Praktek Pertemuan 78

Disusun untuk memenuhi tugas mata kuliah Pemrograman Berorientasi Objek (Teori)

Disusun Oleh:
Radja Restu Arsita (231511061) 2B – D3

Jurusan Teknik Komputer dan Informatika

Politeknik Negeri Bandung

2024

DAFTAR ISI

PEMBAHASAN

1.1 Another type of employee

a. Commission class

```
public class Commission extends Hourly{
    private double totalSales;
    private double commisionRate;

    public Commission(String eName, String eAddress, String ePhone, String
socSecNumber, double rate, double commisionRate) {
        super(eName, eAddress, ePhone, socSecNumber, rate);
        this.commisionRate = commisionRate;
        totalSales = 0;
    }

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

    @Override
    public double pay() {
        double payment = super.pay() + (this.totalSales * commisionRate);
        totalSales = 0;
        return payment;
    }

    @Override
    public String toString() {
        return super.toString() + "\ntotal sales : " + totalSales;
    }
}
```

b. Penjelasan

Menambahkan class commission yang meng extend Hourly dan memiliki 2 instance variable tambahan yaitu totalSales dan juga commisionRate, selain itu ada juga constructor untuk commision yang mana memiliki 5 argumen parent (superclass) yaitu eName, eAddress, ePhone, socSecNumber, rate lalu sebagai constructor untuk instance variable dari commission yaitu commisionRate. Class commission memiliki 3 method yang pertama method addSales untuk menambahkan totalSales dari karyawan hourly dan 2 method yang di override dari superclass nya yaitu pay dan juga toString yang diubah sesuai dengan kebutuhan dari soal.

c. Test

```
staffList[6] = new Commission( eName: "jyoo", eAddress: "666 black deep", ePhone: "666-0000"
                    , socSecNumber: "666-00-0000", rate: 6.25, commisionRate: 0.2);
staffList[7] = new Commission( eName: "kylee", eAddress: "777 white mustang", ePhone: "777-0000"
                    , socSecNumber: "777-00-0000", rate: 9.75, commisionRate: 0.15);
```

```
((Commission) staffList[6]).addHours( moreHours: 35);
((Commission) staffList[6]).addSales( totalSales: 400);
((Commission) staffList[7]).addHours( moreHours: 40);
((Commission) staffList[7]).addSales( totalSales: 950);
```

```
-----------------------------------
Name: jyoo
Address: 666 black deep
Phone: 666-0000
Social Security Number: 666-00-0000
Current hours: 35
total sales : 400.0
Paid: 298.75
-----------------------------------
Name: kylee
Address: 777 white mustang
Phone: 777-0000
Social Security Number: 777-00-0000
Current hours: 40
total sales : 950.0
Paid: 532.5
-----------------------------------
```

d. Penjelasan

Sesuai dengan soal disini saya meningkatkan kuota dari array staffmember menjadi 8 dan juga menambahkan 2 data sesuai dengan nama saya menggunakan nama panggilan saya dan menset current hours dan total sales sesuai dengan kebutuhan soal nya.

1.2 Painting Shapes

a. Class Shape

```
abstract class Shape {
    protected String nameShape;

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

```
    public abstract double area();

    public String toString() {
        return "nameShape = " + nameShape;
    }
}
```

b. Penjelasaan

Sesuai dengan kebutuhan dari soal saya menambah kan instance variable nameShape dan juga abstract method yaitu abstract dan 1 method toString yang digunakan untuk mengembalikan informasi detail mengenai shape.

c. Class Rectangle

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

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

    public double getLength() {
        return length;
    }

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

    public double getWidth() {
        return width;
    }

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

    @Override
    public double area() {
        return this.length * this.width;
    }

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

d. Penjelasan

Ada 2 instance variable dari rectangle yaitu length dan width selain itu class rectangle mengoverride method abstract dari superclass nya yaitu area dan juga toString yang disesuikan dengan konteks dari rectangle.

e. Class Cylinder

```
public class Cylinder extends Shape {
    private double radius;
    private double height;

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

    public double getRadius() {
        return radius;
    }

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

    public double getHeight() {
        return height;
    }

    public void setHeight(double height) {
        this.height = height;
    }

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

    @Override
    public String toString() {
        return super.toString() + " of radius " + radius + " of height " +
height;
    }
}
```

f. Penjelasan

Ada 2 instance variable dari rectangle yaitu radius dan height selain itu class Cylinder mengoverride method abstract dari superclass nya yaitu area dan juga toString yang disesuikan dengan konteks dari.

g. Class Paint

