**LAPORAN PRAKTIKUM**

**PERTEMUAN-8**

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



**Disusun Oleh:**

**Daiva Raditya Pradipa (231511039)**

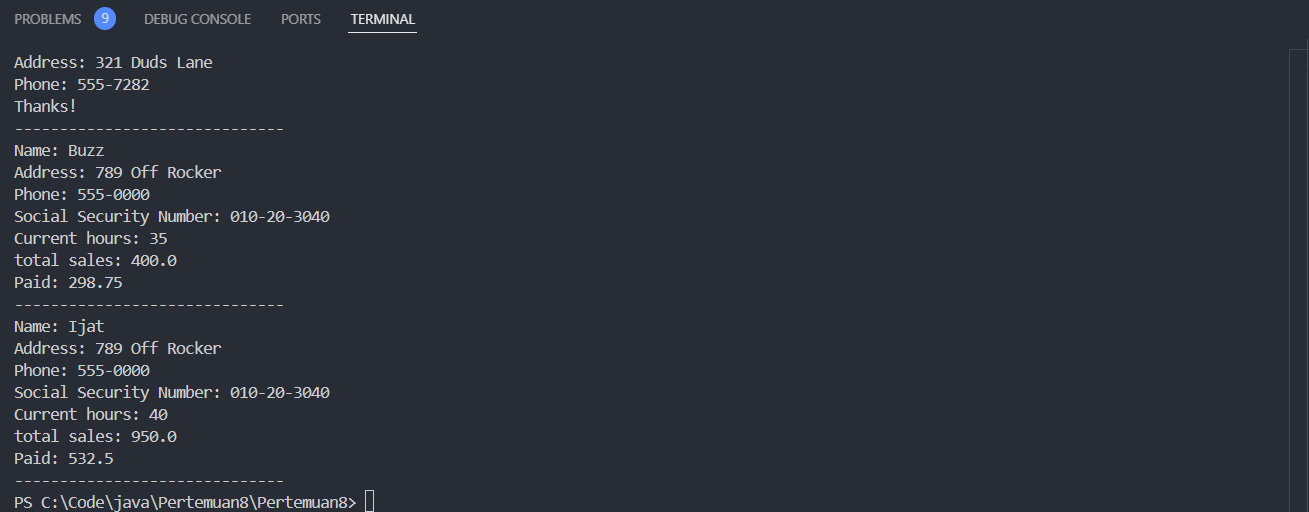
**Jurusan Teknik Komputer dan Informatika**

**Program Studi D-3 Teknik Informatika**

**Politeknik Negeri Bandung**

**2024**

1. **Studi kasus 1**
2. Screenshot Hasil Program



1. Penjelasan

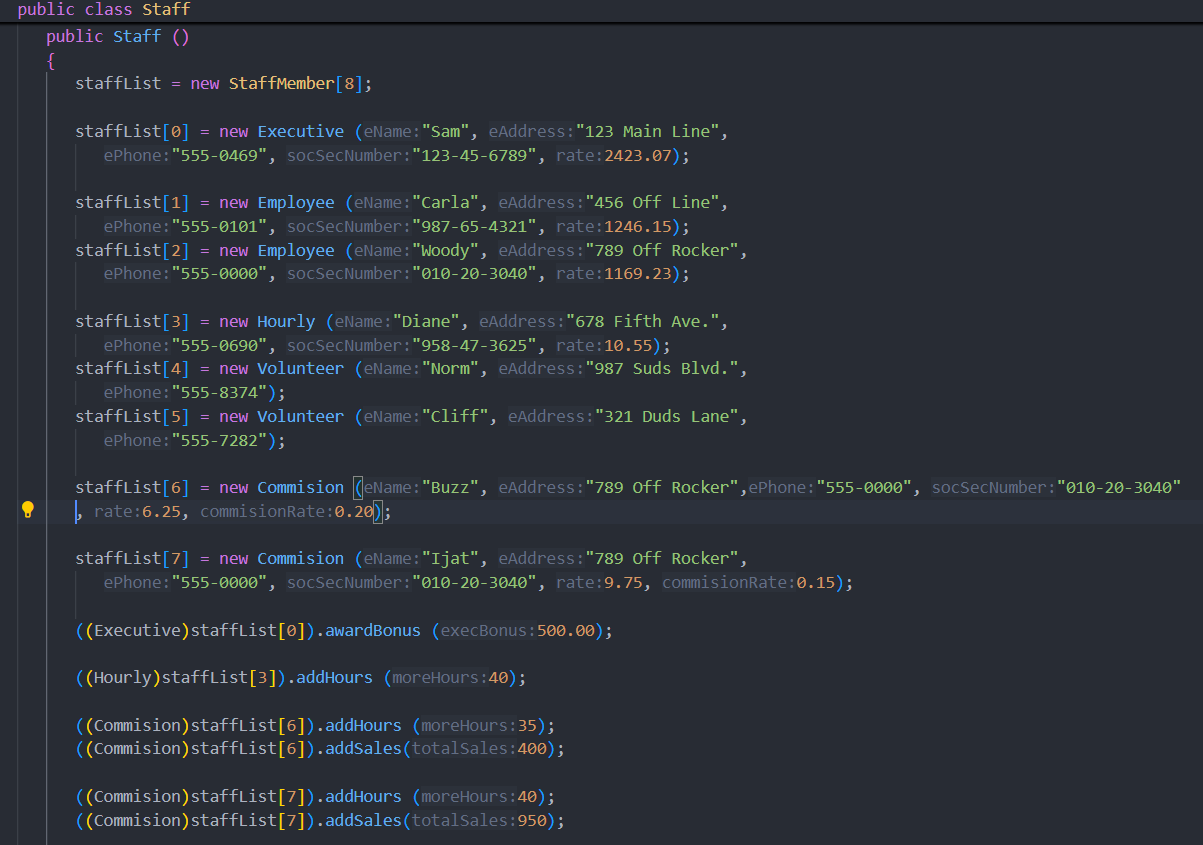
Polymorphism yand digunakan adalah polymorphism inheritance dengan superclass yang menunjuk pada subclass. Hal ini dapat dibuktikan dari file commission.java berikut



