LAPORAN TUGAS MINGGU 7

Polymorphism

Pemorgaman Berorientasi Objek

Resume ini disusun untuk memenuhi Tugas Mata Kuliah Pemrograman Berorientasi Objek

Disusun oleh:
Muhammad Rivan Rivaldi 211511048

PROGRAM STUDI D3 TEKNIK INFORMATIKA
JURUSAN TEKNIK KOMPUTER DAN INFORMATIKA
POLITEKNIK NEGERI BANDUNG
2021

Studi Kasus 1. Another Type of Employee

1. Source Code
   a) Employee.java

```
public class Employee extends StaffMember {
    protected String socialSecurityNumber;
    protected double payRate;

    //-----------------------------------------------------------------
    // Sets up an employee with the specified information.
    //-----------------------------------------------------------------
    public Employee (String eName, String eAddress, String ePhone,String socSecNumber, double rate)
    {
        super (eName, eAddress, ePhone);
        socialSecurityNumber = socSecNumber;
        payRate = rate;
    }

    //-----------------------------------------------------------------
    // Returns information about an employee as a string.
    //-----------------------------------------------------------------
    @Override
    public String toString()
    {
        String result = super.toString ();
        result += "\nSocial Security Number: " + socialSecurityNumber;
        return result;
    }

    //-----------------------------------------------------------------
    // Returns the pay rate for this employee.
    //-----------------------------------------------------------------
    @Override
    public double pay()
    {
        return payRate;
    }
```

   b) Executive.java

```
public class Executive extends Employee {
    private double bonus;

    //-----------------------------------------------------------------
    // Sets up an executive with the specified information.
    //-----------------------------------------------------------------
    public Executive (String eName, String eAddress, String ePhone, String socSecNumber, double rate)
    {
        super (eName, eAddress, ePhone, socSecNumber, rate);
        bonus = 0; // bonus has yet to be awarded
    }

    //-----------------------------------------------------------------
    // Awards the specified bonus to this executive.
    //-----------------------------------------------------------------
    public void awardBonus (double execBonus)
    {
        bonus = execBonus;
    }

    //-----------------------------------------------------------------
    // Computes and returns the pay for an executive, which is the
    // regular employee payment plus a one-time bonus.
    //-----------------------------------------------------------------
    public double pay()
    {
        double payment = super.pay() + bonus;
        bonus = 0;
        return payment;
    }
}
```

c) Firm.java

```java
public class Firm {

	public static void main(String[] args) {
		// TODO Auto-generated method stub

		//-----------------------------------------------------------------
		// Creates a staff of employees for a firm and pays them.
		//-----------------------------------------------------------------
		Staff personnel = new Staff();
		personnel.payday();
	}

}
```

d) Hourly.java

```java
public class Hourly extends Employee {
	private int hoursWorked;

	//-----------------------------------------------------------------
	// Sets up this hourly employee using the specified information.
	//-----------------------------------------------------------------
	public Hourly (String eName, String eAddress, String ePhone,
		String socSecNumber, double rate)
	{
		super (eName, eAddress, ePhone, socSecNumber, rate);
		hoursWorked = 0;
	}

	//-----------------------------------------------------------------
	// Adds the specified number of hours to this employee's
	// accumulated hours.
	//-----------------------------------------------------------------
	public void addHours (int moreHours)
	{
		hoursWorked += moreHours;
	}

	//-----------------------------------------------------------------
	// Computes and returns the pay for this hourly employee.
	//-----------------------------------------------------------------
	public double pay()
	{
		double payment = payRate * hoursWorked;
		hoursWorked = 0;
		return payment;
	}
```

```
    //-----------------------------------------------------------------
    // Returns information about this hourly employee as a string.
    //-----------------------------------------------------------------
    public String toString()
    {
        String result = super.toString();
        result += "\nCurrent hours: " + hoursWorked;
        return result;
    }
}
```

