LAPORAN TUGAS MINGGU 7

PRAKTIKUM POLIMORPHISM

Laporan ini disusun untuk memenuhi tugas mata kuliah Pemrograman Berorientasi Objek

Disusun oleh :

Yumi Febriana 211511063

PROGRAM DIPLOMA III TEKNIK INFORMATIKA
POLITEKNIK NEGERI BANDUNG
BANDUNG
2022

DAFTAR ISI

Daftar isi

LINK GITHUB :

https://github.com/yumifee/PBO-P-W7.git

TASK 1

Another Type of Employee

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.

SOURCE CODE SOAL

```
//********************************************************************
//  Firm.java        Author: Lewis/Loftus
//
//  Demonstrates polymorphism via inheritance.
//********************************************************************

public class Firm
{
   //-----------------------------------------------------------------
   //  Creates a staff of employees for a firm and pays them.
   //-----------------------------------------------------------------
   public static void main (String[] args)
   {
      Staff personnel = new Staff();

      personnel.payday();
   }
}
```

```
//********************************************************************
//  Staff.java       Author: Lewis/Loftus
//
//  Represents the personnel staff of a particular business.
//********************************************************************

public class Staff
{
   StaffMember[] staffList;

   //-----------------------------------------------------------------
   //  Sets up the list of staff members.
   //-----------------------------------------------------------------
   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);
   }

   //-----------------------------------------------------------------
   //  Pays all staff members.
   //-----------------------------------------------------------------
   public void payday ()
   {
      double amount;

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

         amount = staffList[count].pay();  // polymorphic

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

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

```java
//********************************************************************
//  StaffMember.java       Author: Lewis/Loftus
//
//  Represents a generic staff member.
//********************************************************************

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

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

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

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

      return result;
   }

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

```java
//********************************************************************
//  Volunteer.java       Author: Lewis/Loftus
//
//  Represents a staff member that works as a volunteer.
//********************************************************************

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

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

```java
//********************************************************************
//  Employee.java       Author: Lewis/Loftus
//
//  Represents a general paid employee.
//********************************************************************

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

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

      socialSecurityNumber = socSecNumber;
      payRate = rate;
   }

   //-----------------------------------------------------------------
   //  Returns information about an employee as a string.
   //-----------------------------------------------------------------
   public String toString()
   {
      String result = super.toString();

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

      return result;
   }

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

```
//********************************************************************
//  Executive.java       Author: Lewis/Loftus
//
//  Represents an executive staff member, who can earn a bonus.
//********************************************************************

public class Executive extends Employee
{
   private double bonus;

   //-----------------------------------------------------------------
   //  Sets up an executive with the specified information.
   //-----------------------------------------------------------------
   public Executive (String eName, String eAddress, String ePhone,
                     String socSecNumber, double rate)
   {
      super (eName, eAddress, ePhone, socSecNumber, rate);

      bonus = 0;  // bonus has yet to be awarded
   }

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

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

      bonus = 0;

      return payment;
   }
}
```

```java
//********************************************************************
//  Hourly.java       Author: Lewis/Loftus
//
//  Represents an employee that gets paid by the hour.
//********************************************************************

public class Hourly extends Employee
{
   private int hoursWorked;

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

      hoursWorked = 0;
   }

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

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

      hoursWorked = 0;

      return payment;
   }

   //-----------------------------------------------------------------
   //  Returns information about this hourly employee as a string.
   //-----------------------------------------------------------------
   public String toString()
   {
      String result = super.toString();

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

      return result;
   }
}
```

JAWABAN

a. Source Code Setelah di Modifikasi

Firm.java
```
package com.mycompany.kasus1;

/**
 * @author Yumi Febriana
 */
public class Firm
{
   public static void main(String[] args) {
      Staff personnel = new Staff();
      personnel.payday();
   }
}
```

StaffMember.java
```
package com.mycompany.kasus1;

/**
 * @author Yumi Febriana
 */
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;
   }

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

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

      return result;
   }
   public abstract double pay();
}
```

Volunteer.java

```
package com.mycompany.kasus1;

/**
 * @author Yumi Febriana
 */
public class Volunteer extends StaffMember
{
   public Volunteer (String eName, String eAddress, String ePhone)
   {
      super (eName,eAddress,ePhone);
   }
   public double pay()
   {
      return 0.0;
   }
}
```

