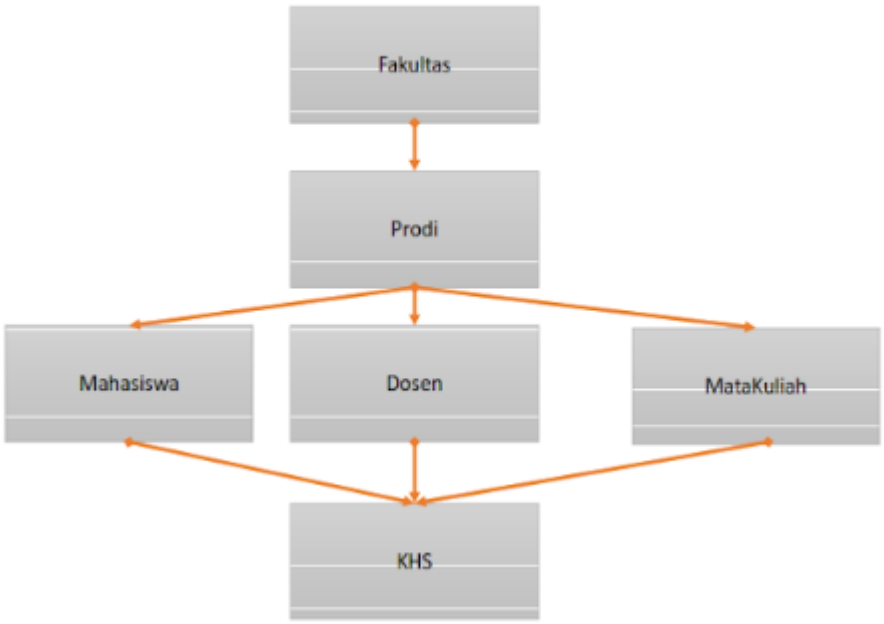
UTS PBO

Nama : Fikri Ainun Najib

Nim :200411100153

Kelas :PBO C



Dari Class Diagram Berikut buatlah Relasi Kelas C++ dengan memberikan contoh dan penjelasan implementasi pada deskripsi kelas tersebut.

1. **Enkapsulasi** Obyek ada suatu cara atau metode yang dimana class pada C++ mengatur property pada class tersebut disembunyikan atau privasi, sehingga jika ingin mengakses proterty tersebut maka dari class itu sendiri dan class lain tidak bisa mengaksesnya. Pada pemrograman C++ enkapsulasi dapat di temukan dengan perintah private pada class, dan jika ingin mengakses property tersebut maka dengan perintah getproperty pada public property.

Contoh disini pada class mahasiswa:

```
class Mahasiswa {
    int counter;
    public:
        string getName();
        string getNim();
    private:
        string nim;
```

```
        string name;
};
```

Pada contoh diatas, property yang disembunyikan yaitu nim dan name, sehingga property tersebut tidak bisa diakses oleh class lain, selain class itu sendiri. Dan untuk mengakasesnya memakai perintah string getname() atau string getnim().

2. **Konstruksi** objek adalah anggota khusus pada class atau function yang dimana akan dijalankan secara otomatis jika property tersebut dipanggil atau pada saat proses instansiasi. Pada Bahasa C++ beragam bentuk property yang dapat dijalankan otomatis Ketika dipanggil seperti cout, object itu sendiri dan sebagainya
Contoh pada class mahasiswa:

```
class Mahasiswa {
    public:
        friend class Prodi;
        explicit Mahasiswa(string nm="", string
nama=""):nim(nm),name(nama){};
};
mhs = new Mahasiswa(nim,name);
```

Pada contoh diatas expilicit akan dijalankan Ketika property tersebut di panggil, yang dimana membentuk objek baru yaitu mahasiswa

