LAPORAN PRAKTIKUM

Pemrograman Berorientasi Objek

Diajukan untuk memenuhi salah satu tugas praktikum mata kuliah Pemrograman berorientasi objek

Disusun Oleh:

Hasbi Andi Muttaqin (231511049)

Jurusan Teknik Komputer dan Informatika

Program Studi D-3 Teknik Informatika

Politeknik Negeri Bandung 2024

DAFTAR ISI

Tanggal praktikum : 17 Oktober 2024

TASK 1

Nama Program : Another type of Employee

Deskripsi Tugas:

Write a class named Commission with the following features:

1. It extends the Hourly class.

```
public class Commission extends Hourly
```

Class Commission menjadi subclass dari class Courly

2. It has two instance variables (in addition to those inherited): one is the total sales the employee has made (type double) and the second is the commission rate for the employee (the commission rate will be type double and will represent the percent (in decimal form) commission the employee earns on sales (so .2 would mean the employee earns 20% commission on sales)).

```
private double totalsales;
private double commissionrate;
```

Attribut di set private untuk membatasi akses serta perubahan nilai-nilai tersebut hanya bisa dilakukan melalui metode di class tersebut.

3. The constructor takes 6 parameters: the first 5 are the same as for Hourly (name, address, phone number, social security number, hourly pay rate) and the 6th is the commission rate for the employee. The constructor should call the constructor of the parent class with the first 5 parameters then use the 6th to set the commission rate.

```
public Commission(String eName, String eAddress, String ePhone, String socSecNumber, double rate,double commissionrate) {
    super(eName, eAddress, ePhone, socSecNumber, rate);
    this.commissionrate = commissionrate;
}
```

Super digunakan untuk memanggil attribute dari parent nya dalam class ini yaitu Attribut dari Hourly

4. One additional method is needed: public void addSales (double totalSales) that adds the parameter to the instance variable representing total sales.

```
public void addSales(double totalsales){
    this.totalsales = this.totalsales + totalsales;
}
```

5. The pay method must call the pay method of the parent class to compute the pay for hours worked then add to that the pay from commission on sales. (See the pay method in the Executive class.) The total sales should be set back to 0

```
@Override
public double pay(){
    double payment = super.pay() + (totalsales * commissionrate);
    totalsales = 0;
    return payment;
}
```

Disini metode dari class hourly di override untuk digunakan di subclass nya yaitu commission, untuk

6. The toString method needs to call the toString method of the parent class then add the total sales to that.

```
@Override
public String toString(){
    String result = super.toString();
    result += "\nTotal Sales: " + totalsales;

    return result;
}
```

To test your class, update Staff.java as follows:

1. Increase the size of the array to 8.

```
staffList = new StaffMember[8];
```

2. Add two commissioned employees to the staffList—make up your own names, addresses, phone numbers and social security numbers.
3. Have one of the employees earn $6.25 per hour and 20% commission and the other one earn $9.75 per hour and 15% commission.
4. For the first additional employee you added, put the hours worked at 35 and the total sales $400; for the second, put the hours at 40 and the sales at $950.

```
staffList[6] = new Commission ("Hasbi", "Ciwaruga",
        "049-2024", "058-49-2024", 6.25,0.2);

staffList[7] = new Commission ("Daffa", "Cimahi",
        "038-2024","058-38-2024", 9.75,0.15);
```

Screenshot

```
------------------------------
Name: Hasbi
Address: Ciwaruga
Phone: 049-2024
Social Security Number: 058-49-2024
Current hours: 35
Total Sales: 400.0
Paid: 298.75
------------------------------
Name: Daffa
Address: Cimahi
Phone: 038-2024
Social Security Number: 058-38-2024
Current hours: 40
Total Sales: 950.0
Paid: 532.5
------------------------------
```

Permasalahan

Terdapat sedikit masalah pada menampilkan output dari commission yang tidak mempunyai nama dan hanya menampilkan total sales dan paid nya

Solusi

Pada Metode toString di Class Commission terdapat sedikit kesalahan, yaitu tidak menambahkan + sebelum =.

```
public String toString(){
    String result = super.toString();
    result += "\nTotal Sales: " + totalsales;

    return result;
}
```

Nama teman yang membantu

Daffa Al Ghifari

TASK 2

Nama Program : Painting Shapes

Deskripsi :

1. Write an abstract class Shape with the following properties:
   An instance variable shapeName of type String
   An abstract method area()
   A toString method that returns the name of the shape

```
public abstract class Shape {
    String shapeName;

     public Shape(String shapeName){
        this.shapeName = shapeName;
    }
```

```
    public abstract double area();


    public String toString(){
        return shapeName;
    }
}
```

2. The file Sphere.java contains a class for a sphere which is a descendant of Shape. A sphere has a radius and its area (surface area) is given by the formula 4*PI*radius^2. Define similar classes for a rectangle and a cylinder. Both the Rectangle class and the Cylinder class are descendants of the Shape class. A rectangle is defined by its length and width and its area is length times width. A cylinder is defined by a radius and height and its area (surface area) is PI*radius^2*height. Define the toString method in a way similar to that for the Sphere class.

