|  |  |  |  |
| --- | --- | --- | --- |
| **PRAKTIKUM PEMROGRAMAN LANJUT** | | |  |
| **LABORATORIUM TELEMATIKA** | | |
|  | DEPARTEMEN TEKNIK KOMPUTER | |
|  | FAKULTAS TEKNOLOGI ELEKTRO  DAN INFORMATIKA CERDAS | |
| **P1** |  | **OBJECT ORIENTED PROGRAMMING BASICS** |

|  |  |
| --- | --- |
| **DATA PRAKTIKAN** | |
| Nama Praktikan |  |
| AKHMAD RIZQULLAH RIDLOHI |
| NRP |
| 5024231037 |
| Nama Kelompok |
| Seraphyne |

|  |  |
| --- | --- |
| **DATA ASISTEN** | |
| Nama |  |
| NRP |  |

|  |  |
| --- | --- |
| **MONITORING PRAKTIKUM** | |
| Tanggal Praktikum | TTD Asisten |
|  | (Jangan lupa mengambil dokumentasi!) |

# Object Oriented Programming Basics

## Advantages of Object Oriented Programming

* **Modularity - Reusability - Data modelling**

Dalam paradigma Object Oriented Programming (OOP) atau pemrograman berorientasi objek, kode akan dipecah menjadi unit-unit terpisah yang dinamakan objek atau biasa disebut class.

Sebagai contoh dalam sistem perbankan, terdapat class seperti *Account* yang memiliki variabel seperti nomor akun dan jumlah uang. Class *Transaction* memiliki variable tipe transaksi, jumlah uang yang ditransaksikan, nomor akun pengirim dan penerima.

Tidak hanya variable, setiap class dapat memiliki fungsinya masing-masing seperti dalam class *Account* terdapat fungsi *deposit* dan *withdraw*, dalam class *Transaction* terdapat fungsi *processTransaction*. Dengan memecah kode program berdasarkan objek, kalian dapat menjalankan banyak fungsionalitas hanya dengan memanggil fungsi tanpa harus mengetik kode berulang.

* **Abstraction**

Konsep abstraksi di OOP digunakan untuk menyembunyikan elemen yang kompleks ataupun tidak perlu untuk ditunjukkan. Terdapat dua abstraksi yakni abstraksi data dan abstraksi process.

Contoh abstraksi data dalam class Account adalah seperti jumlah uang, karena hanya class Account tersebut saja yang berhak untuk mengubah nilai jumlah uang. Class lain seperti *Employee* tidak berhak untuk mengubah nilai jumlah uang suatu akun secara langsung.

Contoh abstraksi fungsi adalah dalam class *Transaction*, sebelum melakukan transaksi akan terdapat banyak proses sebelumnya. Seperti memastikan jumlah uang yang memenuhi, memastikan akun tujuan aktif, dan lainnya. Kalian tidak perlu tahu proses kompleks dibaliknya sehingga proses tersebut akan disembunyikan. Kalian hanya perlu memanggil fungsi *processTransaction*.

* **Inheritance**

Inheritance atau pewarisan adalah konsep di mana sebuah class dapat mewarisi variable dan fungsi dari class lain yang disebut sebagai kelas induk atau superclass. Class yang mewarisi disebut kelas turunan atau subclass. Dengan inheritance, kelas turunan dapat menggunakan dan memperluas fungsionalitas yang telah didefinisikan dalam kelas induk tanpa perlu menulis ulang kode yang sama.

Sebagai contoh terdapat class *Vehicle* sebagai superclass yang memiliki variable merk, warna dan fungsi startEngine. Kemudian terdapat subclass seperti *Car*, *Motorcycle*, dan *Truck*. Setiap subclass tersebut akan memiliki variabel dan fungsi yang sama dengan class *Vehicle*, bahkan dapat diperluas. Contohnya *Car* memiliki 4 roda, *Motorcycle* 2 roda, dan seterusnya.

