1.3, kita diminta untuk mengubah implementasi method toString yang ada pada superclass Circle di dalam subclass Cylinder. Dalam hal ini, implementasi di subclass Cylinder harus memanggil method toString dari superclass Circle, dengan hasil akhir berupa teks seperti: "Cylinder: subclass of " + // menggunakan toString() dari Circle + " height=" + height. Untuk melakukan hal ini, kita dapat menggunakan syntax super.toString() untuk memanggil method toString dari superclass. Dengan demikian, method toString di subclass Cylinder akan menggabungkan informasi dari superclass Circle dan menambahkan detail tinggi (height) yang spesifik untuk tabung.

Masalah:-

Solusi:-

Teman yang membantu: -

4. Task 2.1

```java
package task2;

public class Shape {
    private String color;
    private boolean filled;

    public Shape() {
        color = "green";
        filled = true;
    }

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

    public String getColor() {
        return color;
    }

    public boolean isFilled() {
        return filled;
    }

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

    public void setFilled(boolean filled) {
        this.filled = filled;
    }

    public String toString() {
        return "A Shape with color of " + this.color + " and " + (this.filled ? "filled" : "not filled");
    }
}
```

```java
public class Circle extends Shape { // Save as "Circle.java"
// private instance variable, not accessible from outside this class
    private double radius;


    public Circle() { // 1st (default) constructor
        radius = 1.0;

    }

    /** Constructs a Circle instance with the given radius and default color */
    public Circle(double r) { // 2nd constructor
        radius = r;
    }

    /** Returns the radius */
    public double getRadius() {
        return radius;
    }

    public void setRadius(double radius){
        this.radius = radius;
    }

    /** Returns the area of this Circle instance */
    public double getArea() {
    return radius*radius*Math.PI;
    }

    @Override
    public String toString() {
        return "A Circle with radius="+ this.radius +", which is a subclass of " + super.toString();
    }
}
```

```java
package task2;

public class Rectangle {
    private double width;
    private double length;


    public Rectangle(){
        width = 1.0;
        length = 1.0;
    }

    public Rectangle(double width, double length){
        this.width = width;
        this.length = length;
    }

    public double getWidth(){
        return width;
    }

    public double getLength(){
        return length;
    }

    public void setWidth(double width){
        this.width = width;
    }

    public void setLength(double length){
        this.length = length;
    }

    public double getArea(){
        return width*length;
    }

    @Override
    public String toString(){
        return "A Rectangle with width=" + this.width +"and length= " +this.length+ ", which is a subclass of" +  super.toString();
    }
}
```

```java
package task2;

public class Square  extends Rectangle{

    public Square(){
        super();

    }

    public Square(double side){
        super(side,side);
    }

    @Override
    public void setLength(double side){
        super.setLength(side);
        super.setWidth(side);
    }

    @Override
    public void setWidth(double side){
        super.setLength(side);
        super.setWidth(side);
    }

    @Override
    public String toString(){
        return "A Square with side="+ super.getWidth() +" which is a subclass of " + super.toString();
    }

}
```

Hasil akhir program

```
PS C:\Code\java\Pertemuan6\pertemuan6>  c:; cd 'c:\Code\java\Pertemuan6\pertemuan6'; & 'C:\Program Files\Java\jdk-22\bin\java.exe' '-XX:+ShowCodeDetailsInExceptionMessa
\pertemuan6\bin' 'task2.TestShape'
Shape color: green
Shape filled: true
Updated shape color: red
Updated shape filled: false
Shape toString: A Shape with color of red and not filled

Circle color: green
Circle filled: true
Circle radius: 1.0
Circle area: 3.141592653589793
Circle toString: A Circle with radius=1.0, which is a subclass of A Shape with color of green and filled

Rectangle color: green
Rectangle filled: true
Rectangle area: 1.0
Rectangle width: 1.0
Rectangle length: 1.0
Rectangle toString: A Rectangle with width=1.0and length= 1.0, which is a subclass ofA Shape with color of green and filled

Square color: green
Square filled: true
Square area: 1.0
Square toString: A Square with side=1.0 which is a subclass of A Rectangle with width=1.0and length= 1.0, which is a subclass ofA Shape with color of green and filled
```

Penjelasan:

Pada task 2.1, kita diminta untuk membuat empat kelas dengan ketentuan sebagai berikut:

a. Class `Shape` berfungsi sebagai superclass.
b. Class `Rectangle` dan `Circle` merupakan subclass atau spesialisasi dari kelas `Shape`.
c. Class `Square` merupakan subclass dari kelas `Rectangle`.

