Polymorphic
Pertemuan ke 9

Diajukan untuk memenuhi salat satu tugas praktikum

Mata kuliah PBO

Disusun Oleh:
Alqan Nazra (231511068)

Jurusan Teknik Komputer dan Informatika

Program Studi D-3 Teknik Informatika
Politeknik Negeri Bandung
2024

Kerjakan 2 soal dibawah ini dengan mengikuti ketentuan sebagai berikut:

1. Isi sheet monitoring berdasarkan ketentuan yang ada di sheet tersebut.

2. Source code setiap pengerjaan soal, simpan di Github, lampirkan komentar dari hasil pengerjaan tersebut.

3. Buat laporan hasil pengerjaan berbentuk dokumen, upload laporan di folder Hasil Praktikum di folder hasil praktikum, laporan harus mencakup:

1. Cover.

2. Persoalan yang telah dikerjakan.

Setiap persoalan, harus menjawab beberapa deskripsi berikut ini:

1. Screenshoot hasil akhir program.

2. Screenshoot setiap jawaban soal yang dipertanyakan.

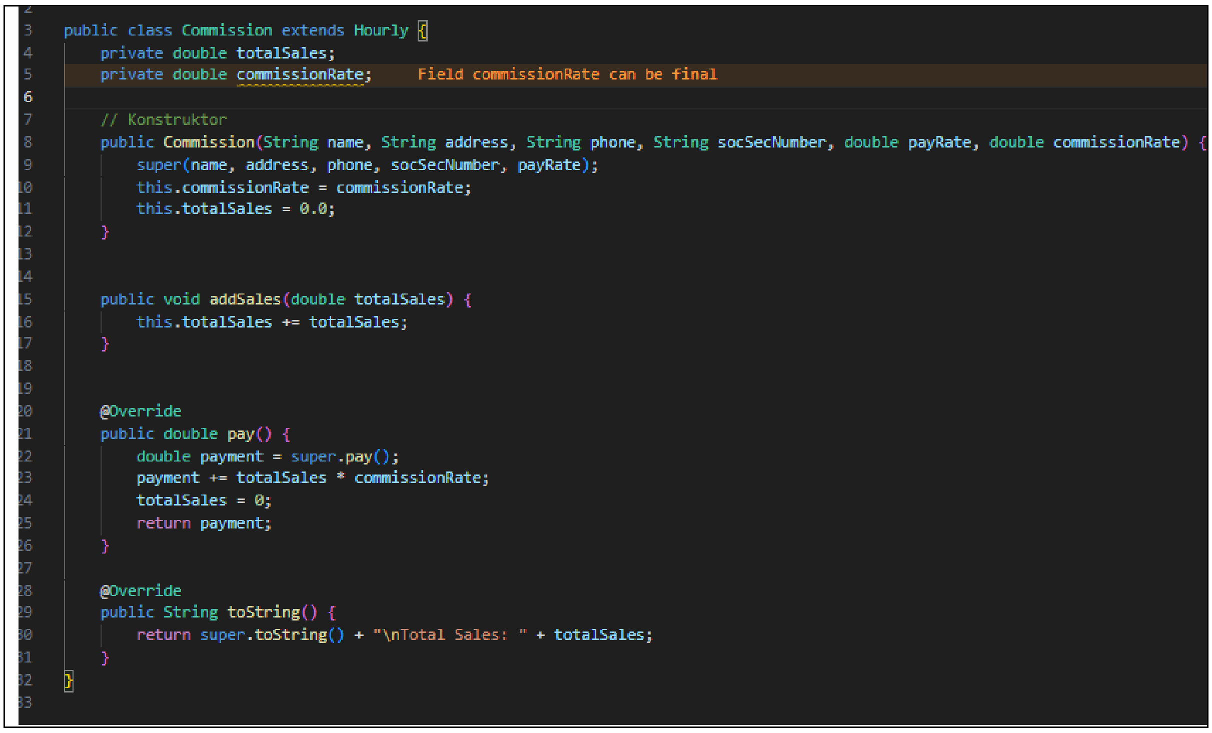
3. Permasalahan yang dihadapi.