e) Staff.java

```
public class Staff {
    StaffMember[] staffList;

    //-----------------------------------------------------------------
    // Sets up the list of staff members.
    //-----------------------------------------------------------------
    public Staff ()

    {
        staffList = new StaffMember[8];
        staffList[0] = new Executive ("Sam", "123 Main Line", "555-0469", "123-45-6789", 2423.07);
        staffList[1] = new Employee ("Carla", "456 Off Line", "555-0101", "987-65-4321", 1246.15);
        staffList[2] = new Employee ("Woody", "789 Off Rocker", "555-0000", "010-20-3040", 1169.23);
        staffList[3] = new Hourly ("Diane", "678 Fifth Ave.", "555-0690", "958-47-3625", 10.55);
        staffList[4] = new Volunteer ("Norm", "987 Suds Blvd.", "555-8374") ;
        staffList[5] = new Volunteer ("Cliff", "321 Duds Lane", "555-7282");
        staffList[6] = new Commission ("Rivan", "Banjaran", "123-1234", "456-89-9874", 6.25, 20);
        staffList[7] = new Commission ("Rivaldi", "Kabupaten Bandung", "321-4321", "259-98-3658", 9.75, 15);

        ((Executive)staffList[0]).awardBonus (500.00);
        ((Hourly)staffList[3]).addHours (40);

        ((Commission)staffList[6]).addHours(35);
((Commission)staffList[6]).addSales(400);
((Commission)staffList[7]).addHours(40);
((Commission)staffList[7]).addSales(550);
}
```

```
    //-----------------------------------------------------------------
    // Pays all staff members.
    //-----------------------------------------------------------------

    public void payday ()
    {
        double amount;
        for (int count=0; count < staffList.length; count++)
        {
            System.out.println (staffList[count]);
            amount = staffList[count].pay(); // polymorphic
            if (amount == 0.0)
                System.out.println ("Thanks!");
            else
                System.out.println ("Paid: " + amount);
                System.out.println ("-----------------------------------");
        }
    }
}
```

f) StaffMember.java

```java
abstract public class StaffMember {
    protected String name;
    protected String address;
    protected String phone;

    //-----------------------------------------------------------------
    // Sets up a staff member using the specified information.
    //-----------------------------------------------------------------
    public StaffMember (String eName, String eAddress, String ePhone)
    {
        name = eName;
        address = eAddress;
        phone = ePhone;
    }

    //-----------------------------------------------------------------
    // Returns a string including the basic employee information.
    //-----------------------------------------------------------------
    public String toString()
    {
        String result = "Name: " + name + "\n";
        result += "Address: " + address + "\n";
        result += "Phone: " + phone;
        return result;
    }

    //-----------------------------------------------------------------
    // Derived classes must define the pay method for each type of
    // employee.
    //-----------------------------------------------------------------
    public abstract double pay();
}
```

g) Volunteer.java

```
public class Volunteer extends StaffMember {
        //-----------------------------------------------------------------
        // Sets up a volunteer using the specified information.
        //-----------------------------------------------------------------
        public Volunteer (String eName, String eAddress, String ePhone)
        {
                super (eName, eAddress, ePhone);
        }

        //-----------------------------------------------------------------
        // Returns a zero pay value for this volunteer.
        //-----------------------------------------------------------------
        public double pay()
        {
                return 0.0;
        }
}
```

