# LAPORAN PRAKTIKUM PERTEMUAN KE 8

*Disusun untuk memenuhi tugas praktikum mata kuliah Pemrograman Berorientasi Objek*


Disusun Oleh:

Dwika Ali Ramdhan (231511042)

2B – D3

Jurusan Teknik Komputer dan Informatika

Politeknik Negeri Bandung

2024

# DAFTAR ISI

**PEMBAHASAN**

## 1.1. Studi Kasus 1

The files Firm.java, Staff.java, StaffMember.java, Volunteer.java, Employee.java, Executive.java, and Hourly.java are from Listings 9.1 – 9.7 in the text. The program illustrates inheritance and polymorphism. In this exercise you will add one more employee type to the class hierarchy (see Figure 9.1 in the text). The employee will be one that is an hourly employee but also earns a commission on sales. Hence the class, which we'll name Commission, will be derived from the Hourly class.

Write a class named Commission with the following features:

- It extends the Hourly class.
- 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)).
- 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.
- One additional method is needed: public void addSales (double totalSales) that adds the parameter to the instance
- variable representing total sales.
- 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 (note: you don't need to set the hoursWorked back to 0—why not?).
- The toString method needs to call the toString method of the parent class then add the total sales to that.

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

- Increase the size of the array to 8.
- Add two commissioned employees to the staffList—make up your own names, addresses, phone numbers and social security numbers. 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.
- 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.

Compile and run the program. Make sure it is working properly. Role.java

### 1.1.1. StaffMember.java

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

    public StaffMember (String eName, String eAddress, String ePhone)
    {
        name = eName;
        address = eAddress;
        phone = ePhone;
```

```
    }

    public String toString()
    {
        String result = "Name: " + name + "\n";

        result += "Address: " + address + "\n";
        result += "Phone: " + phone;

        return result;
    }

    public abstract double pay();
}
```

Kelas abstrak yang mendefinisikan atribut umum seperti name, address, dan phone untuk semua anggota staf. Selain itu, terdapat metode abstrak pay(), yang akan diimplementasikan oleh kelas turunannya. Inheritance digunakan untuk mewariskan properti ini ke kelas turunan. Polimorfisme muncul saat metode pay() diimplementasikan secara berbeda di kelas-kelas turunannya.

### 1.1.2. Employee.java

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

    public Employee (String eName, String eAddress,
                     String ePhone, String socSecNumber,
                     double rate)
    {
        super (eName, eAddress, ePhone);

        socialSecurityNumber = socSecNumber;
        payRate = rate;
    }

    public String toString()
    {
        String result = super.toString();

        result += "\nSocial Security Number: " +
                socialSecurityNumber;

        return result;
    }

    public double pay()
    {
        return payRate;
    }
}
```

Employee adalah subclass dari StaffMember. Kelas ini menambahkan properti socialSecurityNumber dan payRate, serta mengimplementasikan metode pay() untuk mengembalikan payRate sebagai gaji dasar karyawan. Inheritance digunakan untuk mewarisi atribut dasar dari StaffMember, dan polimorfisme digunakan untuk mengimplementasikan metode pay()..

### 1.1.3. Executive.java

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

    public Executive (String eName, String eAddress,
                      String ePhone, String socSecNumber,
```

```
                        double rate)
    {
        super (eName, eAddress, ePhone, socSecNumber, rate);

        bonus = 0;
    }

    public void awardBonus (double execBonus)
    {
        bonus = execBonus;
    }

    public double pay()
    {
        double payment = super.pay() + bonus;

        bonus = 0;

        return payment;
    }
}
```

Kelas ini adalah subclass dari Employee yang menambahkan atribut bonus. Kelas ini mengimplementasikan metode pay() untuk menghitung pembayaran berdasarkan gaji pokok ditambah bonus. Penggunaan inheritance terlihat dengan menggunakan super() untuk memanggil konstruktor dan metode pay() dari kelas Employee. Polimorfisme terjadi ketika pay() di-override dan ditambahkan logika untuk menghitung bonus.

### 1.1.4. Hourly.java

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

    public Hourly(String eName, String eAddress, String ePhone, String
socSecNumber, double rate) {
        super(eName, eAddress, ePhone, socSecNumber, rate);
        hoursWorked = 0;
    }

    public void addHours(int moreHours) {
        hoursWorked += moreHours;
    }

    public double pay() {
        double payment = payRate * hoursWorked;
        hoursWorked = 0;
        return payment;
    }

    public String toString() {
        String result = super.toString();
        result += "\nCurrent hours: " + hoursWorked;
        return result;
    }
}
```

Kelas ini adalah subclass dari Employee yang menambahkan properti hoursWorked dan menyediakan metode untuk menambahkan jam kerja. Metode pay() dihitung berdasarkan jumlah jam kerja dan payRate. Di sini, inheritance digunakan untuk mewarisi properti karyawan, dan polimorfisme muncul karena metode pay() di-override untuk menghitung pembayaran berdasarkan jam kerja.

