Tugas Besar IF2210 Pemrograman Berorientasi Objek

Engimon Factory

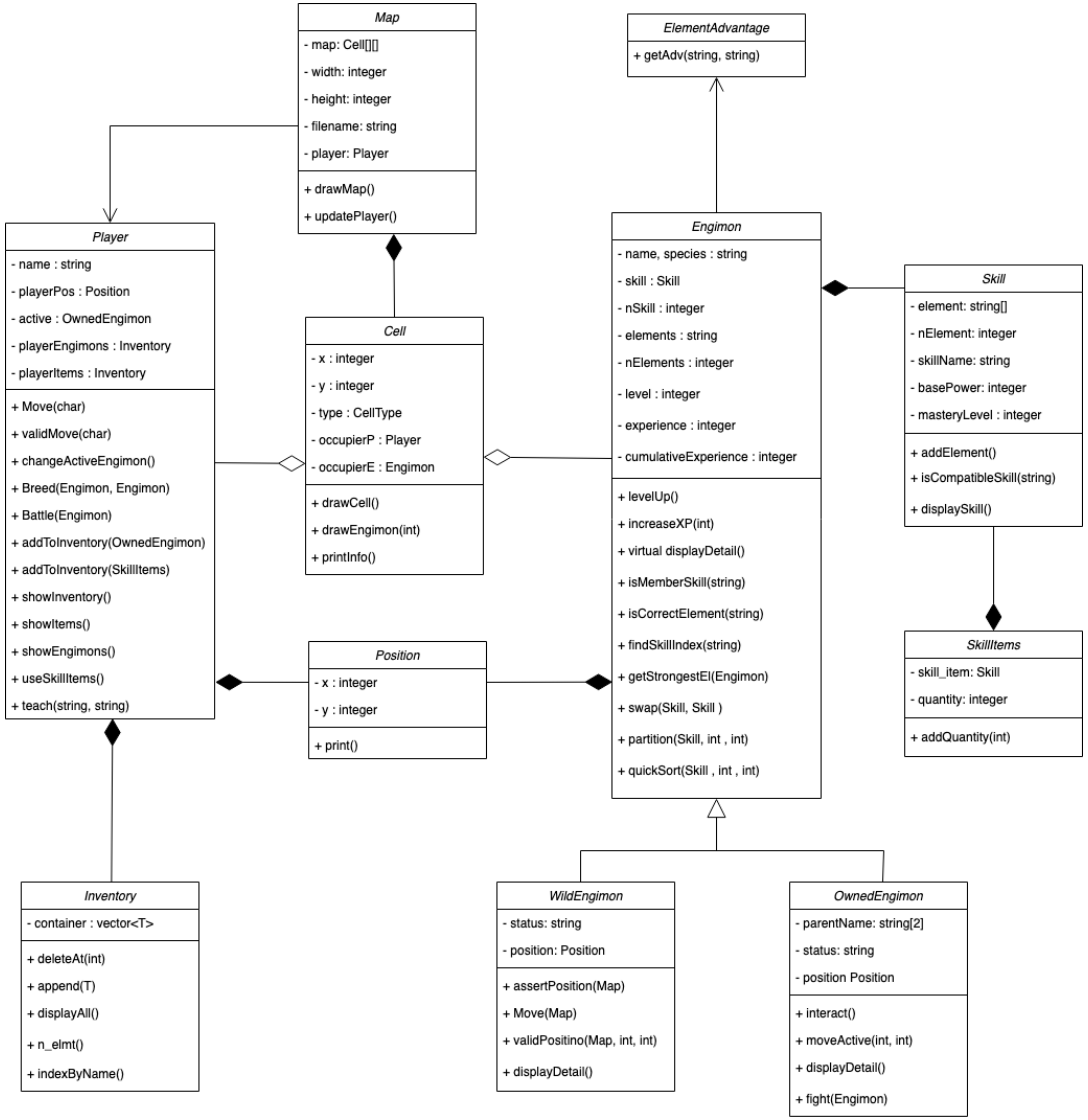
Kelompok Danan For President
Kelas : 02

Helkia Yeremia	13519056
Aditya Bimawan	13519064
Daru Bagus Dananjaya	13519080
Jordan Daniel Joshua	13519098
Aulia Adila	13519100
Shifa Salsabiila	13519106

