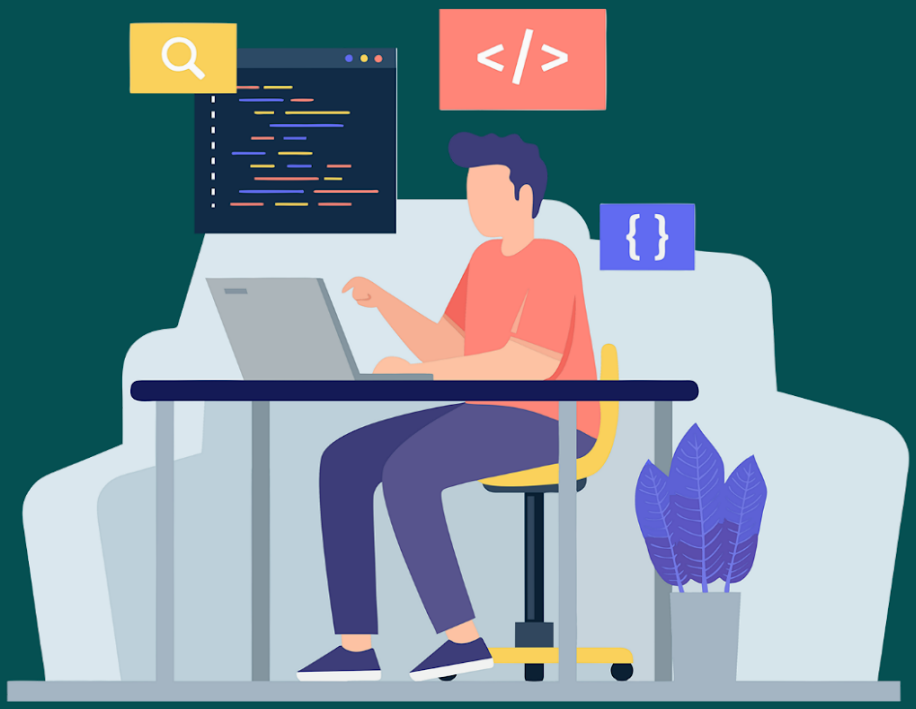
Modul Praktikum
Desain dan Pemrograman Berorientasi Objek

COMPOSITION:
ATRIBUT KOMPOSIT



TIM ASISTEN PEMROGRAMAN
ANGKATAN 12

Departemen Pendidikan Ilmu Komputer
Fakultas Pendidikan Matematika dan Ilmu Pengetahuan Alam
Universitas Pendidikan Indonesia
2023

Composition

Dalam pemrograman berorientasi objek, **komposisi** (*composition*) merupakan suatu konsep dimana suatu kelas menyimpan, mereferensikan, atau menginstansiasi satu atau lebih objek di dalamnya. Istilah yang lebih umum digunakan adalah **atribut komposit**. Berbeda dengan variabel biasa, atribut komposit mempunyai metode-metode yang bisa dipanggil di dalamnya.

Sebagai contoh, ada satu ruangan yang berisi 20 orang. Ruangan itu didefinisikan sebagai kelas, sementara 20 orang yang ada di dalamnya merupakan objek terpisah dan diinstansiasi sebagai *array* tersendiri.

Inheritance v. Composition

Pewarisan dan komposisi merupakan dua konsep utama OOP yang berbeda. **Pewarisan** merupakan hubungan antara dua objek yang memiliki atribut dan metode yang sama beserta tambahan-tambahannya. Hubungan ini lebih sering didefinisikan sebagai "***is a*** *relationship*". Kelas yang menjadi anak bisa disebut penyempurnaan dari kelas orang tua.

Sementara itu, **komposisi** merupakan hubungan antara dua objek yang benar-benar berbeda, tetapi berkaitan agar objek dapat melengkapi suatu kelas yang lainnya. Oleh karena itu, hubungan ini lebih sering didefinisikan sebagai "***has a*** *relationship*". Kelas yang satu hanya menyimpan objek-objek dari kelas lain, bukan perubahan atau penyempurnaan dari objek tersebut.