Employee.java

```
package com.mycompany.kasus1;

/**
 * @author Yumi Febriana
 */
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;
   }
}
```

Executive.java

```
package com.mycompany.kasus1;

/**
 *
 * @author Yumi Febriana
 */
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

```
package com.mycompany.kasus1;

/**
 * @author Yumi Febriana
 */
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;
   }
}
```

Commission.java

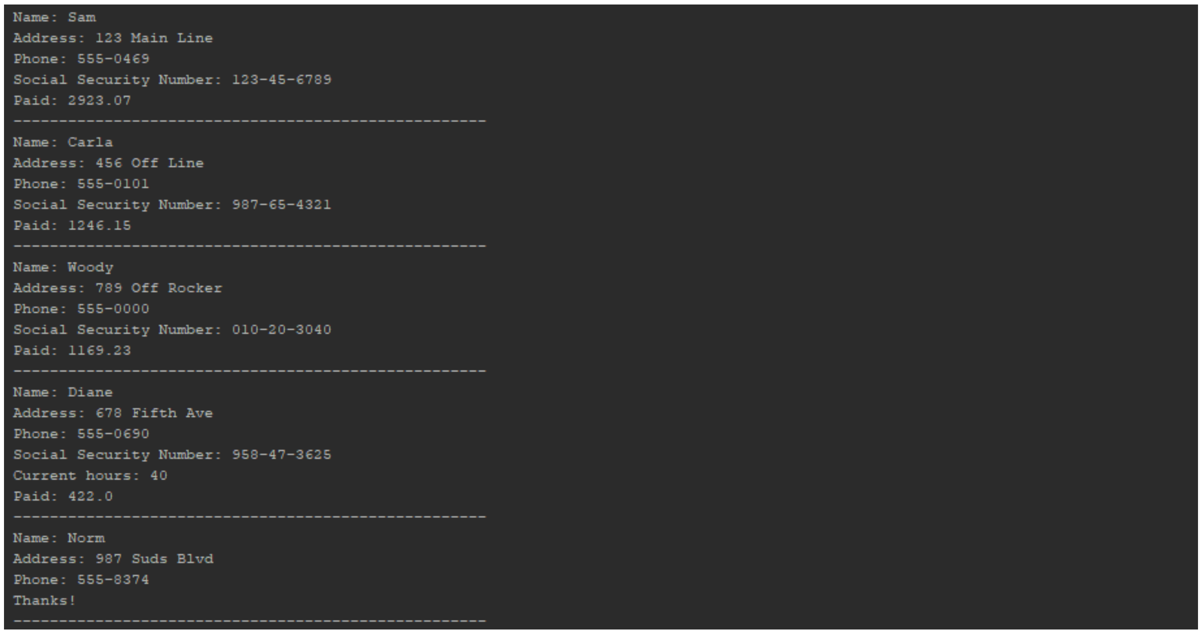
```
package com.mycompany.kasus1;

/**
 * @author Yumi Febriana
 */
public class Commission extends Hourly {
   private double totalSales;
   private double commissionRate;

   public Commission (String eName,String eAddress, String ePhone,
                String socSecNumber,double rate,double commisRate)
   {
      super(eName, eAddress, ePhone, socSecNumber, rate);
      commissionRate = commisRate;
   }

   public void addSales(double totalSales)
   {
      this.totalSales = totalSales*commissionRate;

   }

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

    public String toString()
    {
        String result = super.toString();
        result += "\nTotal Sales: " + totalSales;

