

LAPORAN PRAKTIKUM PERTEMUAN-8

Diajukan untuk memenuhi salah satu tugas praktikum Mata kuliah Pemrograman Berorientasi Objek

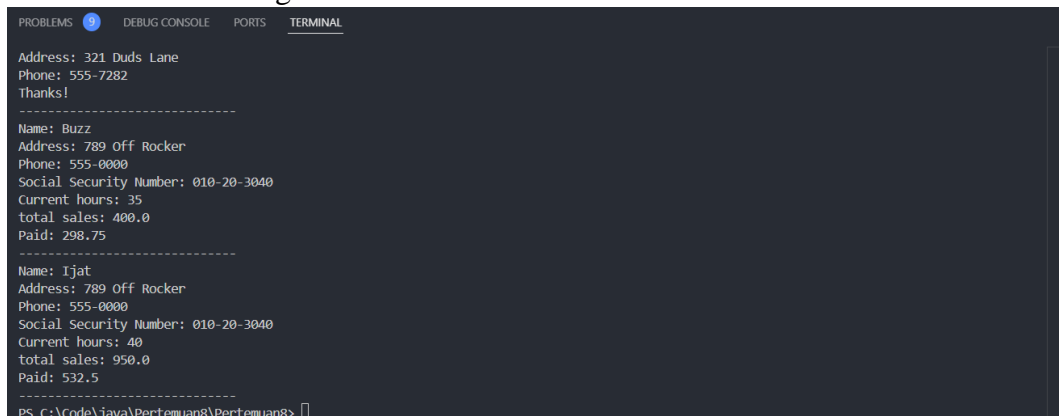


**Disusun Oleh:
Daiva Raditya Pradipa (231511039)**

**Jurusan Teknik Komputer dan Informatika
Program Studi D-3 Teknik Informatika
Politeknik Negeri Bandung
2024**

1. Studi kasus 1

a. Screenshot Hasil Program



The screenshot shows a terminal window with the following output:

```
Address: 321 Duds Lane
Phone: 555-7282
Thanks!
-----
Name: Buzz
Address: 789 Off Rocker
Phone: 555-0000
Social Security Number: 010-20-3040
Current hours: 35
total sales: 400.0
Paid: 298.75
-----
Name: Ijat
Address: 789 Off Rocker
Phone: 555-0000
Social Security Number: 010-20-3040
Current hours: 40
total sales: 950.0
Paid: 532.5
-----
PS C:\Code\java\Pertemuan8\Pertemuan8>
```

b. Penjelasan

Polymorphism yang digunakan adalah polymorphism inheritance dengan superclass yang menunjuk pada subclass. Hal ini dapat dibuktikan dari file `commission.java` berikut

```
1 package studikasu1;
2
3 public class Commission extends Hourly{
4     private double totalSales;
5     private double commissionRate;
6
7     public Commission(String eName, String eAddress, String ePhone, String socSecNumber, double rate, double commissionRate){
8         super (eName, eAddress, ePhone, socSecNumber, rate);
9         this.commissionRate = commissionRate;
10    }
11
12    public void addSales (double totalSales){
13        this.totalSales = this.totalSales + totalSales;
14    }
15
16    @Override
17    public double pay() {
18        double basePay = super.pay();
19        double commission = totalSales * commissionRate;
20        totalSales = 0;
21        return basePay + commission;
22    }
23
24    @Override
25    public String toString()
26    {
27        String result = super.toString();
28
29        result += "\ntotal sales: " + totalSales;
30
31        return result;
32    }
33 }
34
35
```

Dari code ini terdapat beberapa ciri ciri yang menunjukkan bahwa class `commission` menerapkan polymorphism inheritance, diantaranya adalah

- 1) Class tersebut mereferensikan ke class `Hourly` yang dibuktikan dengan syntax “`extends Hourly`”
- 2) Class tersebut menginherit class yang direferensikannya, terbukti dengan penggunaan syntax “`super`” pada constructor dan beberapa method seperti `toString()` dan `pay()` yang menggunakan variable inheritance dan method dari superclassnya yaitu `Hourly`
- 3) Class tersebut mengoveride atau mengganti implementasi method dari superclassnya menjadi implementasi yang lebih spesifik. Contohnya pada

method pay() yang dimana dia mengoveride method pay() dari class Hourly dan mengganti implementasinya menjadi lebih spesifik untuk class commission dalam hal ini untuk total gaji yang terdiri dari gaji pokok ditambah jumlah komisi yang didapatkan dari total penjualan.

