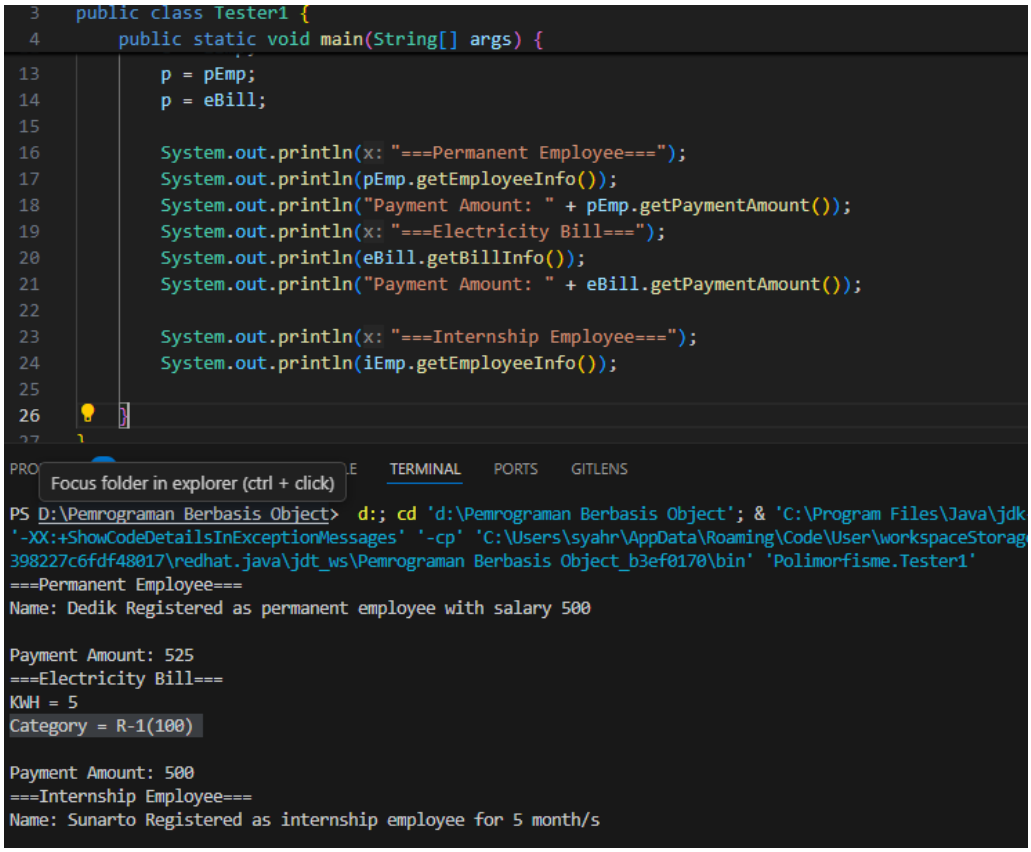
LAPORAN PRAKTIKUM
PEMROGRAMAN BERORIENTASI OBJECT
TUGAS JOBSHEET POLYMORPHISM

Sahrul Ramadhani
244107020058
2H

D-IV Teknik Informatika
Teknologi Informasi
Politeknik Negeri Malang
2025

PERCOBAAN 1

Berikut adalah hasil percobaan 1 setelah saya tambahkan panggil getInfo pada class main