Dekonstruksi Obyek adalah suatu anggota khusus dari suatu class yang dimana anggota khusus tersebut dijalankan secara otomatis dan akan menyebabkan objek atau property akan dihapus, hal ini agar tidak ada penumpukan objek pada memori jika objek tersebut dimasukkan ke dalam memori, pada pemrograman C++ dekonstruksi objek ini dapat ditemukan dengan perintah awalnya “**~**” .
Contoh pada Class Mahasiswa:

```
Class Mahasiswa {
    public:
        virtual ~Mahasiswa();
};
```

Pada contoh diatas, objek mahasiswa diberikan tanda “**~**” , agar tidak ada penumpukan objek /memori pada saat membuat data mahasiswa baru lagi. Dan arti perintah virtual didepannya, objek tersebut dapat didefinisikan ulang pada kelas turunannya.

3. **Agregasi/Koleksi** Obyek adalah hubungan antar objek yang dimana kedua objek tersebut terpisah,namun dihubungkan. Kedua objek tersebut sebenarnya bisa berdiri jadi tidak bergantungan satu sama yang lainnya. Implementasi pada C++ biasanya berhubungan dengan pointer.
Contoh pada class Prodi:

```
class Prodi {
    int counterr;
    public:
        explicit Prodi(string kode="01",string
prodi="Mahasiswa"):code(kode),programs(prodi){};
        Prodi(const Prodi& orig);
        virtual ~Prodi();
        void setMahasiswa(Mahasiswa *mahasiswa);
        void printMhs();
        Mahasiswa *getMahasiswa();
        string getCode();
        string getPrograms();
        void setCounter();
        int getCounter();
        void setListKhs(list<Mahasiswa> lmhs);
        void setListKhs(Mahasiswa mhs);
        void printListKhs();
        list<Mahasiswa> getListKhs();
    private:
        string code;
        string programs;
        Mahasiswa *mahasiswa[1000];
        list<Mahasiswa> lMhs;
};
```

Pada contoh diatas, class prodi membentuk objek baru didalamnya yaitu *mahasiswa yang dimana satu prodi memiliki banyak mahasiswa.

4. **Abstraksi Inheritance** Obyek adalah suatu proses pembentukan kelas baru dengan mewarisi karakteristik yang ada dari parent ke children yang memungkinkan kelas yang terbentuk memiliki sifat yang karakteristik. Namun pada abstrak inheritance, sifat yang diturunkan bersifat abstrak atau tidak jelas. Abstraksi inheritance ini sangat jarang digunakan daripada inheritance sendiri.
Contoh pada Mahasiswa:

```
class sifat: public mahasiswa {}
```

jika inheritance sendiri biasa seperti diatas, yang dimana setiap mahasiswa memiliki sifat tertentu.

5. **Polymorfism Overloading** adalah suatu Teknik untuk membuat suatu kelas baru dengan nama yang sama, namun isi, tipe data dan parameter yang berbeda.overloading lebih digunakan jika ingin sesuatu yang statis. Biasanya pada pemrograman C++ memiliki nama yang sama, namun memiliki return yang berbeda

Contoh pada kelas mahasiswa dan prodi:

```
Mahasiswa *mhs=new Mahasiswa();

Prodi *pd=new Prodi();

mhs = new Mahasiswa("A","Rizal");

pd->setListKhs(*mhs);

mhs = new Mahasiswa("B","CACA");

pd->setListKhs(*mhs);
```

Pada contoh diatas, kelas mahasiswa dapat mengembalikan yang berbeda beda tergantung parameter. Dan dimasukkan pada list prodi.

Overriding Function sama seperti sebelumnya seperti Teknik untuk membuat suatu kelas baru. Namun parameter yang kosong, biasanya digunakan jika sifatnya dinamis. biasanya pada pemrograman C++ dapat di temukan jika ingin menampilkan data.