Asisten Pembimbing :
Muhammad Nurdin Husen

PROGRAM STUDI SARJANA TEKNIK INFORMATIKA
SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG
2021

1. Diagram Kelas



Di antara sejumlah jenis diagram UML, terdapat 2 kategori besar, yaitu *structure diagram* dan *behavioral diagram.* Diagram yang dipilih adalah *Class Diagram*, karena struktur diagram ini sangat cocok dengan paradigma berorientasi objek, yang dapat merepresentasikan kelas dalam sebuah sistem, atribut, operasi dan hubungan antar kelas. Kelebihan diagram kelas yang dipilih terdapat pada kemampuan tipe diagram yang dapat menggambarkan konsep paradigma berorientasi objek dengan baik. Kekurangan diagram ini adalah terbatasnya keterhubungan antar objek atau atribut dalam modul yang dibuat. Kendala yang dialami selama mendesain kelas OOP ini adalah penyesuaian desain modul dan kelas terhadap ketentuan dan kebutuhan Tugas Besar. Terdapat banyak penyesuaian yang harus dilakukan selama proses implementasi kelas dan method berlangsung. Hubungan antar modul yang dibuat harus sedemikian rupa sehingga mencegah redundansi kode, dekomposisi method yang kurang baik, dan struktur kelas yang rumit dan sulit dipahami.

2. Penerapan Konsep OOP

2.1. Inheritance & Polymorphism

```
#ifndef _OWNED_ENGIMON_HPP_
#define _OWNED_ENGIMON_HPP_

#include <string>
#include <iterator>
#include <map>
#include "Position.hpp"
#include "Engimon.hpp"
    You, 2 days ago | 1 author (You)
    class OwnedEngimon : public Engimon {
    private:
        string *parentName;
        string status;
        Position position;
        //mapping species dengan message unik pake stl
        static map<string, string> percakapan;



    public:
        OwnedEngimon();
        OwnedEngimon(string name, string species);
        OwnedEngimon(const OwnedEngimon& oe);
        OwnedEngimon& operator=(const OwnedEngimon& oe);
        ~OwnedEngimon();

        string* getParentName();
        void setParentName(string,string);

        string getStatus();
        void setStatus(string);
```

```
#ifndef _WILD_ENGIMON_HPP_
#define _WILD_ENGIMON_HPP_

#include <string>
#include <iostream>
#include <map>
#include "Engimon.hpp"
#include "Skill.hpp"
#include "Map.hpp"

using namespace std;
// using namespace EngimonFactory;
You, seconds ago | 2 authors (You and others)
class WildEngimon : public Engimon {
   private:
    string status;
    Position position;
    int element2int(string element);
    static map<string, string> spesiesSkill;

   public:
    WildEngimon();
    WildEngimon(string species, string element, int level, int x, int y, Map* m);
    ~WildEngimon();
    void operator=(const WildEngimon&);

    string getStatus();
    void setStatus(string);
```

WildEngimon dan OwnedEngimon merupakan kelas turunan dari Engimon yang merupakan abstrack base class. Inheritance ini dilakukan karena terdapat dua tipe Engimon, yaitu Engimon aktif yang dimiliki player (OwnedEngimon) dan Engimon liar (WildEngimon) yang method dan atributnya diturunkan dari class Engimon. Dengan demikian, tidak perlu ada deklarasi ulang yang dilakukan oleh WildEngimon dan OwnedEngimon (tidak terjadi DRY Don't Repeat Yourself). Inheritance juga memberikan efek polymorphism pada objek-objek kelas turunan, sebab objek-objek tersebut memiliki sifat sebagai kelas tersebut dan sekaligus kelas dasarnya.

```
float Engimon::getStrongestEl(Engimon& enemy) {
    float strongest = ElementAdvantage::getAdv(this->getElements()[0], enemy.getElements()[0]);

    for (int i = 1; i < this->getNElements(); i++) {
        for (int j = 1; j < enemy.getNElements(); j++) {
            float temp = ElementAdvantage::getAdv(this->getElements()[i], enemy.getElements()[j]);
            if (temp > strongest) {
                strongest = temp;
            }
        }
    }

    return strongest;
}
```

Fungsi getStrongestEl di atas memanfaatkan polymorphism dengan menerima parameter berupa *reference* pada *abstract base class* Engimon. Dengan melakukan hal tersebut, fungsi getStrongestEL bisa menerima *derived class* dari Engimon (OwnedEngimon dan WildEngimon) sebagai argumen, dan *derived class* tersebut akan berperilaku selayaknya sebuah Engimon dalam *scope* fungsi getStrongestEl.

