



Nama	:	MAULIA BALQIS ANSYA AULIA
Nim	:	2241720246
Kelas	:	TI – 2I

LEMBAR JAWABAN JOBSHEET-1

Percobaan 1 – Bentuk dasar polimorfisme	
Pertanyaan 4.2	
1	Class yang merupakan turunan dari class Employee adalah: <ul style="list-style-type: none">• PermanentEmployee• InternshipEmployee
2	Class yang implements ke interface Payable adalah: <ul style="list-style-type: none">• PermanentEmployee• ElectricityBill
3	Pada baris ke-10 dan 11 dalam class Tester1 , e dapat diisi dengan objek pEmp (objek dari class PermanentEmployee) dan objek iEmp (objek dari class InternshipEmployee) karena konsep polymorphism. Kedua class tersebut adalah turunan dari class Employee dan implement ke interface Payable , sehingga objek dari class turunan dapat diassign ke variabel yang bertipe class induknya.
4	Pada baris ke-12 dan 13 dalam class Tester1 , p dapat diisi dengan objek pEmp (objek dari class PermanentEmployee) dan objek eBill (objek dari class ElectricityBill) karena keduanya mengimplementasikan interface Payable . Dalam konsep polymorphism, objek dari class yang mengimplementasikan interface dapat diassign ke variabel yang bertipe interface tersebut.
5	Jika ditambahkan sintaks p = iEmp; pada baris 14 dan e = eBill; pada baris 15, akan menyebabkan error. Hal ini karena iEmp bukanlah objek dari class yang



Jurusan Teknologi Informasi Politeknik Negeri Malang

Jobsheet-12 : Polimorfisme

Mata Kuliah Object Oriented Programming

Pengampu: Tim Ajar Object Oriented Programming

Desember 2023

	mengimplementasikan interface Payable , sedangkan eBill mengimplementasikan interface Payable .
6	<p>Kesimpulan tentang konsep/bentuk dasar polimorfisme:</p> <ul style="list-style-type: none">• Polimorfisme memungkinkan objek dari class turunan dapat diakses melalui variabel bertipe class induknya.• Interface dapat digunakan sebagai tipe data, dan objek dari class yang mengimplementasikan interface dapat diakses melalui variabel bertipe interface.• Variabel bertipe class atau interface dapat diisi dengan objek dari class yang mengimplementasikan interface atau turunan dari class tersebut.
Percobaan 2 – Virtual method invocation	
Pertanyaan 5.2	
1	<p>Pemanggilan e.getEmployeeInfo() pada baris 8 dan pEmp.getEmployeeInfo() pada baris 10 menghasilkan hasil yang sama karena e merujuk pada objek pEmp. Meskipun variabel e dideklarasikan sebagai tipe Employee, JVM mengetahui bahwa sebenarnya ia merujuk pada objek dari subclass-nya, yaitu PermanentEmployee. Sehingga, saat memanggil metode getEmployeeInfo(), JVM mengeksekusi versi metode yang ada pada objek aktualnya, bukan tipe deklarasinya.</p>
2	<p>Pemanggilan method e.getEmployeeInfo() disebut sebagai pemanggilan method virtual karena metode yang dipanggil ditentukan pada saat runtime berdasarkan objek aktual yang ditunjuk oleh referensi. Dalam kasus ini, metode yang dipanggil adalah metode dari PermanentEmployee karena objek aktualnya adalah instance dari PermanentEmployee. Sebaliknya, pEmp.getEmployeeInfo() juga disebut sebagai pemanggilan method virtual karena dalam bahasa Java, semua pemanggilan metode pada objek adalah virtual secara default.</p>