Selain itu, pada masing-masing kelas, kita diminta untuk membuat constructor dengan nilai awal (initial value) yang berbeda-beda, tergantung pada jumlah parameter dalam constructor. Jika constructor tidak memiliki parameter, maka atribut class tersebut akan diisi dengan nilai default saat constructor dijalankan. Selain itu, kita juga perlu membuat method getter dan setter di setiap kelas untuk mengakses dan memodifikasi nilai atribut.

Dalam daftar class ini, `Shape` merupakan class yang paling umum (general), yang mewariskan atribut dan method ke setiap class di bawahnya. Class `Shape` berfungsi sebagai superclass yang mendefinisikan fitur umum yang dimiliki oleh semua bentuk (shape), seperti warna dan apakah bentuk tersebut terisi atau tidak. Setiap subclass di level bawah, seperti `Rectangle`, `Circle`, dan `Square`, akan mewarisi atribut dan method dari class `Shape`, serta memiliki atribut dan method spesifik yang unik bagi masing-masing bentuk. Hal ini juga berlaku untuk class Rectangle yang mewariskan atribut dan methodnya pada class Square.

Selain itu, struktur ini menggunakan multi-level single inheritance, di mana subclass seperti `Square` mewarisi atribut dan method dari `Rectangle`, yang dimana Rectangle juga mewarisi atribut dan method dari `Shape`. Dengan begini Square dapat menggunakan method dan atribut yang berasal dari class Rectangle dan Shape.

5. Task 3.1

```
class Employee extends Sortable {
    public void raiseSalary(double byPercent) {
    }

    public int hireYear() {
        return hireyear;
    }

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

Hasil akhir program

```java
package task3;

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

```
PROBLEMS 2  DEBUG CONSOLE  PORTS  TERMINAL

PS C:\Code\java\Pertemuan6\pertemuan6> ^C
PS C:\Code\java\Pertemuan6\pertemuan6>
PS C:\Code\java\Pertemuan6\pertemuan6>  c:; cd 'c:\Code\java\Pertemuan6\pertemuan6'; & 'C:\Program Files\Ja
\pertemuan6\bin' 'task3.EmployeeTest'
Antonio Rossi 2100000.0 1989
Maria Bianchi 2625000.0 1991
Isabel Vidal 3150000.0 1993
PS C:\Code\java\Pertemuan6\pertemuan6>
```

```java
package task3;

public class ManagerTest {
    Run | Debug
    public static void main(String[] args) {
        Employee[] staff = new Employee[3];
        staff[0] = new Employee(n:"Antonio Rossi", s:2000000, day:1, month:10, year:1989);
        staff[1] = new Manager(n:"Maria Bianchi", s:2500000, d:1, m:12, y:1991);
        staff[2] = new Employee(n:"Isabel Vidal", s:3000000, day:1, month:11, year:1993);
        int i;
        for (i = 0; i < 3; i++) staff[i].raiseSalary(byPercent:5);
        for (i = 0; i < 3; i++) staff[i].print();
        // Sortable.shell_sort(staff1);
        // for (int i = 0; i < 3; i++)
        //     staff1[i].print();
    }
```

```
PROBLEMS 2  DEBUG CONSOLE  PORTS  TERMINAL

PS C:\Code\java\Pertemuan6\pertemuan6>  & 'C:\Program Files\Java\jdk-22\bin\java.exe' '-XX:+ShowCodeDetails
Antonio Rossi 2000000.0 1989
Maria Bianchi 2500000.0 1991
Isabel Vidal 3000000.0 1993
PS C:\Code\java\Pertemuan6\pertemuan6> ^C
PS C:\Code\java\Pertemuan6\pertemuan6>
PS C:\Code\java\Pertemuan6\pertemuan6>  c:; cd 'c:\Code\java\Pertemuan6\pertemuan6'; & 'C:\Program Files\Ja
\pertemuan6\bin' 'task3.ManagerTest'
Antonio Rossi 2100000.0 1989
Maria Bianchi 3037500.0 1991
Isabel Vidal 3150000.0 1993
PS C:\Code\java\Pertemuan6\pertemuan6>
```