4. Solusi dari permasalahan yang dihadapi.

5. Nama teman yang membantu memecahkan permasalahan di persoalan ini.

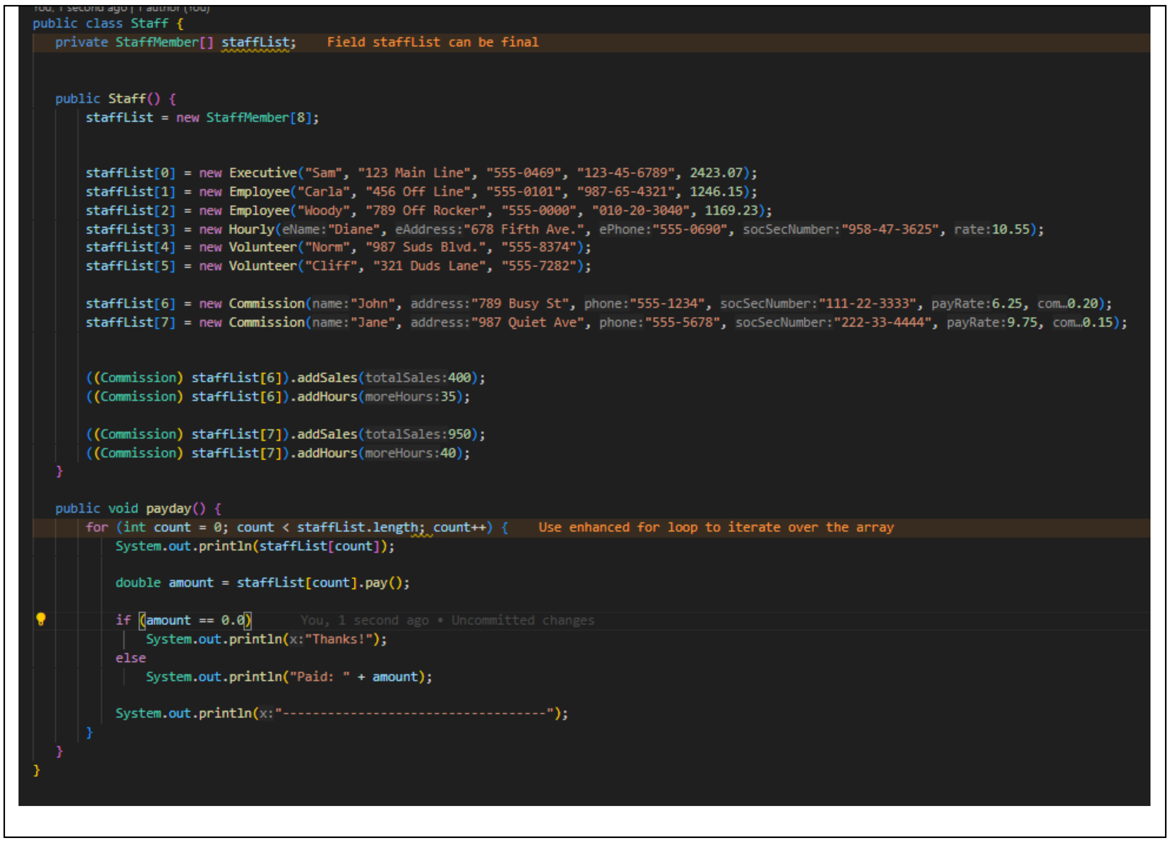
SOAL 1

1. Ini merupakan Isi dari Code Commission



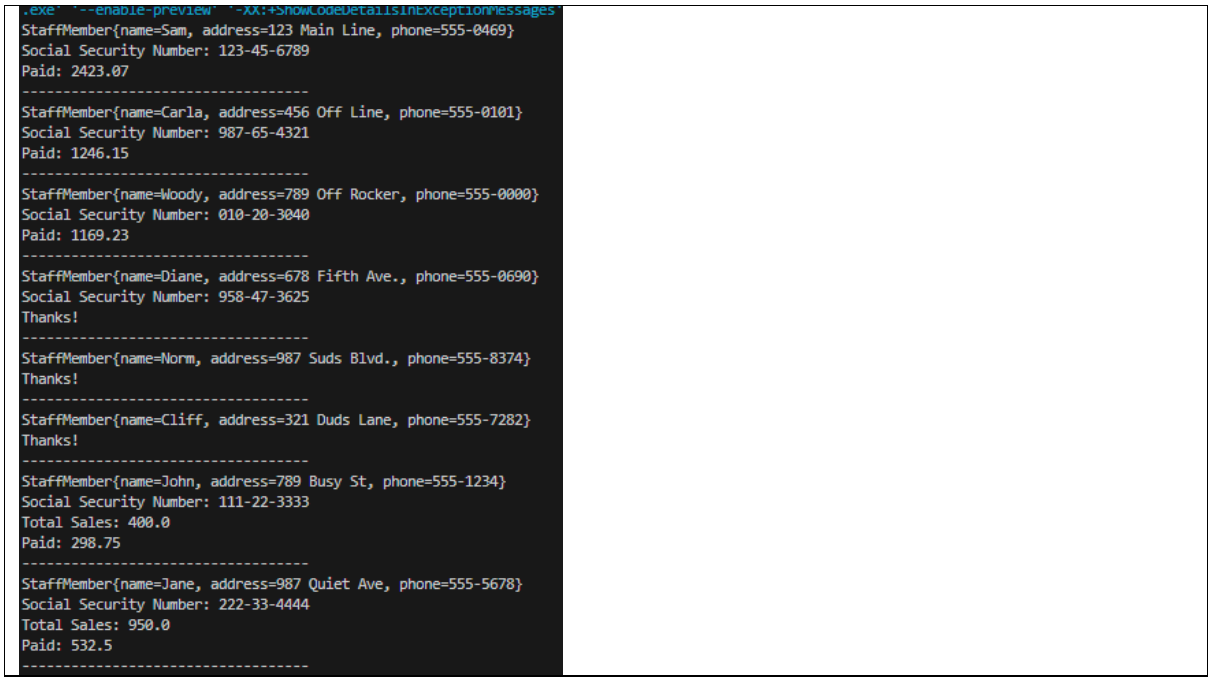
```java
public class Commission extends Hourly {
    private double totalSales;
    private double commissionRate;     Field commissionRate can be final

    // Konstruktor
    public Commission(String name, String address, String phone, String socSecNumber, double payRate, double commissionRate) {
        super(name, address, phone, socSecNumber, payRate);
        this.commissionRate = commissionRate;
        this.totalSales = 0.0;
    }


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


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

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

2. Ini code untuk Staff