        return result;
    }
}
```

Staff.java

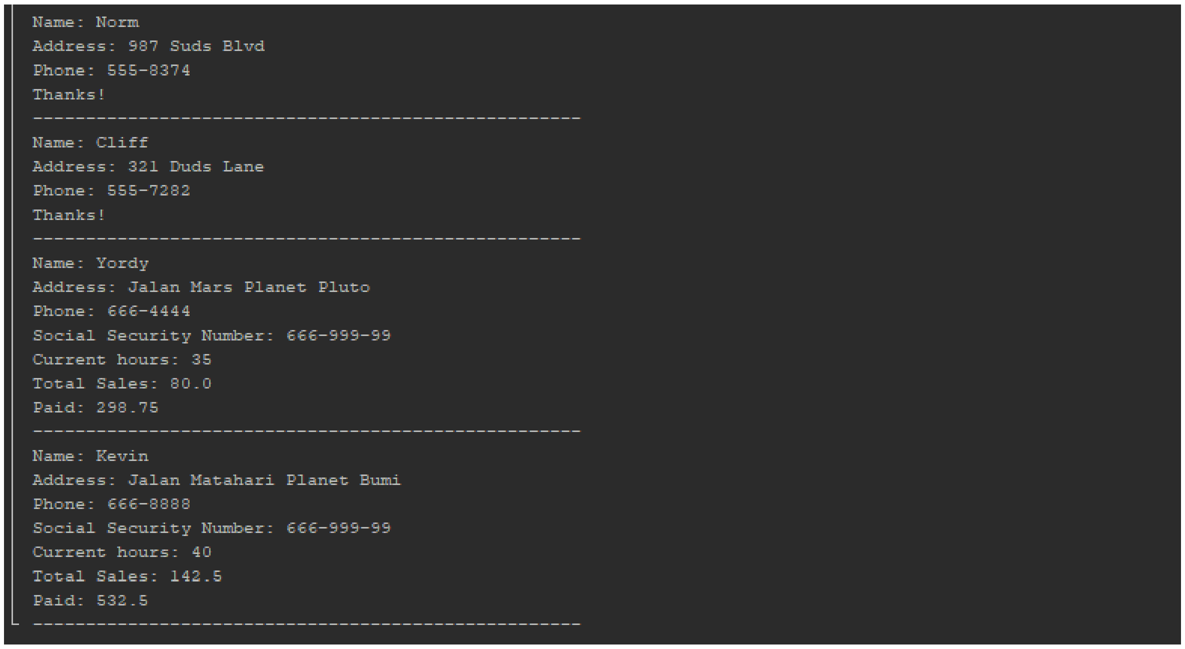
```
package com.mycompany.kasus1;

/**
 * @author Yumi Febriana
 */
public class Staff {
    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("Yordy", "Jalan Mars Planet Pluto",
        "666-4444", "666-999-99", 6.25, 0.2);
        staffList[7] = new Commission("Kevin", "Jalan Matahari Planet Bumi",
        "666-8888", "666-999-99", 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(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("---------------------------------------------------");
      }
   }
}
```

b. Result



```
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: Norm
Address: 987 Suds Blvd
Phone: 555-8374
Thanks!
---------------------------------------------------
Name: Cliff
Address: 321 Duds Lane
Phone: 555-7282
Thanks!
---------------------------------------------------
Name: Yordy
Address: Jalan Mars Planet Pluto
Phone: 666-4444
Social Security Number: 666-999-99
Current hours: 35
Total Sales: 80.0
Paid: 298.75
---------------------------------------------------
Name: Kevin
Address: Jalan Matahari Planet Bumi
Phone: 666-8888
Social Security Number: 666-999-99
Current hours: 40
Total Sales: 142.5
Paid: 532.5
---------------------------------------------------
```

TASK 2

Painting Shapes

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

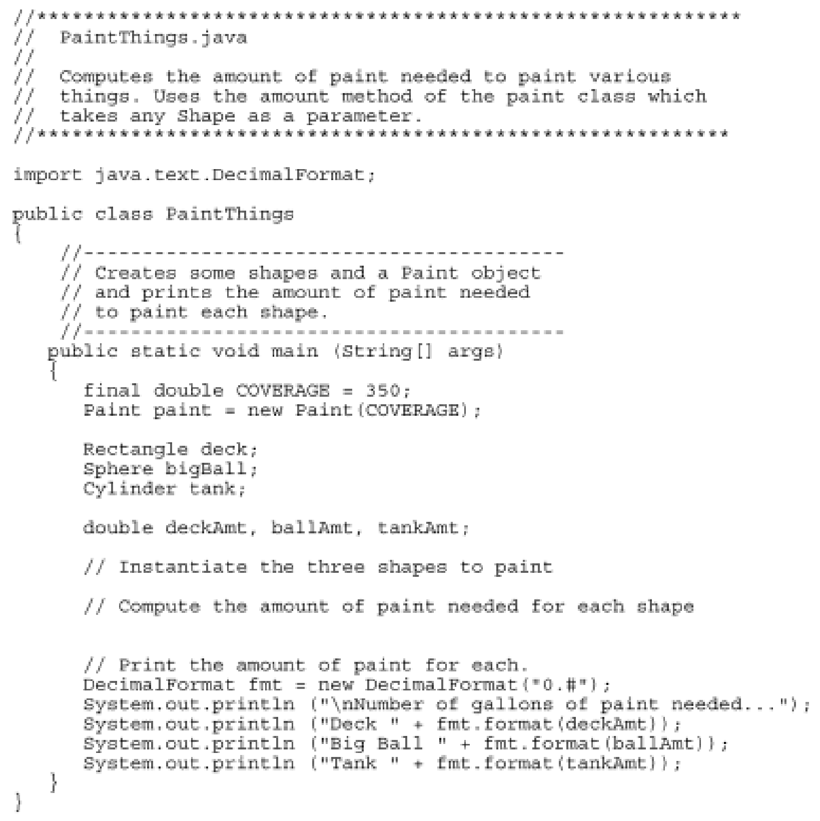
2. The file *Sphere.java* contains a class for a sphere which is a descendant of Shape. A sphere has a radius and its area (surface area) is given by the formula 4*PI*radius^2. Define similar classes for a rectangle and a cylinder. Both the Rectangle class and the Cylinder class are descendants of the 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.

SOURCE CODE SOAL