### 1.1.5. Volunteer.java

```
public class Volunteer extends StaffMember
{

    public Volunteer (String eName, String eAddress,
                      String ePhone)
    {
        super(eName, eAddress, ePhone);
    }

    public double pay()
    {
        return 0.0;
    }
}
```

Volunteer adalah subclass dari StaffMember, tetapi tidak memiliki gaji, sehingga metode pay() selalu mengembalikan 0. Di sini, inheritance digunakan untuk mewarisi atribut dasar, dan polimorfisme digunakan untuk mengimplementasikan metode pay() yang berbeda dari kelas turunan lainnya.

### 1.1.6. Commission.java

```
public class Commission extends Hourly {
    private double totalSales;
    private final double commissionRate;

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

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

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

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

Kelas ini adalah subclass dari Hourly yang menambahkan properti totalSales dan commissionRate. Metode pay() menghitung pembayaran berdasarkan jam kerja ditambah komisi penjualan. Inheritance digunakan untuk mewarisi struktur dari Hourly, sedangkan polimorfisme terlihat ketika metode pay() di-override untuk menambahkan komisi pada pembayaran.

### 1.1.7. Staff.java

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

    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("Dwika", "JL. Riung Damai 07",
"085797759592", "4374274323", 6.25, 0.2);
        staffList[7] = new Commission("Ali", "JL. Riung Damai 01",
"08573472913", "1434234532", 9.75, 0.15);

        ((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(950);
    }

    public void payday ()
    {
        double amount;

        for (StaffMember staffMember : staffList) {
            System.out.println(staffMember);

            amount = staffMember.pay();
            if (amount == 0.0)
                System.out.println("Thanks!");
            else
                System.out.println("Paid: " + amount);

            System.out.println("-------------------------------");
        }
    }
}
```

Kelas ini berisi array dari objek StaffMember, yang diisi dengan berbagai subclass seperti Executive, Employee, Hourly, Volunteer, dan Commission. Pada saat payday(), metode pay() dari masing-masing objek dipanggil secara polimorfik berdasarkan tipe objeknya (entah itu Executive, Hourly, Volunteer, dll.). Hal ini menunjukkan penerapan polimorfisme karena metode pay() di setiap kelas diturunkan memiliki perilaku yang berbeda.

### 1.1.8. Firm.java

```
public class Firm
{
    public static void main (String[] args)
    {
        Staff personnel = new Staff();

        personnel.payday();
```

```
    }
}
```

Kelas ini berfungsi sebagai titik masuk (main class) untuk menjalankan program. Di sini, objek Staff dibuat, dan metode payday() dipanggil untuk memproses pembayaran bagi semua anggota staf. Hal ini menunjukkan bagaimana polimorfisme diterapkan pada runtime, karena metode pay() yang berbeda dipanggil berdasarkan tipe spesifik objek dalam array staffList.

### 1.1.9. Output

```
-----------------------------
Name: Norm
Address: 987 Suds Blvd.
Phone: 555-8374
Thanks!
-----------------------------
Name: Cliff
Address: 321 Duds Lane
Phone: 555-7282
Thanks!
-----------------------------
Name: Dwika
Address: JL. Riung Damai 07
Phone: 085797759592
Social Security Number: 4374274323
Current hours: 35
Total Sales: 400.0
Paid: 298.75
-----------------------------
Name: Ali
Address: JL. Riung Damai 01
Phone: 08573472913
Social Security Number: 1434234532
Current hours: 40
Total Sales: 950.0
Paid: 532.5
-----------------------------
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
-----------------------------
```

## 1.2. Studi Kasus 2

In this lab exercise you will develop a class hierarchy of shapes and write a program that computes the amount of paint needed to paint different objects. The hierarchy will consist of a parent class Shape with three derived classes - Sphere, Rectangle, and Cylinder. For the purposes of this exercise, the only attribute a shape will have is a name and the method of interest will be one that computes the area of the shape (surface area in the case of three-dimensional shapes). Do the following.

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

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.

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.)