```
You, 1 second ago | 1 author (You)
public class Staff {
    private StaffMember[] staffList;    Field staffList can be final


    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(eName:"Diane", eAddress:"678 Fifth Ave.", ePhone:"555-0690", socSecNumber:"958-47-3625", rate: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(name:"John", address:"789 Busy St", phone:"555-1234", socSecNumber:"111-22-3333", payRate:6.25, com…0.20);
        staffList[7] = new Commission(name:"Jane", address:"987 Quiet Ave", phone:"555-5678", socSecNumber:"222-33-4444", payRate:9.75, com…0.15);


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

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

    public void payday() {
        for (int count = 0; count < staffList.length; count++) {    Use enhanced for loop to iterate over the array
            System.out.println(staffList[count]);

            double amount = staffList[count].pay();

            if (amount == 0.0)       You, 1 second ago • Uncommitted changes
                System.out.println(x:"Thanks!");
            else
                System.out.println("Paid: " + amount);

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

Ini Outputnya

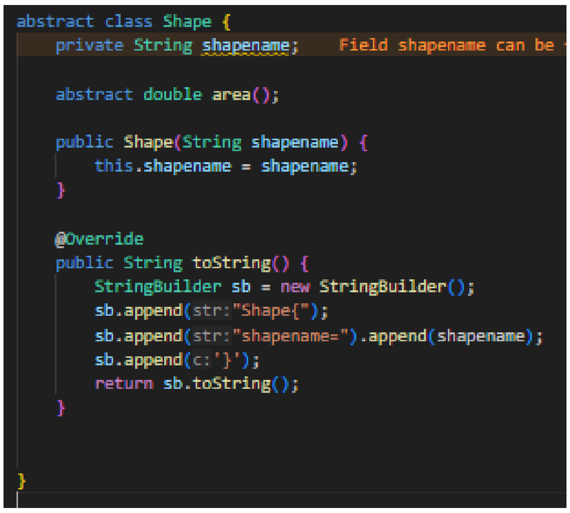


```
.exe' '--enable-preview' '-XX:+ShowCodeDetailsInExceptionMessages'
StaffMember{name=Sam, address=123 Main Line, phone=555-0469}
Social Security Number: 123-45-6789
Paid: 2423.07
-----------------------------------
StaffMember{name=Carla, address=456 Off Line, phone=555-0101}
Social Security Number: 987-65-4321
Paid: 1246.15
-----------------------------------
StaffMember{name=Woody, address=789 Off Rocker, phone=555-0000}
Social Security Number: 010-20-3040
Paid: 1169.23
-----------------------------------
StaffMember{name=Diane, address=678 Fifth Ave., phone=555-0690}
Social Security Number: 958-47-3625
Thanks!
-----------------------------------
StaffMember{name=Norm, address=987 Suds Blvd., phone=555-8374}
Thanks!
-----------------------------------
StaffMember{name=Cliff, address=321 Duds Lane, phone=555-7282}
Thanks!
-----------------------------------
StaffMember{name=John, address=789 Busy St, phone=555-1234}
Social Security Number: 111-22-3333
Total Sales: 400.0
Paid: 298.75
-----------------------------------
StaffMember{name=Jane, address=987 Quiet Ave, phone=555-5678}
Social Security Number: 222-33-4444
Total Sales: 950.0
Paid: 532.5
-----------------------------------
```

Soal 2

2.1 Soal

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



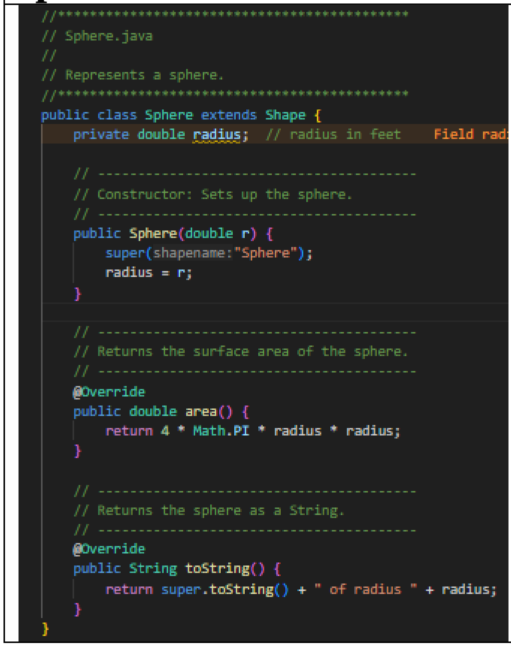
```
abstract class Shape {
    private String shapename;    Field shapename can be 

    abstract double area();

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

    @Override
    public String toString() {
        StringBuilder sb = new StringBuilder();
        sb.append(str:"Shape{");
        sb.append(str:"shapename=").append(shapename);
        sb.append(c:'}');
        return sb.toString();
    }


}
```

2.2 Soal

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.