```java
//*******************************************
//  Sphere.java
//
//  Represents a sphere.
//*******************************************
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;
   }



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

}


//******************************************************
//   Paint.java
//
//   Represents a type of paint that has a fixed area
//   covered by a gallon. All measurements are in feet.
//******************************************************

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
//
//  Computes the amount of paint needed to paint various
//  things. Uses the amount method of the paint class which
//  takes any Shape as a parameter.
//**********************************************************

import java.text.DecimalFormat;

public class PaintThings
{
    //-----------------------------------------
    // Creates some shapes and a Paint object
    // and prints the amount of paint needed
    // to paint each shape.
    //-----------------------------------------
   public static void main (String[] args)
   {
      final double COVERAGE = 350;
      Paint paint = new Paint(COVERAGE);

      Rectangle deck;
      Sphere bigBall;
      Cylinder tank;

      double deckAmt, ballAmt, tankAmt;

      // Instantiate the three shapes to paint

      // Compute the amount of paint needed for each shape


      // 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));
   }
}
```

JAWABAN

a. Source Code hasil modifikasi

PaintThings.java

```
package com.mycompany.paintthings;

import java.text.DecimalFormat;

/**
 * @author Yumi Febriana
 */
public class PaintThings {

    public static void main(String[] args) {
        final double COVERAGE = 350;
        Paint paint = new Paint(COVERAGE);
        Rectangle deck = new Rectangle (20,35);
        Sphere bigBall = new Sphere(15);
        Cylinder tank = new Cylinder (10,30);

        double deckAmt, ballAmt, tankAmt;
        deckAmt = paint.amount(deck);
        ballAmt = paint.amount(bigBall);
        tankAmt = paint.amount(tank);

        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));
    }
}
```

Shape.java

```
package com.mycompany.paintthings;

/**
 * @author Yumi Febriana
 */
abstract public class Shape {
    private String shapeName;
    public Shape (String shapeName){
        this.shapeName = shapeName;
    }
    public abstract double area();
    public String toString(){
        String result = "Shape Namae : " + this.shapeName;
        return result;
    }
}
```

Sphere.java

```
package com.mycompany.paintthings;

/**
 * @author Yumi Febriana
 */
public class Sphere extends Shape{
    private double radius;
    public Sphere (double r){
        super("Sphere");
        radius = r;
    }

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

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

Rectangle.java

```
package com.mycompany.paintthings;
```

```
/**
 * @author Yumi Febriana
 */
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;
    }

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

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

}
```