Implementasi Pewarisan dan Komposisi

Header.hh

```cpp
// Declare library.
#include <iostream>
#include <string>

using namespace std;

/* Composition class.
   We have to put these classes on top because they will be composed in other
   class. C++ reads code from top to bottom, so if you put Human and Adult
   first, the code won't work as they don't know what is a profession / laptop. */

class Profession
{
    private:
        string position;
        int salaryPerMonth;

    public:
        // Constructor.
        Profession();
        Profession(string position, int salaryPerMonth);

        // Getter and Setter.
        string getPosition();
        void setPosition(string position);
        int getSalaryPerMonth();
        void setSalaryPerMonth(int salaryPerMonth);

        // Destructor.
        ~Profession();
};

class Laptop
{
    private:
        string brand;
        int speedMHz;
```

```
    public:
        // Constructor.
        Laptop();
        Laptop(string brand, int speedMHz);

        // Getter and Setter.
        string getBrand();
        void setBrand(string brand);
        int getSpeedMHz();
        void setSpeedMHz(int speedMHz);

        // Destructor.
        ~Laptop();
};

// Base class: Human.
class Human
{
    private:
        string name;
        char gender;

    public:
        // Constructor.
        Human();
        Human(string name, char gender);

        // Getter and setter.
        string getName();
        void setName(string name);
        char getGender();
        void setGender(char gender);

        // Destructor.
        ~Human();
};
```

```

// Derived class: Adult.
class Adult : public Human
{
    private:
        Profession profession;
        Laptop laptop;

    public:
        // Constructor.
        Adult();
        Adult(string name, char gender);
        Adult(string name, char gender, Profession profession, Laptop laptop);

        // Getter and setter.
        Profession getProfession();
        void setProfession(Profession profession);
        Laptop getLaptop();
        void setLaptop(Laptop laptop);

        // Destructor.
        ~Adult();

};
```

human-adult.cpp

```cpp
#include "header.hh"

/* = = HUMAN SECTION = = */

/* Constructor. */

// Empty constructor.
Human::Human()
{
    name = "";
    gender = '-';
}

// Constructor with all attributes.
Human::Human(string name, char gender)
{
    this->name = name;
    this->gender = gender;
}

/* Getter and Setter. */

string Human::getName()
{
    return name;
}

void Human::setName(string name)
{
    this->name = name;
}

char Human::getGender()
{
    return gender;
}

```

```cpp
void Human::setGender(char gender)
{
    this->gender = gender;
}

/* Destructor. */

// Leave it blank for now.
Human::~Human()
{

}

/* = = ADULT SECTION = = */

/* Constructor. */

// Empty constructor.
Adult::Adult() : Human()
{

}

// Constructor with base human attribute.
Adult::Adult(string name, char gender) : Human(name, gender)
{

}

// Constructor with all attributes.
Adult::Adult(string name, char gender, Profession profession, Laptop laptop) :
    Human(name, gender)
{
    this->profession = profession;
    this->laptop = laptop;
}

```

```cpp
/* Getter and Setter. */

Profession Adult::getProfession()
{
    return profession;
}

void Adult::setProfession(Profession profession)
{
    this->profession = profession;
}

Laptop Adult::getLaptop()
{
    return laptop;
}

void Adult::setLaptop(Laptop laptop)
{
    this->laptop = laptop;
}

/* Destructor. */

// Leave it blank for now.
Adult::~Adult()
{

}
```

laptop.cpp

```cpp
#include "header.hh"

/* Constructor. */

// Empty constructor.
Laptop::Laptop()
{
    brand = "";
    speedMHz = 0;
}

// Constructor with all attributes.
Laptop::Laptop(string brand, int speedMHz)
{
    this->brand = brand;
    this->speedMHz = speedMHz;
}

/* Getter and Setter. */

string Laptop::getBrand()
{
    return brand;
}

void Laptop::setBrand(string brand)
{
    this->brand = brand;
}

int Laptop::getSpeedMHz()
{
    return speedMHz;
}

void Laptop::setSpeedMHz(int speedMHz)
{
    this->speedMHz = speedMHz;
}

/* Destructor. */

// Leave it blank for now.
Laptop::~Laptop()
{

}
```

