W7 - FUNDAMENTAL PROGRAMMING STRUCTURES IN JAVA

LAPORAN PRAKTIKUM

Disusun untuk memenuhi tugas Mata Kuliah Pemrograman Berorientasi Objek

Disusun oleh

Nazwa Fitriyani Zahra 211511051

PROGRAM STUDI D3 TEKNIK INFORMATIKA

JURUSAN TEKNIK KOMPUTER DAN INFORMATIKA

POLITEKNIK NEGERI BANDUNG

2022

A. Studi Kasus 1 : Another Type of Employee

//Firm.java

```java
/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template
 */
package anotherEmployee;

/**
 *
 * @author NAZWA FZ
 */
public class Firm {
    public static void main (String[] args){
        Staff personnel = new Staff();

        personnel.payday();
    }
}

```

//Staff.java

```java
/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template
 */
package anotherEmployee;

/**
 *
 * @author NAZWA FZ
 */
public class Staff {
    StaffMember[] staffList;

    public Staff(){
        staffList = new StaffMember[6];

        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");

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

    }

    public void payday(){
        double amount;

        for(int count=0; count < staffList.length; count++){
            System.out.println(staffList[count]);
            amount = staffList[count].pay();

            if(amount == 0.0){
                System.out.println("Thanks !");
            }
            else{
                System.out.println("Paid : " + amount);
            }
            System.out.println("-------------------");
        }
    }
}
```

//StaffMember.java

```java
/**
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template
 */
package anotherEmployee;

/**
 *
 * @author NAZWA FZ
 */
abstract public 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;
    }


```

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

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

        return result;
    }

    public abstract double pay();
}

```

//Employee.java

```java
package anotherEmployee;

/**
 *
 * @author NAZWA FZ
 */
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;
    }

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

        result += "\nSocial Security Number : " + socialSecurityNumber;
        return result;
    }
```

```java
    }
    @Override
    public double pay(){
        return payRate;
    }
```

//Executive.java

```java
 */
package anotherEmployee;

/**
 *
 * @author NAZWA FZ
 */
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;
    }
}
```

//Hourly.java

```java
 */
package anotherEmployee;

/**
 *
 * @author NAZWA FZ
 */
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;
    }
```

```java

    @Override
    public double pay(){
        double payment = payRate * hoursWorked;

        hoursWorked = 0;

        return payment;
    }

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

        result += "\nCurrent hours :" + hoursWorked;

        return result;
    }
```

//Volunteer.java

```java
 */
package anotherEmployee;

/**
 *
 * @author NAZWA FZ
 */
public class Volunteer extends StaffMember {
    public Volunteer (String eName, String eAddress, String ePhone){
        super(eName,eAddress, ePhone);
    }

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

Hasil 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 !
```

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.

//Commission.java

```java
 */
package anotherEmployee;

/**
 *
 * @author NAZWA FZ
 */
public class Commission extends Hourly {

    //2 instance variable
    private double totalSales;
    private double commissionRate;

    public Commission (String eName, String eAddress, String ePhone, String socSecNumber, double rate, double commissionRate){
        super(eName, eAddress, ePhone, socSecNumber, rate);
        //the total sales should be set back to 0
        this.totalSales = 0.0;
        //2 for 20%
        this.commissionRate = commissionRate/10;
    }
```

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

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

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

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.

//Staff.java

```java
 *
 * @author NAZWA FZ
 */
public class Staff {
    StaffMember[] staffList;

    public Staff(){
        //size of array = 8
        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");

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

        //Tambahan
        //First Commission
        staffList [6] = new Commission("Nazwa", "9/159D Dago Babakan", "0877-6368-5268","0877-6368-5268",6.25,2.0);
        ((Hourly)staffList[6]).addHours (35);
        ((Commission)staffList[6]).addSales(400);
        //Second Commission
        staffList [7] = new Commission("Fitriyani", "9/159D Dago Babakan", "0877-6368-5268","0877-6368-5268",9.75,1.5);
        ((Hourly)staffList[7]).addHours (40);
        ((Commission)staffList[7]).addSales(950);
    }
```

