PEMROGRAMAN BERORIENTASI OBJEK

Ulangan Tengah Semester Ganjil

Dosen pengampu :
Yanuar Risah Prayogi S.Kom., M.Kom.

Disusun oleh:

Aaron Febrian Prakoso (3122500060)

2 D3 Teknik Informatika B

PROGRAM STUDI D3 TEKNIK INFORMATIKA

POLITEKNIK ELEKTRONIKA NEGERI SURABAYA

2022 / 2023

Soal dan Jawaban

1. Sebutkan minimal 3 perbedaan overriding dan overloading!
   Jawab:
   a. Dari segi definisi
      - Overriding: Overriding terjadi ketika sebuah subclass menyediakan implementasi ulang (override) dari metode yang sudah didefinisikan dalam superclass. Dalam hal ini, nama metode, tipe parameter, dan tipe kembalian metode harus sama di antara superclass dan subclass.
      - Overloading: Overloading terjadi ketika sebuah kelas memiliki beberapa metode dengan nama yang sama dalam kelas yang sama. Namun, metode-metode tersebut harus memiliki parameter yang berbeda dalam hal jumlah, urutan, atau tipe data.
   b. Dari segi hubungan kelas
      - Overriding: Terkait dengan hubungan kelas hierarki, yaitu antara superclass dan subclass. Metode yang di overriding dalam subclass harus didefinisikan dalam superclass.
      - Overloading: Tidak ada hubungan kelas yang khusus yang diperlukan. Overloading bisa terjadi dalam kelas yang sama tanpa memperhatikan hubungan hierarki.
   c. Dari segi penggunaan parameter
      - Overriding: Metode overriding memiliki parameter yang sama dengan metode yang di overriding dalam superclass. Namun, kita bisa menggunakan parameter tambahan dalam metode subclass jika diperlukan.
      - Overloading: Metode overloading memiliki parameter yang berbeda. Ini berarti jumlah parameter, tipe data, atau urutan parameter bisa berbeda di antara metode-metode yang di-overload.

2. Mengapa sebuah kelas perlu dienkapsulasi?
   Jawab:
   Enkapsulasi sendiri berarti penyembunyian informasi suatu data atau atribut. Jadi alasan kenapa sebuah kelas perlu untuk dienkapsulasi adalah untuk melindungi informasi agar tidak dapat diakses secara sembarangan.

3. Mengapa ada konsep inheritance pada sebuah kelas?
   Jawab:
   Karena dengan menerapkan konsep inheritance, sebuah kelas dapat mempunyai kelas turunan. Biasanya suatu kelas yang mempunyai kelas turunan dinamakan dengan parent class, sedangkan kelas turunannya itu sendiri dinamakan child class/sub-class. Kelas turunan tersebut mewarisi apa-apa yang dimiliki oleh parent-classnya.

4. Apa kegunaan inheritance pada saat kita membuat aplikasi?
   Jawab:

- Pengorganisasian Kode: Inheritance membantu dalam pengorganisasian kode dalam aplikasi. Kita dapat membuat hierarki kelas yang mencerminkan struktur konsep dalam masalah yang kita selesaikan. Ini membuat kode lebih mudah dipahami dan dikelola.
- Penggunaan Ulang (Reuse): Salah satu manfaat utama inheritance adalah kemampuan untuk menggunakan kembali kode yang telah ada. Kita dapat membuat kelas baru yang mewarisi properti dan metode dari kelas yang ada, dan kemudian menambahkan atau mengubah perilaku sesuai kebutuhan tanpa perlu menulis ulang kode yang sama.
- Polimorfisme: Inheritance memungkinkan penggunaan polimorfisme, yang memungkinkan objek dari kelas yang berbeda dalam hierarki untuk digunakan dengan cara yang serupa. Hal ini berguna ketika kita ingin memproses objek dari berbagai jenis dalam cara yang konsisten.
- Menghindari Redundansi: Dengan mewarisi sifat dan perilaku dari kelas dasar, kita dapat menghindari duplikasi kode yang tidak perlu. Jika ada perubahan dalam logika, kita hanya perlu memperbarui kelas dasar.
- Abstraksi: Inheritance memungkinkan kita untuk membuat kelas abstrak yang tidak dapat diinstansiasi sendiri tetapi berfungsi sebagai kerangka dasar untuk kelas turunannya. Ini membantu dalam pemodelan abstrak dan konsep umum dalam aplikasi.
- Penggunaan Kode Eksternal: kita dapat menggunakan kelas-kelas yang telah ada dalam bahasa pemrograman atau pustaka eksternal dan mewarisi fitur-fitur ini untuk mengembangkan aplikasi kita dengan lebih cepat dan efisien.
- Pemisahan Tanggung Jawab: Dengan memisahkan kelas-kelas berdasarkan tanggung jawabnya, kita dapat mengisolasi perubahan yang berkaitan dengan suatu fitur atau komponen tanpa memengaruhi bagian lain dari aplikasi.
- Mempermudah Perawatan: Inheritance dapat membuat perawatan dan pengembangan aplikasi menjadi lebih efisien. Kode yang terorganisir dengan baik lebih mudah dipelihara, diperbaiki, dan ditingkatkan.

5. Studi kasus misalkan saya mau membuat kelas PrinterAIO, printer all in one yang bisa printer, scan, dan fax dengan cara menurunkan dari kelas Printer, kelas Scanner, dan kelas Fax, bagaimana menurut Anda?
   Jawab:
   Membuat kelas PrinterAIO yang mewarisi sifat dari kelas Printer, Scanner, dan Fax adalah pendekatan yang baik dalam pemrograman berorientasi objek, dan ini mencerminkan penggunaan inheritance untuk menciptakan hierarki kelas yang sesuai dengan kebutuhan kita.

   Berikut contoh kerangka dasar untuk implementasi ini:

```
class Printer {
    public void print(String document) {
        // Implementasi mencetak dokumen
    }
}

class Scanner {
    public String scan() {
        // Implementasi pemindaian
```

```
        return "Hasil pemindaian";
    }
}

class Fax {
    public void sendFax(String document) {
        // Implementasi pengiriman fax
    }
}

class PrinterAIO extends Printer, Scanner, Fax {
    // Kelas ini mewarisi sifat dan perilaku dari kelas Printer,
Scanner, dan Fax didefinisikan di kelas dasar sesuai kebutuhan
}
```

Dalam contoh di atas, PrinterAIO mewarisi sifat dan perilaku dari Printer, Scanner, dan Fax. Kita dapat menambahkan metode tambahan atau mengganti metode yang didefinisikan di kelas dasar sesuai kebutuhan untuk memenuhi spesifikasi khusus dari printer all-in-one (AIO) kita.

Ini adalah pendekatan yang baik karena memungkinkan kita untuk mengelola fitur-fitur yang berbeda (mencetak, pemindaian, pengiriman fax) sebagai bagian dari hierarki yang terorganisir dengan baik, dan kita dapat menggunakan kode yang ada dalam kelas Printer, Scanner, dan Fax tanpa perlu menulis ulang. Ini juga mendukung prinsip-prinsip pemrograman berorientasi objek seperti penggunaan kode ulang, polimorfisme, dan abstraksi.