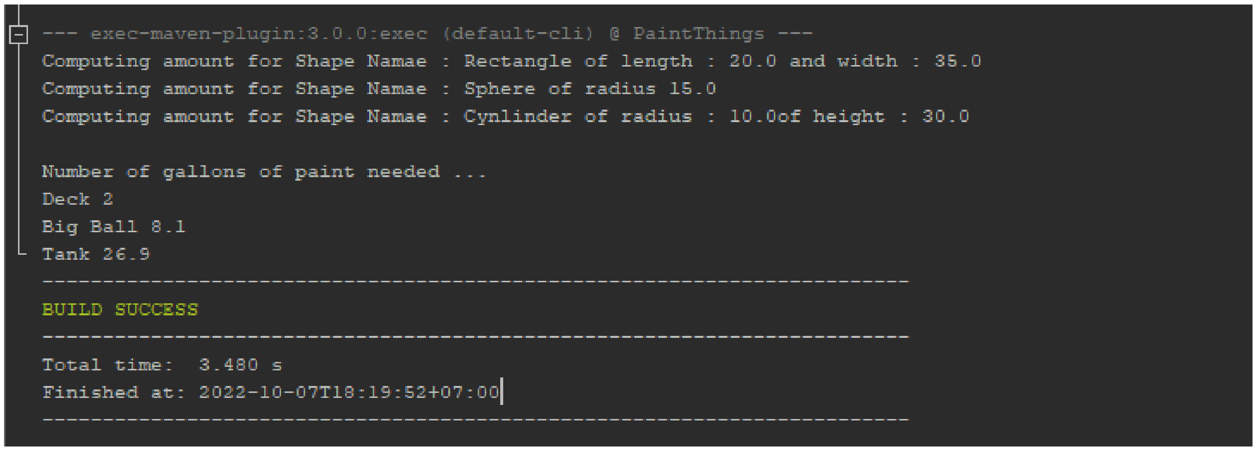
Cylinder.java

```
package com.mycompany.paintthings;

/**
 * @author Yumi Febriana
 */
public class Cylinder extends Shape{
    private double radius;
    private double height;

    public Cylinder(double radius , double height){
        super("Cynlinder");
        this.radius = radius;
        this.height = height;
    }

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

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

Paint.java

```
package com.mycompany.paintthings;

/**
 * @author Yumi Febriana
 */
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;
   }
}
```

b. Result



```
--- exec-maven-plugin:3.0.0:exec (default-cli) @ PaintThings ---
Computing amount for Shape Namae : Rectangle of length : 20.0 and width : 35.0
Computing amount for Shape Namae : Sphere of radius 15.0
Computing amount for Shape Namae : Cynlinder of radius : 10.0of height : 30.0

Number of gallons of paint needed ...
Deck 2
Big Ball 8.1
Tank 26.9
------------------------------------------------------------------------
BUILD SUCCESS
------------------------------------------------------------------------
Total time:  3.480 s
Finished at: 2022-10-07T18:19:52+07:00
------------------------------------------------------------------------
```

Task 3

Polymorphic Sorting

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 the Sorting 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.

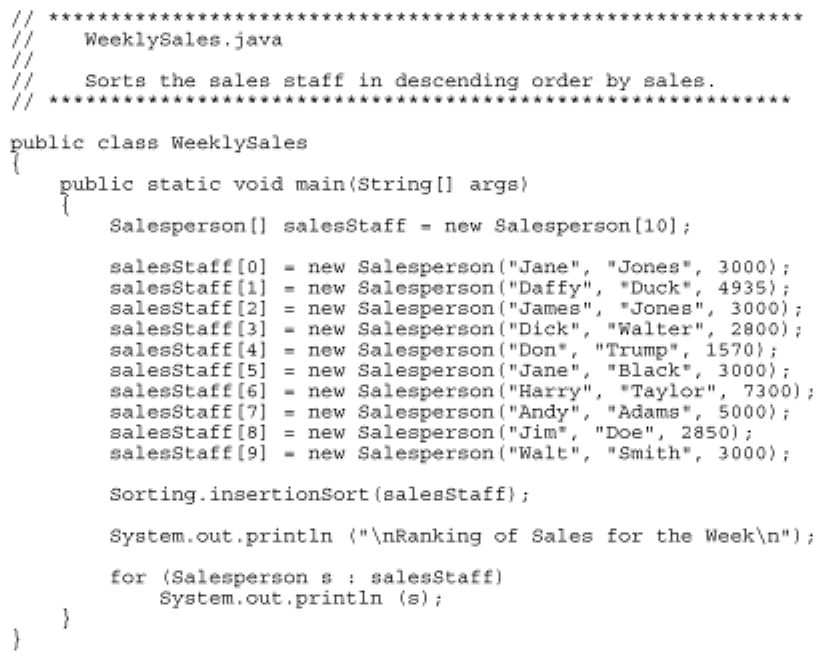
SOURCE CODE SOAL