* **Maintenance**

Dengan paradigma OOP, kalian dapat dengan mudah melakukan maintenance atau pemeliharaan yang merujuk pada proses menjaga, memperbaiki, dan memperbarui perangkat lunak yang sudah ada. Maintenance dalam OOP mencakup berbagai tindakan seperti memperbaiki bug, menambahkan fitur baru, meningkatkan kinerja, dan memperbarui kode agar sesuai dengan perubahan kebutuhan atau spesifikasi, dan lainnya.

## Public Private Class Example

Dalam satu class, kita dapat melakukan abstraksi dengan memberikan hak akses dalam kode class tersebut. Hak akses yang paling sering digunakan adalah *public* dan *private*. Hak akses *public* berarti variable dan fungsi suatu *class* dapat dipanggil di luar class tersebut. Sedangkan hak akses *private* berarti variable dan fungsi class tersebut hanya dapat diakses di dalam class tersebut.

#include <iostream>

class MyClass {

public:

*// Public member variable*

    int publicVar;

*// Public Function*

    void PublicMethod() {

        std::cout << "Public method called." << std::endl;

    }

*// Public Function Calling Private Function*

    void CallPrivate(){

        PrivateMethod();

    }

private:

*// Private member variable*

    int privateVar;

*// Private Function*

    void PrivateMethod() {

        std::cout << "Private method called. :" << publicVar << "-" << privateVar  << std::endl;

    }

};

int main() {

    MyClass obj;

    obj.publicVar = 42; *// Accessing public variable*

    obj.PublicMethod(); *// Calling public method*

    obj.CallPrivate(); *// Example of Methode Caling a private Methode*

    return 0;

}

## Constructor Example

Constructor digunakan untuk menginisialisasi objek dengan nilai awal atau default values, dan atau konfigurasi yang perlu dilakukan untuk sebuah objek saat objek tersebut dibuat.

#include <iostream>

class Person {

public:

    Person() {

        std::cout << "Default constructor called." << std::endl;

    }

    Person(const std::string& name) {

        std::cout << "Parameterized constructor called. Name: " << name << std::endl;

    }

};

int main() {

*// Calls default constructor*

    Person person1;

*// Calls parameterized constructor*

    Person person2("Alice");

    return 0;

}

## Method overloading

Method overloading memungkinkan kalian untuk mendefinisikan beberapa fungsi dengan nama yang sama dalam sebuah class, tetapi dengan parameter yang berbeda. Ini memberikan fleksibilitas dalam kalianan fungsi, karena fungsi yang sama dapat digunakan dengan cara yang berbeda tergantung pada jenis dan jumlah parameter yang diberikan.

#include <iostream>

class Calculator {

public:

    int Add(int a, int b) {

        return a + b;

    }

    double Add(double a, double b) {

        return a + b;

    }

};

int main() {

    Calculator calc;

    std::cout << "Int sum: " << calc.Add(5, 3) << std::endl;

    std::cout << "Double sum: " << calc.Add(2.5, 1.7) << std::endl;

    return 0;

}

**Tugas Pendahuluan**

1. Tandai kode yang salah dengan melingkari kode tersebut dan berikan alasannya! *(dengan bolpoint)*

#include <iostream>

#include <string>

class rockubando {

private:

    int song = 1;

    void play(std::string str) {

        std::cout << str << std::endl;

    }

    void play(std::string str, int num) {

        for (;;++song) {

            std::cout << str << " ";

            if (num<=song) break;

        }

        std::cout << std::endl;

    }

};

int main() {

    rockubando band("kessoku band!");

    band.play("lets rockk!!");

    band.play("bando!", 3);

    return 0;

}

1. Berilah penjelasan per baris pada kode berikut ini! *(ditulis langsung dengan bolpoint di kanan kode)*

#include <iostream>

class kessokuband {

private:

    int songs;

public:

    kessokuband(int count) {

        songs = count;

    }

    void play() {

        for (;songs>0;songs--) {

            std::cout << "kita!";

        }

        std::cout << std::endl;

    }

    void play(std::string songs) {

        std::cout << songs << std::endl;

    }

};

int main() {

    kessokuband orenoband(3);

    orenoband.play("ano bando");

    orenoband.play();

    return 0;

}