1. **DEFINISI MASALAH**
2. Perusahaan NV. Meneer memiliki koperasi karyawan yang memungkinkan karyawannya berbelanja di koperasi tersebut. Tentunya, karyawan tersebut bisa membayar belanjaanya tersebut di akhir bulan melalui pemotongan gaji. Ada 2 kelas yang terlibat disini, Invoice dan Employee. Kedua class tadi mengimplementasikan interface Payable yang mana ia hanya memiliki satu method yang harus diimplementasikan di kedua class, yaitu getPayableAmount(). Program harus bisa mengolah gaji karyawan di akhir bulan beserta invoice belanjaan karyawan yang nantinya gaji karyawan perbulannya dikurang total harga belanjaanya secara polimorfis. Tampilkan informasi dari karyawan tersebut beserta total gaji setelah dipotong hutang belanjaan di koperasi dan tampilkan pula detail belanjaanya secara polimorfis pula.

Attribut dari Invoice: String productName, Integer quantity, Integer pricePerItem Attribut dari Employee: Integer registrationNumber, String name, Integer salaryPerMonth, Invoice[] invoices

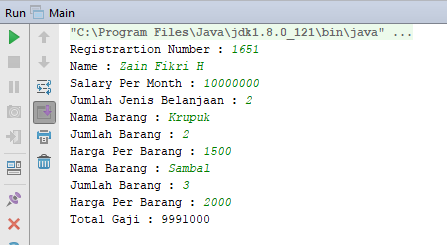
1. **SOURCE CODE**

|  |  |
| --- | --- |
|  | Payable.java |
| 1  2  3  4  5  6 | */\*\*  \* Created by Zain Fikri H on 13/05/2017.  \*/* public interface Payable {  public Integer getPayableAmount(); } |
|  | Invoice.java |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37 | */\*\*  \* Created by Zain Fikri H on 13/05/2017.  \*/* public class Invoice implements Payable {  public String productName;  public Integer quantity;  public Integer pricePerItem;   public String getProductName() {  return productName;  }   public Integer getQuantity() {  return quantity;  }   public Integer getPricePerItem() {  return pricePerItem;  }   public void setProductName(String productName) {  this.productName = productName;  }   public void setQuantity(Integer quantity) {  this.quantity = quantity;  }   public void setPricePerItem(Integer pricePerItem) {  this.pricePerItem = pricePerItem;  }   @Override  public Integer getPayableAmount() {  return getPricePerItem()\*getQuantity();  } } |
|  | Employee.java |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36 | import java.util.ArrayList;  */\*\*  \* Created by Zain Fikri H on 13/05/2017.  \*/* public class Employee implements Payable{  public Integer registrationNumber;  public String name;  public Integer salaryPerMonth;  public static Integer *payable*=0;  public ArrayList<Invoice> invoiceArrayLis= new ArrayList<>();  Invoice invoice = new Invoice();   public Employee(Integer registrationNumber, String name, Integer salaryPerMonth) {  this.registrationNumber = registrationNumber;  this.name = name;  this.salaryPerMonth = salaryPerMonth;  }   public void setInvoice(String nama, Integer quantity, Integer price){  invoice.setProductName(nama);  invoice.setPricePerItem(price);  invoice.setQuantity(quantity);  invoiceArrayLis.add(invoice);  *payable* += invoice.getPayableAmount();  }   @Override  public Integer getPayableAmount() {  System.*out*.print("Total Gaji : ");  return salaryPerMonth-*payable*;  } } |
|  | Main.java |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39 | import java.util.ArrayList; import java.util.Scanner;  */\*\*  \* Created by Zain Fikri H on 14/05/2017.  \*/* public class Main {  public static void main(String[] args) {  Scanner in = new Scanner(System.*in*);  ArrayList<Employee> employees = new ArrayList<>();  System.*out*.print("Registrartion Number : ");  Integer no = in.nextInt();  in.nextLine();  System.*out*.print("Name : ");  String name = in.nextLine();  System.*out*.print("Salary Per Month : ");  Integer salary = in.nextInt();  employees.add(new Employee(no,name,salary));  System.*out*.print("Jumlah Jenis Belanjaan : ");  Integer jenis = in.nextInt();  String nama[]= new String[jenis];  Integer quantity[]= new Integer[jenis];  Integer price[]= new Integer[jenis];  for(int i=0; i<jenis; i++){  System.*out*.print("Nama Barang : ");  nama[i]=in.next();  System.*out*.print("Jumlah Barang : ");  quantity[i]=in.nextInt();  System.*out*.print("Harga Per Barang : ");  price[i]=in.nextInt();  }  for(int i=0;i<nama.length;i++){  employees.get(0).setInvoice(nama[i],quantity[i],price[i]);  }  System.*out*.println(employees.get(0).getPayableAmount());  } } |