3	Virtual method invocation berarti pemanggilan metode yang ditentukan pada saat runtime berdasarkan objek aktual yang ditunjuk oleh referensi, bukan tipe deklarasinya. Disebut "virtual" karena pemilihan metode terjadi secara dinamis pada runtime, yang berbeda dengan metode statis yang ditentukan pada saat kompilasi. Dalam pemrograman berorientasi objek, konsep ini memungkinkan polymorphism, di mana suatu objek dapat memiliki banyak bentuk (tipe), dan pemilihan bentuk (metode) terjadi secara dinamis sesuai dengan objek aktualnya.
<p align="center">Percobaan 3 – Heterogenous Collection</p> <p align="center">Pertanyaan 6.2</p>	
1	Array e pada baris ke-8 dapat diisi dengan objek-objek tipe yang berbeda karena keduanya adalah turunan dari kelas Employee . Dalam pemrograman berorientasi objek, sebuah objek dari subclass dapat dianggap sebagai objek dari superclass, sehingga dapat dimasukkan ke dalam array atau variabel yang bertipe superclass.
2	Array p pada baris ke-9 juga dapat diisi dengan objek-objek tipe yang berbeda karena keduanya mengimplementasikan interface Payable . Dalam pemrograman berorientasi objek, objek dari kelas yang mengimplementasikan suatu interface dapat dianggap sebagai objek dari interface tersebut.
3	Pada baris ke-10, terjadi error karena array e2 adalah array dari tipe Employee , dan ElectricityBill adalah turunan dari Payable , bukan Employee .
<p align="center">Percobaan 4 – Argumen polimorfisme, instanceof dan casting objek</p> <p align="center">Pertanyaan 7.2</p>	
1	Pemanggilan ow.pay(eBill) dan ow.pay(pEmp) dapat dilakukan karena objek eBill dan pEmp sebenarnya adalah turunan dari interface Payable . Keduanya mengimplementasikan interface Payable , sehingga dapat dianggap sebagai objek dengan tipe Payable . Polimorfisme memungkinkan objek yang memiliki tipe turunan dari suatu interface dapat digunakan pada method yang membutuhkan objek dengan tipe tersebut.



2	<p>Tujuan membuat argument bertipe Payable pada method pay() di dalam class Owner adalah untuk memberikan fleksibilitas dan kemampuan polimorfisme (menerima object dari class ElectricityBill dan PermanenetEmployee)</p> <p>. Dengan menggunakan tipe Payable sebagai parameter, method tersebut dapat menerima objek dari kelas mana pun yang mengimplementasikan interface Payable, memungkinkan pemanggilan dengan objek berbagai jenis.</p>
3	<p>Jika ditambahkan perintah ow.pay(iEmp); pada baris terakhir method main(), akan terjadi error karena iEmp adalah objek dari kelas InternshipEmployee, yang tidak mengimplementasikan interface Payable. Sebagai hasilnya, objek tersebut tidak dapat digunakan sebagai argument untuk method pay() yang membutuhkan objek dengan tipe Payable.</p>
4	<p>Sintaks p instanceof ElectricityBill pada baris ke-6 digunakan untuk memeriksa apakah objek yang diterima oleh method pay() adalah instance dari kelas ElectricityBill atau kelas turunannya. Ini digunakan untuk menentukan tindakan yang sesuai dalam method, seperti melakukan casting objek dan menampilkan informasi spesifik.</p>
5	<p>Casting objek (ElectricityBill) p pada baris ke-7 diperlukan karena pada saat kompilasi, tipe objek p diperlakukan sebagai tipe Payable (tipe parameter). Namun, karena kita tahu bahwa objek tersebut sebenarnya merupakan objek dari kelas ElectricityBill, kita melakukan casting untuk mengubah tipe objeknya sehingga dapat diakses dan digunakan sesuai dengan tipe aslinya.</p>
Assignment	
<pre><<interface>> Destroyable</pre>	<pre>Tugas > J Destroyable.java > ... 1 package Tugas; 2 public interface Destroyable { 3 void destroyed(); 4 }</pre>



Jurusan Teknologi Informasi Politeknik Negeri Malang

Jobsheet-12 : Polimorfisme

Mata Kuliah Object Oriented Programming

Pengampu: Tim Ajar Object Oriented Programming

Desember 2023

Barrier

```
Tugas > J Barrier.java > ...
1  package Tugas;
2  public class Barrier implements Destroyable {
3      private int health;
4
5      public Barrier(int strength) {
6          this.health = strength;
7      }
8
9      public void setStrength(int strength){
10         this.health = strength;
11     }
12
13     public int getStrength() {
14         return health;
15     }
16
17     public void destroyed() {
18         health -= 0.1 * health;
19     }
20
21     public String getBarrierInfo() {
22         return "Barrier Strength = " + health + " ";
23     }
24 }
```