h) Commission.java

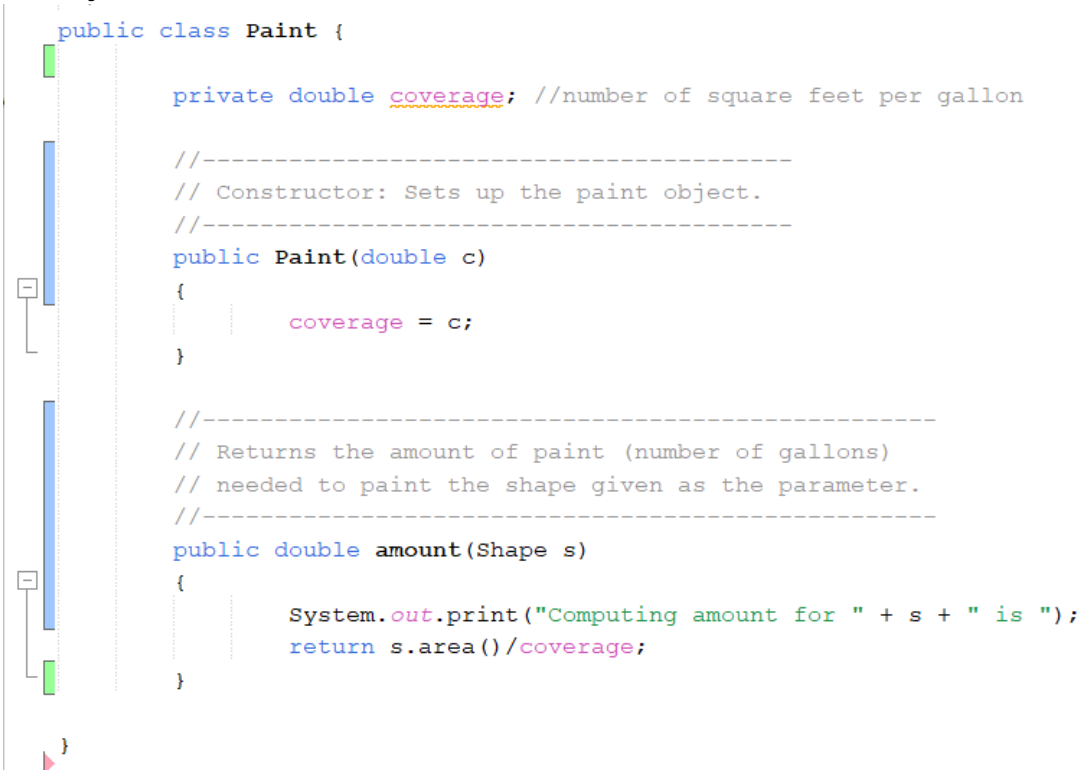
```
public class Commission extends Hourly {
        double totalSales2, commissionRate, payment;

        public Commission(String eName, String eAddress, String ePhone, String socSecNumber, double rate, double commissionRate) {
                super(eName, eAddress, ePhone, socSecNumber, rate);
                this.commissionRate = commissionRate;
                // TODO Auto-generated constructor stub
        }

        public void addSales (double totalSales){
                totalSales2 += totalSales;
        }

        @Override
        public double pay()
        {
                double payment = super.pay() + totalSales2 * commissionRate / 100;
                totalSales2 = 0;
                return payment;
        }

        @Override
        public String toString()
        {
                String result = super.toString ();
                result += "\nTotal Sales: " + totalSales2;
                return result;
        }
}
```

2. Output

```
run:
Name: Sam
Address: 123 Main Line
Phone: 555-0469
Social Security Number: 123-45-6789
Paid: 2923.07
------------------------------------
Name: Carla
Address: 456 Off Line
Phone: 555-0101
Social Security Number: 987-65-4321
Paid: 1246.15
------------------------------------
Name: Woody
Address: 789 Off Rocker
Phone: 555-0000
Social Security Number: 010-20-3040
Paid: 1169.23
------------------------------------
Name: Diane
Address: 678 Fifth Ave.
Phone: 555-0690
Social Security Number: 958-47-3625
Current hours: 40
Paid: 422.0
------------------------------------
Name: Norm
Address: 987 Suds Blvd.
Phone: 555-8374
Thanks!
------------------------------------
Name: Cliff
Address: 321 Duds Lane
Phone: 555-7282
Thanks!
------------------------------------
Name: Rivan
Address: Banjaran
Phone: 123-1234
Social Security Number: 456-89-9874
Current hours: 35
Total Sales: 400.0
Paid: 298.75
------------------------------------
Name: Rivaldi
Address: Kabupaten Bandung
Phone: 321-4321
Social Security Number: 258-98-3658
Current hours: 40
Total Sales: 950.0
Paid: 532.5
------------------------------------
BUILD SUCCESSFUL (total time: 0 seconds)
```