Contoh pada kelas mahasiswa,dengan situasi pada vector berisikan data:

```
int main(){
     Mahasiswa *mhs=new Mahasiswa();
     vector<Mahasiswa> vecMhs;
     cout<<"Nim : "<<vecMhs[i].getNim()<<endl;
     cout<<"Name : "<<vecMhs[i].getName()<<endl;
   return 0;
}
#output
Input Nim : 01
Input Name : fikri
```

FULL CODE

```
#include <iostream>
#include <list>
#include <vector>
#include <algorithm>
using namespace std;

// make class khs ,prodi, and mahasiswa
class Khs {
    public:
        explicit Khs(string nm="", string mk="", string nilai="",string
dosen=""):nim(nm),course(mk),grade(nilai),namadosen(dosen){};
        // Khs(const Khs& orig);
        virtual ~Khs();
        friend class Mahasiswa;
        string getCourse();
        string getGrade();
        string getNim();
        string getNamadosen();
        void setCourse(string c);
        void setGrade(string g);
    private:
            string nim;
            string course;
            string grade;
            string namadosen;
};


class Mahasiswa {
    int counter;
    public:
        friend class Prodi;
        // Mahasiswa();
        explicit Mahasiswa(string nm="", string nama=""):nim(nm),name(nama){};
        Mahasiswa(const Mahasiswa& orig);
        virtual ~Mahasiswa();
        void setKhs(Khs *khs);
        void printKhs();
        Khs *getKhs();
        string getName();
        string getNim();
        void setName(string n);
```

```
        void setNim(string n);
        void setCounter();
        int getCounter();
        void setListKhs(list<Khs> lkhs);
        void setListKhs(Khs khss);
        void printListKhs();
        list<Khs> getListKhs();
    private:
        string nim;
        string name;
        Khs *khs[1000];
        list<Khs> lKhs;
};

class Dosen {
    int counterrr;
    public:
        explicit Dosen(string nm="", string nama=""):nidn(nm),name(nama){};
        Dosen(const Dosen& orig);
        virtual ~Dosen();
        void setKhs(Khs *khs);
        void printKhs();
        Khs *getKhs();
        string getName();
        string getNim();
        void setName(string n);
        void setNim(string n);
        void setCounter();
        int getCounter();
        void setListKhs(list<Khs> lkhs);
        void setListKhs(Khs khss);
        void printListKhs();
        list<Khs> getListKhs();
    private:
        string nidn;
        string name;
        Khs *khs[1000];
        list<Khs> lKhs;
};



class Prodi {
    int counterr;
    public:
```

```
        explicit Prodi(string kode="01",string
prodi="Mahasiswa"):code(kode),programs(prodi){};
        Prodi(const Prodi& orig);
        virtual ~Prodi();
        void setMahasiswa(Mahasiswa *mahasiswa);
        void printMhs();
        Mahasiswa *getMahasiswa();
        string getCode();
        string getPrograms();
        void setCounter();
        int getCounter();
        void setListKhs(list<Mahasiswa> lmhs);
        void setListKhs(Mahasiswa mhs);
        void printListKhs();
        list<Mahasiswa> getListKhs();
    private:
        string code;
        string programs;
        Mahasiswa *mahasiswa[1000];
        list<Mahasiswa> lMhs;
};

class Fakultas {
    int counterrrr;
    public:
        explicit Fakultas( string nm="", string kode="", string program="" ,
string fakultas=""): nim(nm), code(kode),caca(program) ,faculty(fakultas){};
        Fakultas(const Fakultas& orig);
        virtual ~Fakultas();
        void setProdi(Prodi *prodi);
        void printProdi();
        // Prodi *getProdi();
        string getNim();
        string getCode();