Dari code ini terdapat beberapa ciri ciri yang menunjukan 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.

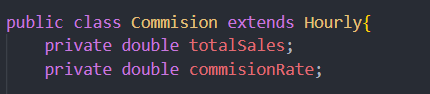
Kemudian pada file Staff.java yang dimana pada studi kasus ini merupakan file untuk menginstansikan object object dari class yang ada seperti Hourly, Comission, dll.



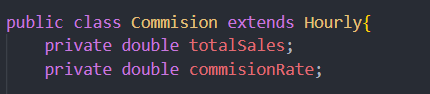
Dari file ini, kita bisa lihat terdapat syntax yang menunjukan bahwa polymorphism yang diterapkan adalah polymorphism dengan tipe superclass yang menunjuk ke subclass. Hal ini dibuktikan dengan penggunaan tipe data superclass untuk menunjuk ke objek-objek dari subclass. Hal ini terlihat pada deklarasi 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.

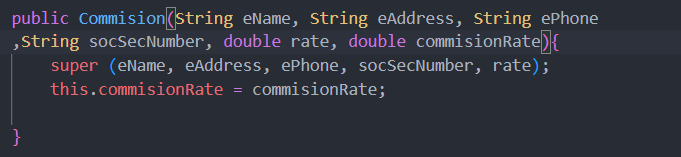
1. Jawaban Soal
2. Write a class named Commission with the following features
   1. It extends the Hourly class.



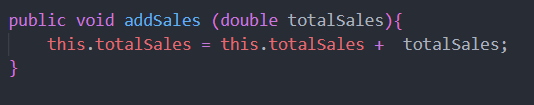
* 1. It has two instance variables (in addition to those inherited): one is the total sales the employee has made (type double) and the second is the commission rate for the employee (the commission rate will be type double and will represent the percent (in decimal form) commission the employee earns on sales (so .2 would mean the employee earns 20% commission on sales)).



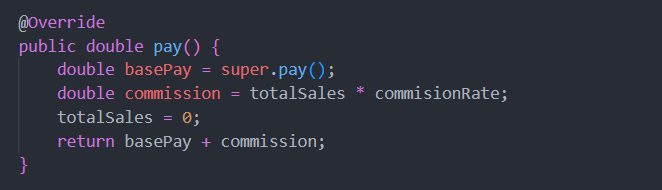
* 1. The constructor takes 6 parameters: the first 5 are the same as for Hourly (name, address, phone number, social security number, hourly pay rate) and the 6th is the commission rate for the employee. The constructor should call the constructor of the parent class with the first 5 parameters then use the 6th to set the commission rate.



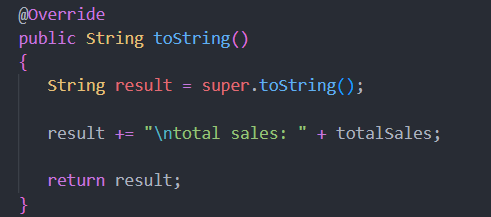
* 1. One additional method is needed: public void addSales (double totalSales) that adds the parameter to the instance variable representing total sales.