```
//********************************************************************
//  Sorting.java       Author: Lewis/Loftus
//
//  Demonstrates the selection sort and insertion sort algorithms.
//********************************************************************

public class Sorting
{
   //-----------------------------------------------------------------
   //  Sorts the specified array of objects using the selection
   //  sort algorithm.
   //-----------------------------------------------------------------
   public static void selectionSort (Comparable[] list)
   {
      int min;
      Comparable temp;

      for (int index = 0; index < list.length-1; index++)
      {
         min = index;
         for (int scan = index+1; scan < list.length; scan++)
            if (list[scan].compareTo(list[min]) < 0)
```

```
               min = scan;

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

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

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

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

```
// *******************************************************
//   Numbers.java
//
//   Demonstrates selectionSort on an array of integers.
// *******************************************************

import java.util.Scanner;

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 ();
    }
}
```

```java
// *******************************************************
//   Salesperson.java
//
//   Represents a sales person who has a first name, last
//   name, and total number of sales.
// *******************************************************

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()));
    }

    //--------------------------------------------------
    //  Order is based on total sales with the name
    //  (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
//
//    Sorts the sales staff in descending order by sales.
// **********************************************************

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);
    }
}
```

JAWABAN

a. Source Code Hasil Modifikasi

Sorting.java

```
package com.mycompany.kasus3;

/**
 * @author Yumi Febriana
 */
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 && key.compareTo(list[position-1]) > 0){
                list[position] = list[position-1];
            position--;
         }
         list[position] = key;
      }
   }
}
```

SalesPerson.java

```
package com.mycompany.kasus3;

/**
 * @author Yumi Febriana
 */
public class SalesPerson implements Comparable{
   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 + " , " + firstName + ": \t" + totalSales;
   }

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

   public int compareTo(Object other) {
            // TODO Auto-generated method stub
            int result = 0;
            if (totalSales > ((Salesperson)other).totalSales) result = 1;
            else if(totalSales < ((Salesperson)other).totalSales) result = -1;
            else {
                    if (lastName.compareTo(((Salesperson) other).getLastName()) < 0)
            return 1;
            else
            return -1;
                    }
            return result;
   }

   public String getFirstName(){
      return firstName;
   }

   public String getLastName(){
      return lastName;
```

```
    }

    public int getSales(){
        return totalSales;
    }
}
```

Numbers.java

```
package com.mycompany.kasus3;

import java.util.Scanner;

/**
 * @author Yumi Febriana
 */
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.insertionSort(intList);
        System.out.println("\nYour numbers in  sorted order ...");
        for (int i = 0 ; i < size ; i++)
            System.out.println(intList[i] + " ");
        System.out.println();

    }
}
```

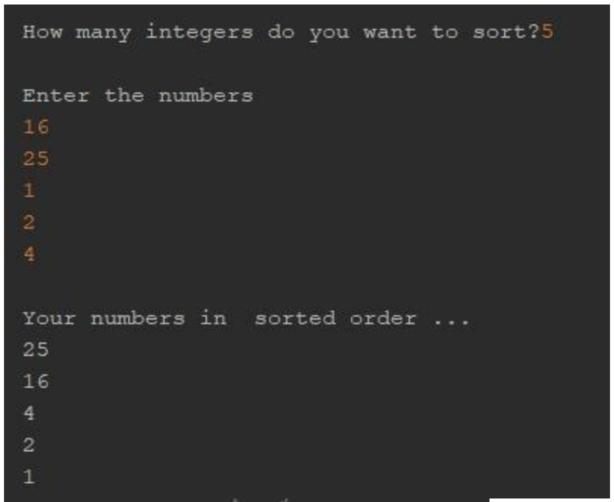
Strings.java

```
package com.mycompany.kasus3;

import java.util.Scanner;

/**
 * @author Yumi Febriana
 */
public class Strings {
    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.insertionSort(stringList);
        System.out.println("\nYour numbers in sorted order ...");
        for (int i = 0 ; i < size ; i++)
            System.out.println(stringList[i] + " ");
        System.out.println ();
        scan.close();


    }
}
```