Kemudian pada file Staff.java yang dimana pada studi kasus ini merupakan file untuk menginstansikan object object dari class yang ada seperti Hourly, Comission, dll.

```
public class Staff
{
    public Staff ()
    {
        staffList = new StaffMember[8];

        staffList[0] = new Executive (eName:"Sam", eAddress:"123 Main Line",
            ePhone:"555-0469", socSecNumber:"123-45-6789", rate:2423.07);

        staffList[1] = new Employee (eName:"Carla", eAddress:"456 Off Line",
            ePhone:"555-0101", socSecNumber:"987-65-4321", rate:1246.15);
        staffList[2] = new Employee (eName:"Woody", eAddress:"789 Off Rocker",
            ePhone:"555-0000", socSecNumber:"010-20-3040", rate:1169.23);

        staffList[3] = new Hourly (eName:"Diane", eAddress:"678 Fifth Ave.",
            ePhone:"555-0690", socSecNumber:"958-47-3625", rate:10.55);
        staffList[4] = new Volunteer (eName:"Norm", eAddress:"987 Suds Blvd.",
            ePhone:"555-8374");
        staffList[5] = new Volunteer (eName:"Cliff", eAddress:"321 Duds Lane",
            ePhone:"555-7282");

        staffList[6] = new Commission (eName:"Buzz", eAddress:"789 Off Rocker", ePhone:"555-0000", socSecNumber:"010-20-3040",
            rate:6.25, commissionRate:0.20);

        staffList[7] = new Commission (eName:"Ijat", eAddress:"789 Off Rocker",
            ePhone:"555-0000", socSecNumber:"010-20-3040", rate:9.75, commissionRate:0.15);

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

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

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

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

Dari file ini, kita bisa lihat terdapat syntax yang menunjukkan bahwa polymorphism yang diterapkan adalah polymorphism dengan tipe superclass yang menunjuk ke subclass. Hal ini dibuktikan dengan penggunaan tipe data superclass untuk menunjuk ke objek-objek dari subclass. Hal ini terlihat pada deklarasi array staffList yang bertipe StaffMember, yang merupakan superclass dari berbagai kelas lain (level tertinggi). Meskipun bertipe StaffMember, masing-masing elemen array staffList diisi dengan objek dari subclass seperti Commission, Volunteer, Executive, dan Employee.

Dengan kata lain, polymorphism diterapkan ketika objek dari berbagai subclass diakses melalui referensi superclass. Sehingga meskipun staffList bertipe StaffMember, objek-objek yang spesifik seperti Commission atau Executive dapat diinstansiasi dan digunakan di dalamnya.

c. Jawaban Soal

- 1) Write a class named Commission with the following features
 - a. It extends the Hourly class.

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

- b. 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)).

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

- c. 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 Commision(String eName, String eAddress, String ePhone
,String socSecNumber, double rate, double commissionRate){
    super (eName, eAddress, ePhone, socSecNumber, rate);
    this.commisionRate = commissionRate;
}
```

- d. 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;
}
```

- e. 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?)

