

Tugas Pendahuluan: Modul 8

NIM: 105223002

Nama: Gerald Eberhard

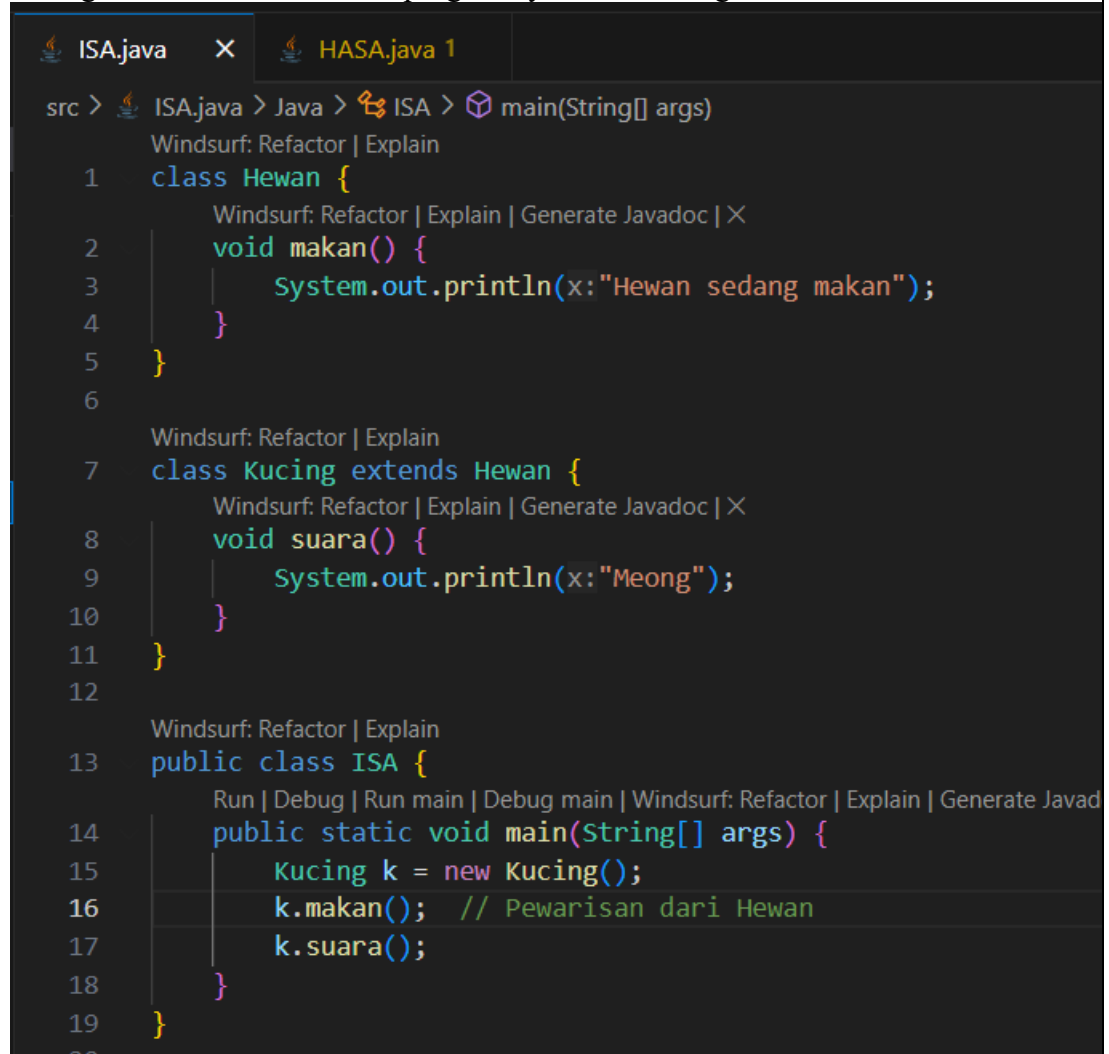
Instruksi: Jawablah pertanyaan-pertanyaan berikut dengan jelas dan rinci. Gunakan contoh kode jika diperlukan untuk memperjelas jawaban Anda. Kumpulkan dengan format **TP8_NIM_PrakPBO.pdf**. Deadline, Kamis 15 Mei 2025, pukul 13:00 (sebelum praktikum). Link pengumpulan: <https://forms.gle/AcPBWgx7is21tJUi7>

Soal

1. Jelaskan konsep IS-A relationship dalam Java! Berikan contoh implementasinya dalam bentuk kode program sederhana. Apa manfaat dari penggunaan hubungan ini dalam pemrograman berorientasi objek?