```
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.nameShape);
        return s.area() / this.coverage;
    }
}
```

h. Penjelasan

Memperbaiki nilai yang di return dari method amount yaitu menambahkan formula nilai return nya menjadi area dari shape yang dibagi oleh variable instance dari paint yaitu coverage.

i. Class paintThings

```
import java.text.DecimalFormat;

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

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

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

        DecimalFormat fmt = new DecimalFormat("0.#");
        System.out.println("\n Number of gallons of paint needed ...");
        System.out.println("Deck " + fmt.format(deckAmt));
        System.out.println("BigBall " + fmt.format(bigBallAmt));
        System.out.println("tank " + fmt.format(tankAmt));
    }
}
```

j. Penjelasan

Sesuai dengan kriteria yang diberikan oleh soal no 4 dimana kita harus membuat class paintThings yang didalam nya ada 3 object yaitu deck,bigball dan tank yang mana memiliki nilai yang disesuaikan dengan soal selain itu nilai coverage nya di set 350 dan hasil dari perhitungan method amount akan disimpan ke dalam sebuah variable dan nanti akan di tampilkan

k. Output

```
 Number of gallons of paint needed ...
Deck 2
BigBall 8.1
tank 26.9
```

1.3 Polymorphic Sorting

a. Soal no 1 dan 2

Untuk mengatasi masalah yang disebutkan di soal nomor 1 kita perlu mengubah tipe variable dari intList menjadi variable Integer karena Comparable merupakan subclass dari Comparable.

b. Soal no 3

```
import java.util.Scanner;

public class Strings
{
    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();
        strList = new String[size];

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

        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();
    }
}
```

c. Penjelasan

Membuat fungsi sorting untuk String yang serupa dengan sorting untuk integer disini saya hanya mengubah variable dari intList menjadi strList yang merupakan variable kumpulan String (*array of string*).

d. Soal no 4

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

        while (position > 0 && key.compareTo(list[position - 1]) > 0)
        {
            list[position] = list[position - 1];
            position--;
        }

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

e. Penjelasan

Disini saya merubah operator pembanding dari while nya untuk menjadi lebih dari agar order sorting menjadi descending.

f. Soal no 6

```
public class Salesperson implements Comparable<Salesperson>
{
    private String firstName, lastName;
    private int totalSales;

    public Salesperson(String first, String last, int sales)
    {
        firstName = first;
        lastName = last;
        totalSales = sales;
    }

    public String toString()
    {
        return lastName + ", " + firstName + ": \t" + totalSales;
    }

    public boolean equals(Object other)
    {
        return (lastName.equals(((Salesperson)other).getLastName()) &&
                firstName.equals(((Salesperson)other).getFirstName()));
    }

    public int compareTo(Salesperson other)
    {
        if (this.totalSales < other.getSales())
            return -1;
        else if (this.totalSales > other.getSales())
            return 1;
        else
            return (lastName + firstName).compareTo(other.getLastName() +
other.getFirstName());
    }

    public String getFirstName()
```

```
    {
        return firstName;
    }

    public String getLastName()
    {
        return lastName;
    }

    public int getSales()
    {
        return totalSales;
    }
}
```

g. Output

```
Ranking of Sales for the Week

Taylor, Harry:  7300
Adams, Andy:    5000
Duck, Daffy:    4935
Smith, Walt:    3000
Jones, Jane:    3000
Jones, James:   3000
Black, Jane:    3000
Doe, Jim:   2850
Walter, Dick:   2800
Trump, Don:     1570
```

h. Penjelasan

Sesuai soal disini sales nya diurutkan dari yang terbesar ke yang terkecil (descending) dan juga ketika ada kondisi dimana nilai sales nya sama maka nama yang sesuai alphabeth lebih rendah akan berada diatas (descending).

LESSON LEARNED

Dari tugas praktek PBO week 9 ini saya belajar dan mengimplementasikan penerapan mengenai materi Polymorphism.

REFERENSI

W3Schools. (n.d.). *Java Polymorphism*. Diakses pada 18 Oktober 2024, dari https://www.w3schools.com/java/java_polymorphism.asp

GeeksforGeeks. (n.d.). *Polymorphism in Java*. Diakses pada 18 Oktober 2024, dari https://www.geeksforgeeks.org/polymorphism-in-java/

LAMPIRAN