Nama	:	Rahayu Amaliyah Kamis
NIM	:	20220040248
Kelas	:	TI22A
Mata Kuliah	:	Pemrograman Berorientasi Object
Tugas Sesi	:	Alun Sujjada, S.Kom, M.T

Rincian Jawaban :

- Percobaan 1

  Analisis :

  Ketika menggunakan this.x di dalam metode Info, itu mengacu pada nilai x dari objek yang sedang diproses saat ini, yaitu objek Child yang baru dibuat.

  Saat menggunakan super.x di dalam metode Info, itu merujuk pada nilai x dari kelas induk, yaitu kelas Parent. Variabel x di dalam metode Info (yang merupakan parameter metode) akan menerima nilai yang diberikan saat metode tersebut dipanggil (dalam kasus ini, 20).

  Output :

  1. Nilai x sebagai parameter = 20
  2. Data member x di class child = 10
  3. Data member x di class parent = 5

- Percobaan 2

  Alasan terjadinya kesalahan kode beserta solusinya:

  Kesalahan muncul dalam kode tersebut karena kelas Manajer mencoba mengakses variabel nama yang telah dinyatakan sebagai private di kelas Pegawai. Dalam bahasa pemrograman Java, ketika sebuah variabel dideklarasikan sebagai private, itu hanya dapat diakses secara langsung oleh kelas yang mendefinisikannya. Kelas lain tidak dapat mengaksesnya secara langsung. Untuk memperbaiki kesalahan tersebut, variabel nama di kelas Pegawai harus diakses menggunakan metode getNama() dan diatur nilainya menggunakan metode setNama(String nama). Dalam kelas Manajer, metode isiData(String n, String d) digunakan untuk menetapkan nilai nama menggunakan setter yang tersedia di kelas Pegawai.

  Analisis kode ini adalah sebagai berikut:

  Kode tersebut menunjukkan penerapan konsep pewarisan (inheritance) dimana kelas Manajer mewarisi sifat-sifat (variabel dan metode) dari kelas Pegawai. Prinsip enkapsulasi diterapkan dengan mendeklarasikan

variabel nama sebagai private, sehingga hanya dapat diakses melalui metode getter dan setter yang telah tersedia. Kelas Manajer menambahkan fitur tambahan berupa variabel departemen, yang tidak ada di kelas Pegawai. Metode isiData(String n, String d) digunakan untuk mengisi data sekaligus menetapkan nilai nama dan departemen dari objek kelas Manajer.

- Percobaan 3

  Alasan terjadinya kesalahan dalam kode program dan solusinya adalah sebagai berikut:

  Pada percobaan ketiga, terjadi kesalahan karena saat membuat kelas turunan (Child) dari kelas induk (Parent), Java mencoba memanggil konstruktor dari kelas induk (Parent2). Namun, dalam kelas Parent2, tidak ada konstruktor yang didefinisikan, yang menyebabkan terjadinya kesalahan. Solusi untuk mengatasi kesalahan ini adalah dengan menambahkan konstruktor public Parent2 di dalam kelas Parent2.

  Analisis kode tersebut adalah sebagai berikut:

  Kelas Parent2 berperan sebagai kelas dasar atau induk. Meskipun dalam contoh ini kelas tersebut kosong, secara konseptual, kelas ini dapat berisi atribut atau metode yang akan diwarisi oleh kelas anaknya. Kelas Child2 merupakan turunan dari kelas Parent2, ditunjukkan dengan penggunaan kata kunci extends dalam deklarasi kelas. Dengan demikian, kelas Child2 mewarisi semua sifat dari kelas Parent2. Variabel x adalah variabel anggota (instance variable) dari kelas Child2, yang diinisialisasi dengan nilai awal 5. Setiap objek kelas Child2 yang dibuat akan memiliki variabel ini.

  Konstruktor kelas Child2 adalah metode khusus yang memiliki nama yang sama dengan nama kelasnya. Dalam konstruktor ini, variabel x diinisialisasi dengan nilai 5.