2.2. Method/Operator Overloading

```
void addToInventory(OwnedEngimon el);
void addToInventory(SkillItems el);
```

Penerapan function overloading terdapat pada contoh kode diatas, di mana prosedur addToInventory di overloading dengan 2 jenis parameter berbeda, yaitu OwnedEngimon dan SkillItems. Keuntungan penggunaan method overloading adalah deklarasi method yang tidak perlu berulang, sehingga dapat digunakan untuk beberapa jenis parameter berbeda.

2.3. Template & Generic Classes

```
#ifndef _INVENTORY_HPP_
#define _INVENTORY_HPP_

#include <string>
#include <iostream>
#include <vector>
#include "SkillItems.hpp"
using namespace std;

You, 3 days ago | 1 author (You)
template <class T>
class Inventory {
    public:
        T operator[](int i){
            return container.at(i);
        }
        T deleteAt(int i){
            T temp = container.at(i);
            container.erase(container.begin()+i);
            return temp;
        }
        void append(T el){
            container.push_back(el);
        }
```

```
template<>
inline int Inventory<SkillItems>::n_elmt() const{
    int sum = 0;
    for (int i = 0; i < container.size(); i++) {
        sum += container.at(i).getQuantity();
    }

    return sum;
}
```

```
class Player {
    private :
        string name;
        Position playerPos;
        OwnedEngimon active;
        // int activeIndex;
        Inventory<OwnedEngimon> playerEngimons;
        Inventory<SkillItems> playerItems;
```

Berikut adalah contoh template & generic class, yang dapat memberi kemampuan untuk dapat *pass* tipe data sebagai parameter, sehingga tidak perlu menuliskan kode yang sama untuk tipe data yang berbeda. Pada class Inventory, tipe data yang disimpan dalam container bisa beragam, sehingga konsep template class ini sangat cocok untuk diaplikasikan. Manfaat dari penggunaan template class dapat dilihat pada class Player. Dengan membuat class Inventory sebagai template class, player dapat memiliki dua inventory yang masing-masing menyimpan class berbeda, yaitu OwnedEngimon dan SkillItems.

Pada template class juga bisa dilakukan specialization, yaitu mendefinisikan perilaku khusus pada class tertentu. Pada class Inventory dibuat specialization pada class SkillItems. Diimplementasikan sebuah specialization karena SkillItems memiliki field quantity, dalam menghitung jumlah element tidak bisa hanya menggunakan size dari vector container.

2.4. Exception

```
Map::Map(string _filename, Player& _player) {
    player = _player;
    filename = _filename;
    string line_input;
    int i = 0;

    fstream file(filename);
    if (!file) {
        throw "File not found!";
    }

    while (file && i < MAP_HEIGHT - 1) {
        getline(file, line_input);
        if (line_input.size() > MAP_WIDTH) {
            throw "Board size invalid";
        }
        for (size_t j = 0; j < line_input.size(); j++) {
            CellType type = getType(line_input[j]);
            Cell cell(i, j, type);
            map[i][j] = cell;
        }
        i++;
    }
}
```

```
int main() {

    try {

        Player p("danan");
```

```
    catch (const char* c) {
        cout << "Exception: " << c << endl;
    }

    return 0;
}
```

Exception merupakan sebuah konsep yang dapat meng-handle error dengan ‘elegan’. t*ry* merepresentasikan block kode yang melempar exception. *catch* merepresentasikan block kode yang akan di eksekusi ketika sebuah exception dilempar. *throw* adalah untuk melempar exception. Dengan konsep ini, programmer dapat meng-handle berbagai jenis error secara terpisah sesuai keinginan. Jenis error dapat dikelompokkan dalam jenis-jenis yang diinginkan. Bagian kode di atas menunjukkan *throw exception*, kemudian *try catch* di implementasikan pada program main.

2.5. C++ Standard Template Library

```
template <class T>
class Inventory {
    private:
        vector<T> container;
```

Standard Template Library (STL) adalah set dari template class yang disediakan oleh C++ yang menyediakan struktur data dan fungsi yang umum digunakan, seperti list, stack, array, dll. STL vector digunakan dalam Inventory karena dengan memanfaatkan vector, tidak perlu membuat banyak implementasi, seperti ukuran yang dinamis, penggeseran elemen ke kiri setiap ada penghapusan di tengah, dan pemanggilan dtor, cctor, dan assignment operator pada penambahan, penghapusan, dan akses elemen.

