NAMA : SILVIA PRADA APRILIA
NIM : 2041720141
KELAS : TI 2C
MATERI : POLIMORFISME

Percobaan 1

1. Class apa sajakah yang merupakan turunan dari class **Employee**?
   Jawab: Class yang merupakan turunan dari class employe adalah Class InternshipEmployee dan Class PermanentEmployee.

2. Class apa sajakah yang implements ke interface **Payable**?
   Jawab: Class yang implements ke Interface Payable adalah Class PermanentEployee dan ElectricityBill.

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 bisa diinisialisasi dengan objek pEmp dan iEmp karena variabel e bertipe Employee dinama Class Epmloyee adalah super class dari PermenentEmployee dan InternshipEmployee. Maka dari itu variabel e bisa diinisialisasikan dengan pEmp yang merupakan objek dari Class PermanentEmployee dan iEmp yang merupakan objek dari Class InternshipEmployee.

4. Perhatikan class **Tester1**, baris ke-12 dan 13. Mengapa **p**, bisa diisi denganobjek**pEmp** (merupakan objek dari class **PermanentEmployee**) dan objek **eBill** (merupakan objek dari class **ElectricityBill**) ?
   Jawab: Variabel p bisa diinisialisasikan dengan objek pEmp dan eBill karena variabel p bertipe Payable dimana Class PermanentEmployee dan ElectricityBill meng-implements Interface Payable. Maka dari itu variabel p bisa diinisialisasikan dengan pEmp yang merupakan objek dari Class PermanentEmployee dan eBill yang merupakan objek dari Class ElevtricityBill.

5. Coba tambahkan sintaks:
   p = iEmp;
   e = eBill;
   pada baris 14 dan 15 (baris terakhir dalam method **main**) ! Apa yang menyebabkan error?
   Jawab: Baris 14 error karena p bertipe Payable dan iEmp merupakan objek dari Class InternshipEmployee dimana InternshipEployee tidak implements interface Payable sehingga tidak ada hubungan antara Class InternshipEmployee dan interface Payable. Maka dari itu variabel p tidak bisa diinisialisasikan dengan objek iEmp. Sedangkan Baris 15 error karena e bertipe Employee dan eBill merupakan objek dari Class ElectricityBill dimana ElectricityBill bukan child class dari Class Employee. sehingga tidak ada hubungan antara Class ElectricityBill dan

Class Employee. Maka dari itu variabel e tidak bisa diinisialisasikan dengan objek eBill.

6. Ambil kesimpulan tentang konsep/bentuk dasar polimorfisme!
   Jawab: Dari hasil percobaan tersebut dapat diambil kesimpulan bahwa, dalam konsep polimorfisme variabel dengan tipe class super dapat diinisialisasikan dengan objek dari class child dan variabel yang bertipe interface dapat diinisialisasikan dengan objek dari class yang meng-implements interface tersebut.

Percobaan 2

1. Perhatikan class **Tester2** di atas, mengapa pemanggilan **e.getEmployeeInfo()** pada baris 8 dan **pEmp.getEmployeeInfo()** pada baris 10 menghasilkan hasil sama?
   Jawab: e.getEmployeeInfo() dan pEmp.getEmployeeInfo menghasilkan hasil yang sama karena variabel e diinisialisasikan dengan objek pEmp yang artinya saat dijalankan nantinya akan memanggi method getEmployeeInfo() dari Class PermanentEmployee.

2. Mengapa pemanggilan method **e.getEmployeeInfo()** disebut sebagai pemanggilan method virtual (virtual method invication), sedangkan **pEmp.getEmployeeInfo()** tidak?
   Jawab: Pemanggilan method e.getEmployeeInfo() dikatakan sebagai pemanggilan method virtual dikarenakan saat pemanggilan method e.getEmployeeInfo() compiler akan mengenali method e.getEmployeeInfo() pada class Employee karena variabel e bertipe Employee tetapi saat dijalankan justru method getEmployee() dari class PermanentEmployee karena variabel e diinisialisasi dengan objek pEmp dari class PermanentEmployee.

3. Jadi apakah yang dimaksud dari virtual method invocation? Mengapa disebut virtual?
   Jawab: Kesimpulan dari hasil percobaan yang telah dilakukan virtual method invocation adalah pemanggilan method overriding dari suatu objek polimorfisme yang diantara method yang dikenali oleh compiler dan method yang dijalankan berbeda.