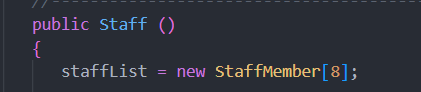
* 1. The pay method must call the pay method of the parent class to compute the pay for hours worked then add to that the pay from commission on sales. (See the pay method in the Executive class.) The total sales should be set back to 0 (note: you don't need to set the hoursWorked back to 0—why not?)



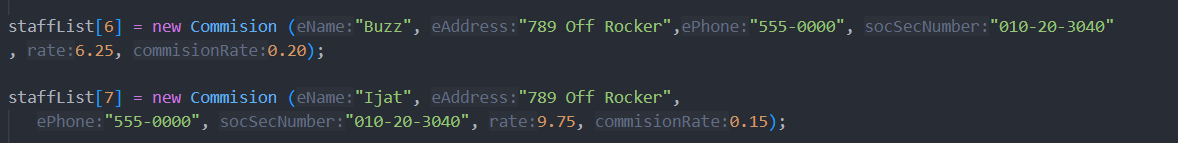
* 1. The toString method needs to call the toString method of the parent class then add the total sales to that.



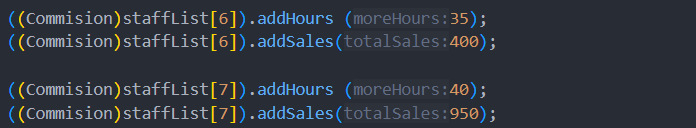
1. To test your class, update Staff.java as follows:
   1. Increase the size of the array to 8



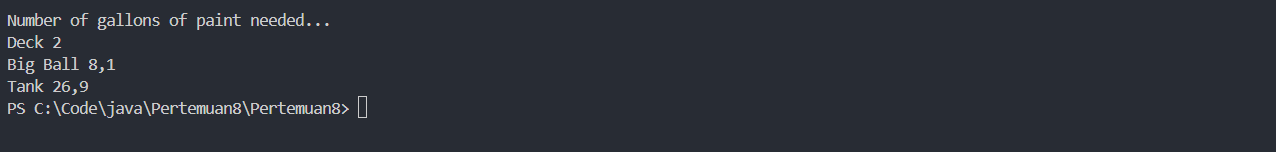
* 1. Add two commissioned employees to the staffList—make up your own names, addresses, phone numbers and social security numbers. Have one of the employees earn $6.25 per hour and 20% commission and the other one earn $9.75 per hour and 15% commission



* 1. 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.

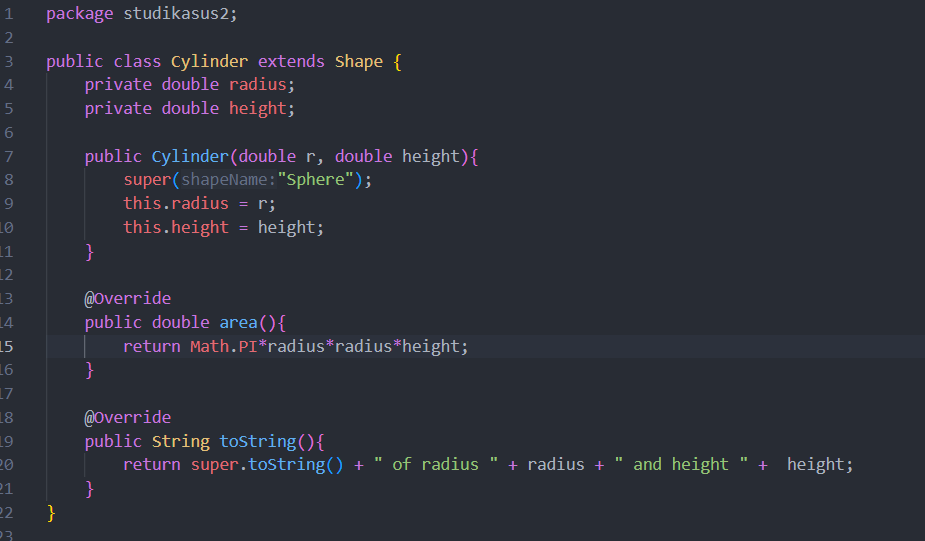


1. Studi kasus 2
2. Screenshot hasil program



1. Penjelasan

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



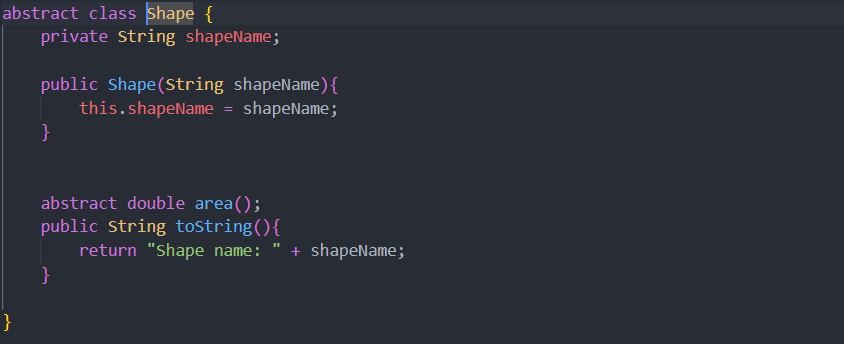
Dari code ini terdapat beberapa ciri ciri yang menunjukan bahwa class commission menerapkan polymorphism inheritance, diantaranya adalah

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” .