Jawab:

Konsep IS – A ini pada dasarnya ya menandakan bahwa kelas ini mewarisi sifat-sifat dari kelas lain. Contohnya jika kucing mewarisi sifat dari kelas Hewan maka kucing IS – A Hewan. Contoh programnya adalah sebagai berikut:



```
src > ISA.java > Java > ISA > main(String[] args)
Windsurf: Refactor | Explain
1 class Hewan {
2     void makan() {
3         System.out.println(x:"Hewan sedang makan");
4     }
5 }
6
Windsurf: Refactor | Explain
7 class Kucing extends Hewan {
8     void suara() {
9         System.out.println(x:"Meong");
10    }
11 }
12
Windsurf: Refactor | Explain
13 public class ISA {
14     public static void main(String[] args) {
15         Kucing k = new Kucing();
16         k.makan(); // Pewarisan dari Hewan
17         k.suara();
18     }
19 }
```

Manfaat dari menggunakan konsep ini adalah codenya dapat digunakan kembali dan mengurangi perulangan kode.

2. Apa yang dimaksud dengan Has-A relationship dalam Java? Jelaskan dengan contoh kode dan bandingkan perbedaannya dengan IS-A relationship!

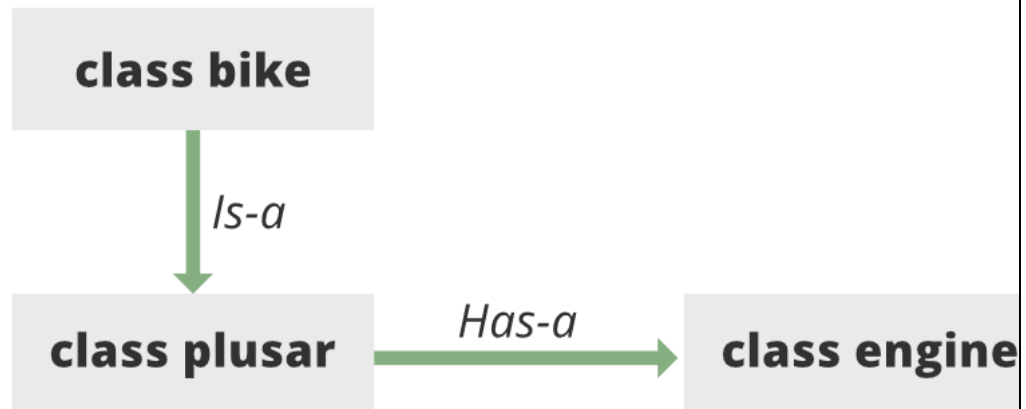
Jawab:

Kalo tadi IS – A bicara tentang adalah apa. Kalo sekarang Has – A. Kita akan bicara tentang punya apa. contohnya adalah Mobil mempunyai Mesin. Jadi konsep Has – A ini melambangkan kepemilikan. Biasanya Has – A ini bisa berupa hubungan komposisi atau agregasi. Berikut ini contoh codingannya:

```
src > ISA.java > HASA.java 1 X
Windsurf: Refactor | Explain
1 class Mesin {
  Windsurf: Refactor | Explain | Generate Javadoc | X
2   void nyalakan() {
3     System.out.println(x:"Mesin menyala");
4   }
5 }
6
Windsurf: Refactor | Explain
7 class Mobil {
8   Mesin mesin = new Mesin(); // Mobil HAS-A Mesin
9
10  void hidupkanMobil() {
11    mesin.nyalakan();
12  }
13 }
14
Windsurf: Refactor | Explain
15 class HASA {
  Run | Debug | Run main | Debug main | Windsurf: Refactor | Explain | Generate Java
16  public static void main(String[] args) {
17    Mobil mobil = new Mobil();
18    mobil.hidupkanMobil();
19  }
20 }
```