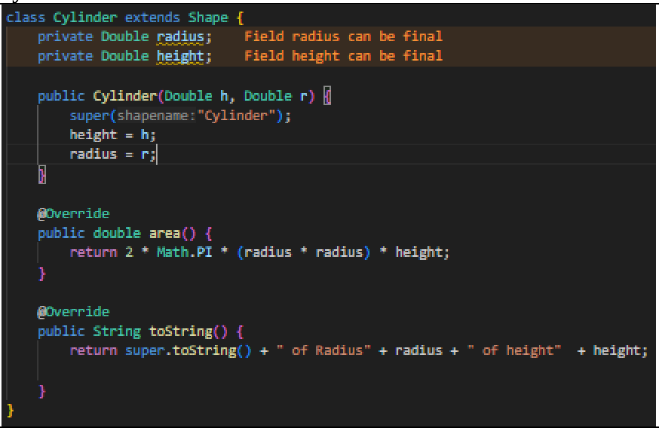
Output atau jawaban

sphere



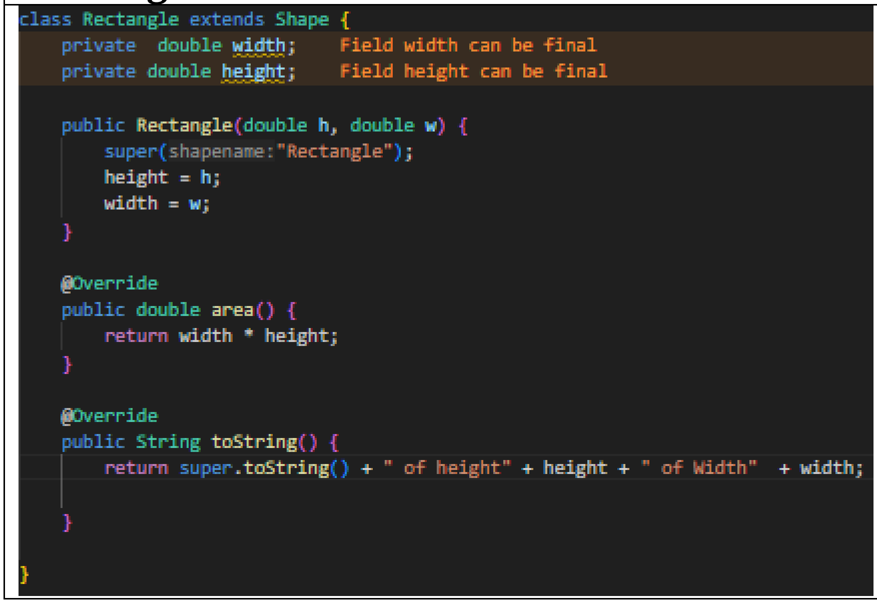
```
//*********************************************
// Sphere.java
//
// Represents a sphere.
//*********************************************
public class Sphere extends Shape {
    private double radius;  // radius in feet    Field rad

    // ---------------------------------------
    // Constructor: Sets up the sphere.
    // ---------------------------------------
    public Sphere(double r) {
        super(shapename:"Sphere");
        radius = r;
    }

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

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

cylinder



```
class Cylinder extends Shape {
    private Double radius;    Field radius can be final
    private Double height;    Field height can be final

    public Cylinder(Double h, Double r) {
        super(shapename:"Cylinder");
        height = h;
        radius = r;
    }

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

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

    }
}
```

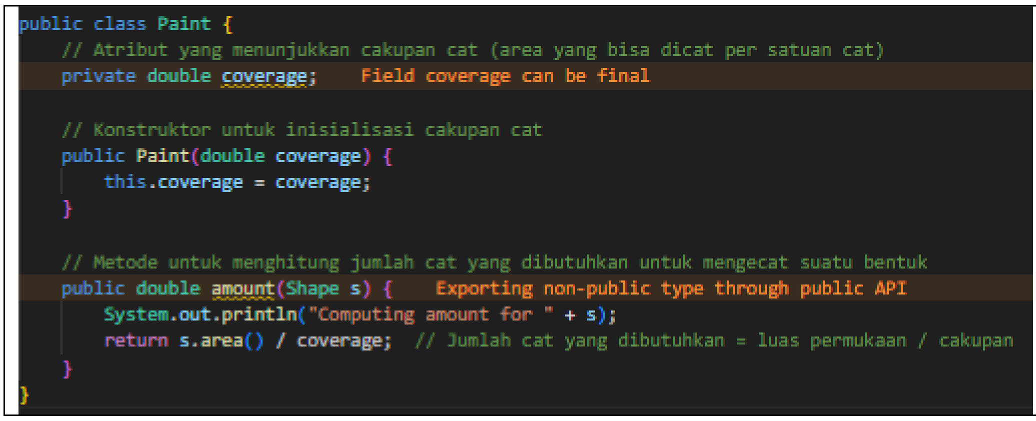
Rectangle



```
class Rectangle extends Shape {
    private  double width;     Field width can be final
    private double height;     Field height can be final

    public Rectangle(double h, double w) {
        super(shapename:"Rectangle");
        height = h;
        width = w;
    }