4. The file PaintThings.java contains a program that computes the amount of paint needed to paint various shapes. A paint object has been instantiated. Add the following to complete the program:

- 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.
- Make the appropriate method calls to assign the correct values to the three amount variables.
- Run the program and test it. You should see polymorphism in action as the amount method computes the amount of paint for various shapes.

### 1.2.1. Shape.java

```
public abstract class Shape {
    protected String shapeName;

    public Shape(String name){
        shapeName = name;
    }

    public abstract double area();

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

File ini mendefinisikan kelas abstrak Shape, yang merupakan superclass untuk semua bentuk geometri yang digunakan. Kelas ini memiliki atribut shapeName untuk menyimpan nama bentuk. Kelas ini juga mendeklarasikan metode abstrak area() yang akan diimplementasikan oleh subclass, serta metode toString() untuk mengembalikan nama bentuk. Karena kelas ini abstrak, objek dari kelas Shape tidak dapat dibuat secara langsung.

### 1.2.2. Cylinder.java

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

    public Cylinder(double hei, double rad){
        super("Cylinder");
        height = hei;
        radius = rad;
    }

    @Override
    public double area(){
        return 2 * Math.PI * radius * (radius + height);
    }

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

Kelas ini merupakan subclass dari Shape, yang mewakili bentuk silinder. Kelas ini menambahkan atribut height dan radius yang digunakan untuk menghitung luas permukaan silinder. Konstruktornya memanggil konstruktor superclass (Shape) dengan parameter "Cylinder" untuk mengatur nama bentuk. Metode area() di-override untuk menghitung luas permukaan silinder menggunakan rumus: 2πr(r+h)2 \pi r (r + h)2πr(r+h). Metode toString() juga di-override untuk menampilkan detail tambahan seperti radius dan tinggi silinder. Inheritance digunakan untuk mewarisi atribut dan metode dasar dari Shape, sedangkan polymorphism terjadi ketika metode area() dan toString() di-override sesuai dengan karakteristik bentuk silinder.

### 1.2.3. Rectangle.java

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

    public Rectangle (double len, double wid){
        super("Rectangle");
        length = len;
        width = wid;
    }

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

    public String toString(){
        return super.toString() + " panjang " + length + " lebar " + width;
    }
}
```

Kelas ini mewakili bentuk persegi panjang dan juga merupakan subclass dari Shape. Ia memiliki atribut tambahan yaitu length dan width untuk menghitung luas permukaan persegi panjang. Konstruktornya memanggil konstruktor superclass dengan nama "Rectangle". Metode area() di-override untuk menghitung luas persegi panjang dengan rumus: length×width. Seperti kelas lainnya, metode toString() juga di-override untuk menampilkan informasi panjang dan lebar. Inheritance diterapkan untuk mewarisi metode dan atribut dari Shape, sedangkan polymorphism terjadi melalui penggantian metode area() dan toString().

### 1.2.4. Sphere.java

```
public class Sphere extends Shape{
    private double radius;

    public Sphere(double rad){
        super("Sphere");
        radius = rad;
    }

    @Override
    public double area(){
        return 4 * Math.PI * Math.pow(radius, 2);
    }

    public String toString(){
        return super.toString() + " dengan Radius " + radius;
    }
}
```

Kelas ini mewakili bentuk bola (sphere) dan juga merupakan subclass dari Shape. Atribut yang digunakan adalah radius untuk menghitung luas permukaan bola. Konstruktornya memanggil konstruktor superclass dengan nama "Sphere". Metode area() di-override untuk menghitung luas permukaan bola dengan rumus: 4πr24 \pi r^24πr2. Metode toString() juga di-override untuk menambahkan informasi radius bola ke output. Inheritance memungkinkan Sphere mewarisi metode dasar dari Shape, sementara polymorphism muncul dalam penggantian metode area() dan toString().

### 1.2.5. Paint.java

```
public class Paint {
    private double coverage;

    public Paint(double c) {
        coverage = c;
    }

    public double amount(Shape s) {
        System.out.println("Computing amount for " + s);
        return s.area() / coverage;
    }
}
```

Kelas ini bertanggung jawab untuk menghitung jumlah cat yang diperlukan untuk mengecat suatu bentuk. Kelas ini memiliki atribut coverage, yaitu berapa banyak area yang dapat ditutupi oleh satu galon cat. Metode amount() menerima objek Shape sebagai parameter, memanfaatkan polymorphism di sini. Karena parameter bertipe Shape, objek dari semua subclass Shape (seperti Rectangle, Cylinder, dan Sphere) dapat digunakan di sini. Metode ini memanggil metode area() dari bentuk yang diteruskan untuk menghitung jumlah cat yang dibutuhkan berdasarkan luas permukaan dan cakupan cat.