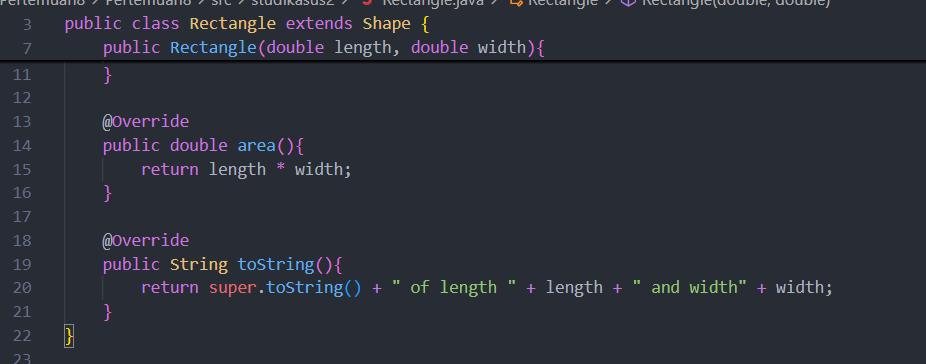
Kemudian pada file PaintThings.java yang dimana pada studi kasus ini merupakan main file untuk menjalankan program, sekaligus untuk menginstansikan object dari class seperti Sphere, Cylinder, dan Rectangle.

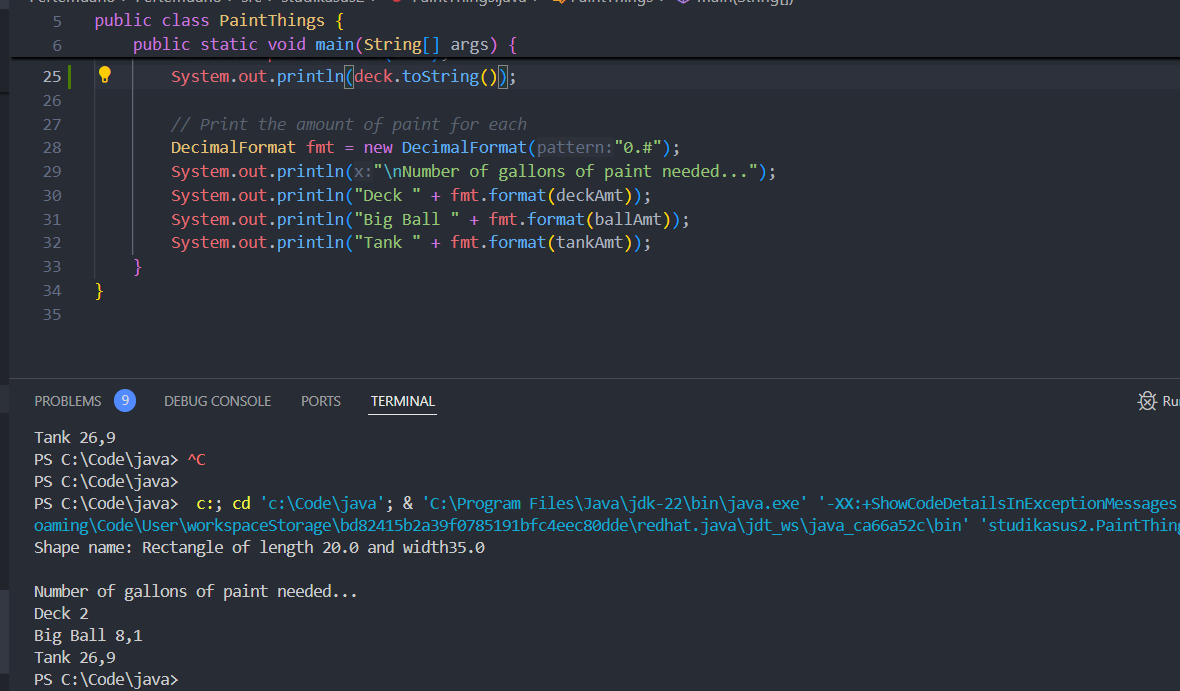


Dari code ini bis kita lihat bahwa polymorphism yang diterapkan adalah polymorphism dengan tipe superclass yang menunjuk ke subclass (sama dengan studi kasus pertama). Dimana Shape yang merupakan superclass dijadikan tipe data dari variabel yang akan menampung objek dari subclass Shape yaitu Rectangle,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