```
@Override
public double pay() {
    double basePay = super.pay();
    double commission = totalSales * commisionRate;
    totalSales = 0;
    return basePay + commission;
}
```

- f. 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;
}
```

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

- a. Increase the size of the array to 8

```
public Staff ()
{
    staffList = new StaffMember[8];
}
```

- b. 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

```
staffList[6] = new Commision (eName:"Buzz", eAddress:"789 Off Rocker",ePhone:"555-0000", socSecNumber:"010-20-3040"
, rate:6.25, commisionRate:0.20);

staffList[7] = new Commision (eName:"Ijat", eAddress:"789 Off Rocker",
ePhone:"555-0000", socSecNumber:"010-20-3040", rate:9.75, commisionRate:0.15);
```

- c. 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.

```
((Commision)staffList[6]).addHours (moreHours:35);
((Commision)staffList[6]).addSales(totalSales:400);

((Commision)staffList[7]).addHours (moreHours:40);
((Commision)staffList[7]).addSales(totalSales:950);
```

2. Studi kasus 2

- a. Screenshot hasil program

```
Number of gallons of paint needed...
Deck 2
Big Ball 8,1
Tank 26,9
PS C:\Code\java\Pertemuan8\Pertemuan8> 
```

- b. Penjelasan

Polymorphism yang digunakan adalah polymorphism inheritance. Hal ini dapat dibuktikan dari file Cylinder.java berikut

```

1 package studikassus2;
2
3 public class Cylinder extends Shape {
4     private double radius;
5     private double height;
6
7     public Cylinder(double r, double height){
8         super(shapeName:"Sphere");
9         this.radius = r;
10        this.height = height;
11    }
12
13    @Override
14    public double area(){
15        return Math.PI*radius*radius*height;
16    }
17
18    @Override
19    public String toString(){
20        return super.toString() + " of radius " + radius + " and height " + height;
21    }
22 }
23

```

Dari code ini terdapat beberapa ciri ciri yang menunjukan bahwa class commission menerapkan polymorphism inheritance, diantaranya adalah

- 1) Class tersebut mereferensikan ke class Shape yang dibuktikan dengan syntax “extends Shape”
- 2) Class tersebut menginherit class yang direferensikannya, terbukti dengan penggunaan syntax “super” pada constructor dan beberapa method seperti toString() yang menggunakan variable inheritance dan method dari superclassnya yaitu Shape
- 3) Class tersebut mengoveride atau mengganti implementasi method dari superclassnya menjadi implementasi yang lebih spesifik. Contohnya pada method toString() yang dimana dia mengoveride method toString() dari class Shape dan mengganti implementasinya menjadi lebih spesifik untuk class Cylinder dalam hal ini untuk menampilkan nama shape yang menggunakan method toString() dari class Shape yang digabungkan dengan statement “ ” of radius ” + radius + ” and height ” + height” .

Kemudian pada file PaintThings.java yang dimana pada studi kasus ini merupakan main file untuk menjalankan program, sekaligus untuk menginstansikan object dari class seperti Sphere, Cylinder, dan Rectangle.

```

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

        Shape deck;
        Shape bigBall;
        Shape tank;

        double deckAmt, ballAmt, tankAmt;

        // Instantiate the three shapes to paint
        deck = new Rectangle(length:20,width:35);
        bigBall = new Sphere(r:15);
        tank = new Cylinder(r:10,height:30);

        // Compute the amount of paint needed for each shape
        deckAmt = paint.amount(deck);
        ballAmt = paint.amount(bigBall);
        tankAmt = paint.amount(tank);

        // Print the amount of paint for each
        DecimalFormat fmt = new DecimalFormat(pattern:"0.#");
        System.out.println(x:"\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));
    }
}

```

Dari code ini bis kita lihat bahwa polymorphism yang diterapkan adalah polymorphism dengan tipe superclass yang menunjuk ke subclass (sama dengan studi kasus pertama). Dimana Shape yang merupakan superclass dijadikan tipe data dari variabel yang akan menampung objek dari subclass Shape yaitu Rectangle, Sphere, dan Cylinder. Meskipun variable tersebut bertipekan Shape yang memiliki method toString() sebagai berikut

```

abstract class Shape {
    private String shapeName;

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

    abstract double area();
    public String toString(){
        return "Shape name: " + shapeName;
    }
}