1. **PEMBAHASAN**

|  |  |
| --- | --- |
|  | MainPerpus.java |
|  |  |
|  | Kategori.java |
|  |  |
|  | Buku.java |
|  |  |

1. **SCREENSHOT PROGRAM**

****

1. **PRAKTIKUM**

|  |  |
| --- | --- |
|  | MainPerpus.java |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43 | package praktikumpl;  public interface Colorable {  public void howToColor();  }  public interface Comparable {  public void compareTo(Object obj);  }  public class Rectangle implements Colorable, Comparable { // class rectanggle  private String warna;  private intkategori;  public Rectangle() {  }  public Rectangle(String warna) {  this.warna = warna;  }  public void howToColor() {  if (this.warna == null) {  System.out.println("tidakadawarna, warnabangunkotakmasih polos");  } else {  System.out.println("bangunkotaksudahdiwarnaidenganwarna " + this.warna);  }  }  public void compareTo(Object obj) {  this.kategori = (int) obj;  if (this.kategori == 0) {  System.out.println("ukuran cat yang cocokuntukbangunkotakdenganukurankategori " + this.kategori + " yaitu 2.5L");  } else {  System.out.println("ukuran cat yang cocokuntukbangunkotakdenganukurankategori " + this.kategori + " yaitu 6.5L");  }  }  public static void main(String[] args) {  Rectangle kotak1 = new Rectangle("merah");  Rectangle kotak2 = new Rectangle();  Rectangle kotak3 = new Rectangle();  kotak1.howToColor();  kotak2.howToColor();  kotak3.compareTo(4);  }  } |

1. Lakukan percobaan diatas dan benahi jika menemukan kesalahan serta jelaskan!

*Jawaban:*

Dari hsail percobaan terdapat error karena terdapat lebih dari 1 class public dalam satu file java. Hal tersebut bias diselesaikan dengan menghilangkan public pada interface. Atau bias membuatkan file baru untuk setiap interfacenya.

1. Apakah class yang berbentuk Interface bisa diinstansiasi menjadi sebuah objek? Jelaskan alasannya!

*Jawaban:*

Class yang berbentuk Interface tidak dapat diinstansiasi karena sebenarnya Interface sama halnya dengan calss abstract yang hanya memiliki method tanpa ada isinya. Karena Interface adalah class induk yang isi methodnya hanya dapat diisi pada class yang mengimplementasikannya.

1. Apakah suatu class dapat mengimplementasi class interface yang jumlahnya lebih dari satu? Jelaskan alasannya!

*Jawaban:*

Suatu class dapat mengimplementasikan class interface lebih dari satu. Karena pada prinsipnya suatu class dapat memiliki sifat yang sama dengan class interfacenya.