Jadi sekilas saja yang paling kelihatan adalah IS – A itu pasti berbicara tentang konsep Inheritance atau pewarisan tapi kalo konsep HAS – A itu berbicara tentang hubungan antara 2 kelas tetapi bukan merupakan pewarisan. Mungkin akan lebih

mudah dipahami melalui gambar berikut ini:



Nah nampaknya sudah cukup jelas ya. Jadi IS-A itu hubungan pewarisan sifat. Sedangkan HAS-A itu hubungan antar kelas yang tidak saling mewariskan sifat. Hubungannya bisa dalam bentuk komposisi, agregasi, dan asosiasi lainnya.

3. Kata kunci instanceof sering digunakan dalam Java. Jelaskan fungsi dari keyword ini dan berikan contoh penggunaannya! Dalam situasi seperti apa instanceof sebaiknya digunakan?

Jawab:

Kata kunci instance of adalah kata kunci untuk memastikan apakah suatu objek adalah instansi dari kelas tertentu:

```
src > Instanceof.java > Java > Instanceof > main(String[] args)
1 class Hewan {}
2 class Kucing extends Hewan {}
3 class Benda {}
4
5 public class Instanceof {
6     public static void main(String[] args) {
7         Hewan h = new Kucing();
8         System.out.println(h instanceof Kucing); // true
9         System.out.println(h instanceof Hewan); // true
10    }
11 }
12
```

```
PS C:\Users\Gerald Eberhard\Documents\sekolah\Uper\semester 4\PBO\PRAKTIKUM\wee
'-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'C:\Users\Gerald Eberhard\Do
true
true
PS C:\Users\Gerald Eberhard\Documents\sekolah\Uper\semester 4\PBO\PRAKTIKUM\wee
```

Nah, instanceof ini akan digunakan untuk mengecek terkait. Saat menggunakan pewarisan dan kita perlu memastikan tipe objek sebenarnya sebelum melakukan

casting. Atau dalam implementasi polimorfisme, seperti saat menangani banyak subclass dengan satu referensi superclass.

4. Java mendukung beberapa tipe pewarisan. Jelaskan tipe-tipe pewarisan dalam Java dan berikan contoh kode untuk masing-masingnya! Sertakan penjelasan tentang pewarisan tunggal, pewarisan multilevel, dan pewarisan hierarkis. Mengapa Java tidak mendukung pewarisan berganda secara langsung?

Jawab:

Ada beberapa tipe pewarisan diantaranya berikut:

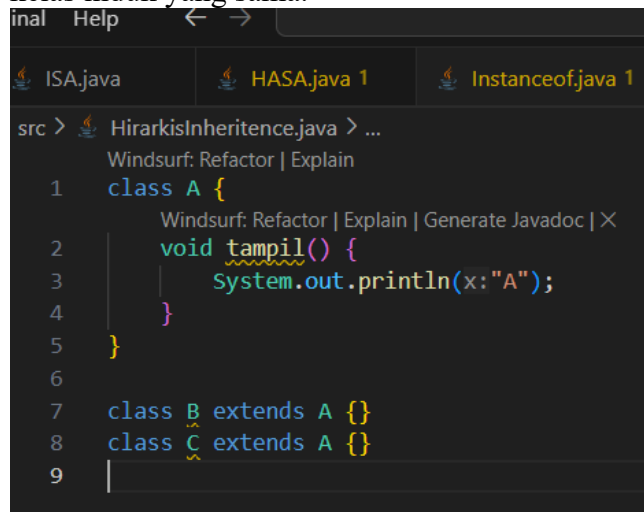
Pewarisan Tunggal (Single Inheritance): Dalam pewarisan ini, sebuah kelas hanya mewarisi dari satu kelas induk.

```
SingleInheritance.java > Java > B
Windsurf: Refactor | Explain
1 class A {
2     void tampil() {
3         System.out.println(x:"A");
4     }
5 }
6
7 class B extends A {
8     void tampilB() {
9         System.out.println(x:"B");
10    }
11 }
12
```