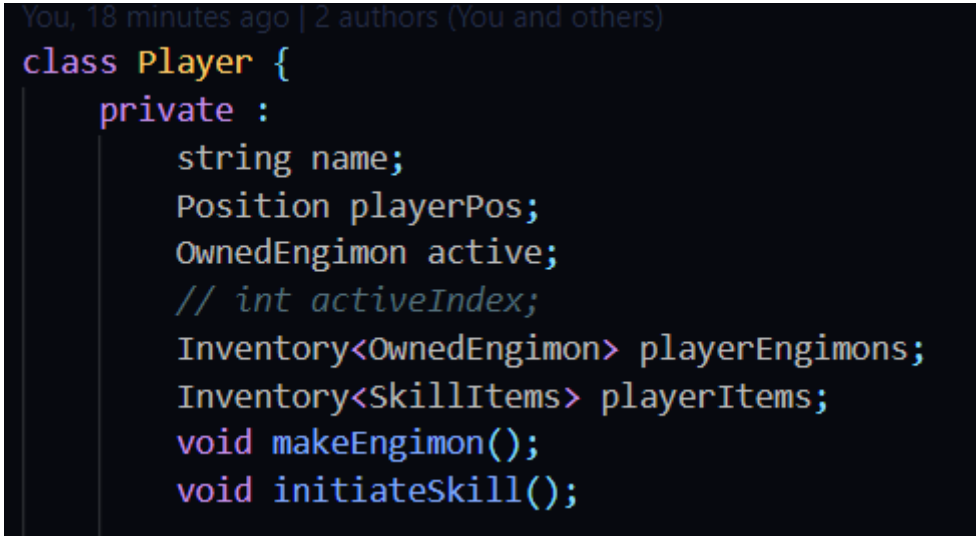
2.6. Konsep OOP lain

1. Abstract Base Class

Class Engimon merupakan penerapan dari Abstrack Base Class karena memiliki fungsi *pure virtual* pada displayDetail(), getStatus(), dan setStatus(string). Ketiga fungsi/prosedur tersebut belum diketahui implementasinya pada *base class*, dan akan diimplementasikan di *derived class*nya, yaitu pada WildEngimon dan OwnedEngimon.

2. Composition

Sebuah kelas (misal A) memiliki kelas lain (misal B, yang dianggap sebagai objek) sebagai *data member* kelas (A) tersebut. Dapat dikatakan bahwa A mengandung B, sehingga kelas A tidak dapat hidup tanpa B. Penerapan konsep ini terdapat pada class Player dan class Inventory, di mana Player mengandung Inventory, sehingga kedua objek tersebut memiliki hubungan *composition*. Begitupula dengan class Position dan OwnedEngimon yang turut meng-komposisi class Player.



Selain hubungan *composition* yang terdapat dalam class Player, konsep ini juga diterapkan pada class Map. Map dikomposisi (*contains*) oleh Cell, sehingga cell akan hilang jika Map dihapus.

```
class Map {
private:
    Cell map[MAP_HEIGHT][MAP_WIDTH];
    int width;
    int height;
    string filename;
    Player player;
```

3. Association

Association adalah sebuah relasi atau hubungan antar dua atau lebih objek di mana masing-masing objek memiliki *lifetime* masing-masing, serta tidak ada hubungan kepemilikan antar objek tersebut. Contohnya adalah class Map yang memiliki relasi dengan class Player, dengan tujuan untuk mengambil data objek Player tanpa memiliki *ownership* dari Player.

```
class Map {
private:
    Cell map[MAP_HEIGHT][MAP_WIDTH];
    int width;
    int height;
    string filename;
    Player player;
```