### 1.2.6. PaintThings.java

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

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

        double deckAmount = paint.amount(deck);
        double ballAmount = paint.amount(bigBall);
        double tankAmount = paint.amount(tank);

        System.out.printf("Paint required for deck: %.2f gallons\n",
deckAmount);
        System.out.printf("Paint required for big ball: %.2f gallons\n",
ballAmount);
        System.out.printf("Paint required for tank: %.2f gallons\n",
tankAmount);
    }
}
```

Ini adalah kelas utama yang mengimplementasikan logika program. Objek Paint dibuat dengan 350 kaki persegi per galon cat. Tiga objek bentuk dibuat: Rectangle, Sphere, dan Cylinder. Karena semua objek tersebut adalah subclass dari Shape, mereka dapat disimpan dalam variabel bertipe Shape, menunjukkan penggunaan polymorphism. Program ini kemudian menghitung dan mencetak jumlah cat yang dibutuhkan untuk mengecat tiap-tiap bentuk dengan memanggil metode amount() dari objek Paint. Hasilnya menunjukkan jumlah galon yang dibutuhkan untuk mengecat masing-masing bentuk.

### 1.2.7. Output

```
Computing amount for Rectangle panjang 20.0 lebar 35.0
Computing amount for Sphere dengan Radius 15.0
Computing amount for Cylinder radius 30.0 height 10.0
Paint required for deck: 2,00 gallons
Paint required for big ball: 8,08 gallons
Paint required for tank: 21,54 gallons
```

## 1.3. Studi Kasus 3

The file Sorting.java contains the Sorting class from Listing 9.9 in the text. This class implements both the selection sort and the insertion sort algorithms for sorting any array of Comparable objects in ascending order. In this exercise, you will use theSorting class to sort several different types of objects.

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.
2. Try to compile Numbers.java and see what the error message is. The problem involves the difference between primitive data and objects. Change the program so it will work correctly (note: you don't need to make many changes - the autoboxing feature of Java 1.5 will take care of most conversions from int to Integer).
3. 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.
4. 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.
5. 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).
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.
7. OPTIONAL: Modify WeeklySales.java so the salespeople are read in rather than hardcoded in the program.

### 1.3.1. Salesperson.java

```
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 + ", " + lastName + ": " + totalSales;
    }

    public int compareTo(Salesperson other){
        if (this.totalSales != other.totalSales){
            return  other.totalSales - this.totalSales;
        } else {
            return other.lastName.compareTo(this.lastName);
        }
    }
}
```

Kelas ini merepresentasikan data seorang salesperson yang mencakup atribut firstName, lastName, dan totalSales. Kelas ini mengimplementasikan interface Comparable<Salesperson>, yang berarti setiap objek Salesperson dapat dibandingkan satu sama lain berdasarkan kriteria yang telah ditentukan. Metode compareTo() di-override untuk menentukan urutan salesperson berdasarkan total penjualan (totalSales), dan jika total penjualannya sama, dibandingkan dengan nama belakang mereka. Ini adalah contoh polimorfisme karena metode compareTo() memberikan implementasi yang berbeda sesuai dengan cara Salesperson

dibandingkan.

### 1.3.2. Sorting.java

```
public class Sorting {

    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;
                }
            }

            temp = list[min];
            list[min] = list[index];
            list[index] = temp;
        }
    }

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

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

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

Kelas ini berisi dua metode untuk mengurutkan array: selectionSort() dan insertionSort(). Kedua metode ini bekerja dengan array dari objek-objek yang mengimplementasikan interface Comparable, seperti Integer, String, atau objek lain yang mengimplementasikan compareTo(). Dengan menggunakan polimorfisme, kedua metode sorting ini dapat menerima array yang berisi berbagai tipe data yang bisa dibandingkan, selama mereka mengimplementasikan Comparable. Ini menunjukkan polimorfisme karena array dapat diisi dengan berbagai tipe data (Integer, String, atau Salesperson), dan metode compareTo() yang sesuai akan dipanggil saat membandingkan elemen-elemen dalam array.

