|  |  |  |
| --- | --- | --- |
| **Nama** | : | Vincentius Leonanda Prabowo |
| **Nim** | : | 2341720149 |
| **Kelas** | : | TI-2A |

**LEMBAR JAWABAN PERCOBAAN 1**

|  |  |
| --- | --- |
| **Soal No** | **Jawaban** |
| 1 | **Class apa sajakah yang merupakan turunan dari class Employee?**   * PermanentEmployee dan InternshipEmployee. Kedua kelas ini mungkin memperluas (extend) class Employee |
| 2 | **Class apa sajakah yang implements ke interface Payable?**   * Class yang mengimplementasikan interface Payable adalah PermanentEmployee dan ElectricityBill. |
| 3 | **Mengapa e bisa diisi dengan objek pEmp (objek dari class PermanentEmployee) dan objek iEmp (objek dari class InternshipEmployee)?**   * Variabel e memiliki tipe data Employee, yang merupakan supertype dari PermanentEmployee dan InternshipEmployee. Dalam Java, tipe Employee dapat merujuk pada objek dari kelas turunannya, yaitu PermanentEmployee atau InternshipEmployee, karena aturan pewarisan yang mengizinkan referensi tipe parent untuk menunjuk ke objek child class. Ini adalah salah satu bentuk polimorfisme. |
| 4 | **Mengapa p bisa diisi dengan objek pEmp (objek dari class PermanentEmployee) dan objek eBill (objek dari class ElectricityBill)?**   * Variabel p bertipe Payable, yang memungkinkan p merujuk pada objek apa pun yang mengimplementasikan interface Payable, termasuk pEmp (objek PermanentEmployee) dan eBill (objek ElectricityBill). Karena keduanya mengimplementasikan Payable, variabel bertipe Payable dapat merujuk pada kedua objek ini. |
| 5 | **Mengapa terjadi error pada penambahan p = iEmp; dan e = eBill;?**   * Kesalahan ini terjadi karena iEmp (objek InternshipEmployee) tidak mengimplementasikan Payable, sehingga tidak kompatibel dengan tipe Payable yang diharapkan untuk p. Demikian juga, eBill (objek ElectricityBill) bukan subclass dari Employee, jadi tidak bisa ditugaskan ke variabel bertipe Employee. Ini adalah aturan tipe dalam Java yang mengharuskan jenis yang kompatibel. |
| 6 | **Kesimpulan tentang konsep/bentuk dasar polimorfisme:**   * Polimorfisme dalam Java memungkinkan variabel dari suatu tipe parent (superclass atau interface) untuk merujuk pada objek dari tipe child (subclass atau implementasi interface). Hal ini memungkinkan objek-objek yang berbeda untuk digunakan secara bergantian selama mereka memiliki kompatibilitas tipe, baik melalui pewarisan (inheritance) atau implementasi interface. Polimorfisme memberikan fleksibilitas dalam program untuk memproses objek yang berbeda dengan cara yang seragam, misalnya, dengan memanggil metode yang sama pada tipe variabel yang lebih umum. |

**LEMBAR JAWABAN Percobaan 2**

|  |  |
| --- | --- |
| **Soal No** | **Jawaban** |
| 1 | **Mengapa pemanggilan e.getEmployeeInfo() pada baris 8 dan pEmp.getEmployeeInfo() pada baris 10 menghasilkan hasil yang sama?**  Kedua pemanggilan ini menghasilkan hasil yang sama karena e dan pEmp merujuk pada objek yang sama, yaitu instance dari PermanentEmployee. Meskipun variabel e bertipe Employee, objek yang sebenarnya dirujuk oleh e adalah PermanentEmployee, sehingga metode getEmployeeInfo() yang dipanggil pada kedua variabel tersebut berasal dari class PermanentEmployee. Dalam Java, metode yang dipanggil didasarkan pada tipe objek yang sebenarnya (actual object type), bukan tipe variabel referensinya, sehingga hasilnya sama. |
| 2 | **Mengapa pemanggilan metode e.getEmployeeInfo() disebut sebagai pemanggilan metode virtual (virtual method invocation), sedangkan pEmp.getEmployeeInfo() tidak?**  Pemanggilan metode e.getEmployeeInfo() disebut virtual method invocation karena e bertipe Employee, tetapi memanggil metode yang diimplementasikan di subclass PermanentEmployee. Dalam Java, metode yang dipanggil bergantung pada tipe objek aktual (PermanentEmployee), bukan tipe referensi variabel (Employee). Java menggunakan virtual method invocation untuk menentukan metode mana yang akan dieksekusi pada runtime, bukan pada waktu kompilasi.  Sebaliknya, pEmp.getEmployeeInfo() tidak dianggap sebagai virtual method invocation karena variabel pEmp bertipe PermanentEmployee, dan metode getEmployeeInfo() langsung diambil dari class yang sama, tanpa perlu mekanisme runtime untuk menentukan implementasi yang tepat. |
| 3 | **Apa yang dimaksud dengan virtual method invocation? Mengapa disebut virtual?**  **Virtual method invocation** adalah mekanisme di mana metode yang dipanggil bergantung pada tipe objek sebenarnya (actual object type) dari instance, bukan tipe referensi variabel yang memanggil metode tersebut. Dalam Java, ketika kita memanggil suatu metode melalui referensi yang bertipe superclass atau interface, JVM menentukan implementasi metode yang benar pada runtime berdasarkan tipe objek aktual, bukan tipe variabelnya. Disebut "virtual" karena implementasi metode yang akan dijalankan tidak ditentukan secara tetap pada saat kompilasi tetapi dipilih secara dinamis pada runtime sesuai dengan tipe objek yang sebenarnya.  Virtual method invocation memungkinkan fleksibilitas dan polimorfisme, memungkinkan metode yang sesuai dipanggil pada objek-objek yang berbeda meskipun referensinya berasal dari superclass atau interface. |