Penjelasan:

Pada task 3.1, terdapat dua kelas, yaitu Employee dan Manager, di mana Employee berfungsi sebagai superclass bagi Manager dan mewariskan atribut serta method-nya kepada kelas Manager. Dalam kelas Manager, terdapat method raiseSalary yang sama namanya dengan yang ada di kelas Employee, namun dengan implementasi yang berbeda.

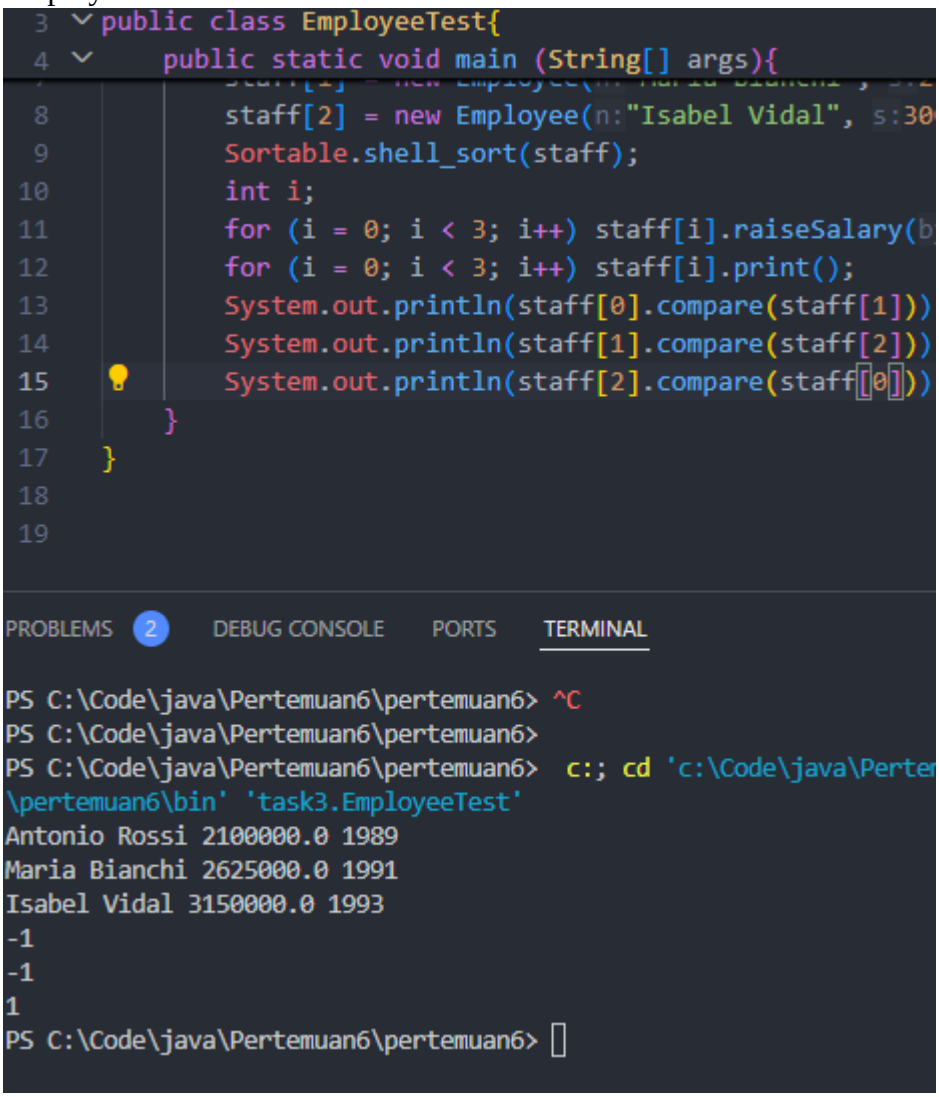
Meskipun method raiseSalary di kelas Manager tidak menggunakan overriding, saat kita memanggil method tersebut pada objek Manager, implementasi yang baru tetap dijalankan. Hal ini terjadi karena ketika objek yang dibuat adalah tipe Manager, maka Java akan menggunakan method raiseSalary yang ada pada kelas Manager meskipun tidak ada syntax @overriding, karena objek tersebut adalah instance dari subclass yang memiliki implementasi sendiri.