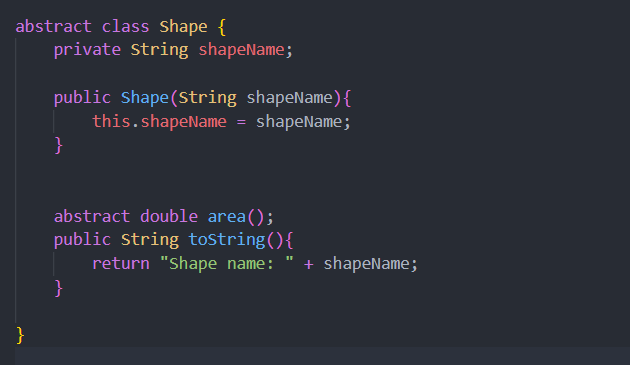
Namun saat salah satu object tersebut dipanggil, dia akan tetap memanggil method dari class yang diinstansikan tersebut, sebagai contoh disini saya akan coba memanggil method to string dari object deck dengan method toString() sebagai berikut.



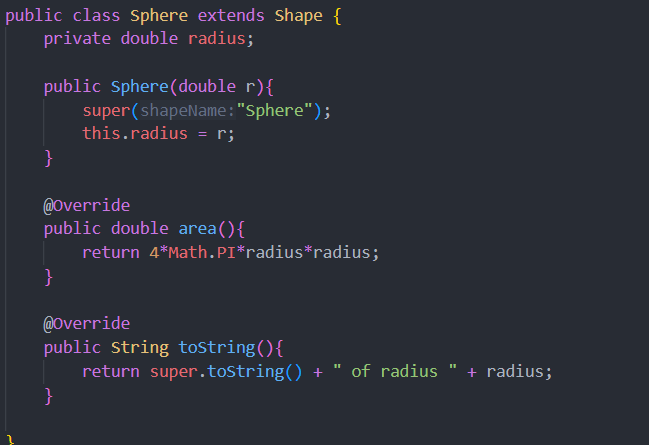


Bisa dilihat pada terminal output, saat menjalankan method toString dari object deck yang diinstansikan dari class Rectangle, dia memanggil method toString() dari class Rectangle dan bukan dari class Shape.

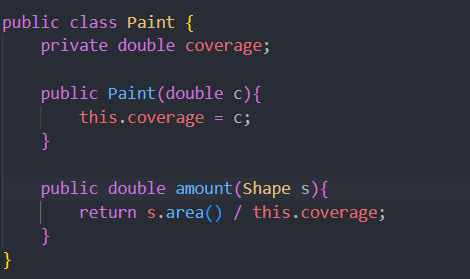
1. Jawaban Soal
2. Write an abstract class Shape with the following properties: An instance variable shapeName of type String An abstract method area() A toString method that returns the name of the shape



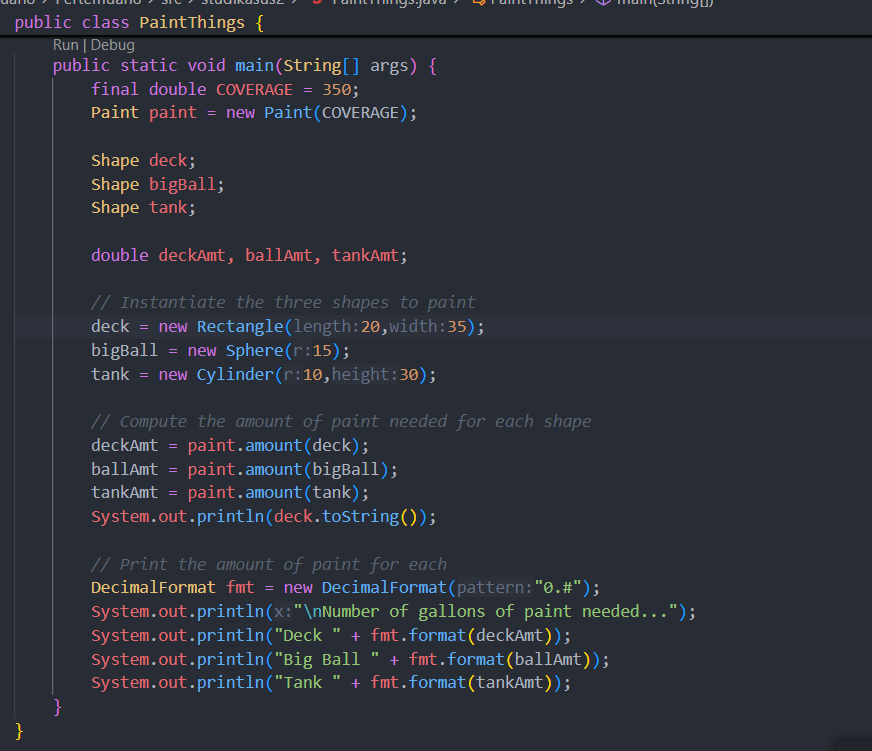
1. The file Sphere.java contains a class for a sphere which is a descendant of Shape. A sphere has a radius and its area (surface area) is given by the formula 4\*PI\*radius^2. Define similar classes for a rectangle and a cylinder. Both the Rectangle class and the Cylinder class are descendants of the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