        string getFaculty();
        string getProdi();
        string getCaca();
        void setCounter();
        int getCounter();
        void setListKhs(list<Prodi> lprodi);
        void setListKhs(Prodi prodi);
        void printListKhs();
        list<Prodi> getListKhs();
    private:
        string nim;
```

```
        string code;
        string faculty;
        string prody;
        string caca;
        Prodi *prodi[1000];
        list<Prodi> lProdi;
};

// make function for class khs
string Khs::getCourse(){
    return course;
}
string Khs::getNim(){
    return nim;
}

string Khs::getNamadosen(){
    return namadosen;
}

string Khs::getGrade(){
    return grade;
}
Khs::~Khs(){
}


void Khs::setCourse(string c){
    course=c;
}

void Khs::setGrade(string g){
    grade=g;
}

// make function for class mahasiswa
// Mahasiswa::Mahasiswa(){
//     counter=0;
// }

Mahasiswa::Mahasiswa(const Mahasiswa& orig){
    this->nim=orig.nim;
    this->name=orig.name;
}
```

```
Mahasiswa::~Mahasiswa(){
}



void Mahasiswa::setKhs(Khs *khs){
    this->khs[counter]=khs;
    counter++;
}

void Mahasiswa::printKhs(){
    for(int i=0;i<counter;i++){
        cout<<"Course : "<<khs[i]->getCourse()<<endl;
        cout<<"Grade : "<<khs[i]->getGrade()<<endl;
    }
}

Khs *Mahasiswa::getKhs(){
    return *khs;
}

void Mahasiswa::setName(string n){
    name=n;
}

void Mahasiswa::setNim(string n){
    nim=n;
}
string Mahasiswa::getName(){
    return name;
}

string Mahasiswa::getNim(){
    return nim;
}

void Mahasiswa::setCounter(){
    counter++;
}

int Mahasiswa::getCounter(){
    return counter;
}

void Mahasiswa::setListKhs(list<Khs> lkhs){
```

```
    lKhs=lkhs;
}

void Mahasiswa::setListKhs(Khs khss){
    lKhs.push_back(khss);
}

void Mahasiswa::printListKhs(){

    for(list<Khs>::iterator ti = lKhs.begin();ti!=lKhs.end();++ti){
        cout<<"Course : "<<ti->getCourse()<<endl;
        cout<<"Grade : "<<ti->getGrade()<<endl;
    }
}

list<Khs> Mahasiswa::getListKhs(){
    return lKhs;
}
// make function for class dosen
Dosen::Dosen(const Dosen& orig){
    this->nidn=orig.nidn;
    this->name=orig.name;
}

Dosen::~Dosen(){
}

void Dosen::setKhs(Khs *khs){
    this->khs[counterrr]=khs;
    counterrr++;
}

void Dosen::printKhs(){
    for(int i=0;i<counterrr;i++){
        cout<<"Course : "<<khs[i]->getCourse()<<endl;
        cout<<"Grade : "<<khs[i]->getGrade()<<endl;
    }
}

Khs *Dosen::getKhs(){
    return *khs;
}

void Dosen::setName(string n){
    name=n;
```

```
}

void Dosen::setNim(string n){
    nidn=n;
}

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

string Dosen::getNim(){
    return nidn;
}

void Dosen::setCounter(){
    counterrr++;
}

int Dosen::getCounter(){
    return counterrr;
}

void Dosen::setListKhs(list<Khs> lkhs){
    lKhs=lkhs;
}

void Dosen::setListKhs(Khs khss){
    lKhs.push_back(khss);
}

void Dosen::printListKhs(){

    for(list<Khs>::iterator ti = lKhs.begin();ti!=lKhs.end();++ti){
        cout<<"Course : "<<ti->getCourse()<<endl;
        cout<<"Grade : "<<ti->getGrade()<<endl;
    }
}

list<Khs> Dosen::getListKhs(){
    return lKhs;
}



// make function for class prodi
```

```
Prodi::Prodi(const Prodi& orig){
    this->code=orig.code;
    this->programs=orig.programs;
}

Prodi::~Prodi(){
}

void Prodi::setMahasiswa(Mahasiswa *mahasiswa){
    this->mahasiswa[counterr]=mahasiswa;
    counterr++;
}

string Prodi::getCode(){
    return code;
}

string Prodi::getPrograms(){
    return programs;
}

void Prodi::setCounter(){
    counterr++;
}
int Prodi::getCounter(){
    return counterr;