```java
    public void payday(){
        double amount;

        for(int count=0; count < staffList.length; count++){
            System.out.println(staffList[count]);
            amount = staffList[count].pay();

            if(amount == 0.0){
                System.out.println("Thanks !");
            }
            else{
                System.out.println("Paid : " + amount);
            }
            System.out.println("-------------------");
        }
    }
}
```

Hasil 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 : Nazwa
Address : 9/159D Dago Babakan
Phone : 0877-6368-5268
Social Security Number : 0877-6368-5268
Current hours :35
Total sales = 400.0
Paid : 298.75
-------------------
Name : Fitriyani
Address : 9/159D Dago Babakan
Phone : 0877-6368-5268
Social Security Number : 0877-6368-5268
Current hours :40
Total sales = 950.0
Paid : 532.5
-------------------
BUILD SUCCESSFUL (total time: 0 seconds)
```

B. Studi Kasus 2 : Painting Shapes

//Sphere.java

```java
 */
package PaintingShapes;

/**
 *
 * @author NAZWA FZ
 */
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 the surface area of the sphere.
    //----------------------------------------
    public double area()
    {
        return 4*Math.PI*radius*radius;
    }

```

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

//Paint.java

```java
 */
package PaintingShapes;

/**
 *
 * @author NAZWA FZ
 */
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);
        return 0;
    }
}
```

//PaintThings.java

```java
 */
package PaintingShapes;

/**
 *
 * @author NAZWA FZ
 */
public class PaintThings {
    public static void main (String[] args){
        final double COVERAGE = 350;
        Paint paint = new Paint(COVERAGE);
        Rectangle deck;
        Sphere bigBall;
        Cylinder tank;

        double dectAmt, ballAmt, tankAmt;

        DecimalFormal fmt = DecimalFormat("0.#");
        System.out.println ("\nNumber of gallons of paint neeeded...");
        System.out.println ("Deck" + fmt.format(deckAmt));
        System.out.println("Big Ball " + fmt.format (ballAmt));
        System.out.println("Tank " + fmt.format(tankAmt));
    }
}
```

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

```java
 */
package PaintingShapes;

/**
 *
 * @author NAZWA FZ
 */
abstract public class Shape
{
    //instance variable called shapeName
    private String shapeName;

    //constructor that return shapeName
    public Shape (String shapeName)
    {
        this.shapeName = shapeName;
    }
    //abstract methode area
    public abstract double area();
    public  String toString()
    {
        String result = "Shape Name : "+this.shapeName;
        return result;
    }
}

```

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.

//Rectangle.java

```java
 *
 * @author NAZWA FZ
 */
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;
    }

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

    @Override
    public String toString()
    {
        String result = super.toString() + " of length : "+ length+" and widht: "+width;
        return result;
    }
```

//Cylinder.java

```java
 *
 * @author NAZWA FZ
 */
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;
    }

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

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

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.)

```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.println ("Computing amount for " + s);
        return (s.area()/ coverage);
    }
}
```

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.

```java
 */
package PaintingShapes;
import java.text.DecimalFormat;
/**
 *
 * @author NAZWA FZ
 */
public class PaintThings {
 public static void main (String[] args)
 {
    final double COVERAGE = 350;
    Paint paint = new Paint(COVERAGE);
    Rectangle deck = new Rectangle(25,35);
    Sphere bigBall = new Sphere(15);
    Cylinder tank = new Cylinder(10,30);

    double deckAmt, ballAmt, tankAmt;
    // Instantiate the three shapes to paint

    // 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("0.#");
    System.out.println ("\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));
 }
```

Hasil Output :

```
Commission.java   Output - W7 (run)   Sphere.java   Sphere.java   Paint.java

run:
Computing amount for Shape Name : Rectangle of length : 25.0 and widht: 35.0
Computing amount for Shape Name : Sphere of radius 15.0
Computing amount for Shape Name : Cylinder of radius : 10.0 of height: 30.0

Number of gallons of paint needed...
Deck 2.5
Big Ball 8.1
Tank 26.9
BUILD SUCCESSFUL (total time: 0 seconds)
```

C. Studi Kasus 3 : Polymorphic Sorting

//Sorting.java

```java
 */
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;
        }
    }

    //
```

```java
    //-----------------------------------------------------------------
    // 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;
        }
    }
```

//Numbers.java