Studi Kasus 2. Painting Shapes

1. Source Code
   a) 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.print("Computing amount for " + s + " is ");
        return s.area()/coverage;
    }

}
```

b) PaintThings.java

```java
public class PaintThings {

        //----------------------------------------
        // Creates someshapes and a Paint object
        // and prints the amount of paintneeded
        // to paint each shape.
        //----------------------------------------

        public static void main(String[] args) {
                final double COVERAGE = 350;
                double coveraged;
                Paint _paint = new Paint(10);

                Shape deck = new Retangle(20, 35);
                System.out.println(deck.toString());
                coveraged = _paint.amount(deck);
                System.out.println(coveraged + "\n");

                Shape bigBall = new Sphere(15);
                System.out.println(((Sphere)bigBall).toString());
                coveraged = _paint.amount(bigBall);
                System.out.println(coveraged + "\n");

                Shape tank = new Cylinder(10, 30);
                System.out.println(((Cylinder)tank).toString());
                coveraged = _paint.amount(tank);
                System.out.println(coveraged + "\n");
        }

}
```

c) Sphere.java

```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 thesurface area of the sphere.
    //-----------------------------------------
    @Override
    public double area()
    {
        return 4*Math.PI*radius*radius;
    }

    //-----------------------------------
        // Returns the sphere as aString.
    //-----------------------------------
    @Override
    public String toString()
    {
        return super.toString() + " of radius " + radius;
    }
}
```

d) Cylinder.java

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

        public Cylinder(double r, double h) {
                super("Cylinder");
        radius = r;
        height = h;
        }

        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 4*Math.PI*radius*radius*height;
        }

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

e) Retangle.java

```java
public class Retangle extends Shape {
        private double width;
        private double length;

        public Retangle(double w, double l) {
                super("Rectangle");
        width = w;
        length = l;
        }

        public double getWidth() {
                return width;
        }

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

        public double getLength() {
                return length;
        }

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

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

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

f) Shape.java

```java
public abstract class Shape {
    private String shapeName;

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

    public String getShapeName() {
        return shapeName;
    }

    public void setShapeName(String shapeName) {
        this.shapeName = shapeName;
    }

    abstract double area();

    public String toString() {
        return "Name of the shape is " + shapeName;
    }
}
```

2. Ouput

```
run:
Name of the shape is Rectangle of width 20.0 of length 35.0
Computing amount for Name of the shape is Rectangle of width 20.0 of length 35.0 is 70.0

Name of the shape is Sphere of radius 15.0
Computing amount for Name of the shape is Sphere of radius 15.0 is 282.7433388230814

Name of the shape is Cylinder of radius 10.0 of height 30.0
Computing amount for Name of the shape is Cylinder of radius 10.0 of height 30.0 is 3769.9111843077517

BUILD SUCCESSFUL (total time: 0 seconds)
```