```

Namun saat salah satu object tersebut dipanggil, dia akan tetap memanggil method dari class yang diinstansikan tersebut, sebagai contoh disini saya akan coba memanggil method to string dari object deck dengan method toString() sebagai berikut.

```

3 public class Rectangle extends Shape {
7     public Rectangle(double length, double width){
11     }
12
13     @Override
14     public double area(){
15         return length * width;
16     }
17
18     @Override
19     public String toString(){
20         return super.toString() + " of length " + length + " and width" + width;
21     }
22 }
23

```

```

5 public class PaintThings {
6     public static void main(String[] args) {
25     System.out.println(deck.toString());
26
27     // Print the amount of paint for each
28     DecimalFormat fmt = new DecimalFormat(pattern:"0.#");
29     System.out.println("Number of gallons of paint needed...");
30     System.out.println("Deck " + fmt.format(deckAmt));
31     System.out.println("Big Ball " + fmt.format(ballAmt));
32     System.out.println("Tank " + fmt.format(tankAmt));
33     }
34 }
35

```

PROBLEMS 9 DEBUG CONSOLE PORTS TERMINAL

```

Tank 26,9
PS C:\Code\java> ^C
PS C:\Code\java>
PS C:\Code\java> c:; cd 'c:\Code\java'; & 'C:\Program Files\Java\jdk-22\bin\java.exe' '-XX:+ShowCodeDetailsInExceptionMessages
oaming\Code\User\workspaceStorage\bd82415b2a39f0785191bfc4eec80dde\redhat.java\jdt_ws\java_ca66a52c\bin' 'studikasus2.PaintThin
Shape name: Rectangle of length 20.0 and width35.0

Number of gallons of paint needed...
Deck 2
Big Ball 8,1
Tank 26,9
PS C:\Code\java>

```

Bisa dilihat pada terminal output, saat menjalankan method toString dari object deck yang diinstansikan dari class Rectangle, dia memanggil method toString() dari class Rectangle dan bukan dari class Shape.

c. Jawaban Soal

- a. 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


```

abstract class Shape {
    private String shapeName;

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

    abstract double area();
    public String toString(){
        return "Shape name: " + shapeName;
    }
}

```

- b. 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 $4 \cdot \pi \cdot \text{radius}^2$. 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 $\pi \cdot \text{radius}^2 \cdot \text{height}$. Define the toString method in a way similar to that for the Sphere class.

```

public class Sphere extends Shape {
    private double radius;

    public Sphere(double r){
        super(shapeName:"Sphere");
        this.radius = r;
    }

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

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

```

- c. 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;

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

    public double amount(Shape s){
        return s.area() / this.coverage;
    }
}
```

- d. 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.

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

        Shape deck;
        Shape bigBall;
        Shape tank;

        double deckAmt, ballAmt, tankAmt;

        // Instantiate the three shapes to paint
        deck = new Rectangle(length:20,width:35);
        bigBall = new Sphere(r:15);
        tank = new Cylinder(r:10,height:30);

        // Compute the amount of paint needed for each shape
        deckAmt = paint.amount(deck);
        ballAmt = paint.amount(bigBall);
        tankAmt = paint.amount(tank);
        System.out.println(deck.toString());

        // Print the amount of paint for each
        DecimalFormat fmt = new DecimalFormat(pattern:"0.##");
        System.out.println(x:"\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));
    }
}
```