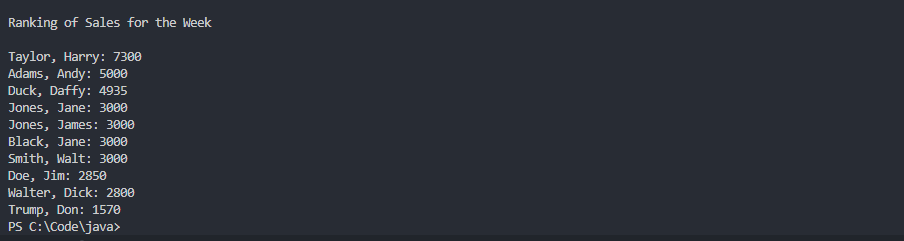
1. The file Paint.java contains a class for a type of paint (which has a "coverage" and a method to compute the amount of paint needed to paint a shape). Correct the return statement in the amount method so the correct amount will be returned. Use the fact that the amount of paint needed is the area 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.)



1. The file PaintThings.java contains a program that computes the amount of paint needed to paint various shapes. A paint object has been instantiated. Add the following to complete the program: Instantiate the three shape objects: deck to be a 20 by 35 foot rectangle, bigBall to be a sphere of radius 15, and tank to be a cylinder of radius 10 and height 30. Make the appropriate method calls to assign the correct values to the three amount variables. Run the program and test it. You should see polymorphism in action as the amount method computes the amount of paint for various shapes.