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

    @Override
    public String toString() {
        return super.toString() + " of height" + height + " of Width"  + width;

    }

}
```

2.3 Soal

The file Paint.java contains a class for a type of paint (which has a "coverage" and a method to compute the amount of paint needed to paint a shape). Correct the return statement in the amount method so the correct amount will be returned. Use the fact that the amount of paint needed is the area of the shape divided by the coverage for the paint.
(NOTE: Leave the print statement - it is there for illustration purposes, so you can see the method operating on different types of Shape objects.)



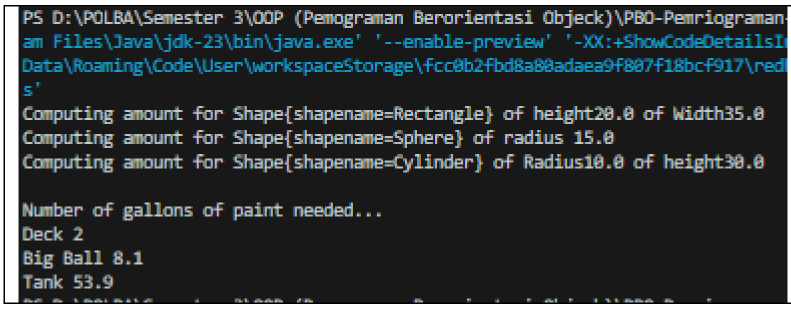
```
public class Paint {
    // Atribut yang menunjukkan cakupan cat (area yang bisa dicat per satuan cat)
    private double coverage;     Field coverage can be final

    // Konstruktor untuk inisialisasi cakupan cat
    public Paint(double coverage) {
        this.coverage = coverage;
    }

    // Metode untuk menghitung jumlah cat yang dibutuhkan untuk mengecat suatu bentuk
    public double amount(Shape s) {     Exporting non-public type through public API
        System.out.println("Computing amount for " + s);
        return s.area() / coverage;  // Jumlah cat yang dibutuhkan = luas permukaan / cakupan
    }
}
```

2.4 Soal

he 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:

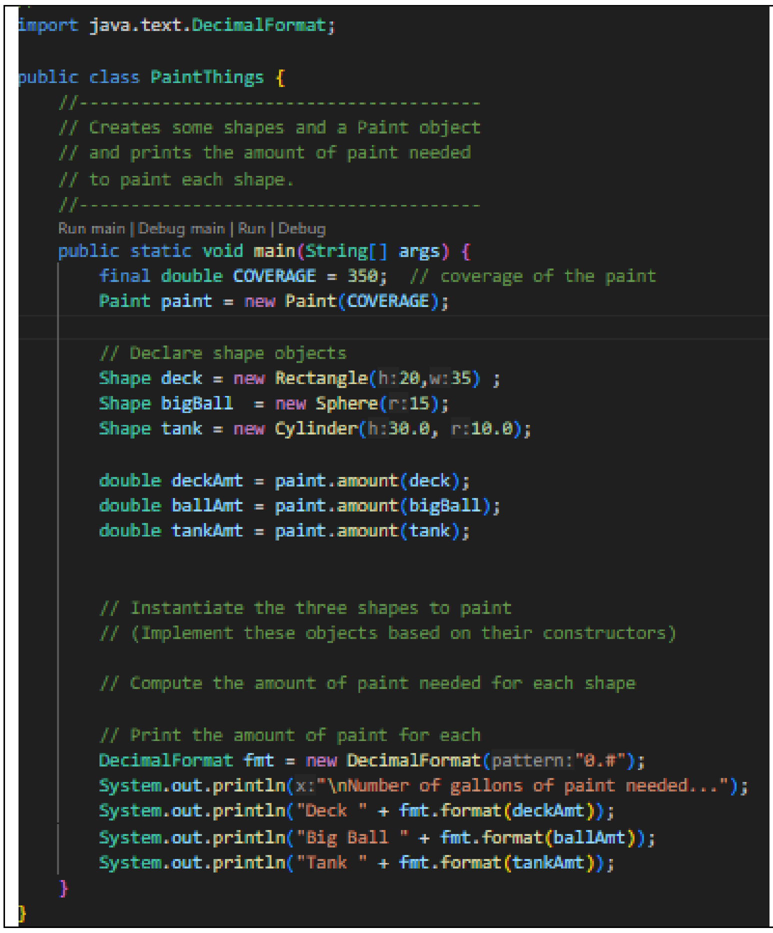
- **Instantiate the three shape objects: deck to be a 20 by 35 foot rectangle, bigBall to be a sphere of radius 15, and tank to be a cylinder of radius 10 and height 30.**
- **Make the appropriate method calls to assign the correct values to the three amount variables.**
- **Run the program and test it. You should see polymorphism in action as the amount method computes the amount of paint for various shapes**

Output



```
PS D:\POLBA\Semester 3\OOP (Pemograman Berorientasi Objeck)\PBO-Pemriograman-
am Files\Java\jdk-23\bin\java.exe' '--enable-preview' '-XX:+ShowCodeDetailsI
Data\Roaming\Code\User\workspaceStorage\fcc0b2fbd8a80adaea9f807f18bcf917\red
s'
Computing amount for Shape{shapename=Rectangle} of height20.0 of Width35.0
Computing amount for Shape{shapename=Sphere} of radius 15.0
Computing amount for Shape{shapename=Cylinder} of Radius10.0 of height30.0