3. Studi kasus 3

a. Screenshot hasil program

Ranking of Sales for the Week

Taylor, Harry: 7300
Adams, Andy: 5000
Duck, Daffy: 4935
Jones, Jane: 3000
Jones, James: 3000
Black, Jane: 3000
Smith, Walt: 3000
Doe, Jim: 2850
Walter, Dick: 2800
Trump, Don: 1570
PS C:\Code\java>

b. Penjelasan

Polymorphism yang digunakan adalah polymorphism interface dengan superclass yang menunjuk pada subclass. Hal ini dapat dibuktikan dari file SalesPerson.java berikut

```
studikasuk3 > J SalesPerson.java > SalesPerson > gettotalSales()
package studikasuk3;

public class SalesPerson implements Comparable {
    private String firstName, lastName;
    private int totalSales;

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

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

    @Override
    public int compareTo(Object obj) {
        if (obj instanceof SalesPerson) {
            SalesPerson other = (SalesPerson) obj;
            return Integer.compare(this.totalSales, other.getTotalSales());
        } else {
            throw new ClassCastException(s:"Invalid object type for comparison");
        }
    }

    @Override
    public String toString() {
        return firstName + " " + lastName + ": " + totalSales;
    }

    public String getFirstName() {
        return firstName;
    }
}
```

Dari code ini terdapat beberapa ciri ciri yang menunjukkan bahwa class commission menerapkan polymorphism inheritance, diantaranya adalah

- 1) Class tersebut mengimplemen interface Comparable yang dibuktikan dengan syntax “implements Comparable”
- 2) Class tersebut mengoveride method toString() dan compareTo() yang merupakan method dari interface Comparable.

c. Jawaban Soal

- 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.

Hal ini dikarenakan tipe data yang digunakan untuk menampung bilangan yang di sort yaitu array intList menggunakan tipe data int yang merupakan tipe data

primitive. Sedangkan method SelectionSort() menerima nilai parameter array dengan tipe Comparable yang merupakan tipe data object. Maka dari itu kita harus mengcasting tipe data int menjadi integers yang merupakan tipe data objek.

- 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).

```
public class Numbers {  
    Run | Debug  
    public static void main(String[] args) {  
        Integer[] intlist;  
        int size;  
  
        Scanner scan = new Scanner(System.in);  
        System.out.print(s:"\nHow many Integers do you want to sort?");  
        size = scan.nextInt();  
        intlist = new Integer[size];  
  
        System.out.println(x:"\n Enter the Numbers....");  
        for (int i = 0; i < size; i++){  
            intlist[i] = scan.nextInt();  
        }  
        Sorting.selectionSort(intlist);  
        Sorting.insertionSort(intlist);  
  
        System.out.println(x:"\nYour numbers in sorted order...");  
        for(int i = 0 ; i < size; i++){  
            System.out.print(intlist[i] + " ");  
        }  
        System.out.println();  
    }  
}
```

- 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.

```

public class Strings {
    Run | Debug
    public static void main(String[] args) {
        String[] strlist;
        int size;

        Scanner scan = new Scanner(System.in);
        System.out.print(s:"\nHow many String do you want to sort?");
        size = scan.nextInt();
        strlist = new String[size];

        System.out.println(x:"\n Enter the String....");
        for (int i = 0; i < size; i++){
            strlist[i] = scan.next();
        }
        Sorting.selectionSort(strlist);

        System.out.println(x:"\nYour String in sorted order...");
        for(int i = 0 ; i < size; i++){
            System.out.print(strlist[i] + " ");
        }
        System.out.println();
    }
}

```

- 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.

```

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;
    }
}

```

How many Integers do you want to sort?5

Enter the Numbers....

5

3

1

2

4

Your numbers in selection sorted order...

1 3 4 2 5

Your numbers in insertion sorted order...

5 4 3 2 1

PS C:\Code\java> A

- 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).

```
@Override
public int compareTo(Object obj) {
    if (obj instanceof SalesPerson) {
        SalesPerson other = (SalesPerson) obj;
        if (this.totalSales < other.getTotalSales()) {
            return -1;
        } else if (this.totalSales > other.getTotalSales()) {
            return 1;
        } else {
            return 0;
        }
    } else {
        throw new ClassCastException(s:"Invalid object type for comparison");
    }
}
```

- 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.

```
Ranking of Sales for the Week

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

Link github: [Tugas-PBO/Week-8 at main · RaditZX/Tugas-PBO \(github.com\)](https://github.com/RaditZX/Tugas-PBO)