Nama	:	MILA YUNITA
Nim	:	2041720027
Kelas	:	TI-2C
Absen	:	12

A. Percobaan 1 - Bentuk dasar polimorfisme

Pertanyaan

1. Class apa sajakah yang merupakan turunan dari class Employee?

   Jawab:

   Class yang merupakan turunan Class Employee yakni Class InternshipEmployee dan Class PermanentEmployee.

2. Class apa sajakah yang implements ke interface Payable?

   Jawab:

   Class yang implement ke interface Payable yakni Class PermanentEmployee dan Class ElectrycityBill.

3. Perhatikan class Tester1, baris ke-10 dan 11. Mengapa e, bisa diisi dengan objek pEmp (merupakan objek dari class PermanentEmployee) dan objek iEmp (merupakan objek dari class InternshipEmploye) ?

   Jawab:

   Variabel e (bertipe data references Employee) bisa diisi dengan objek pEmp(objek class PermanentEmployee) dan objek iEmp(objek class InternshipEmployee) karena Class PermanentEmployee dan Class

InternshipEmployee merupakan child dari class Employee. Sehingga atribut yang bertipe data Employee dapat diisi dengan objek dari class turunannya(Class PermanentEmployee dan Class InternshipEmployee).

4. Perhatikan class Tester1, baris ke-12 dan 13. Mengapa p, bisa diisi dengan objek pEmp (merupakan objek dari class PermanentEmployee) dan objek eBill (merupakan objek dari class ElectricityBill) ?

   Jawab:

   Variabel p dapat diisi dengan objek pEmp (merupakan objek dari class PermanentEmployee) dan objek eBill (merupakan objek dari class ElectricityBill) karena p merupakan variabel yang bertipe data references Payable yang mana class PermanentEmployee dan class ElectricityBill merupakan class yang implements ke class Payable. Sehingga objek dari class PermanentEmployee dan class ElectricityBill dapat diisikan ke varibel yang bertipe data references Payable.

5. Coba tambahkan sintaks: p = iEmp; e = eBill; pada baris 14 dan 15 (baris terakhir dalam method main) ! Apa yang menyebabkan error?

   Jawab:

   Yang menyebabkan error adalah class IntershipEmployee(yang memiliki objek iEmp) tidak implements ke class Payable sehingga objek iEmp tidak dapat diisikan ke variabel p yang bertipe data references Payable. Kemudian objek eBill juga tidak dapat diisikan ke variabel e yang bertipe data references Employee karena class ElectricityBill(yang memiliki objek eBill) bukan subclass atau *child* dari class Employee. Di bawah ini hasil tangkapan layar kode program yang ditambahkan sintaks: p = iEmp; e = eBill; pada baris 14 dan 15.

```
12  public class Tester1 {
13      public static void main(String[] args) {
14          PermanentEmployee pEmp = new PermanentEmployee("Dedik", 500);
15          InternshipEmployee iEmp = new InternshipEmployee("Sunarto", 5
16          ElectricityBill eBill = new ElectricityBill(5, "A-1");
17
18          Employee e;
19          Payable p;
20          e = pEmp;
21          e = iEmp;
22          p = pEmp;
23          p = eBill;
            p = iEmp;
            e = eBill;
26      }
27  }
```

6. Ambil kesimpulan tentang konsep/bentuk dasar polimorfisme!

Jawab:

Kesimpulan tentang konsep bentuk dasar polimorfisme yakni polimorfisme dapat diterapkan pada relasi inheritance dan juga interface. Dimana jika suatu variabel ataupun objek dideklarasikan dari sebuah superclass(hubungan inheritance), maka objek tersebut dapat diberi isian objek milik subclassnya. Kemudian jika suatu variabel ataupun objek dideklarasikan dari sebuah interface class, maka objek tersebut dapat diberi isian objek milik class yang implements ke class interface tersebut.

B. Percobaan 2 - Virtual method invocation

Pertanyaan

1. Perhatikan class Tester2 di atas, mengapa pemanggilan e.getEmployeeInfo() pada baris 8 dan pEmp.getEmployeeInfo() pada baris 10 menghasilkan hasil sama?

   Jawab:

   Pemanggilan e.getEmployeeInfo() pada baris 8 dan pEmp.getEmployeeInfo() pada baris 10 menghasilkan hasil sama karena pada kode program baris ke-7 objek polimorfisme e telah disikan objek milik Class PermanentEmployee sehingga JVM akan menjalankan method yang ada di class PermanentEmployee walaupun compiler mengenali method getEmployeeInfo() sebagai method milik class Employee. Kemudian pemanggilan pEmp.getEmployeeInfo() pada baris 10 juga akan memanggil method getEmployeeInfo() yang ada di class PermanentEmployee. Sehingga kedua baris program tersebut akan menghasilkan output yang sama karena memanggil method dari class yang sama.

2. Mengapa pemanggilan method e.getEmployeeInfo() disebut sebagai pemanggilan method virtual (virtual method invication), sedangkan pEmp.getEmployeeInfo() tidak?