Number of gallons of paint needed...
Deck 2
Big Ball 8.1
Tank 53.9
```

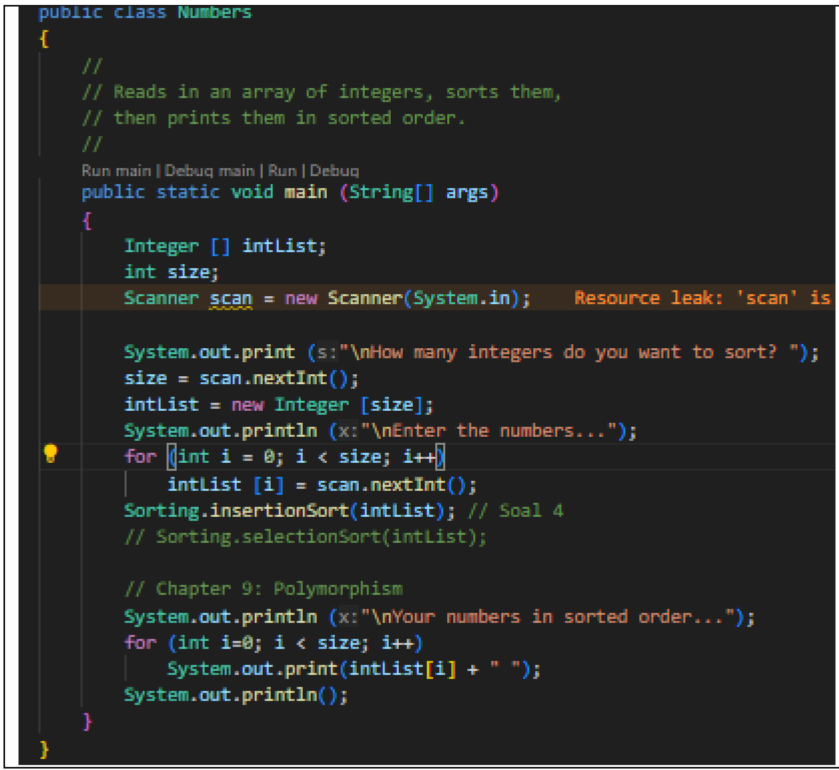
Code



```java
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.
    //----------------------------------------
    Run main | Debug main | Run | Debug
    public static void main(String[] args) {
        final double COVERAGE = 350;  // coverage of the paint
        Paint paint = new Paint(COVERAGE);

        // Declare shape objects
        Shape deck = new Rectangle(h:20,w:35) ;
        Shape bigBall  = new Sphere(r:15);
        Shape tank = new Cylinder(h:30.0, r:10.0);

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


        // Instantiate the three shapes to paint
        // (Implement these objects based on their constructors)

        // Compute the amount of paint needed for each shape

        // Print the amount of paint for each
        DecimalFormat fmt = new DecimalFormat(pattern:"0.#");
        System.out.println(x:"\nNumber of gallons of paint needed...");
        System.out.println("Deck " + fmt.format(deckAmt));
        System.out.println("Big Ball " + fmt.format(ballAmt));
        System.out.println("Tank " + fmt.format(tankAmt));
    }
}
```

3.1 Soal

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.

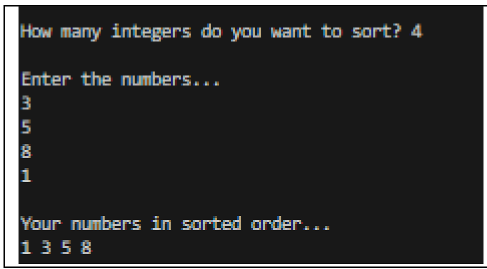


```
public class Numbers
{
    //
    // Reads in an array of integers, sorts them,
    // then prints them in sorted order.
    //
    Run main | Debug main | Run | Debug
    public static void main (String[] args)
    {
        Integer [] intList;
        int size;
        Scanner scan = new Scanner(System.in);    Resource leak: 'scan' is

        System.out.print (s:"\nHow many integers do you want to sort? ");
        size = scan.nextInt();
        intList = new Integer [size];
        System.out.println (x:"\nEnter the numbers...");
        for (int i = 0; i < size; i++)
            intList [i] = scan.nextInt();
        Sorting.insertionSort(intList); // Soal 4
        // Sorting.selectionSort(intList);

