Pewarisan, Abstrak, & Interface

|  |
| --- |
| No: 4  Materi: Abstract dan Interface  Poin: 15 |
| Terdapat sebuah interface “InterfesA” dengan method “methodNSatu”, “methodNDua” dan “methodNTiga”. InterfesA mengambil sifat-sifat dari interface “InterfesB” yang mempunyai method “methodASatu” dan “methodADua”. Selain itu terdapat interface “InterfesC” dengan method berupa function “methodB” yang mengembalikan tipe integer dan konstanta (nama konstanta bebas) dengan tipe data String yang berisikan kata “Halo...”.  Kelas “AsliA” mengimplements “InterfesA” dan “InterfesC” dengan method tambahan “MethodRSatu”. Selain itu, terdapat kelas abstrak “AbstrakB” dengan method abstrak “methodAbstA” berupa prosedur, “methodAbstB” berupa function dengan tipe data balikan integer, dan method non-abstrak “methodRealB” yang merupakan function dengan tipe data balikan integer, “methodRealB” memiliki sebuah parameter masukan “x” dengan tipe data integer.  Kelas “AbstrakA” merupakan kelas yang meng-extends “AbstrakB” dengan method abstrak “methodAbstC” dan meng-override “methodRealB” dan “methodAbstB”.  Kelas “AsliB” meng-extends kelas “AbstrakA” dengan method terkandung adalah method yang wajib di-override oleh kelas ini. Pada kelas “AsliB” terdapat atribut static “statikSatu” (tipe String) dengan visibilitas protected dan “statikDua” dengan tipe integer. Kelas “AsliB” merupakan kelas dengan keyword final. Selain itu terdapat atribut “atributReal” dengan tipe “AsliA”. Hubungan antara kelas “AsliB” dan “AsliA” adalah whole-part (aggregation).  Tentukan method yang harus di-override oleh kelas “AsliA” dan “AsliB”. |

Kasus di atas bisa ditentukan dengan mengetahui sifat dari sebuah abstract class dan interface.

Sifat dari sebuah abstract class:

1. Setiap abstract class boleh memilii method abstract;
2. Method abstract harus di-override oleh class yang meng-extends class abstract.

Sifat dari interface:

1. Setiap interface hanya memiliki method tanpa body method;
2. Setiap class yang mengimplements interface harus meng-override method pada interface yang dimaksud.

Berdasarkan sifat di atas didapat kesimpulan bahwa:

|  |  |
| --- | --- |
| AsliA harus meng-override:   1. methodNSatu 2. methodNDua 3. methodNTiga 4. methodASatu 5. methodADua 6. methodB | AsliB harus meng-override:   1. methodAbstA 2. methodAbstC |

Berdasarkan hal di atas, didapat kesimpulan bahwa setiap method abstract dan method pada interface harus diturunkan oleh sub class-nya. Jika tidak di-override, kewajiban akan berlanjut pada sub class berikutnya.

|  |
| --- |
| No: 2  Materi: Inheritance  Poin: 15 |
| Diketahui sebuah program memiliki kelas-kelas dengan kode sebagai berikut  class Bird{  protected String tweet;  public Bird(){  System.out.println(“This is Bird”);  }  }  class Parrot extends Bird{  public Parrot(String tweet){  super.tweet = tweet;  System.out.println(“This is Parrot Species”);  }  public void tweet(){  System.out.println(tweet);  }  }  Jika terdapat sebuah driver class sebagai berikut,  public class Main {  public static void main(String[] args) {  Parrot p = new Parrot("I'm a Parrot");  p.tweet();  }  }  Apakah program dapat dijalankan? Jika dapat, tuliskan dan jelaskan keluaran dari program di atas.  Dan… jika terdapat sebuah driver class sebagai berikut,  public class Main {  public static void main(String[] args) {  Bird p = new Parrot("I'm a Parrot");  p.tweet();  }  }  Apakah program dapat dijalankan? Jika dapat, tuliskan dan jelaskan keluaran dari program di atas. |

Pertanyaan pertama, apakah kode

|  |
| --- |
| public class Main {  public static void main(String[] args) {  Parrot p = new Parrot("I'm a Parrot");  p.tweet();  }  } |

Dapat dijalankan? Jawabannya adalah dapat dijalankan dengan hasil akhir

|  |
| --- |
| This is Bird  This is Parrot Species  I'm a Parrot |

Dua baris pertama didapat dari pemanggilan konstruktor class Bird dan Parrot. Walaupun pemanggilan dilakukan terhadap konstruktor dari class Parrot, konstruktor super class (kelas induk) akan tetap dipanggil sebelum memanggil konstruktor sub class. Baris terakhir (I'm a Parrot) didapat dari pemanggilan method “tweet” pada class Parrot.