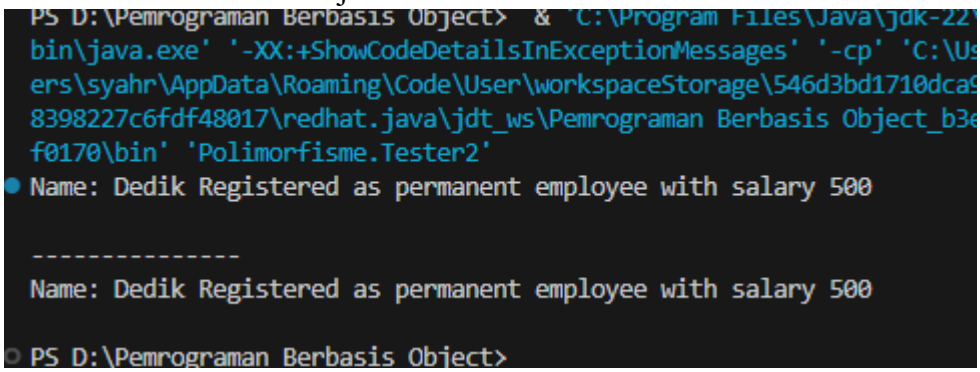


JAWABAN PERTANYAAN PERCOBAAN 1

1. Ada 2 class yang jadi turunan dari Employee yaitu PermanentEmployee dan InternshipEmployee
2. Ada 2 class yang implements payable yaitu PermanentEmployee dan ElectricityBill.
3. Karena e di deklarasikan dengan tipe Employee, di java tuh var bertipe superclasss boleh mereferensi object dari subclass manapun. Biasa di sebut dengan upcasting.
4. Karena p di deklarasikan bertipe Payable (iunterface) kedua object pEmp dan eBill sama sama mengimplementasikan payable, jadi keduanya valid jika bertipe Payable
5. p = iEmp; eror karena iEmpt bertipe InternshipEmployee dia tidak mengimplement ke interface payable.
   Sedangkan e = eBill; eror karena e bertipe employee, sedangkan eBill bertipe ElectricityBill yangbukan subclass dari Employee.
6. Konsep dasar polimorfisme adalah kemampuan sebuah variabel referensi untuk mereferensi berbagai bentuk objek selama objek tersebut masih berada dalam satu hubungan tipe, baik melalui inheritance maupun interface. Jadi mereka bisa tetap saling mereferensi dengan syarat mereka masih terhubung dengan sebagai super/sub class atau interface.

PERCOBAAN 2

Berikut hasil Run Tester2.java

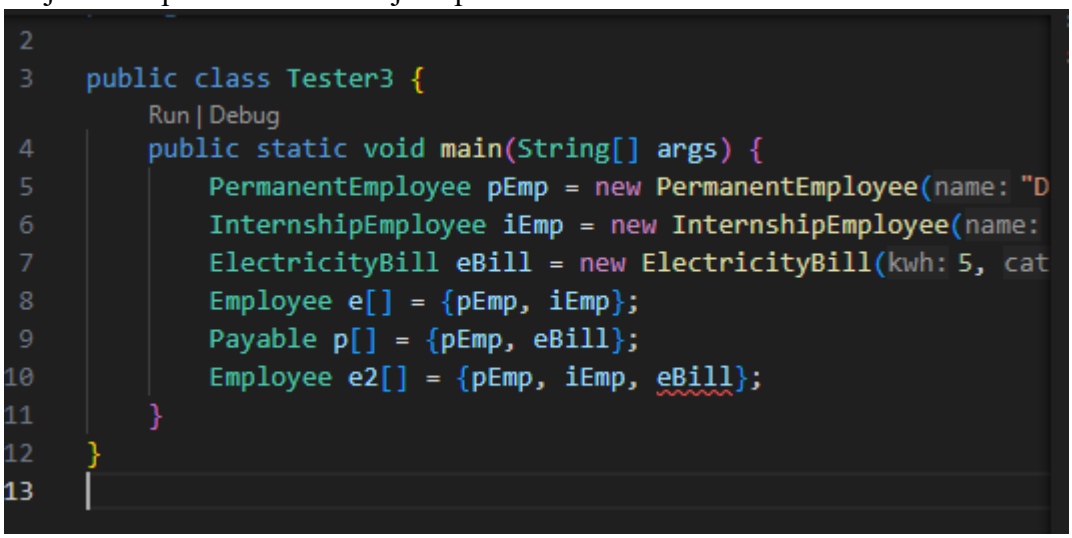


JAWABAN PERTANYAAN PERCOBAAN 2

1. Sama karena objek yang di isi kan sama. Yaitu objek pEmp dari class PermanentEmployee, sedangkan e adalah bertipe Employee (parent nya) dia juga di isi dengan objek pEmpt. jadi Java akan ekseskusi method milik class permanentEmployee karena objek asli adalah miliknya. Meskipun tipe reff nya beda, method yang di panggil tetap berasal dari objek yang sama. Jadi hasilnya sama.
   Maaf Agak mbulet ya penjelasan saya hehe.
2. e.getEmployeeInfo() disebut virtual method invocation karena tipe referensi e adalah Employee (superclass). Sedangkan pEmp.getEmployeeInfo() bukan virtual method invocation karena tipe referensi dan tipe objeknya sama-sama PermanentEmployee, sehingga sejak compile time sudah jelas method mana yang akan dipanggil. Tidak ada proses pemilihan method secara dinamis oleh JVM.
3. Virtual method invocation adalah mekanisme di mana method yang dijalankan ditentukan berdasarkan tipe objek sebenarnya saat runtime, bukan berdasarkan tipe variabel referensinya. Disebut virtual karena pada saat compile time method yang akan dipanggil belum ditentukan secara pasti. Compiler hanya memverifikasi bahwa method tersebut ada, tetapi JVM-lah yang memilih implementasi method mana yang benar-benar dijalankan ketika program berjalan.