}

Mahasiswa *Prodi::getMahasiswa(){
    return *mahasiswa;
}
// get mahasiswa

void Prodi::printMhs(){
    for(int i=0;i<counterr;i++){
        cout<<"Nim : "<<mahasiswa[i]->getNim()<<endl;
        cout<<"Name : "<<mahasiswa[i]->getName()<<endl;
    }
}

void Prodi::setListKhs(list<Mahasiswa> lmhs){
    lMhs=lmhs;
}

void Prodi::setListKhs(Mahasiswa mhs){
```

```
    lMhs.push_back(mhs);
}

void Prodi::printListKhs(){
    for(list<Mahasiswa>::iterator it=lMhs.begin();it!=lMhs.end();++it){
        cout<<"Nim : "<<it->getNim()<<endl;
        cout<<"Name : "<<it->getName()<<endl;
    }
}

list<Mahasiswa> Prodi::getListKhs(){
    return lMhs;
}

// make function for class fakultas
Fakultas::Fakultas(const Fakultas& orig){
    this->nim=orig.nim;
    this->code=orig.code;
    this->faculty=orig.faculty;
    this->prody=orig.prody;
    this->caca=orig.caca;
}

Fakultas::~Fakultas(){
}


// get nim di faktultas
// void Fakultas::setProdi(Prodi *prodi){
//     this->prodi[counter]=prodi;
//     counter++;
// }

string Fakultas::getNim(){
    return nim;
}

string Fakultas::getCode(){
    return code;
}

string Fakultas::getFaculty(){
    return faculty;
}
string Fakultas::getCaca(){
```

```
    return caca;
}

string Fakultas::getProdi(){
    return prody;
}

void Fakultas::setCounter(){
    counterrrr++;
}


int Fakultas::getCounter(){
    return counterrrr;
}

void Fakultas::setListKhs(list<Prodi> lprodi){
    lProdi=lprodi;
}

void Fakultas::setListKhs(Prodi prodi){
    lProdi.push_back(prodi);
}

void Fakultas::printListKhs(){
    for(list<Prodi>::iterator it=lProdi.begin();it!=lProdi.end();++it){
        cout<<"Code : "<<it->getCode()<<endl;
        cout<<"Programs : "<<it->getPrograms()<<endl;
    }
}

list<Prodi> Fakultas::getListKhs(){
    return lProdi;
}


// make main function

int main(){
    Khs *khs=new Khs();
    Mahasiswa *mhs=new Mahasiswa();
    Dosen *dsn=new Dosen();
    Prodi *prodi=new Prodi();
    Fakultas *fakultas=new Fakultas();
```

```
    int pilih,n,pilih2,cek=0,pilih3,iter;
    string nim,name,matkul,nilai,nidn,namaDosen,code,pd,faculty;

    vector<Khs> vecKhs;
    vector<Mahasiswa> vecMhs;
    vector<Fakultas> vecFakultas;

    while(true){
        cout <<"1.Input Data Khs"<<endl;
        cout <<"2.Tampil Data Khs"<<endl;
        cout <<"3.Exit"<<endl;
        cout <<"Pilih : ";
        cin >> pilih;
        if(pilih==1){
            cout<<"1. Input Data Mahasiswa + khs"<<endl;
            cout<<"2. Input Data Khs saja"<<endl;
            cout<<"Pilih : ";
            cin>>pilih2;
            if(pilih2==1){
                // input data mahasiswa , nim, name and for  khs matkul and nilai
                cout<<"Input Nim : ";
                cin>>nim;
                cout<<"Input Name : ";
                cin>>name;
                // mengecek data
                while(cek < vecMhs.size()){
                    if(vecMhs[cek].getNim()==nim){
                        cek = 100;
                        break;
                    };
                    cek++;
                };
                if(cek ==100){
                        cek = 0;
                        cout<<"\n--Data Sudah Ada--\n"<<endl;
                }else{
                    mhs = new Mahasiswa(nim,name);
                    vecMhs.push_back(*mhs);
                    cout << "Input Data Mahasiswa Berhasil" << endl;
                    cout << "Input semester : ";
                    cin >> code;
                    cout << "Input Prodi : ";
                    cin >> pd;
                    cout << "Input Fakultas : ";