### 1.3.3. Numbers.java

```
import java.util.Scanner;

public class Numbers {
    public static void main(String[] args) {
        Scanner scan = new Scanner(System.in);

        System.out.println("How many numbers do you want to sort?");
        int numItems = scan.nextInt();
        Integer[] numbers = new Integer[numItems];

        System.out.println("Enter the numbers:");
        for (int i = 0; i < numItems; i++) {
            numbers[i] = scan.nextInt();  // Autoboxing akan mengubah int
menjadi Integer
        }
```

```
        Sorting.selectionSort(numbers);

        System.out.println("Your numbers in sorted order:");
        for (int number : numbers) {
            System.out.print(number + " ");
        }
    }
}
```

Kelas ini menerima input dari pengguna berupa sejumlah angka yang akan diurutkan. Input ini disimpan dalam array Integer, yang kemudian diurutkan menggunakan metode selectionSort() dari kelas Sorting. Karena Integer mengimplementasikan interface Comparable, metode compareTo() yang ada pada Integer akan dipanggil untuk membandingkan elemen-elemen dalam array selama proses sorting. Di sini, polimorfisme terjadi ketika metode selectionSort() dipanggil dengan array Integer, di mana metode compareTo() dari Integer yang digunakan.

### 1.3.4. Strings.java

```
import java.util.Scanner;

public class Strings {
    public static void main(String[] args) {
        Scanner scan = new Scanner(System.in);

        System.out.println("How many strings do you want to sort?");
        int numItems = scan.nextInt();
        scan.nextLine();  // Consume newline

        String[] strings = new String[numItems];

        System.out.println("Enter the strings:");
        for (int i = 0; i < numItems; i++) {
            strings[i] = scan.nextLine();
        }

        Sorting.selectionSort(strings);

        System.out.println("Your strings in sorted order:");
        for (String str : strings) {
            System.out.print(str + " ");
        }
    }
}
```

Program ini menerima input berupa string dari pengguna, menyimpannya dalam array String, dan kemudian mengurutkannya menggunakan metode selectionSort() dari kelas Sorting. Sama seperti dalam kasus angka, String juga mengimplementasikan interface Comparable, sehingga metode compareTo() yang ada pada String digunakan selama proses sorting. Polimorfisme diterapkan di sini karena selectionSort() dapat bekerja dengan array String dan memanfaatkan metode compareTo() milik kelas String.

### 1.3.5. weeklySales.java

```
import java.util.Arrays;
import java.util.Scanner;

public class weeklySales {
    public static void main(String[] args){
        Scanner scan = new Scanner(System.in);

        System.out.println("Masukkan total sales: ");
        int numSalesPeople = scan.nextInt();
        scan.nextLine();

        Salesperson[] salespeople = new Salesperson[numSalesPeople];
```

```
        for (int i = 0; i < numSalesPeople; i++){
            System.out.println("Masukkan first name, last name, dan jumlah
sales");
            String firstName = scan.next();
            String lastName =  scan.next();
            int totalSales = scan.nextInt();

            salespeople[i] = new Salesperson(firstName, lastName, totalSales);
        }

        Arrays.sort(salespeople);

        System.out.println("sales staff in order of sales from most to least :
");
        for (Salesperson sp : salespeople){
            System.out.println(sp);
        }
    }
}
```

Program ini mengumpulkan informasi tentang beberapa salesperson, termasuk nama depan, nama belakang, dan total penjualan mereka, lalu menyimpannya dalam array Salesperson. Setelah semua data salesperson dimasukkan, array tersebut diurutkan menggunakan metode Arrays.sort(), yang memanfaatkan metode compareTo() dari kelas Salesperson untuk menentukan urutan berdasarkan total penjualan. Di sini, polimorfisme terjadi ketika Arrays.sort() memanggil metode compareTo() yang diimplementasikan pada objek Salesperson. Program ini kemudian mencetak daftar salesperson dalam urutan penjualan terbanyak hingga yang paling sedikit.

### 1.3.6. Output

```
How many numbers do you want to sort?
5
Enter the numbers:
9
2
7
5
4
Your numbers in sorted order:
2 4 5 7 9
```

```
How many strings do you want to sort?
5
Enter the strings:
kafe
cafe
roar
paw
kim
Your strings in sorted order:
cafe kafe kim paw roar
```

```
Masukkan total sales:
5
Masukkan first name, last name, dan jumlah sales
Ali Ramdhan 11
Masukkan first name, last name, dan jumlah sales
Ramdhan Ali 4
Masukkan first name, last name, dan jumlah sales
Ridho Ali 9
Masukkan first name, last name, dan jumlah sales
dika kari 60
Masukkan first name, last name, dan jumlah sales
Dear all 88
sales staff in order of sales from most to least :
all, all: 88
```