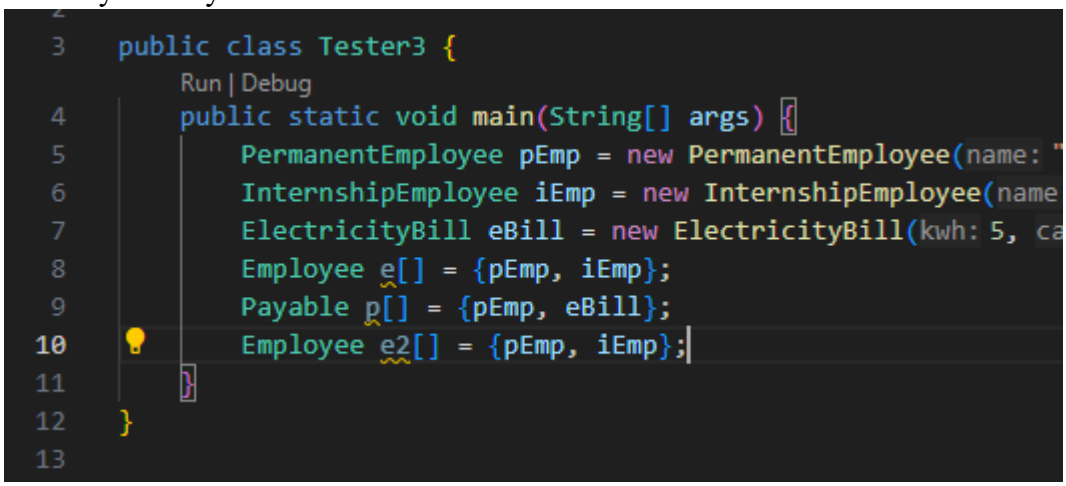
PERCOBAAN 3 – HETEROGENOUS COLLECTION

Terjadi eror pada kode tester3.java pada baris ke 10



JAWABAN PERTANYAAN PERCOBAAN 3

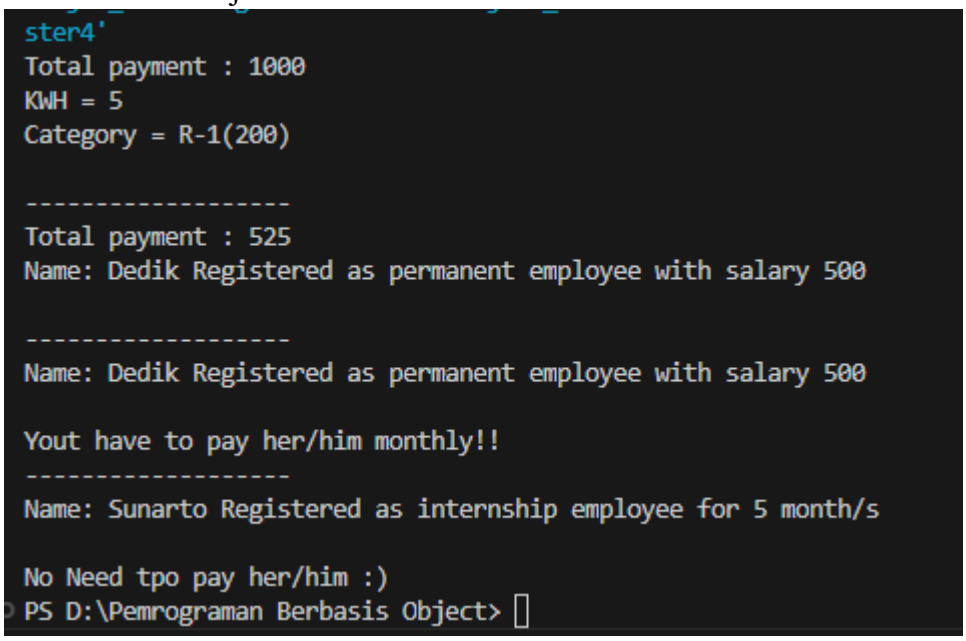
1. e dideklarasikan sebagai array bertipe Employee[] dan mereka berdua adalah subclass dari Employee jadi mereka bisa diupcast ke Employee dan masuk arraynya.
2. p dideklarasikan sebagai array bertipe Payable[] sebagai interface, sama saja sih seperti jawbaan no 1 mereka berdua adalah class yang mengimplement interface payable jadinya bisa dan valid dimasukan ke array p
3. eror, karena ada oobjek yang tidak sesuai dengan tipe eleemn arraynya, yaitu mencoba memasukkan ElectricityBill ke dalam Employee[], eror akan hilang ketika hapus object eBill
   akan saya coba ya



Nah bisa.