                    cin >> faculty;
```

```
                fakultas = new Fakultas(code,nim,pd,faculty);
                vecFakultas.push_back(*fakultas);
                cout << "Input Berapa Jumlah KHS : ";
                cin >> n;
                for(int i=0;i<n;i++){
                    cout<<"Input Matkul : ";
                    cin>>matkul;
                    cout<<"Input Nilai : ";
                    cin>>nilai;
                    cout <<"Input Nama Dosen : ";
                    cin >> namaDosen;
                    khs = new Khs(nim,matkul,nilai,namaDosen);
                    vecKhs.push_back(*khs);
                }
                cout << "Input Data KHS Berhasil \n" << endl;
            }
        }else if(pilih2==2){
            cout << "Input Nim : ";
            cin >> nim;
            cout << "Input Semester : ";
            cin >> code;
            cout << "Input Prodi : ";
            cin >> pd;
            cout << "Input Fakultas : ";
            cin >> faculty;
            fakultas = new Fakultas(code,nim,pd,faculty);
            vecFakultas.push_back(*fakultas);

            cout << "Input Berapa Jumlah KHS : ";
            cin >> n;
            for(int i=0;i<n;i++){
                cout<<"Input Matkul : ";
                cin>>matkul;
                cout<<"Input Nilai : ";
                cin>>nilai;
                cout <<"Input Nama Dosen : ";
                cin >> namaDosen;
                khs = new Khs(nim,matkul,nilai,namaDosen);
                vecKhs.push_back(*khs);
            }
            cout << "Input Data KHS Berhasil \n" << endl;
        }
    }else if(pilih==2){
        // Tampilkan data mahasiswa, khs, dosen, prodi, fakultas
        cout <<"Data Mahasiswa : "<<endl;
```

```
            if(vecMhs.size()==0){
                cout<<"Data Kosong"<<endl;
            }else{
                for(int i=0;i<vecMhs.size();i++){
                    cout<<"Nim : "<<vecMhs[i].getNim()<<endl;
                    cout<<"Name : "<<vecMhs[i].getName()<<endl;
                    cout<<"KHS : "<<endl;
                    for(int j=0;j<vecKhs.size();j++){
                        if(vecMhs[i].getNim()==vecKhs[j].getNim()){
                            cout<<"Matkul : "<<vecKhs[j].getCourse()<<endl;
                            cout<<"Nilai : "<<vecKhs[j].getGrade()<<endl;
                            cout<<"Nama Dosen :
"<<vecKhs[j].getNamadosen()<<endl;
                        }
                    }
                    cout<<"Fakultas : "<<endl;
                    for(int x=0;x<vecFakultas.size();x++){
                        if(vecMhs[i].getNim()==vecFakultas[x].getCode()){
                            cout<<"Semester : "<<vecFakultas[x].getNim()<<endl;
                            cout<<"Prodi : "<<vecFakultas[x].getCaca()<<endl;
                            cout<<"Fakultas :
"<<vecFakultas[x].getFaculty()<<endl;
                        }
                    }
                }
            }
        }else{
            break;
        }
    };
    return 0;
}
```

Output

```
1. Input Data Mahasiswa + khs
2. Input Data Khs saja
Pilih : 1
Input Nim : 01
Input Name : fikri
Input Data Mahasiswa Berhasil
Input semester : 3
Input Prodi : ti
Input Fakultas : teknik
Input Berapa Jumlah KHS : 1
Input Matkul : pbo
Input Nilai : a
Input Nama Dosen : herma
Input Data KHS Berhasil
1.Input Data Khs
2.Tampil Data Khs
3.Exit
Pilih : 2
Data Mahasiswa :
Nim : 01
Name : fikri
KHS :
Matkul : pbo
Nilai : a
Nama Dosen : herma
Fakultas :
01
Semester : 3
Prodi : ti
Fakultas : teknik
```