        // Chapter 9: Polymorphism
        System.out.println (x:"\nYour numbers in sorted order...");
        for (int i=0; i < size; i++)
            System.out.print(intList[i] + " ");
        System.out.println();
    }
}
```

Saya merubah dari Int ke Integer karena tipe Integer ini dapat menampung data koleksi yang hanya dapat menyimpan data objek yang membuat cocok menggunkana arraylist serta Integer menyediakan metode untuk konversi,perbandingan dan manipulasi data

3.2 Soal

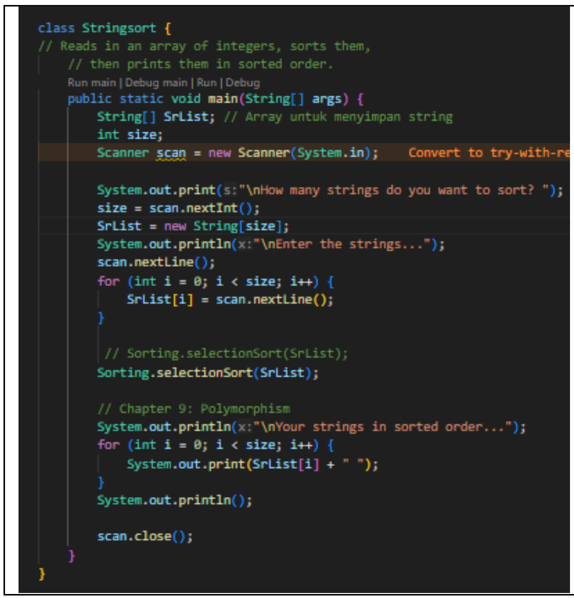
Try to compile Numbers.java and see what the error message is. The problem involves the difference between primitive data and objects. Change the program so it will work correctly (note: you don't need to make many changes - the autoboxing feature of Java 1.5 will take care of most conversions from int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



```
How many integers do you want to sort? 4

Enter the numbers...
3
5
8
1

Your numbers in sorted order...
1 3 5 8
```

3.3 Soal

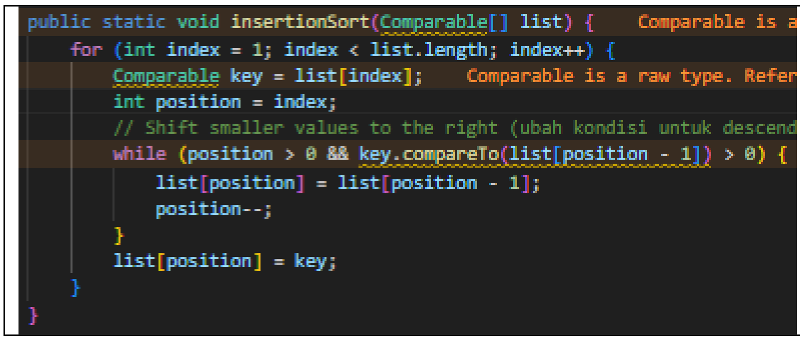
rite a program Strings.java, similar to Numbers.java, that reads in an array of String objects and sorts them. You may just copy and edit Numbers.java.



```
class Stringsort {
// Reads in an array of integers, sorts them,
    // then prints them in sorted order.
    Run main | Debug main | Run | Debug
    public static void main(String[] args) {
        String[] SrList; // Array untuk menyimpan string
        int size;
        Scanner scan = new Scanner(System.in);    Convert to try-with-re

        System.out.print(s:"\nHow many strings do you want to sort? ");
        size = scan.nextInt();
        SrList = new String[size];
        System.out.println(x:"\nEnter the strings...");
        scan.nextLine();
        for (int i = 0; i < size; i++) {
            SrList[i] = scan.nextLine();
        }

         // Sorting.selectionSort(SrList);
        Sorting.selectionSort(SrList);

        // Chapter 9: Polymorphism
        System.out.println(x:"\nYour strings in sorted order...");
        for (int i = 0; i < size; i++) {
            System.out.print(SrList[i] + " ");
        }
        System.out.println();

        scan.close();
    }
}
```

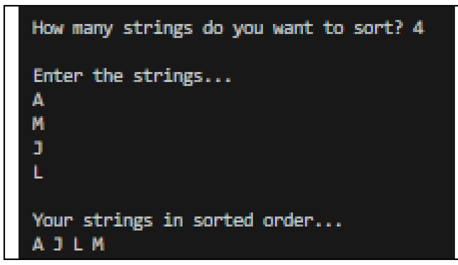
Soal 3.4

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

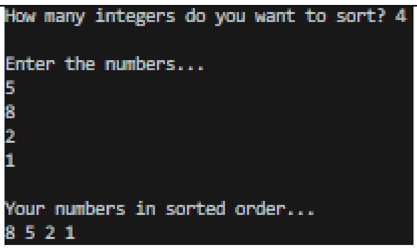