Pertanyaan kedua, apakah kode

|  |
| --- |
| public class Main {  public static void main(String[] args) {  Bird p = new Parrot("I'm a Parrot");  p.tweet();  }  } |

Dapat dijalankan? Jawabannya adalah tidak dapat dijalankan.

Hal ini dikarenakan baris kode p.tweet(). Kode tersebut mengakibatkan error saat kompilasi, karena method tweet dimiliki oleh class Parrot, sedangkan pada saat membentuk objek “p”, tipe yang dipakai adalah tipe “Bird”. Pada class “Bird” tidak terdapat method tweet, sehingga akan terjadi gagal kompilasi.

|  |
| --- |
| Nomor: 2  Tipe: Interface |
| Seorang engineer ingin membuat lampu dengan 1 saklar/tombol. Sebelum membuat rangkaian lampu tersebut, ia membuat simulasinya dalam sebuah program dengan konsep OO. Lampu tersebut dibagi ke dalam 2 tipe, tipe 2 state dan tipe 3 state. Tipe 2 state memiliki 2 tingkat nyala lampu, tingkat 0 (mati), dan tingkat 2 (menyala). Tipe 3 state memiliki 3 tingkat nyala lampu, tingkat 0 (mati), tingkat 1 (menyala redup), dan tingkat 2 (menyala).  Tipe 2 state memiliki algoritma sebagai berikut ketika tombol ditekan:   1. Cek apakah lampu menyala atau tidak menyala, 2. jika menyala, matikan lampu, 3. jika tidak menyala, nyalakan lampu   Tipe 3 state memiliki algoritma sebagai berikut ketika tombol ditekan:   1. Cek apakah lampu menyala, menyala redup atau mati, 2. jika lampu mati, hidupkan dengan nyala redup 3. jika lampu menyala redup, tambahkan intensitas penyalaan, 4. jika lampu menyala, matikan lampu tersebut.   Dikarenakan ingin menggunakan konsep interface, berikut adalah class diagram yang dirancang untuk membuat simulasi tersebut:    Note: Sebelum dijalankan/di-klik, pada kondisi awal, lampu dalam kondisi mati.  Contoh dari Driver Class adalah sebagai berikut:   |  | | --- | | public class Main {  public static void main(String[] args) {  System.out.println("Lampu Dua State");  LampuDuaState ld = new LampuDuaState();  System.out.println("Klik Pertama");  ld.klik();  System.out.println("Klik Kedua");  ld.klik();  System.out.println("Klik Ketiga");  ld.klik();  System.out.println("\n\nLampu Tiga State");  LampuTigaState lt = new LampuTigaState();  System.out.println("Klik Pertama");  lt.klik();  System.out.println("Klik Kedua");  lt.klik();  System.out.println("Klik Ketiga");  lt.klik();  System.out.println("Klik Keempat");  lt.klik();  }    } |   Output:    Kodekanlah kasus di atas. Jika tidak mengikuti contoh class diagram, silakan cantumkan class diagram menurut rancangan masing-masing. |

Solusi:

|  |
| --- |
| public interface Lampu {  public static final int MATI = 0;  public static final int REDUP = 5;  public static final int NYALA = 10;  public void klik();  } |

|  |
| --- |
| public class LampuDuaState implements Lampu {  private int nyala = 0;  @Override  public void klik() {  if (nyala == MATI) {  System.out.println("Kondisi Awal: LAMPU MATI");  nyala = NYALA;  System.out.println("Kondisi Akhir: LAMPU MENYALA");  }else{  System.out.println("Kondisi Awal: LAMPU MENYALA");  nyala = MATI;  System.out.println("Kondisi Akhir: LAMPU MATI");  }  }  } |

|  |
| --- |
| public class LampuTigaState implements Lampu {  private int nyala = 0;  @Override  public void klik() {  if (nyala == MATI) {  System.out.println("Kondisi Awal: LAMPU MATI");  nyala = REDUP;  System.out.println("Kondisi Akhir: LAMPU MENYALA REDUP");  } else if (nyala == REDUP) {  System.out.println("Kondisi Awal: LAMPU MENYALA REDUP");  nyala = NYALA;  System.out.println("Kondisi Akhir: LAMPU MENYALA");  } else{  System.out.println("Kondisi Awal: LAMPU MENYALA");  nyala = MATI;  System.out.println("Kondisi Akhir: LAMPU MATI");  }  }  } |