Studi Kasus 3. Polymorphic Sorting

1. Source Code
   a) Numbers

```
public class Numbers
{
        // --------------------------------------------
        // Reads in an array of integers, sorts them,
        // then prints them in sorted order.
        // --------------------------------------------
        public static void main (Strings[] 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.selectionSort(intList);

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

   b) Salesperson

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

    //-----------------------------------------------------------
    // Constructor: Sets up the sales person object with
    // the given data.
    //-----------------------------------------------------------
    public Salesperson (String first, String last, int sales)
    {
        firstName = first;
        lastName = last;
        totalSales = sales;
    }

    //-------------------------------------------
    // Returns the sales person as a string.
    //-------------------------------------------
    public String toString()
    {
        return lastName + ", " + firstName + ": \t" + totalSales;
    }

    //-------------------------------------------
    // Returns true if the sales people have
    // the same name.
    //-------------------------------------------
    public boolean equals (Salesperson s)
    {
        return (lastName.equals(s.getLastName()) && firstName.equals(s.getFirstName()));
    }
```

```
    //-------------------------------------------------
    // Order is based on total sales with the name
    // (last, then first) breaking a tie.
    //-------------------------------------------------
    public int compareTo(Salesperson s)
    {
        int result=-200;

        if(s.totalSales == this.totalSales)
        {
            if(equals(s))
                result = 0;
            else
            {
                if(s.firstName != firstName)
                {
                    result = (s.firstName).compareTo(firstName);
                    if(result > 0)
                        result=-1;
                }
                else
                {
                    result = (s.lastName).compareTo(lastName);
                    if(result > 0)
                        result=-1;
                }
            }
        }
        else
            if(s.totalSales < this.totalSales)
                result = 1;
            else
                result = -1;

            return result;
    }
    //-------------------------
    // First name accessor.
    //-------------------------
    public String getFirstName()
    {
        return firstName;
    }

    //-------------------------
    // Last name accessor.
    //-------------------------
    public String getLastName()
    {
        return lastName;
    }

    //-------------------------
    // Total sales accessor.
    //-------------------------
    public int getSales()
    {
        return totalSales;
    }
}
```

c) Sorting

```
public class Sorting
{
        //-----------------------------------------------------------------
        // Sorts the specified array of objects using the selection
        // sort algorithm.
        //-----------------------------------------------------------------
        public static void selectionSort (Comparable[] list)
        {
                int min;
                Comparable temp;
                for (int index = 0; index < list.length-1; index++)
                {
                        min = index;
                        for (int scan = index+1; scan < list.length; scan++){
                                if (list[scan].compareTo(list[min]) < 0){
                                        min = scan;
                                }
                        }

                        // Swap the values
                        temp = list[min];
                        list[min] = list[index];
                        list[index] = temp;
                }
        }

        //-----------------------------------------------------------------
        // Sorts the specified array of objects using the insertion
        // sort algorithm.
        //-----------------------------------------------------------------
        public static void insertionSort (Comparable[] list)
        {
                for (int index = 1; index < list.length; index++)
                {
                        Comparable key = list[index];
                        int position = index;

                        // Shift larger values to the right
                        while (position > 0 && key.compareTo(list[position-1]) > 0)
                        {
                                list[position] = list[position-1];
                                position--;
                        }
                        list[position] = key;
                }
        }
}
```

d) WeeklySales

```
public class WeeklySales
{
        public static void main(Strings[] args)
        {
                Salesperson[] salesStaff = new Salesperson[10];
                salesStaff[0] = new Salesperson("Jane", "Jones", 3000);
                salesStaff[1] = new Salesperson("Daffy", "Duck", 4935);
                salesStaff[2] = new Salesperson("James", "Jones", 3000);
                salesStaff[3] = new Salesperson("Dick", "Walter", 2800);
                salesStaff[4] = new Salesperson("Don", "Trump", 1570);
                salesStaff[5] = new Salesperson("Jane", "Black", 3000);
                salesStaff[6] = new Salesperson("Harry", "Taylor", 7300);
                salesStaff[7] = new Salesperson("Andy", "Adams", 5000);
                salesStaff[8] = new Salesperson("Jim", "Doe", 2850);
                salesStaff[9] = new Salesperson("Walt", "Smith", 3000);

                Sorting.insertionSort(salesStaff);

                System.out.println ("\nRanking of Sales for the Week\n");
                for (Salesperson s : salesStaff){
                        System.out.println (s);
                }
        }
}
```

e) Strings

```
public class Strings {
        public static void main(String[] args)
    {

        String[] StringList;
        int size;

        Scanner scan = new Scanner(System.in);

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

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

        Sorting.insertionSort(StringList);

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

2. Ouput

```
run:

How many Strings do you want to sort? 1

Enter the strings...
Rivan

Your numbers in sorted order...
Rivan
BUILD SUCCESSFUL (total time: 7 seconds)
```