1. Ubah source code diatas menjadi proses meminta inputan dari user dan buat menjadi interaktif!

*Jawaban:*

public static void main(String[] args) {  
 Scanner in = new Scanner(System.*in*);  
 System.*out*.println("Masukkan Warna Kotak 1 : ");  
 String warnaKotak1=in.next();  
 System.*out*.println("Masukkan Warna Kotak 2 : ");  
 String warnaKotak2=in.next();  
 System.*out*.println("Masukkan Warna Kotak 3 : ");  
 String warnaKotak3=in.next();  
 Rectangle kotak1 = new Rectangle(warnaKotak1);  
 Rectangle kotak2= new Rectangle(warnaKotak2);  
 Rectangle kotak3 = new Rectangle(warnaKotak3);  
 kotak1.howToColor();  
 kotak2.howToColor();  
 kotak3.compareTo(4);  
}

1. Buat objek selain objek diatas dengan menggunakan method yang berbeda dengan yang diatas! (min.1 contoh)

*Jawaban:*

public void luas(Object obj){  
 this.kategori = (int) obj;  
 if(this.kategori == 0) {  
 System.*out*.println("Jenis luas kotak dengan ukuran kategori " +this.kategori+ " adalah luas kecil");  
 }  
 else {  
 System.*out*.println("Jenis luas kotak dengan ukuran kategori "+this.kategori+" adalah luas besar");  
 }  
}  
public static void main(String[] args) {  
 Scanner in = new Scanner(System.*in*);  
 System.*out*.println("Masukkan Warna Kotak 1 : ");  
 String warnaKotak1=in.next();  
 System.*out*.println("Masukkan Warna Kotak 2 : ");  
 String warnaKotak2=in.next();  
 System.*out*.println("Masukkan Warna Kotak 3 : ");  
 String warnaKotak3=in.next();  
 Rectangle kotak1 = new Rectangle(warnaKotak1);  
 Rectangle kotak2= new Rectangle(warnaKotak2);  
 Rectangle kotak3 = new Rectangle(warnaKotak3);  
 kotak1.howToColor();  
 kotak2.howToColor();  
 kotak3.compareTo(4);  
}

1. **KESIMPULAN**

* Apa itu interface, polymorphism hanya memungkinkan untuk mewarisi sifat dari satu kelas abstrak saja, namun tidak jarang juga diperlukan untuk mewarisi sifat lebih dari satu class abstrak, atau biasa disebut juga dengan multiple inheritance. Dalam Java, sebuah class hanya bisa mewarisi sifat dari sebuah superclass atau yang biasa disebut dengan single inheritance. Sebuah Java class tersebut tidak mendukung prinsip multiple inheritance. Dalam memenuhi prinsip multiple inheritance, pada Java terdapat Interfaces yang memiliki aspek seperti multiple inheritance namun berbeda dengan abstract class.
* Syntax untuk mendeklarasikan interface adalah sebagai berikut:

Modifier interface Interface\_name {

/\*\* Konstanta \*/

/\*\* Abstract Method \*/

}

Sebuah kelas dapat mengimplementasikan sebuah interface dengan menggunakan kata kunci implements sebagai berikut.

Modifier class class\_name implements Interface\_name {

/\*\* Attribute \*/

/\*\* Method \*/

}

* Interface dan polimorfisme terdapat perbedaan yaitu pada method interface tidak punya tubuh, sebuah interface hanya dapat mendefinisikan konstanta dan interface tidak langsung mewariskan hubungan dengan class istimewalainnya,, mereka didefinisikan secara independent.Dapat diimplementasikan lebih dari satu interface (multiple inheritance) dengan memisahkan nama dari setiap interface dengan tanda koma.
* Java sendiri telah menyediakan class interface yaitu Appendable, autocloseable, charsequence, cloneable, comparable, iterable, readable, runable, dll.
* Interface digunakan ketika kita menginginkan sebuah method dengan sifat yang sama tapi berjalan pada class yang berbeda dan tidak terdapat hubungan. Atau jika suatu class ingin memiliki lebih dari satu class induk atau multiple inheritance.