**TUGAS : POLYMORPHISM**

**PERTEMUAN 8**

**PEMROGRAMAN BERORIENTASI OBJEK (TE)**

A blue and orange logo

Description automatically generated

Disusun oleh :

Muhammad Wildan Gumilang (231511087)

**PROGRAM DIPLOMA III TEKNIK INFORMATIKA**

**JURUSAN TEKNIK KOMPUTER DAN INFORMATIKA**

**POLITEKNIK NEGERI BANDUNG**

# **DAFTAR ISI**

[**DAFTAR ISI** 2](#_Toc180128048)

[**Studi Kasus 1** 3](#_Toc180128049)

[**Studi Kasus 2** 3](#_Toc180128050)

[**Studi Kasus 3** 3](#_Toc180128051)

# **Studi Kasus 1**

**Commision.java**

public class Commision extends Hourly

{

    private double totalSales;

    private double commisionRate;

    public Commision(String eName, String eAddress, String ePhone, String socSecNumber, double rate, double commisionRate) {

        super(eName, eAddress, ePhone, socSecNumber, rate);

        this.commisionRate = commisionRate;

    }

    public void addSales(double totalSales)

    {

        this.totalSales = totalSales;

    }

    @Override

    public double pay() {

        double payment = super.pay() + (totalSales \* commisionRate);

        totalSales = 0;

        return payment;

    }

    @Override

    public String toString()

   {

      String result = super.toString();

      result += "\nTotal Sales : " + totalSales;

      return result;

   }

}

**Staff.java**

private StaffMember[] staffList;

   //----------------------------------------------------------

   //  Constructor: 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 Commision("Wildan", "Sukajadi", "089874893", "00000", 6.25, 0.20);

      staffList[7] = new Commision("ALvito", "Cimahi", "0999", "0888", 9.75, 0.15);

      ((Executive)staffList[0]).awardBonus (500.00);

      ((Hourly)staffList[3]).addHours (40);

      ((Commision)staffList[6]).addHours (35);

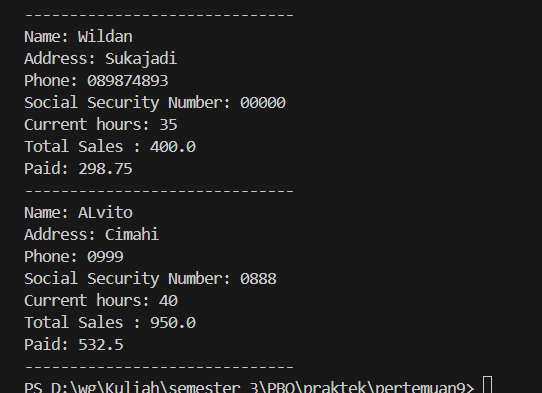
      ((Commision)staffList[6]).addSales(400);

      ((Commision)staffList[7]).addHours (40);

      ((Commision)staffList[7]).addSales(950);

   }

**Output dari dua karyawan yang ditambahkan :**

****

Polimorphism yang terjadi dalam program ini adalah ketika objek dari subclass (Commission) adalah sebagai objek dari superclassnya (Hourly atau Employee). Meskipun objek Commission disimpan dalam array StaffMember[] tapi ketika metode pay() dipanggil yang dijalankan adalah implementasi pay() milik Commission. Bukan Hourly ataupun Employee.

# **Studi Kasus 2**

**Shape.java**

public abstract class Shape {

    protected String shapeName;

    public Shape(String shapeName) {

        this.shapeName = shapeName;

    }

    public abstract double area();

    public String toString() {

        return "shapeName : " + shapeName + " ";

    }

}

**Sphere.java**

public class Sphere extends Shape {

    private double radius;

    public Sphere(String shapeName, double radius) {

        super("Sphere");

        this.radius = radius;

    }

    @Override

    public double area()

    {

        return 4\*Math.PI\*radius\*radius;

    }

    @Override

    public String toString() {

        return super.toString() + " of radius " + radius;

    }

}

**Paint.java**

public class Paint {

    private double coverage;

    public Paint(double c) {

        this.coverage = c;

    }

    public double amount(Shape s)

    {

        System.out.println("Computing amount for " + s);

        return s.area() / coverage;

    }

}

**PaintThings.java**

import java.text.DecimalFormat;

public class PaintThings {

    public static void main(String[] args) {

        final double COVERAGE = 350;

        Paint paint = new Paint(COVERAGE);

        Shape deck = new Rectangle("Persegi Panjang", 20, 35);

        Shape bigBall = new Sphere("Bola", 15);

        Shape tank = new Cylinder("Tabung", 10, 30);

        double dekAmt = 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(dekAmt));

        System.out.println("Big Ball " + fmt.format(ballAmt));

        System.out.println("Tank " + fmt.format(tankAmt));

    }

}

**Output :**

**A screen shot of a computer

Description automatically generated**

Polimorphism yang terjadi pada program ini adalah pada method amout dalam class Paint yang dapat digunakan pada beberapa objek dari subclassnya (Sphere, Rectangle, dan Cylinder) yang semuanya adalah turunan dari abstract class Shape. Meskipun objek tersebut berbeda kelas turunan, metode amount() dapat memanggil metode area(). Sehingga, meskipun amount() dipanggil pada objek-objek yang berbeda seperti Sphere, Rectangle, atau Cylinder, program akan secara memanggil metode area() yang sesuai untuk setiap bentuk, tanpa harus mengetahui tipe objek pada saat program dikompilasi.

# **Studi Kasus 3**

**Numbers.java**

import java.util.Scanner;

public class Numbers {

    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.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();

    }

}

**Function insertionSort**

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;

        }

    }

**weeklySales.java**

import java.util.Scanner;

public class WeeklySales {

    public static void main(String[] args) {

        Scanner scan = new Scanner(System.in);

        System.out.print("How many salespeople? ");

        int numSalespeople = scan.nextInt();

        SalesPerson[] salesPeople = new SalesPerson[numSalespeople];

        for (int i = 0; i < numSalespeople; i++) {

            System.out.print("Enter first name: ");

            String firstName = scan.next();

            System.out.print("Enter last name: ");

            String lastName = scan.next();

            System.out.print("Enter total sales: ");

            int totalSales = scan.nextInt();

            salesPeople[i] = new SalesPerson(firstName, lastName, totalSales);

        }

        Sorting.insertionSort(salesPeople);

        System.out.println("\nSalespeople sorted by sales (descending):");

        for (SalesPerson person : salesPeople) {

            System.out.println(person);

        }

        scan.close();

    }

}

**Output :**

A screen shot of a computer

Description automatically generated

Polimorphism pada program ini aadalah yaitu pada method compareTo yang diimplementasikan dalam kelas Salesperson. Yaitu pada saat method Sorting.insertionSort() memanggil method compareTo pada berbagai objek Salesperson. Sehingga hal ini menunjukkan bahwa satu method insertionSort dapat digunakan untuk berbagai tipe objek yang mengimplementasikan Comparable.