4. Aggregation

Aggregation adalah sebuah relasi antar dua atau lebih objek dengan masing-masing objek memiliki *lifetime* masing-masing, namun ada kepemilikan (*ownership*) antar objek. Contoh implementasi konsep ini terdapat dalam class Cell. Terdapat atribut occupierP dengan tipe objek Player dan occupierE yang merupakan pointer terhadap objek Engimon. Pada implementasi ini, cell memiliki *ownership* terhadap occupierP

dan occupierE, namun keduanya tidak saling menghambat siklus hidup satu sama lain. Hal ini berarti jika Cell dihapus, maka occupierP dan occupierE tetap akan *exist*. Begitu pula jika occupierP atau occupierE dihapus, maka Cell juga akan tetap *exist*.

```
class Cell {
    private:
        int x; //x coordinate
        int y; //y coordinate
        CellType type; //Grassland or sea
        Player occupierP;
        Engimon* occupierE;
```

3. Bonus Yang dikerjakan

3.1. Bonus yang diusulkan oleh spek

3.1.1. Dual Element Breeding

Pada dual element breeding, pendekatan yang kami gunakan yaitu mengambil elemen pertama dari masing-masing parent, kemudian melakukan breeding seperti biasa. Apabila hasil dari dual element breeding menghasilkan engimon yang hanya memiliki 1 elemen yaitu elemen dari induk yang memiliki dual element maka spesies dari elemen tersebut tidak mengikuti spesies induk yang dual element melainkan digenerate random sesuai dengan spesies yang cocok dengan elemen engimon tersebut.

3.1.2. Purely Random Wild Engimon Generation

Posisi, level, dan spesies dari engimon digenerate di awal program secara random menggunakan library cstdlib dan ctime. Adapun spesies dari engimon digenerate berdasarkan lokasi dari Engimon. Engimon yang posisinya terdapat di air memiliki jenis spesies yang berelemen air atau es. Sedangkan engimon yang memiliki posisi di grassland adalah spesies engimon yang memiliki elemen api, listrik, dan tanah.

4. Pembagian Tugas

Modul (dalam poin spek)	Designer	Implementer
1.a. Engimon memiliki bermacam-macam spesies	13519056	13519056
1.b. Suatu spesies memiliki tipe elemen yang konsisten	13519064	13519064
1.c. Setiap spesies memiliki beberapa informasi wajib	13519080	13519080
1.d. Engimon level up	13519098	13519098
1.e. Engimon mati jika mencapai nilai maksimum cumulative experience	13519100	13519100
1.f. Minimal dibuat 1 jenis spesies untuk setiap elemen dan untuk setiap kombinasi elemen	13519106	13519106
2.a. Memiliki banyak jenis skill	13519056	13519056
2.b. Setiap skill memiliki informasi wajib	13519064	13519064
2.c. Setiap spesies engimon memiliki satu skill bawaan yang unik	13519080	13519080
3.a. Player memiliki beberapa commands yang bisa dipilih pemain	13519098	13519098
3.b. Player memiliki inventory.	13519100	13519100
3.c. Player memiliki satu active engimon.	13519106	13519106
4.a. Menghitung power	13519056	13519056

4.b. Menampilkan total power level dari kedua engimon yang bertarung	13519064	13519064
4.c. Element advantage	13519080	13519080
4.d. Kondisi jika engimon player kalah	13519098	13519098
4.e. Kondisi jika engimon player menang	13519100	13519100
4.f. Battle antara multiple engimon	13519106	13519106
5.a. Memulai breeding	13519056	13519056
5.b. Update level parent setelah breeding	13519064	13519064
5.c. Player memberi nama anak hasil breeding	13519080	13519080
5.d Inherit Skill	13519098	13519098
5.e. Resulting Child Species & Element	13519100	13519100
6.a. Engimon dikeluarkan dalam peta dengan huruf besar	13519106	13519106
6.b . Engimon dikeluarkan dalam peta dengan huruf kecil	13519056	13519056
6.c. Huruf dari engimon ditampilkan tergantung elemennya	13519064	13519064
6.d. Peta berbentuk tile dengan 2 lingkungan	13519080	13519080
6.e Player menempati tile apapun	13519098	13519098
6.f. Engimon liar bergerak secara random setiap X turn	13519100	13519100
6.g. Maksimal engimon yang dapat di spawn	13519106	13519106

6.h Engimon yang ada dispawn secara random	13519056	13519056
6.i. Engimon bergerak secara random di peta sesuai wilayah	13519064	13519064
6.j Load peta melalui file eksternal	13519080	13519080
6.k Penanganan kasus player dan wild bergerak ke tile yang sama	13519098	13519098
Bonus 1: Dual Element Breeding	13519100	13519100
Bonus 2 : Purely Random Wild Engimon Generation	13519106	13519106