**LEMBAR JAWABAN Percobaan 3**

|  |  |
| --- | --- |
| **Soal No** | **Jawaban** |
| 1 | **Mengapa array e bisa diisi dengan objek-objek dengan tipe yang berbeda, yaitu objek pEmp (objek dari PermanentEmployee) dan objek iEmp (objek dari InternshipEmployee)?**  Array e dideklarasikan dengan tipe Employee. Karena PermanentEmployee dan InternshipEmployee adalah subclass dari Employee, maka objek pEmp dan iEmp dapat disimpan dalam array e. Ini adalah penerapan polimorfisme dalam Java, di mana referensi bertipe superclass (Employee) dapat merujuk ke objek dari subclass-nya (PermanentEmployee dan InternshipEmployee), sehingga array e dapat menampung kedua objek tersebut. |
| 2 | **Mengapa array p bisa diisi dengan objek-objek dengan tipe yang berbeda, yaitu objek pEmp (objek dari PermanentEmployee) dan objek eBill (objek dari ElectricityBill)?**  Array p bertipe Payable, yang merupakan sebuah interface. Karena PermanentEmployee dan ElectricityBill keduanya mengimplementasikan interface Payable, objek pEmp dan eBill dapat disimpan dalam array p. Java memungkinkan array yang bertipe interface untuk menyimpan objek apa pun yang mengimplementasikan interface tersebut. |
| 3 | **Mengapa terjadi error pada baris ke-10?**  Error pada baris ke-10 terjadi karena array e2 dideklarasikan dengan tipe Employee, tetapi salah satu elemen yang ingin dimasukkan ke dalam array e2 adalah eBill, yang bertipe ElectricityBill. ElectricityBill tidak memiliki hubungan pewarisan atau keturunan dengan Employee, sehingga tidak bisa ditempatkan dalam array bertipe Employee. Di Java, objek harus sesuai dengan tipe atau subclass dari tipe arraynya untuk dapat dimasukkan ke dalam array tersebut, dan dalam kasus ini eBill tidak memenuhi kriteria tersebut karena tidak berhubungan langsung dengan Employee. |

**LEMBAR JAWABAN Percobaan 4**

|  |  |
| --- | --- |
| **Soal No** | **Jawaban** |
| 1 | **Mengapa pemanggilan ow.pay(eBill) dan ow.pay(pEmp) bisa dilakukan, padahal parameter metode pay() di kelas Owner bertipe Payable?**  Kelas Owner memiliki metode pay() dengan parameter bertipe Payable. Karena ElectricityBill dan PermanentEmployee masing-masing mengimplementasikan interface Payable, objek eBill (dari ElectricityBill) dan pEmp (dari PermanentEmployee) bisa diterima sebagai argumen metode pay(). Ini adalah contoh polimorfisme, di mana objek dari kelas yang mengimplementasikan interface tertentu dapat digunakan sebagai argumen bertipe interface tersebut.. |
| 2 | **Apa tujuan membuat parameter pay() bertipe Payable di dalam kelas Owner?**  Menjadikan parameter pay() bertipe Payable memungkinkan metode ini menerima objek apa pun yang mengimplementasikan interface Payable, tanpa mempermasalahkan tipe spesifik kelasnya. Dengan kata lain, tujuan ini adalah untuk menerapkan prinsip *interface-based polymorphism*, sehingga Owner dapat melakukan pembayaran terhadap objek yang memenuhi kontrak Payable, baik itu tagihan listrik (ElectricityBill) atau gaji karyawan (PermanentEmployee). |
| 3 | **Mengapa ow.pay(iEmp); menghasilkan error jika ditambahkan di baris terakhir metode main()?**  InternshipEmployee tidak mengimplementasikan interface Payable, sehingga objek iEmp tidak dapat diterima sebagai argumen untuk metode pay() di kelas Owner, yang membutuhkan objek bertipe Payable. Oleh karena itu, menambahkan ow.pay(iEmp); menghasilkan error kompilasi karena iEmp tidak kompatibel dengan tipe parameter Payable. |
| 4 | **Apa tujuan dari sintaks p instanceof ElectricityBill di baris ke-6 kelas Owner?**  Ekspresi p instanceof ElectricityBill memeriksa apakah objek p adalah instance dari ElectricityBill. Pemeriksaan ini diperlukan untuk memastikan bahwa objek p memang merupakan ElectricityBill sebelum melakukan casting, guna menghindari error ClassCastException. Jika p adalah ElectricityBill, maka casting yang dilakukan setelahnya akan aman. |
| 5 | **Mengapa casting objek diperlukan pada baris ke-7 di kelas Owner (ElectricityBill eb = (ElectricityBill) p)?**  Casting objek p yang bertipe Payable ke ElectricityBill diperlukan untuk mengakses metode atau atribut spesifik ElectricityBill yang tidak ada di interface Payable. Karena metode pay() menerima parameter Payable, objek p harus di-cast ke ElectricityBill agar metode Owner dapat memperlakukan p sebagai objek ElectricityBill dan mengakses metode khususnya, seperti getBasePrice() atau properti khusus lainnya |

**LEMBAR JAWABAN TUGAS**

|  |  |
| --- | --- |
| **Soal No** | **Jawaban** |
| 1 |  |
| 2 |  |
| 3 |  |