

**PEMPROGRAMAN BERORIENTASI OBJEK**  
**LAPORAN TUGAS PRAKTIKUM KE-8**  
**TOPIK : Polymorphism**



**Disusun Oleh:**  
**Alya Angraini (221511042)**

**KELAS 2B**

**JURUSAN TEKNIK KOMPUTER DAN INFORMATIKA PROGRAM**  
**STUDI DIPLOMA III TEKNIK INFORMATIKA POLITEKNIK NEGERI**  
**BANDUNG**  
**2023**

## Another Type of Employee

Tulis kelas bernama Commission dengan fitur berikut:

- Turunan kelas Hourly.

```
public class Commission extends Hourly
{
    . . . . .
```

- Memiliki dua variabel instan (selain variabel yang diwariskan): satu adalah total penjualan yang dilakukan karyawan (tipe ganda) dan yang kedua adalah tingkat komisi untuk karyawan (tingkat komisi akan diketik ganda dan akan mewakili persen (dalam bentuk desimal) komisi yang diperoleh karyawan dari penjualan (jadi 0,2 berarti karyawan memperoleh 20% komisi penjualan)).

```
private double totalSales;
private double commission_rate;
```

- Konstruktor mengambil 6 parameter: 5 parameter pertama sama dengan Hourly (nama, alamat, nomor telepon, jaminan sosial nomor, tingkat gaji per jam) dan yang ke-6 adalah tingkat komisi untuk karyawan. Konstruktor harus memanggil konstruktor dari kelas induk dengan 5 parameter pertama kemudian gunakan parameter ke-6 untuk menetapkan tingkat komisi.

```
public Commission(String eName, String eAddress, String ePhone,
    String socSecNumber, double rate, double commission_rate)
{
```

- Diperlukan satu metode tambahan: public void addSales (double totalSales) yang menambahkan parameter ke instance variabel yang mewakili total penjualan.

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

- Metode pembayaran harus memanggil metode pembayaran kelas induk untuk menghitung pembayaran jam kerja, lalu menambahkannya membayar dari komisi penjualan. (Lihat metode pembayaran di kelas Eksekutif.) Total penjualan harus dikembalikan ke 0 (catatan: Anda tidak perlu menyetel jam Bekerja kembali ke 0—mengapa tidak?). bisa jadi untuk konsistensi Data mereset "jam Bekerja" ke 0 setiap kali metode pay() dipanggil dapat menghasilkan penggandaan data yang tidak konsisten dan dapat membuat perhitungan gaji menjadi rumit.

```

@Override
public double pay()
{
    double basePay = super.pay();
    double commissionPay = totalSales * commission_rate;
    totalSales = 0;

    return basePay + commissionPay ;
}

```

- Metode toString perlu memanggil metode toString dari kelas induk lalu menambahkan total penjualan ke dalamnya.

```

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

    result += "\nTotal Sales: " + totalSales;

    return result;
}

```

Untuk menguji kelas Anda, perbarui Staff.java sebagai berikut:

- Tingkatkan ukuran array menjadi 8.

```

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

```

- Tambahkan dua karyawan yang ditugaskan ke Daftar Staf—buatlah nama, alamat, nomor telepon, dan media sosial Anda sendiri nomor keamanan. Mintalah salah satu karyawan memperoleh \$6,25 per jam dan komisi 20% dan yang lainnya memperoleh \$9,75 per jam dan komisi 15%.

```

staffList[6] = new Commission ("Alya", "425 Rocket Line",
    "555-7252", "782-83-2345", 6.25 ,0.2);

staffList[7] = new Commission ("Aruru", "496 Rock Bottom",
    "555-7412", "776-25-0545", 9.75 ,0.15);

```

- Untuk karyawan tambahan pertama yang Anda tambahkan, cantumkan jam kerja 35 dan total penjualan \$400; untuk yang kedua, letakkan jam pada 40 dan penjualan pada \$950.

```

((Hourly)staffList[5]).addHours (35);
((Commission)staffList[6]).addSales(400);
((Hourly)staffList[6]).addHours (35);
((Commission)staffList[7]).addSales(950);
((Hourly)staffList[7]).addHours (40);
}

```

OUTPUT:

-----  
Name: Alya  
Address: 425 Rocket Line  
Phone: 555-7252  
Social Security Number: 782-83-2345  
Current hours: 35  
Total Sales: 400.0  
Paid: 298.75  
-----

Name: Aruru  
Address: 496 Rock Bottom  
Phone: 555-7412  
Social Security Number: 776-25-0545  
Current hours: 40  
Total Sales: 950.0  
Paid: 532.5  
-----