   Class Cylinder

```
public class Cylinder extends Shape{
    double radius;
    double heigth;

    public Cylinder(double radius, double heigth){
        super("Cylinder");
        this.radius =radius;
        this.heigth = heigth;
    }

    @Override
    public double area(){
        return Math.PI * (radius * radius) *heigth;
    }

    @Override
    public String toString(){
        return super.toString() + " with radius " + radius + " and height " +
heigth + " area " + area();
    }
}
```

   Class Rectangle

```
public class Rectangle extends Shape {
    private double length, width;

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

    @Override
    public double area(){
        return length *width;
```

```
    }

    @Override
    public String toString(){
        return super.toString() + " with length " + length + " and width "
+width;
    }
}
```

Kedua class diatas merupakan subclass dari class Shape dan sama sama mewarisi atribut dan method nya

3. The file Paint.java contains a class for a type of paint (which has a "coverage" and a method to compute the amount of paint needed to paint a shape). Correct the return statement in the amount method so the correct amount will be returned. Use the fact that the amount of paint needed is the area of the shape divided by the coverage for the paint. (NOTE: Leave the print statement - it is there for illustration purposes, so you can see the method operating on different types of Shape objects.)

```
public class Paint {
    private double coverage; //number of square feet per gallon

//Sets up the paint object.

    public Paint(double coverage) {
        this.coverage = coverage;
    }


//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 s.area()/coverage;

    }
}
```

Test Program :

1. Instantiate the three shape objects: deck to be a 20 by 35 foot rectangle, bigBall to be a sphere of radius 15, and tank to be a cylinder of radius 10 and height 30.
2. Make the appropriate method calls to assign the correct values to the three amount variables.
3. Run the program and test it. You should see polymorphism in action as the amount method computes the amount of paint for various shapes.

```
public class PaintThings {

    public static void main(String[] args) {
        final double COVERAGE = 350;
        Paint paint = new Paint(COVERAGE);

        Shape deck = new Rectangle(20,35);
        Shape bigBall = new Sphere(15);
        Shape tank = new Cylinder( 10,30);
```

```
        double deckAmt = paint.amount(deck);
        double ballAmt = paint.amount(bigBall);
        double tankAmt = paint.amount(tank);

        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));

    }
}
```

Variabel deck bigBall dan tank bertipe shape, dengan masing masing merujuk kepada objek yang berbeda dimana dek merujuk rectangle bigball merujuk sphere dan tank merujuk cylinder. Ini memungkinkan penggunaan merode yang sama yaitu paint.amoutn pada objek objek yang berbeda.

Screenshot

```
Computing amount for Rectangle with length 20.0 and width 35.0
Computing amount for Sphere of radius 15.0
Computing amount for Cylinder with radius 10.0 and height 30.0 area 9424.77796076938

Number of gallons of paint needed ...
Deck 2
Big Ball 8.1
Tank 26.9

Process finished with exit code 0
```

Permasalahan

-

Solusi

-

Nama teman yang membantu

-

TASK 3

Nama Program : Polymorphic Sorting

Deskripsi :

1. The file Numbers.java reads in an array of integers, invokes the selection sort algorithm to sort them, and then prints the sorted array. Save Sorting.java and Numbers.java to your directory. Numbers.java won't compile in its current form. Study it to see if you can figure out why.

```
public static void main (String[] args)
    {
        Integer[] intList;
        int size;
```

```
        Scanner scan = new Scanner(System.in);

        System.out.print ("\nHow many integers do you want to sort? ");
        size = scan.nextInt();
        intList = new Integer[size];

        System.out.println ("\nEnter the numbers...");
        for (int i = 0; i < size; i++)
            intList[i] = scan.nextInt();

        Sorting.insertionSort(intList);

        System.out.println ("\nYour numbers in sorted order...");
        for (int i = 0; i < size; i++)
            System.out.print(intList[i] + " ");
        System.out.println();
    }
}
```