Pada kelas ManagerTest, kita membuat tiga objek Employee yang disimpan dalam bentuk array. Pada salah satu elemen array, kita mengisi dengan objek Manager (lebih tepatnya pada elemen kedua). Ini menunjukkan bahwa objek Manager dapat diperlakukan sebagai objek Employee karena Manager adalah subclass dari Employee. Dengan cara ini, kita dapat memanfaatkan polymorphism di mana metode yang dipanggil tergantung pada tipe objek yang sebenarnya, bukan pada tipe referensinya.

Maka dari itu pada saat object dengan nama Maria Bianchi dilakukan raiseSalary dan di print pada file class EmployeeTest dan ManagerTest memiliki hasil yang berbeda. Karena object dengan nama Maria Bianchi pada file class EmployeeTest merupakan object instansi dari class Employee maka dari itu dia menjalankan method raiseSalary() dari class Employee. Sedangkan object dengan nama Maria Bianchi pada file class ManagerTest meerupakan instalnsi dari class Manger maka dari itu dia menjalankan method raiseSalary() dari class Manager.

Pertanyaan:

1. When Sortable extended to Employee class, the method compare will be implemented. Please try the codes above. Call the method compare, in EmployeeTest class



2. Imagine that we want to order the Managers in a similar way : class Managers extends Employee extends Sortable It will be work? What is your solution?

   Untuk mencoba membuat sebuah class manager yang merupakan subclass dari sortable dan juga Employee dengan syntax class Managers extends Employee extends Sortable tidak akan bekerja, mengapa demikian? Hal ini dikarenakan java tidak memperbolehkan penulisan pewarisan 2 supperclass pada satu subclass dengan cara seperti ini, java akan memberikan pesan error pada saat menuliskan kedua syntax tersebut.

   Untuk Solusi dari saya adalah kita tidak perlu membuat subclass Manager yang merupakan subclass dari superclass Employee dan Sortable, mengapa begitu? Hal ini karena class employee sendiri telah menjadi subclass dari class Sortable maka dari itu kita hanya perlu membuat class manager mejadi subclass dari class Employee. Dengan demikian class Manager tetap bisa menagakses dan menerima warisan atirbut dan method dari kedua class tersebut.

   Namun, misal terdapat case dimana class Employee bukan merupakan subclass dari Sortable, kita dapat menggunakan konsep interface. Dalam hal ini, baik Employee maupun Sortable dapat memiliki metode dengan nama yang sama, seperti compare. Dengan mengimplementasikan interface Sortable di kelas Employee, Untuk contoh implmentasinya adalah sebagai berikut

```
abstract  interface Sortable{
    public abstract int compare(Sortable b);
    public static void shell_sort(Sortable[] a){
        for (int i = 0; i < a.length - 1; i++) {
            for (int j = i + 1; j < a.length; j++) {
                if (a[i].compare(a[j]) > 0) {
                    Sortable temp = a[i];
                    a[i] = a[j];
                    a[j] = temp;
                }
            }
        }
    }
}
```

```java
class Manager extends Employee implements Sortable {
    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);
    }

    @Override
    public int compare(Sortable o) {
        if (o instanceof Manager) {
            Manager eb = (Manager) o;
            if (super.getSalary() < eb.getSalary())
                return -1;
            if (super.getSalary() > eb.getSalary())
                return +1;
        } else if (o instanceof Employee) {
            Employee other = (Employee) o;
            return other.compare((Employee)o);
        }
        return 0;
    }

    public String getSecretaryName() {
        return secretaryName;
    }

    private String secretaryName;
}
```

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

Masalah:-

Solusi:-

Teman yang membantu: -

Link github source code week-6 Tugas-PBO/Week-6 at main · RaditZX/Tugas-PBO (github.com)