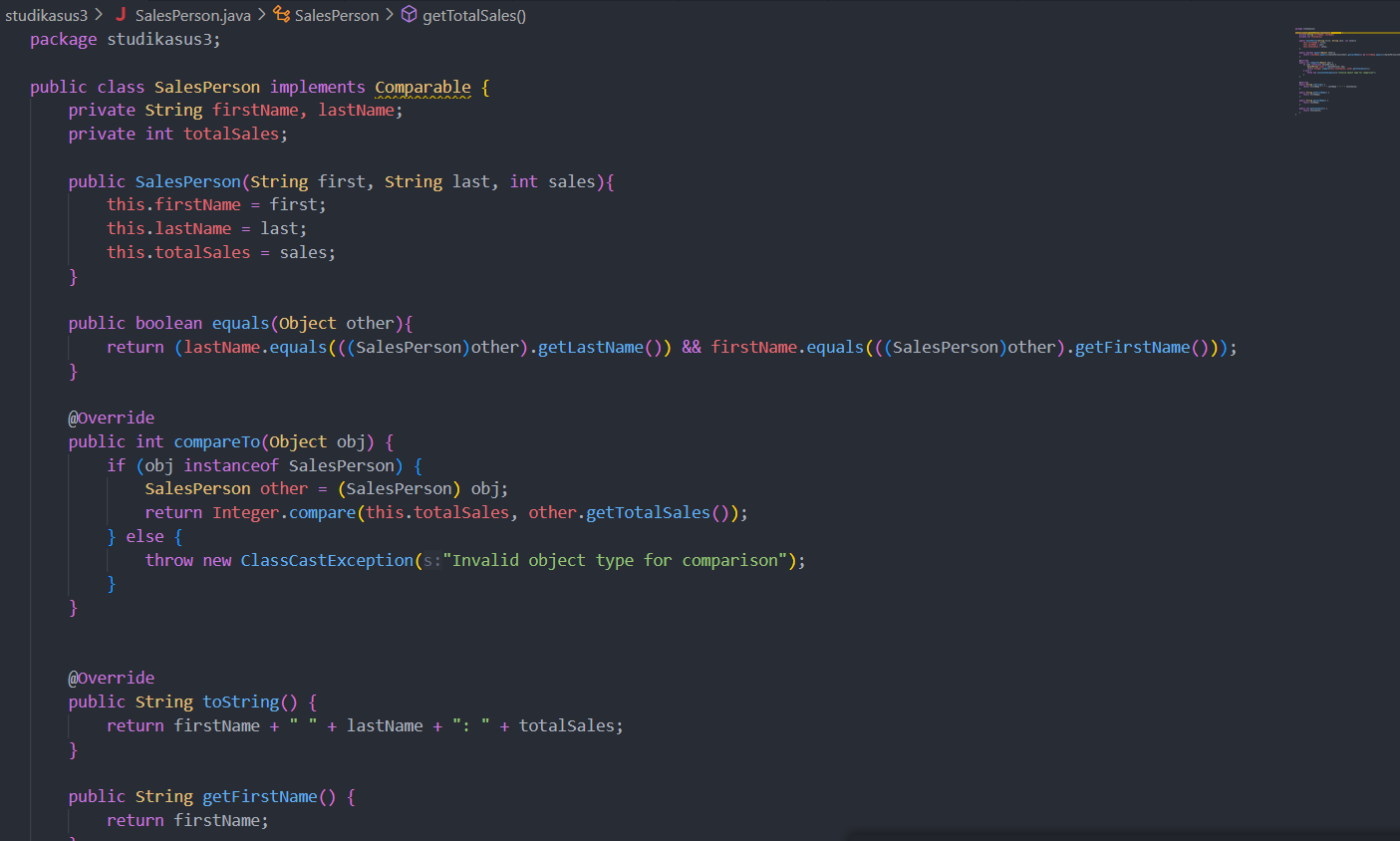


1. Studi kasus 3
2. Screenshot hasil program



1. Penjelasan

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

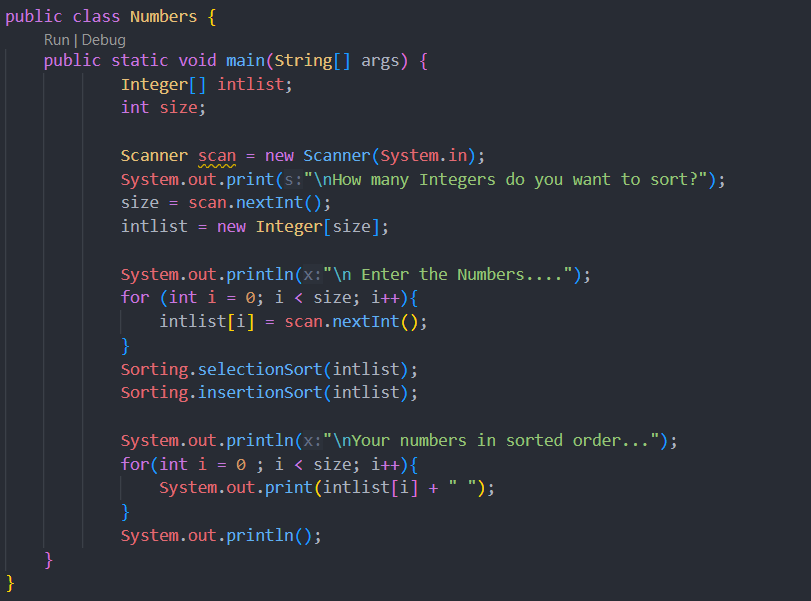


Dari code ini terdapat beberapa ciri ciri yang menunjukan bahwa class commission menerapkan polymorphism inheritance, diantaranya adalah

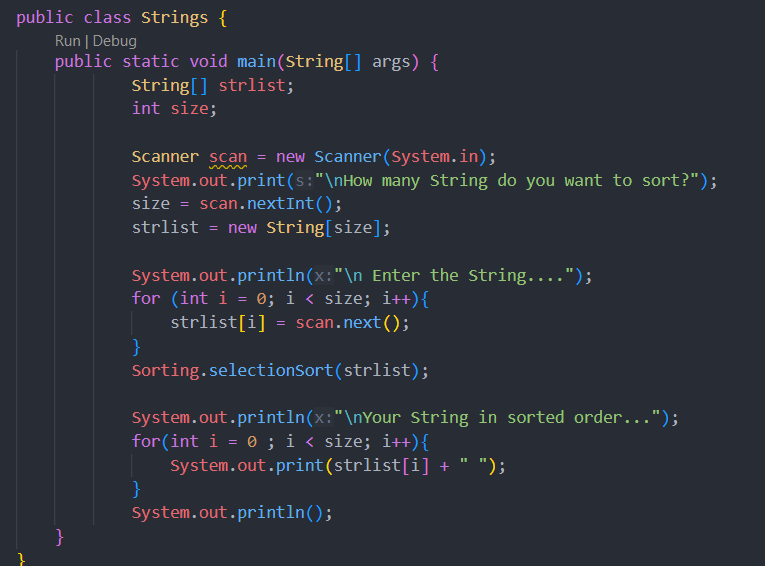
1. Class tersebut mengimplemen interface Comparable yang dibuktikan dengan syntax “implements Comparable”
2. Class tersebut mengoveride method toString() dan compareTo() yang merupakan method dari interface Comparable.
3. Jawaban Soal
4. The file Numbers.java reads in an array of integers, invokes the 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.