profession.cpp

```cpp
#include "header.hh"

/* Constructor. */

// Empty constructor.
Profession::Profession()
{
    position = "";
    salaryPerMonth = 0;
}
```

```cpp

// Constructor with all attributes.
Profession::Profession(string position, int salaryPerMonth)
{
    this->position = position;
    this->salaryPerMonth = salaryPerMonth;
}

/* Getter and Setter. */

string Profession::getPosition()
{
    return position;
}

void Profession::setPosition(string position)
{
    this->position = position;
}

int Profession::getSalaryPerMonth()
{
    return salaryPerMonth;
}

void Profession::setSalaryPerMonth(int salaryPerMonth)
{
    this->salaryPerMonth = salaryPerMonth;
}

/* Destructor. */

// Leave it blank for now.
Profession::~Profession()
{

}
```

main.cpp

```cpp
#include "header.hh"

int main()
{
    // So, what kind of codes are they?
    // Well, figure it out by yourself!

    Laptop laptop("ASUS", 3200);
    Profession profession("Programmer", 10000000);

    Adult adult("Rain", 'L', profession, laptop);

    // Print.
    cout << "Name : " << adult.getName() << '\n';
    cout << "Gender : " << adult.getGender() << "\n\n";
    cout << adult.getName() << "'s job is : "
            << adult.getProfession().getPosition() << '\n';
    cout << adult.getName() << "'s salary per year is : "
            << (adult.getProfession().getSalaryPerMonth() * 12) << '\n';
    cout << adult.getName() << "'s laptop and its speed is : "
            << adult.getLaptop().getBrand() << " - "
            << adult.getLaptop().getSpeedMHz() << '\n';

    return 0;
}
```

Latihan

Buatlah program berbasis OOP menggunakan bahasa pemrograman C++ dan Python yang mengimplementasikan konsep *inheritance* dan *composition* pada kelas-kelas tersebut:

- **Mahasiswa**: NIM, nama, jenis_kelamin, fakultas, prodi
- **Human**: NIK, nama, jenis_kelamin
- **SivitasAkademik:** asal_universitas, email_edu
- **Dosen**: NIP, nama, jenis_kelamin, fakultas, prodi, pend_terakhir, keahlian
- **Course**: nama_matakuliah, dosen, mahasiswa, prodi
- **Program Studi**: nama_prodi, kode, mahasiswa, dosen

Note.

- Boleh menambahkan properti/atribut baru
- Tampilkan data selengkap-lengkapnya dalam bentuk list/tabel
- Program dikumpulkan pada repository GitHub yang dibuat public dengan nama "LATIHAN3DPBO2023"
  - Hanya program pada branch **main** yang akan dinilai dan diperiksa
  - Jika waktu pengumpulan sudah habis dan ingin mengupdate kode program, update pada branch lain karena mengupdate branch Main setelah waktu pengumpulan terlewat maka program tidak akan dinilai
- Struktur folder

```
- CPP
  - program
  - screenshot
+ Python
+ PHP
README.md
```

- File README berisi desain program, penjelasan alur, dan dokumentasi saat program dijalankan (screenshot/screen record)
- Submit link repository pada form berikut: https://forms.gle/rvb1hKxbQVuYNbhKA

Penutup

Terima kasih atas kerja sama seluruh pihak yang membantu dalam penyusunan modul ini, semoga apa yang telah didapatkan bisa bermanfaat di masa yang akan datang.

Daftar Pustaka

Sukamto, Rosa A. (2018). *Pemrograman Berorientasi Objek*. Bandung: Modula.

Asisten Pemrograman 11. (2022). *Modul Desain dan Pemrograman Berorientasi Objek*.