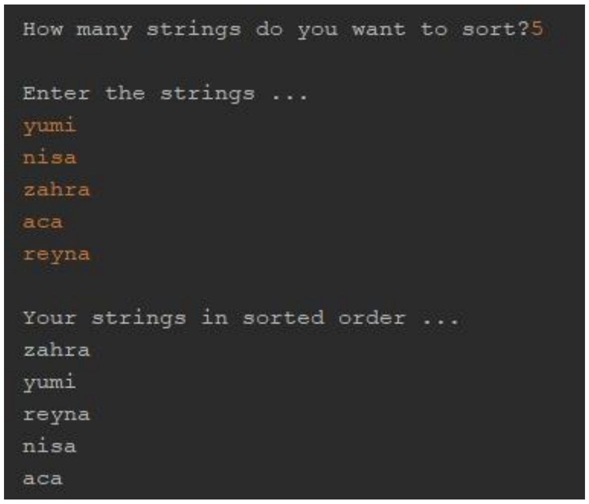
WeeklySales.java

```
package com.mycompany.kasus3;
import java.util.Scanner;

/**
 * @author Yumi Febriana
 */
public class WeeklySales {

    public static void main(String[] args) {
        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++){
            System.out.print("\nFirst Name [" + i + "] : ");
            String firstName = scan.next();
            System.out.print("\nLast Name [" + i + "] : ");
            String lastName = scan.next();
            System.out.print("\nTotal Sales [" + i + "] : ");
            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();
    }
}
```

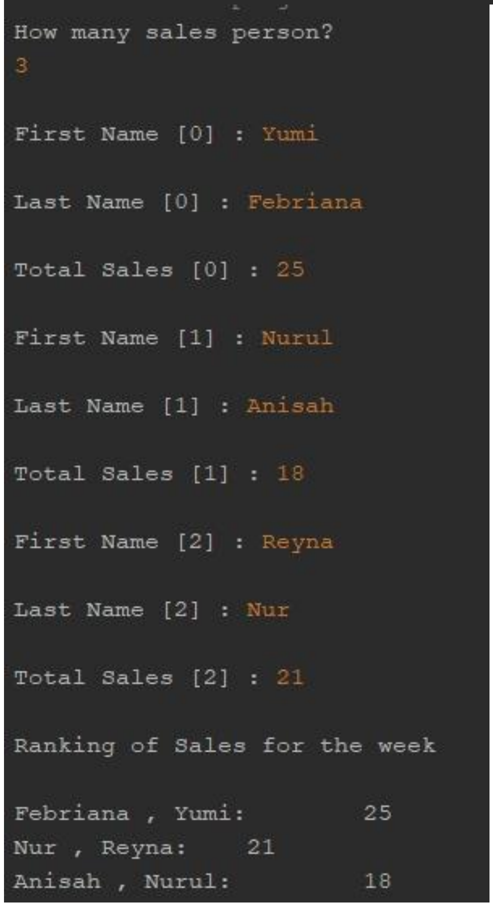
b. Result



```
How many integers do you want to sort?5

Enter the numbers
16
25
1
2
4

Your numbers in  sorted order ...
25
16
4
2
1
```



```
How many strings do you want to sort?5

Enter the strings ...
yumi
nisa
zahra
aca
reyna

Your strings in sorted order ...
zahra
yumi
reyna
nisa
aca
```



```
How many sales person?
3

First Name [0] : Yumi

Last Name [0] : Febriana

Total Sales [0] : 25

First Name [1] : Nurul

Last Name [1] : Anisah

Total Sales [1] : 18

First Name [2] : Reyna

Last Name [2] : Nur

Total Sales [2] : 21

Ranking of Sales for the week

Febriana , Yumi:        25
Nur , Reyna:    21
Anisah , Nurul:         18
```

c. Problem
   Pada bagian sorting ranking of slaes for the week saya belum bisa membenarkan agar sorting sesuai dengan ranking
d. Solution
   Membenarkan program pada sales person dengan memperbaiki algoritma untuk sortir
e. Teman yang membantu
   -