## Painting Shapes

1. Tulis kelas abstrak Shape dengan properti berikut:
  - Variabel instan shapeName bertipe String
  - Area metode abstrak()
  - Metode toString yang mengembalikan nama bentuk

```
public abstract class Shape {  
    private String shapeName;  
  
    public Shape(String shapeName)  
    {  
        this.shapeName = shapeName;  
    }  
  
    public abstract double area();  
  
    public String toString()  
    {  
        return shapeName;  
    }  
}
```

2. File Sphere.java berisi kelas untuk sphere yang merupakan turunan dari Shape. sphere mempunyai jari-jari dan luasnya (luas permukaan) diberikan dengan rumus  $4 \cdot \pi \cdot \text{radius}^2$ . Tentukan kelas serupa untuk persegi panjang dan silinder. Keduanya Kelas Rectangle dan kelas Cylinder merupakan turunan dari kelas Shape. Sebuah persegi panjang ditentukan oleh panjangnya dan lebarnya dan luasnya adalah panjang dikali lebar. Sebuah silinder ditentukan oleh jari-jari dan tinggi, dan luasnya (luas permukaan) adalah  $\pi \cdot \text{radius}^2 \cdot \text{tinggi}$ . Definisikan metode toString dengan cara yang mirip dengan kelas Sphere.

```

1 package Kasus2;
2
3 public class Sphere extends Shape {
4     private double radius; //radius in feet
5
6     //-----
7     // Constructor: Sets up the sphere.
8     //-----
9     public Sphere(double r)
10    {
11        super("Sphere");
12        radius = r;
13    }
14
15    //-----
16    // Return the surface area of the sphere.
17    //-----
18    public double area()
19    {
20        double area = 4 * Math.PI * Math.pow(radius, 2);
21        return area;
22    }
23
24    //-----
25    // Return the surface area of the sphere.
26    //-----
27    public String toString()
28    {
29        return super.toString() + " of radius " + radius;
30    }
31 }
32

```

```

3 public class Cylinder extends Shape {
4     private double height;
5     private double radius;
6
7     //-----
8     //  Constructor: Sets up the rectangle.
9     //-----
10 public Cylinder(double r, double h)
11 {
12     super("Cylinder");
13     radius = r; // call superclass constructor Circle(r)
14     height = h;
15 }
16
17 //-----
18 //  Return the surface area of the sphere.
19 //-----
20 public double area()
21 {
22     double area = Math.PI * radius * radius * height;
23     return area;
24 }
25
26 //-----
27 //  Return the surface area of the sphere.
28 //-----
29 public String toString()
30 {
31     return super.toString() + " height: " + height + " and radius " + radius;
32 }
33 }
34

```

```

2
3 public class Rectangle extends Shape {
4     private double width;
5     private double length;
6
7     //-----
8     //  Constructor: Sets up the rectangle.
9     //-----
10    public Rectangle(double r, double e)
11    {
12        super("Rectangle");
13        width = r;
14        length = e;
15    }
16
17    //-----
18    //  Return the surface area of the sphere.
19    //-----
20    public double area()
21    {
22        double area = width * length;
23        return area;
24    }
25
26    //-----
27    //  Return the surface area of the sphere.
28    //-----
29    public String toString()
30    {
31        return super.toString() + " width: " + width + " and length: " + length;
32    }
33 }
34

```

3. File Paint.java berisi kelas untuk jenis cat (yang memiliki "cakupan" dan metode untuk menghitung jumlahnya cat yang dibutuhkan untuk melukis suatu bentuk). Perbaiki pernyataan pengembalian dalam metode jumlah sehingga jumlah yang benar akan diperoleh kembali. Gunakan fakta bahwa jumlah cat yang dibutuhkan adalah luas bentuk dibagi dengan cakupan cat. (CATATAN: Tinggalkan pernyataan cetak - pernyataan tersebut ada di sana untuk tujuan ilustrasi, sehingga Anda dapat melihat metode yang dijalankan berbagai jenis objek Bentuk.)

```

public double amount(Shape s)
{
    System.out.println("Computing amount for " + s);
    double area = s.area();
    // Return Use the fact that the amount of paint needed
    // is the area of the shape divided by the coverage for the paint
    return area / coverage;
}

```

4. File PaintThings.java berisi program yang menghitung jumlah cat yang dibutuhkan untuk melukis berbagai bentuk. A objek cat telah dipakai. Tambahkan yang berikut ini untuk menyelesaikan program:
  - Buat instance tiga objek bentuk: dek menjadi persegi panjang berukuran 20 kali 35 kaki, bigBall menjadi bola berjari-jari 15, dan tank menjadi silinder dengan jari-jari 10 dan tinggi 30.