```
public static void insertionSort(Comparable[] list) {    Comparable is a
    for (int index = 1; index < list.length; index++) {
        Comparable key = list[index];    Comparable is a raw type. Refer
        int position = index;
        // Shift smaller values to the right (ubah kondisi untuk descend
        while (position > 0 && key.compareTo(list[position - 1]) > 0) {
            list[position] = list[position - 1];
            position--;
        }
        list[position] = key;
    }
}
```

Output



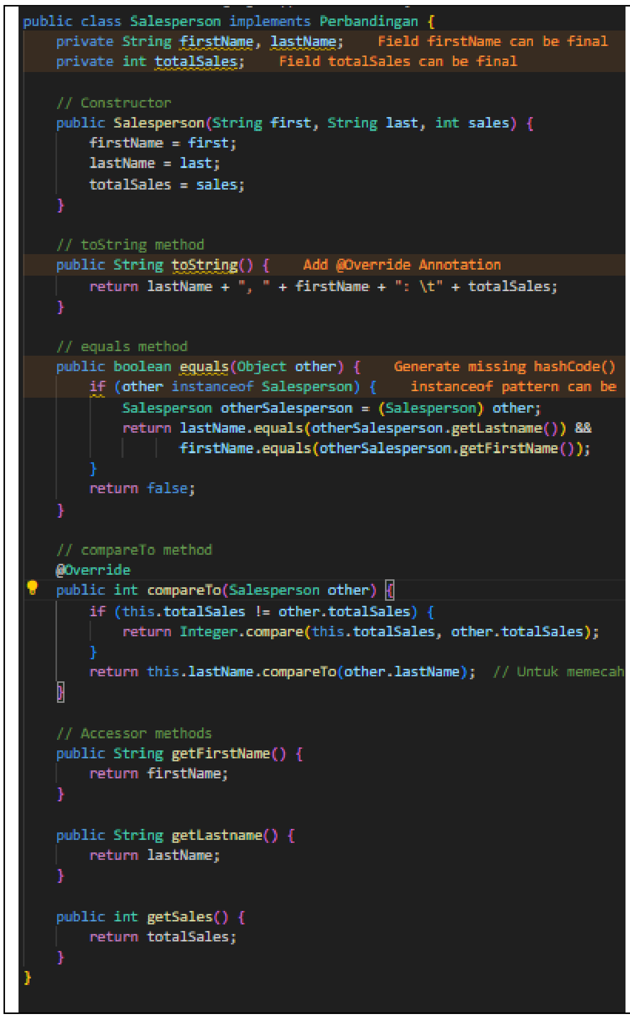
```
How many strings do you want to sort? 4

Enter the strings...
A
M
J
L

Your strings in sorted order...
A J L M
```



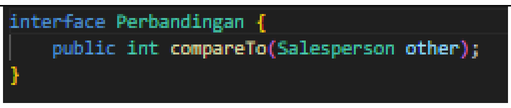
```
How many integers do you want to sort? 4

Enter the numbers...
5
8
2
1

Your numbers in sorted order...
8 5 2 1
```

3.5 soal

he 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).



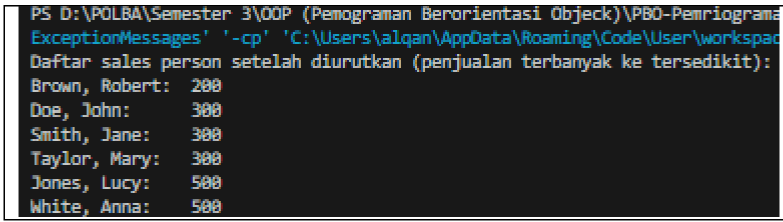
```java
public class Salesperson implements Perbandingan {
    private String firstName, lastName;    Field firstName can be final
    private int totalSales;    Field totalSales can be final

    // Constructor
    public Salesperson(String first, String last, int sales) {
        firstName = first;
        lastName = last;
        totalSales = sales;
    }

    // toString method
    public String toString() {    Add @Override Annotation
        return lastName + ", " + firstName + ": \t" + totalSales;
    }

    // equals method
    public boolean equals(Object other) {    Generate missing hashCode()
        if (other instanceof Salesperson) {    instanceof pattern can be
            Salesperson otherSalesperson = (Salesperson) other;
            return lastName.equals(otherSalesperson.getLastname()) &&
                    firstName.equals(otherSalesperson.getFirstName());
        }
        return false;
    }

    // compareTo method
    @Override
    public int compareTo(Salesperson other) {
        if (this.totalSales != other.totalSales) {
            return Integer.compare(this.totalSales, other.totalSales);
        }
        return this.lastName.compareTo(other.lastName);  // Untuk memecah
    }

    // Accessor methods
    public String getFirstName() {
        return firstName;
    }

    public String getLastname() {
        return lastName;
    }

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



```java
interface Perbandingan {
    public int compareTo(Salesperson other);
}
```

3.6 Soal

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.



```
PS D:\POLBA\Semester 3\OOP (Pemograman Berorientasi Objeck)\PBO-Pemriograma
ExceptionMessages' '-cp' 'C:\Users\alqan\AppData\Roaming\Code\User\workspac
Daftar sales person setelah diurutkan (penjualan terbanyak ke tersedikit):
Brown, Robert:  200
Doe, John:      300
Smith, Jane:    300
Taylor, Mary:   300
Jones, Lucy:    500
White, Anna:    500
```

Permasalah yang dihadapi

Saya tidak mengerti apa yang harus dilakukan dan langkah apa yang saya harus ambil yang membuat pengerjaan saya ada beberapa tidak sesuai dengan ekspetasi soal dan membuat saya tidak yakin dengan hasil dari output dan code

Solusi

Saya melakukan translate pada deply yang merupakan software translate yang dapat mentranslate dengan akurat dan bila saya masih belum mengerti maka saya akan menggunkan chatGPT untuk mengetahui langkah selanjutnya dan memastikan bahwa code saya sesuai dengan

Link Git

https://github.com/AlqanNazra/PBO-Pemriogsraman-Berorientasi-Objeck-.git