PERCOBAAN 4

Hasil run Tester4.java

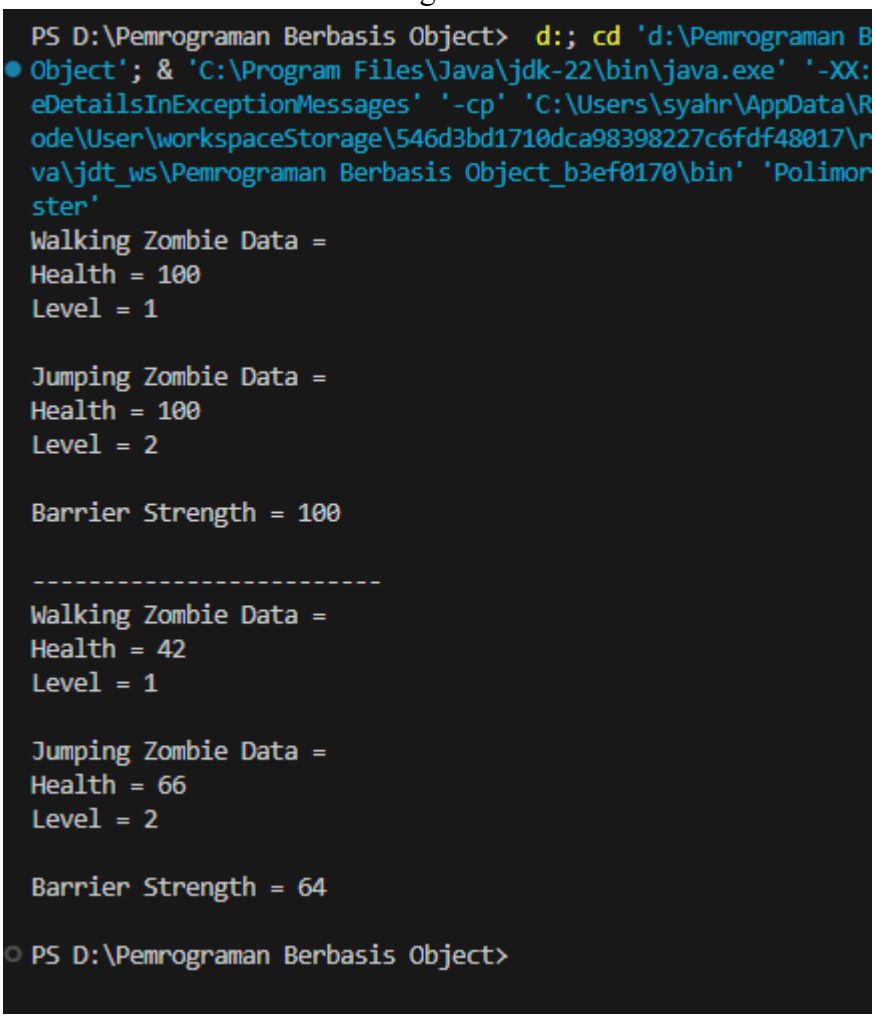


JAWABAN PERTANYAAN PERCOBAAN 4

1. Karena kedua objek tersebut mengimplement interface Payable. keduanya secara tipe kompatibel dengan parameter pay(Payable p)
2. parameter bertipe Payable, pay() dapat menerima berbagai objek yang bisa “dibayar” tanpa tergantung pada kelas konkret mereka. Owner cukup tahu cara memanggil operasi umum yang ada di Payable. sementara implementasi detail pembayaran terserah masing-masing kelas (PermanentEmployee, ElectricityBill) sehingga ini membuat class Owner sangat fleksibel hanya memproses pembayaran yang harus di lakukan.
3. Terjadi eror, Karena InternshipEmployee tidak mengimplement Payable. jadi objek iEmp tidak kompatibel dengan parameter Payable pada pay()
4. instanceof dipakai untuk mengecek tipe dari objek p sebelum melakukan operasi yang spesifik untuk ElectricityBill, tujuannya untuk memastikan dulu bahwa p memang benar-benar sebuah ElectricityBill agar casting aman dan tidak menimbulkan ClassCastException.
5. agar variabel eb berjenis ElectricityBill dapat mengakses method atau atribut yang tidak didefinisikan dalam interface Payable. Tanpa casting, dari tipe Payable hanya bisa memanggil method yang ada di interface.

TUGAS

Berikut adalah hasil run tester tugas



Semua kode saya lampirkan di folder yang sama dengan laporan ini. Sesuai diagram UML. Terimakasih.