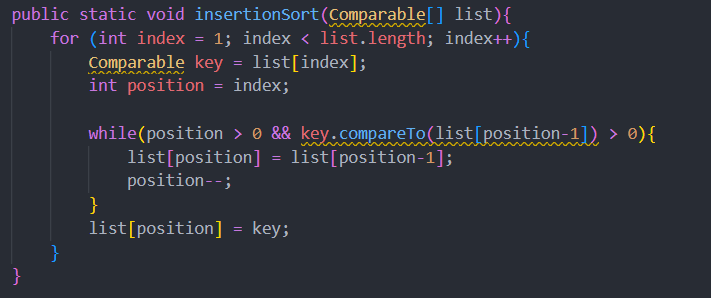
1. Try to compile Numbers.java and see what the error message is. The problem involves the difference between primitive data and objects. Change the program so it will work correctly (note: you don't need to make many changes - the autoboxing feature of Java 1.5 will take care of most conversions from int to Integer).

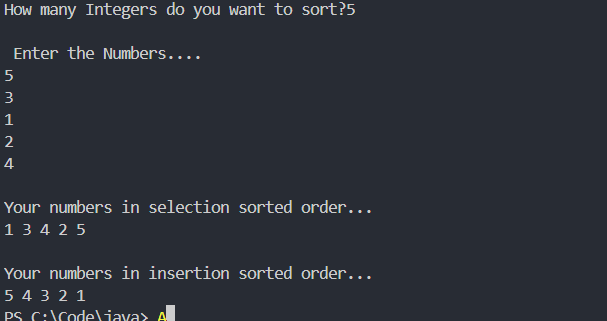


1. Write a program Strings.java, similar to Numbers.java, that reads in an array of String objects and sorts them. You may just copy and edit Numbers.java.

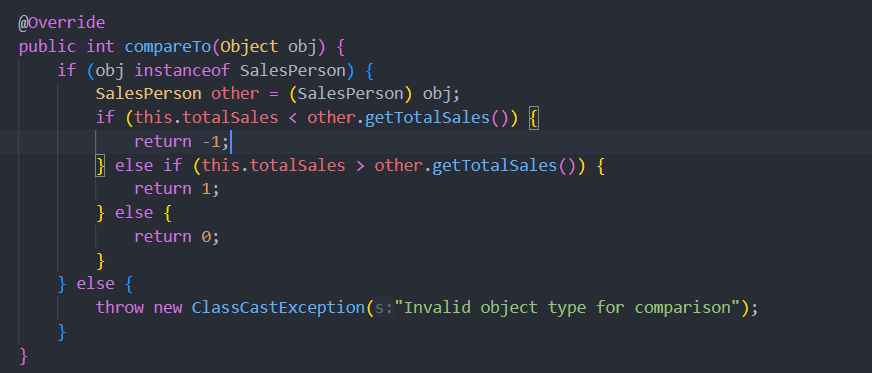


1. Modify the insertionSort algorithm so that it sorts in descending order rather than ascending order. Change Numbers.java and Strings.java to call insertionSort rather than selectionSort. Run both to make sure the sorting is correct.

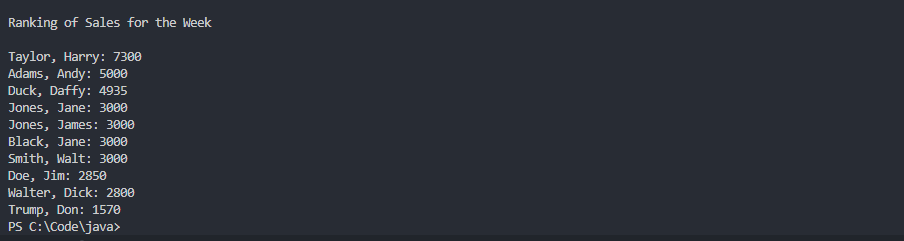




1. The file Salesperson.java partially defines a class that represents a sales person. This is very similar to the Contact class in Listing 9.10. However, a sales person has a first name, last name, and a total number of sales (an int) rather than a first name, last name, and phone number. Complete the 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).



1. 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.



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