Jurusan Teknologi Informasi Politeknik Negeri Malang

Jobsheet-12 : Polimorfisme

Mata Kuliah Object Oriented Programming

Pengampu: Tim Ajar Object Oriented Programming

Desember 2023

Zombie

Tugas > **J** Zombie.java > ...

```
1  package Tugas;
2  public class Zombie implements Destroyable {
3      protected int health;
4      protected int level;
5
6      public Zombie(int health, int level) {
7          this.health = health;
8          this.level = level;
9      }
10
11     public void heal(){
12     }
13
14     public void destroyed(){
15     }
16
17     public String getZombieInfo() {
18         return "\nHealth = " + health + "\nLevel = " + level;
19     }
20 }
```



Jurusan Teknologi Informasi Politeknik Negeri Malang

Jobsheet-12 : Polimorfisme

Mata Kuliah Object Oriented Programming

Pengampu: Tim Ajar Object Oriented Programming

Desember 2023

Walking
Zombie

```
Tugas > J WalkingZombie.java > ...
1  package Tugas;
2  public class WalkingZombie extends Zombie {
3      public WalkingZombie(int health, int level) {
4          super(health, level);
5      }
6
7      public void heal() {
8          if (level == 1) {
9              health += 0.1 * health;
10         } else if (level == 2) {
11             health += 0.2 * health;
12         } else if (level == 3) {
13             health += 0.3 * health;
14         }
15     }
16
17     public void destroyed() {
18         health -= 0.15 * health;
19     }
20
21     public String getZombieInfo() {
22         return "Walking Zombie Data = " + super.getZombieInfo();
23     }
24 }
```



Jurusan Teknologi Informasi Politeknik Negeri Malang

Jobsheet-12 : Polimorfisme

Mata Kuliah Object Oriented Programming

Pengampu: Tim Ajar Object Oriented Programming

Desember 2023

Jumping Zombie	<pre>Tugas > J JumpingZombie.java > ... 1 package Tugas; 2 public class JumpingZombie extends Zombie { 3 public JumpingZombie(int health, int level) { 4 super(health, level); 5 } 6 7 public void heal() { 8 if (level == 1) { 9 health += 0.3 * health; 10 } else if (level == 2) { 11 health += 0.4 * health; 12 } else if (level == 3) { 13 health += 0.5 * health; 14 } 15 } 16 17 public void destroyed() { 18 health -= 0.01 * health; 19 } 20 21 public String getZombieInfo() { 22 return "Jumping Zombie Data = " + super.getZombieInfo(); 23 } 24 }</pre>
Plant	<pre>Tugas > J Plant.java > ... 1 package Tugas; 2 public class Plant { 3 public void doDestroy(Destroyable d) { 4 d.destroyed(); 5 } 6 }</pre>



Jurusan Teknologi Informasi Politeknik Negeri Malang


Jobsheet-12 : Polimorfisme

Mata Kuliah Object Oriented Programming

Pengampu: Tim Ajar Object Oriented Programming

Desember 2023

Tester

Tugas >  Tester.java > ...

```
1 package Tugas;
2 public class Tester {
    Run | Debug
3     public static void main(String[] args) {
4         WalkingZombie wz = new WalkingZombie(health:100, level:1);
5         JumpingZombie jz = new JumpingZombie(health:100, level:2);
6         Barrier b = new Barrier(strength:100);
7         Plant p = new Plant();
8
9         System.out.println(" " + wz.getZombieInfo());
10        System.out.println(" " + jz.getZombieInfo());
11        System.out.println(" " + b.getBarrierInfo());
12        System.out.println(x:"-----");
13
14        for (int i = 0; i < 4; i++) { //Destroy the enemies 4 times
15            p.doDestroy(wz);
16            p.doDestroy(jz);
17            p.doDestroy(b);
18        }
19
20        System.out.println(" " + wz.getZombieInfo());
21        System.out.println(" " + jz.getZombieInfo());
22        System.out.println(" " + b.getBarrierInfo());
23    }
24 }
```



Jurusan Teknologi Informasi Politeknik Negeri Malang

Jobsheet-12 : Polimorfisme

Mata Kuliah Object Oriented Programming

Pengampu: Tim Ajar Object Oriented Programming

Desember 2023

Output	<pre>86d6c5eea35db914139a9c36470a Walking Zombie Data= Health = 100 Level = 1 Jumping Zombie Data = Health = 100 Level = 2 Barrier Strength = 100 ----- Walking Zombie Data= Health = 51 Level = 1 Jumping Zombie Data = Health = 96 Level = 2 Barrier Strength = 64 PS C:\OOP\Week12> </pre>
--------	---