Percobaan 3

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 bisa diisi dengan objek-objek dengan tipe yang berbeda karena adanya konsep polimorfisme sehingga array bisa dibuat heterogen. Array e bisa

diisi dengan objek pEmp dan iEmp karena pEmp dan iEmp merupakan objek yang dari class child Employee.

2. Perhatikan juga baris ke-9, mengapa array **p** juga biisi dengan objek-objek dengan tipe yang berbeda, yaitu objek **pEmp** (objek dari **PermanentEmployee**) dan objek **eBill** (objek dari **ElectricityBilling**) ?
   Jawab: Array p bisa diisi dengan objek-objek dengan tipe yang berbeda karena adanya konsep polimorfisme sehingga array bisa dibuat heterogen. Array e bisa diisi dengan objek pEmp dan eBill karena pEmp dan eBill merupakan objek yang dari class yang implements interface Payable.

3. Perhatikan baris ke-10, mengapa terjadi error?
   Jawab: Pada baris ke-10 terjadi error dikarenakan terdapat objek eBill dalam elemen array. Objek eBill tidak dapat dimasukkan ke array e2 karena array e2 bertipe Employee, sedangkan objek eBill bukan merupakan child class dari Employee, sehingga akan terjadi error jika tetap dimasukkan ke array e2.

Percobaan 4

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 karena objek eBill berasal dari class ElectricityBill dan objek pEmp berasal dari class PermanentEmploye dimana class tersebut adalah class implements dari interface Payable. Sehingga method pay tetap bisa dijalankan walau berparameter objek lain selain dari Payable namun masih merupakan class yang implements ke Payable.

2. Jadi apakah tujuan membuat argument bertipe **Payable** pada method **pay()** yang ada di dalam class **Owner**?
   Jawab: Tujuan dari membuat arguments bertipe Payable adalah agar method pay() dapat diakses oleh class child/ class implements dari interface Payable. Sehingga tidak perlu membuat method yang berbeda.

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

NAMA : SILVIA PRADA APRILIA
NIM : 2041720141
KELAS : TI 2C
MATERI : POLIMORFISME

```
3   public class Tester4 {
4       public static void main(String[] args) {
5           Owner ow = new Owner();
6           ElectricityBill eBill = new ElectricityBill(5, "R-1");
7           ow.pay(eBill);//pay for electricity bill
8           System.out.println("------------------------------");
9
10          PermanentEmployee pEmp = new PermanentEmployee("Dedik", 500);
11          ow.pay(pEmp);//pay for permanent employee
12          System.out.println("------------------------------");
13
14          InternshipEmployee iEmp = new InternshipEmployee("Sunarto", 5);
15          ow.showMyEmployee(pEmp);//show permanent employee info
16          System.out.println("------------------------------");
17          ow.showMyEmployee(iEmp);//show internship employee info
18
            ow.pay(iEmp);
20      }
21  }
```

Mengapa terjadi error?
Jawab: Setelah ditambahkan ow.pay(iEmp) terjadi error karena parameter method pay() bertipe Payable. Sedangkan iEmp bertipe IntershipEmployee yang tidak implements ke interface Payable dan tidak memiliki hubungan dengan interface Payable. Sehingga tidak bisa dijadikan parameter ke method pay() dan akhirnya terjadi error.

4. Perhatikan class **Owner**, diperlukan untuk apakah sintaks **p instanceof ElectricityBill** pada baris ke-6 ?
   Jawab: Sintaks p instance of ElectricityBill digunakan untuk mengecek apakah p diinisialisasikan dengan objek dari class ElectricityBill. Apabila benar makan akan mengembalikan nilai true dan apabila tidak maka akan mengembalikan nilai 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: Ya. Casting objek(ELectricityBill eb = (ELectricityBill) p) diperlukan karena objek eb bertipe ElectricityBill sedangkan p yang akan diinisialisasikan ke eb bertipe Payable sehingga diperlukan Downcasting untuk mengubah tipe data Payable ke ElectricityBill untuk menginisialisasikan objek eb.

NAMA : SILVIA PRADA APRILIA
NIM : 2041720141
KELAS : TI 2C
MATERI : POLIMORFISME

Tugas Praktikum