   Jawab:

   Pemanggilan method e.getEmployeeInfo() disebut sebagai pemanggilan method virtual karena pada kode tersebut terjadi pemanggilan method overriding dari objek polimorfisme. Objek polimorfisme yang dimaksud adalah objek e. Dimana saat kode program tersebut dijalankan, compiler akan mengenali method getEmployeeInfo() yang ada di class Employee, tapi yang JVM justru menjalankan method getEmployeeInfo() yang ada di class PermanentEmployee. Sedangkan pEmp.getEmployeeInfo() tidak disebut sebagai pemanggilan method

virtual karena yang dikenali compiler dan yang dijalankan oleh JVM sama-sama method getEmployeeInfo() yang ada di class PermanentEmployee.

3. Jadi apakah yang dimaksud dari virtual method invocation? Mengapa disebut virtual?

   Jawab:

   Virtual method invocation terjadi ketika ada pemanggilan overriding method dari suatu objek polimorfisme. Disebut virtual karena antara method yang dikenali oleh compiler dan method yang dijalankan oleh JVM berbeda.

Percobaan 3 - Heterogenous Collection

Pertanyaan

1. Perhatikan array e pada baris ke-8, mengapa ia bisa diisi dengan objek-objek dengan tipe yang berbeda, yaitu objek pEmp (objek dari PermanentEmployee) dan objek iEmp (objek dari InternshipEmployee) ?

   Jawab:

   Array e dapat diisi dengan objek pEmp (objek dari class PermanentEmployee) dan objek iEmp (objek dari InternshipEmployee) karena array e dideklarasikan dari class Employee yang merupakan superclass dari class PermanentEmployee dan class InternshipEmployee.

2. Perhatikan juga baris ke-9, mengapa array p juga bisa diiisi dengan objek-objek dengan tipe yang berbeda, yaitu objek pEmp (objek dari PermanentEmployee) dan objek eBill (objek dari ElectricityBilling) ?

   Jawab:

   Array p dapat diisi dengan objek pEmp (objek dari PermanentEmployee) dan objek eBill (objek dari ElectricityBilling) karena array p dideklarasikan dari class Payable yang merupakan class interface yang diimplementasikan oleh class PermanentEmployee dan class ElectricityBilling.

3. Perhatikan baris ke-10, mengapa terjadi error?

   Jawab:

   Pada baris ke-10 terjadi error karena array e2 dideklarasikan dari class Employee yang merupakan superclass dari class PermanentEmployee(memiliki objek pEmp) dan class IntershipEmployee(memiliki objek iEmp). Sedangkan objek eBill merupakan objek milik classElectricityBill yang bukan subclass dari class Employee.

C. Percobaan 4 - Argumen polimorfisme, instanceof dan casting objek

Pertanyaan

1. Perhatikan class Tester4 baris ke-7 dan baris ke-11, mengapa pemanggilan ow.pay(eBill) dan ow.pay(pEmp) bisa dilakukan, padahal jika diperhatikan method pay() yang ada di dalam class Owner memiliki argument/parameter bertipe Payable? Jika diperhatikan lebih detil eBill merupakan objek dari ElectricityBill dan pEmp merupakan objek dari PermanentEmployee?

   Jawab:

   Pemanggilan ow.pay(eBill) dan ow.pay(pEmp) dapat dilakukan pada method pay() yang memiliki argument bertipe Payable karena eBill merupakan objek dari class ElectricityBill dan pEmp merupakan objek dari class PermanentEmployee yang mana kedua class tersebut merupakan class yang mengimplementasikan class interface Payable. Sehingga objek dari kedua class tersebut dapat diisikan pada parameter yang bertipe data Payable.

2. Jadi apakah tujuan membuat argument bertipe Payable pada method pay() yang ada di dalam class Owner?

   Jawab:

   Tujuan membuat argument bertipe Payable pada method pay() yang ada di dalam class Owner adalah agar method tersebut dapat menerima objek dari class yang membutuhkan proses pembayaran(class yang mengimplementasi class interface Payable), misal class PermanentEmployee yang terdapat gaji karyawan tetap yang harus dibayar dan class ElectricityBill yang terdapat tagihan pembayaran listrik.

3. Coba pada baris terakhir method main() yang ada di dalam class Tester4 ditambahkan perintah ow.pay(iEmp); Mengapa terjadi error?

   Jawab:

   Error tersebut terjadi karena objek iEmp merupakan objek milik class InternshipEmployee yang tidak mengimplentasikan class interface Payable. Sehingga objek tersebut tidak dapat diisikan ke parameter method yang bertipe data parameter Payable.

4. Perhatikan class Owner, diperlukan untuk apakah sintaks p instanceof ElectricityBill pada baris ke-6 ?

   Jawab:

   Sintaks p intanceof ElectricityBill digunakan untuk mengecek apakah objek p merupakan hasil instansiasi dari class ElectricityBill. Hasil dari sintaks tersebut berupa nilai boolean(true false).

5. Perhatikan kembali class Owner baris ke-7, untuk apakah casting objek disana (ElectricityBill eb = (ElectricityBill) p) diperlukan ? Mengapa objek p yang bertipe Payable harus di-casting ke dalam objek eb yang bertipe ElectricityBill ?

   Jawab:

   Casting objek pada kode program (ElectricityBill eb = (ElectricityBill) p) diperlukan untuk mengubah objek superclass (p) menjadi objek dari subclass(eb). Objek p yang bertipe data Payable harus di-casting ke dalam objek eb yang bertipe data Electricity karena diperlukan pengaksesan method atau behaviour yang lebih spesifik dari class ElectricityBill yakni untuk mengakses method getBillInfo() yang ada di class ElectricityBill .