Pewarisan Multilevel: Dalam tipe ini, sebuah kelas mewarisi dari kelas lain, dan kelas tersebut juga menjadi induk bagi kelas berikutnya.

```
src > MultipleInheritance.java > ...
Windsurf: Refactor | Explain
1 class A {
2     void tampil() {
3         System.out.println(x:"A");
4     }
5 }
6
7 class B extends A {
8     void tampilB() {
9         System.out.println(x:"B");
10    }
11 }
12
13 class C extends B {
14     void tampilC() {
15         System.out.println(x:"C");
16     }
17 }
18
```

Pewarisan Hierarkis: Dalam pewarisan ini, beberapa kelas anak mewarisi dari satu kelas induk yang sama.



```
src > HirarkisInheritance.java > ...  
Windsurf: Refactor | Explain  
1 class A {  
2     void tampil() {  
3         System.out.println(x:"A");  
4     }  
5 }  
6  
7 class B extends A {}  
8 class C extends A {}  
9
```

Java tidak mengizinkan pewarisan ganda untuk menghindari diamond problem, yaitu konflik jika dua superclass memiliki method dengan nama yang sama. Solusinya adalah dengan menggunakan interface.

5. Jelaskan perbedaan antara agregasi dan komposisi dalam konteks relasi antar objek dalam Java! Berikan contoh implementasi kode untuk masing-masing relasi tersebut. Kapan sebaiknya menggunakan agregasi dan kapan menggunakan komposisi?

Jawab:

Agregasi itu memiliki hubungan yang lemah, objek tanpa induk, dan ditandai referensi biasa. Sedangkan Komposisi hubungannya kuat, objek bergantung pada induknya, Biasanya diinisialisasi dalam konstruktor.

Agregasi:

1 Instanceof.java 1 SingleInheritance.java 3 MultipleInheritance.java 3

src > Agregasi.java > Java > Dosen > Dosen(String nama)

Windsurf: Refactor | Explain

```
1 class Dosen {
2     String nama;
3     Dosen(String nama) {
4         this.nama = nama;
5     }
6 }
7
8 Windsurf: Refactor | Explain
9 class Universitas {
10     Dosen dosen;
11     Universitas(Dosen dosen) {
12         this.dosen = dosen;
13     }
14 }
15
16 Windsurf: Refactor | Explain
17 public class Agregasi {
18     Run | Debug | Run main | Debug main | Windsurf: Refactor | Explain | Generate Java
19     public static void main(String[] args) {
20         Dosen dosen = new Dosen(nama:"Budi");
21         Universitas universitas = new Universitas(dosen);
22     }
23 }
```

Komposisi:

```
terminal Help ← → Pretest
- java 1 Instanceof.java 1 SingleInheritance.java 3 MultipleInheritance
src > Komposisi.java > Java > Komposisi
Windsurf: Refactor | Explain
1 class Jantung {
2     void detak() {
3         System.out.println(x:"Jantung berdetak...");
4     }
5 }
6
Windsurf: Refactor | Explain
7 class Manusia {
8     private Jantung jantung;
9
10    Manusia() {
11        jantung = new Jantung(); // Komposisi
12    }
13
14    void hidup() {
15        jantung.detak();
16    }
17 }
18
Windsurf: Refactor | Explain
19 class Komposisi {
20     public static void main(String[] args) {
21         Manusia manusia = new Manusia();
22         manusia.hidup();
23     }
24 }
```

Jadi kapan akan kita gunakan?

Gunakan agregasi jika objek bisa hidup mandiri (misalnya, dosen bisa berpindah universitas). dan Gunakan komposisi jika objek tidak bisa berdiri sendiri tanpa objek induknya (misalnya, jantung tidak bisa hidup tanpa manusia).

Referensi

- [1]. [Relasi Antar Kelas – Agregasi – Rahmadya Trias Handayanto](#)
- [2]. [What is Is-A-Relationship in Java? | GeeksforGeeks](#)
- [3]. [Operator instanceof di Java](#)
- [4]. [Tutorial OOP Java: Cara Membuat Pewarisan Class \(Inheritance\)](#)
- [5]. [Perbedaan Asosiasi, Agregasi dan Komposisi pada Object Oriented Programming – AdnanSetiawan's Blog](#)