Ini dikarenakan kesalahan penggunaan tipe data yang digunakan, tipe yang digunakan adalah int sedangkan tipe data yang seharusnya adalah Integer, pada java keduanya itu berbeda, tipe data int merupakan tipe data primitive, sedangkan tipe data Integer merupakan wrapper yang mempresentasikan tipe primitif integer ke dalam suatu objek

4. Write a program Strings.java, similar to Numbers.java, that reads in an array of String objects and sorts them. You may just copy and edit Numbers.java.

```
import java.util.Scanner;

public class Strings
{
    //-----------------------------------------------------------------
    // Reads in an array of strings, sorts them,
    // then prints them in sorted order.
    //-----------------------------------------------------------------
    public static void main (String[] args)
    {
        String[] strList;
        int size;

        Scanner scan = new Scanner(System.in);

        System.out.print ("\nHow many strings do you want to sort? ");
        size = scan.nextInt();
        scan.nextLine(); // Membersihkan input buffer
        strList = new String[size];

        System.out.println ("\nEnter the strings...");
        for (int i = 0; i < size; i++)
            strList[i] = scan.nextLine();

        Sorting.insertionSort(strList);

        System.out.println ("\nYour strings in sorted order...");
        for (int i = 0; i < size; i++)
            System.out.print(strList[i] + " ");
        System.out.println();
```

```
    }
}
```

5. Modify the insertionSort algorithm so that it sorts in descending order rather than ascending order. Change Numbers.java and Strings.java to call insertionSort rather than selectionSort. Run both to make sure the sorting is correct.

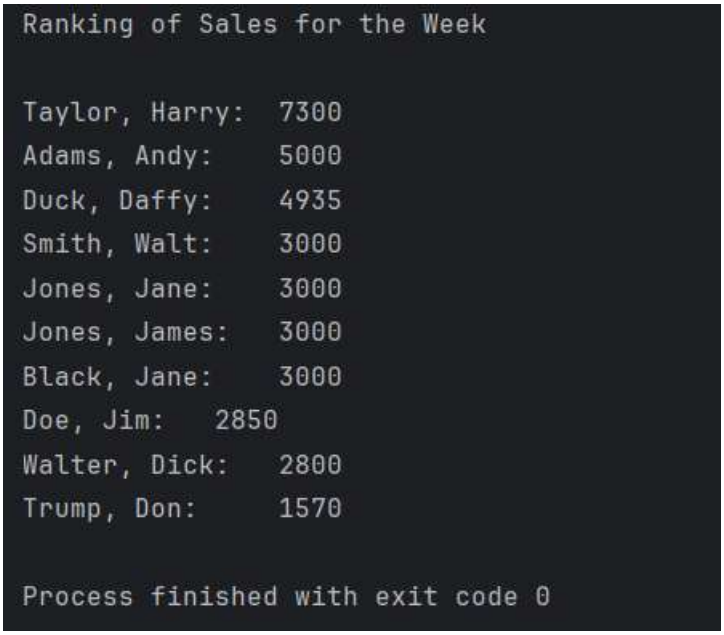
```
   public static void insertionSort (Comparable[] list)
{
    for (int index = 1; index < list.length; index++)
    {
        Comparable key = list[index];
        int position = index;

        // Ubah tanda perbandingan agar sorting menurun
        while (position > 0 && key.compareTo(list[position-1]) > 0)
        {
            list[position] = list[position-1];
            position--;
        }

        list[position] = key;
    }
}
```

6. The file Salesperson.java partially defines a class that represents a sales person. This is very similar to the Contact class in Listing 9.10. However, a sales person has a first name, last name, and a total number of sales (an int) rather than a first name, last name, and phone number. Complete the compareTo method in the Salesperson class. The comparison should be based on total sales; that is, return a negative number if the executing object has total sales less than the other object and return a positive number if the sales are greater. Use the name of the sales person to break a tie (alphabetical order).

```
public int compareTo(Object other)
{
    int result;

    // Compare total sales first
    result = totalSales - ((Salesperson)other).getSales();

    // If sales are equal, break the tie by comparing last name and then first
name
    if (result == 0)
    {
        result = lastName.compareTo(((Salesperson)other).getLastName());
        if (result == 0)
            result = firstName.compareTo(((Salesperson)other).getFirstName());
    }

    return result;
}
```

6. The file WeeklySales.java contains a driver for testing the compareTo method and the sorting (this is similar to Listing 9.8 in the text). Compile and run it. Make sure your compareTo method is correct. The sales staff should be listed in order of sales from most to least with the four people having the same number of sales in reverse alphabetical order.

Screenshot



Permasalahan

-

Solusi

-

Nama teman yang membantu

-

LINK GITHUB

https://github.com/Yorubreak/Pemrograman-Berorientasi-Objek.git