- Lakukan pemanggilan metode yang sesuai untuk menetapkan nilai yang benar ke tiga variabel jumlah.
- Jalankan program dan ujilah. Anda akan melihat polimorfisme beraksi saat metode jumlah menghitung jumlah cat untuk berbagai bentuk.

```
//*****
//PaintThings.java
//
//Compute the amount of paint needed to paint various
//things. uses the amount method of the paint class which
//takes any shape as a parameter.
//*****

public class PaintThings {

    //-----
    // Create 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
        deck = new Rectangle(20, 35);
        bigBall = new Sphere(15);
        tank = new Cylinder(10, 30);

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

        //print the amount of paint for each
        DecimalFormat fmt = new DecimalFormat("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));
    }
}
```

OUTPUT:

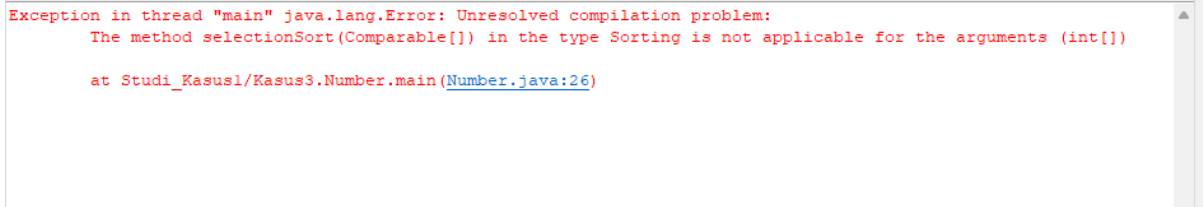
```
Computing amount for Rectangle width: 20.0 and length: 35.0
Computing amount for Sphere of radius 15.0
Computing amount for Cylinder height: 30.0 and radius 10.0

Number of gallons of paint needed...
Deck 2
Big Ball 8.1
Tank 26.9
```

Program ini akan menghitung dan mencetak jumlah cat yang diperlukan untuk melukis tiga bentuk yang berbeda (persegi panjang, bola, dan silinder) menggunakan polimorfisme, di mana metode amount() dari kelas Paint menerima berbagai bentuk sebagai parameter.

## Polymorphic Sorting

1. File Numbers.java dibaca dalam array bilangan bulat, memanggil algoritma pengurutan pemilihan untuk mengurutkannya, dan kemudian mencetak array yang diurutkan. Simpan Sorting.java dan Numbers.java ke direktori Anda. Numbers.java tidak dapat dikompilasi saat ini membentuk. Pelajarilah untuk mengetahui apakah Anda dapat mengetahui alasannya.



```
Exception in thread "main" java.lang.Error: Unresolved compilation problem:
    The method selectionSort(Comparable[]) in the type Sorting is not applicable for the arguments (int[])
    at Studi_Kasus1/Kasus3.Number.main(Number.java:26)
```

2. Coba kompilasi Numbers.java dan lihat pesan kesalahannya. Masalahnya melibatkan perbedaan antara data dan objek primitif. Ubah programnya agar berfungsi dengan benar (catatan: tidak perlu membuat banyak perubahan - fitur autoboxing Java 1.5 akan menangani sebagian besar konversi dari int ke Integer).

```
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];
}
```

Menjadi:

```
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]; //menggunakan object

System.out.println ("\nEnter the numbers...");
for (int i = 0; i < size; i++)
    intList[i] = scan.nextInt();
```

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

```
Enter the numbers...
```

```
56735
```

```
873
```

```
2
```

```
75
```

```
20
```

```
|
```

```
Your numbers in sorted order...
```

```
56735 873 75 20 2
```

3. Tulis program Strings.java, mirip dengan Numbers.java, yang membaca array objek String dan mengurutkannya. Anda boleh saja menyalin dan mengedit Numbers.java.

```
public class Strings
{
    //-----
    // Reads in an array of strings, sorts them alphabetically,
    // 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 strings do you want to sort? ");
        size = scan.nextInt();
        stringList = new String[size];

        System.out.println("\nEnter the strings...");
        for (int i = 0; i < size; i++)
            stringList[i] = scan.next();

        // Use insertion sort to sort strings alphabetically
        //dec
        Sorting.insertionSort(stringList);
        //asc
        Sorting.selectionSort(stringList);

        System.out.println("\nYour strings in sorted order (alphabetically)...");
        for (int i = 0; i < size; i++)
            System.out.print(stringList[i] + " ");
        System.out.println();
    }
}
```

4. Ubah algoritma insertionSort sehingga mengurutkannya dalam urutan menurun, bukan dalam urutan menaik. Mengubah Numbers.java dan Strings.java untuk memanggil insertionSort daripada choiceSort. Jalankan keduanya untuk memastikan penyortirannya berhasil benar.

```

//-----
// 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;
    }
}

```

Menjadi

```

//-----
// Sort 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;
    }
}