- Percobaan 4

  Analisis Utama:

  Kode tersebut mengilustrasikan penerapan konsep pewarisan (inheritance) di mana kelas Manager mewarisi karakteristik dan perilaku yang dimiliki oleh kelas Employee. Pada kelas Manager, konstruktor memanfaatkan konstruktor dari kelas Employee dengan menggunakan kata kunci super(). Setiap konstruktor memiliki peran yang berbeda bergantung pada jumlah parameter yang disediakan dan nilai-nilai yang ingin diinisialisasi. Kode tersebut juga menampilkan penerapan overloading konstruktor untuk memberikan kemampuan fleksibilitas dalam pembuatan objek.

- Percobaan 5

  Analisis Komprehensif:

  Kode tersebut menerapkan konsep pewarisan, di mana kelas turunan (SadObject dan HappyObject) menerima sifat dan perilaku dari kelas induk (MoodyObject). Selain itu, terdapat polimorfisme di sini, di mana objek m yang berasal dari tipe MoodyObject dapat merujuk ke objek dari kelas turunan yang berbeda (SadObject dan HappyObject), dan perilaku metode yang dipanggil akan bervariasi tergantung pada objek sebenarnya yang ditunjuk oleh variabel referensi m. Terjadi overriding metode di kelas turunan (SadObject dan HappyObject) di mana metode getMood() digantikan untuk mengembalikan mood yang sesuai. Implementasi dari metode laugh() dan cry() berbeda di setiap kelas turunan, menunjukkan fleksibilitas dalam implementasi sesuai dengan kebutuhan masing-masing kelas.

- Percobaan 6

  Analisis Komprehensif:

  Pada bagian ini, kode tersebut menjelaskan konsep pewarisan di mana kelas B mewarisi atribut dan perilaku dari kelas A.Saat objek dari kelas A atau kelas B dibuat, konstruktor kelas A akan dieksekusi terlebih dahulu.Penggunaan super() dalam konstruktor kelas B memungkinkan untuk memanggil konstruktor kelas induk sebelum melakukan inisialisasi kelas turunan. Setelah konstruktor kelas B selesai dieksekusi, nilai dari variabel anggota var_a dan var_b diubah, menunjukkan bahwa kelas turunan dapat mengakses dan memanipulasi variabel anggota dari kelas induk. Secara keseluruhan, kode tersebut memberikan contoh sederhana tentang bagaimana pewarisan berfungsi dalam bahasa pemrograman Java dan bagaimana konstruktor kelas turunan dapat memanggil konstruktor kelas induk.

- Percobaan 7

  Pada bagian ini, kode tersebut menggambarkan konsep pewarisan, di mana kelas Anak mewarisi atribut dan perilaku dari kelas Bapak. Metode show_variabel() di kelas Anak melakukan overriding terhadap metode yang sama di kelas Bapak, yang berarti metode yang digunakan adalah milik kelas Anak, bukan kelas Bapak. Override method memungkinkan kelas turunan untuk memberikan implementasi yang berbeda terhadap metode yang sudah didefinisikan di kelas induknya. Ketika metode show_variabel() dipanggil pada objek objectAnak, yang dieksekusi adalah metode yang ada di kelas Anak, sehingga mencetak nilai dari variabel a, b, dan c. Hal ini menunjukkan bagaimana pewarisan memungkinkan penggunaan ulang kode dan memfasilitasi pengembangan perangkat lunak yang lebih terstruktur dan terorganisir.

- Percobaan 8

  Bagian ini menunjukkan konsep pewarisan dalam pemrograman berorientasi objek (OOP) di Java, di mana kelas Baby mewarisi atribut dan perilaku dari kelas Parent2. Konstruktor Baby memanggil konstruktor superclass Parent2 menggunakan kata kunci super(), sehingga konstruktor dari kelas Parent2 akan dieksekusi terlebih dahulu sebelum konstruktor dari kelas Baby. Dengan demikian, penggunaan super() memungkinkan untuk memanggil konstruktor superclass dan melakukan inisialisasi dari kelas induk sebelum melakukan inisialisasi dari kelas turunannya. Meskipun tidak ada overriding method dalam contoh ini, tetapi jika diperlukan, method dapat ditambahkan di kelas Parent2 yang kemudian di-override oleh kelas Baby.