```java
public class Numbers {
    // ------------------------------------------
    // Reads in an array of integers, sorts them,
    // then prints them in sorted order.
    // ------------------------------------------
    public static void main (String[] args)
    {
        int[] 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 int[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 ();
        scan.close();
    }
}
```

//Salesperson

```java
public class Salesperson implements Comparable
{
    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 (Object other)
    {
        return (lastName.equals(((Salesperson)other).getLastName()) &&
        firstName.equals(((Salesperson)other).getFirstName()));
    }
```

```java
    // (last, then first) breaking a tie.
    //-----------------------------------------------------
    public int compareTo(Object other)
    {
        int result;
        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;
    }
}
```

//WeeklySales.java

```java
*/
public class WeeklySales {
    public static void main(String[] 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);

    }
}
```

Hasil Output error :

```
public double area()
{
PaintingShapes.Sphere
Output - W7 (run)
run:
Exception in thread "main" java.lang.RuntimeException: Uncompilable code - variable result might not have been initialized
        at PolimorphicSorting.Salesperson.compareTo(Salesperson.java:1)
        at PolimorphicSorting.Sorting.insertionSort(Sorting.java:45)
        at PolimorphicSorting.WeeklySales.main(WeeklySales.java:28)
C:\Users\NAZWA FZ\AppData\Local\NetBeans\Cache\14\executor-snippets\run.xml:111: The following error occurred while executing this line:
C:\Users\NAZWA FZ\AppData\Local\NetBeans\Cache\14\executor-snippets\run.xml:68: Java returned: 1
BUILD FAILED (total time: 1 second)
```

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.

Modify code :

//Number.java

```java
public class Numbers
{
    // ------------------------------------------
    // Reads in an array of integers, sorts them,
    // then prints them in sorted order.
    // ------------------------------------------
    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);
        Sorting.insertionSort(intList);
        System.out.println ("\nYour numbers in sorted order...");
        for (int i = 0; i < size; i++)
            System.out.print(intList[i] + " ");
        System.out.println ();
        scan.close();
    }
}
```

//Salesperson.java

```java
public class Salesperson implements Comparable
{
    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;
    }
```

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

    //-----------------------------------------------------
    // Order is based on total sales with the name
    // (last, then first) breaking a tie.
    //-----------------------------------------------------
    public int compareTo(Object other)
    {
        int result;
        if(totalSales>((Salesperson)other).totalSales) result = 1;
        else if (totalSales<((Salesperson)other).totalSales) result = -1;
        else
        {
            result = firstName.compareTo(((Salesperson)other).firstName);
            if(result == 0) result = lastName.compareTo(((Salesperson)other).lastName);
        }
        return result;
    }

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

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

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

//Sorting.java

```java
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;
        }
    }
```

```java
        //-----------------------------------------------------------------
        // 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;
            }
        }
    }
```

//Strings.java

```java
 */
public class Strings
{
    // ---------------------------------------------
    // Reads in an array of integers, sorts them,
    // then prints them in sorted order.
    // ---------------------------------------------
    public static void main (String[] args)
    {
        String[] stringList;
        int size;
        Scanner scan = new Scanner(System.in);
        System.out.print ("\nHow many integers do you want to sort? ");
        size = scan.nextInt();
        stringList = new String[size];
        System.out.println ("\nEnter the numbers...");
        for (int i = 0; i < size; i++)
            stringList[i] = scan.next();
        // Sorting.selectionSort(stringList);
        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 ();
        scan.close();
    }
}
```

//WeeklySales.java

```java
*/
public class WeeklySales
{
    public static void main(String[] 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);
//          Sorting.selectionSort(salesStaff);
//          System.out.println ("\nRanking of Sales for the Week\n");
//          for (Salesperson s : salesStaff)
//          System.out.println (s);

        Scanner scan = new Scanner(System.in);
        System.out.println("How many sales person? ");
        int size = scan.nextInt();
        Salesperson[] salesStaff = new Salesperson[size];
        for(int i=0;i<size;i++)
        {
            String firstName = scan.next();
            String lastName = scan.next();
            int totalSales = scan.nextInt();
            salesStaff[i] = new Salesperson(firstName, lastName, totalSales);
        }
        Sorting.insertionSort(salesStaff);
        System.out.println ("\nRanking of Sales for the Week\n");
        for (Salesperson s : salesStaff)
        System.out.println (s);
        scan.close();


    }
}
```