```

OUTPUT:

- Selection

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

```
Enter the numbers...
```

```
987
```

```
76
```

```
5
```

```
3
```

```
|
```

```
Your numbers in sorted order...
```

```
3 5 76 987
```

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

```
Enter the strings...
```

```
alya
```

```
hehe
```

```
aku
```

```
tehe
```

```
Your strings in sorted order (alphabetically)...
```

```
aku alya hehe tehe
```

- insertion

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

```
Enter the numbers...
```

```
34
```

```
90
```

```
65
```

```
8
```

```
|
```

```
Your numbers in sorted order...
```

```
90 65 34 8
```

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

```
Enter the strings...
```

```
abjad buba caci dadu
```

```
Your strings in sorted order (alphabetically)...
```

```
dadu caci buba abjad
```

5. File Salesperson.java mendefinisikan sebagian kelas yang mewakili staf penjualan. Ini sangat mirip dengan Kontak kelas di Listing 9.10. Namun, staf penjualan memiliki

nama depan, nama belakang, dan jumlah total penjualan (int). Selain nama depan, nama belakang, dan nomor telepon. Selesaikan metode `bandingkanTo` di kelas `Tenaga Penjualan`. Itu perbandingan harus didasarkan pada total penjualan; yaitu mengembalikan angka negatif jika objek pelaksana memiliki total penjualan lebih kecil dari objek lainnya dan menghasilkan angka positif jika penjualannya lebih besar. Gunakan nama staf penjualan untuk putusn dasi (urutan abjad).

```
//-----
// Order is based on total sales with the name
// (last, then first) breaking a tie.
//-----
public int compareTo(Salesperson other) {
    if (this.totalSales < other.totalSales) {
        return 1;
    } else if (this.totalSales > other.totalSales) {
        return -1;
    } else {
        // If total sales are the same, use the name to break the tie in descending alphabetical order
        int nameComparison = other.lastName.compareTo(this.lastName); // Compare in descending order
        if (nameComparison != 0) {
            return nameComparison;
        } else {
            return other.firstName.compareTo(this.firstName); // Compare in descending order
        }
    }
}
```

- File `WeeklySales.java` berisi driver untuk menguji metode perbandingan dan penyortiran (ini mirip dengan Cantumkan 9.8 dalam teks). Kompilasi dan jalankan. Pastikan metode `bandingkanTo` Anda benar. Staf penjualan seharusnya diurutkan berdasarkan urutan penjualan dari yang terbanyak hingga yang terkecil dengan empat orang tersebut memiliki jumlah penjualan yang sama secara terbalik Sesuai abjad.

```
Sorting.insertionSort(salesStaff);
```

Menjadi:

```
Sorting.selectionSort(salesStaff);
System.out.println ("\nRanking of Sales for the Week\n");
```

```
Ranking of Sales for the Week
```

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

Polymorp dalam kasus ini pada dimana dengan menggunakan antarmuka Comparable, metode selectionSort dan insertionSort dapat bekerja dengan berbagai jenis objek yang dapat dibandingkan tanpa perlu mengetahui jenis objek yang sesungguhnya tanpa harus mengubah implementasi metode itu sendiri.

```
//-----  
public static void selectionSort (Comparable[] list)  
{  
public static void insertionSort (Comparable[] list) {  
    for (int index =1; index < list.length ; index++)  
    {
```

## 7. Modify

```
1 package Kasus3;  
2 import java.util.Scanner;  
3  
4 public class ModifyWeeklySales {  
5     public static void main(String[] args) {  
6         Scanner scan = new Scanner(System.in);  
7  
8         System.out.print("How many salespeople do you want to rank? ");  
9         int numSalespeople = scan.nextInt();  
10        scan.nextLine(); // Consuming the newline character  
11  
12        Salesperson[] salesStaff = new Salesperson[numSalespeople];  
13  
14        for (int i = 0; i < numSalespeople; i++) {  
15            System.out.print("Enter first name for salesperson " + (i + 1) + ": ");  
16            String firstName = scan.nextLine();  
17  
18            System.out.print("Enter last name for salesperson " + (i + 1) + ": ");  
19            String lastName = scan.nextLine();  
20  
21            System.out.print("Enter total sales for salesperson " + (i + 1) + ": ");  
22            int totalSales = scan.nextInt();  
23            scan.nextLine(); // Consuming the newline character  
24  
25            salesStaff[i] = new Salesperson(firstName, lastName, totalSales);  
26        }  
27  
28        Sorting.insertionSort(salesStaff);  
29  
30        System.out.println("\nRanking of Sales for the Week\n");  
31  
32        for (Salesperson s : salesStaff) {  
33            System.out.println(s);  
34        }  
35    }  
36 }  
37
```