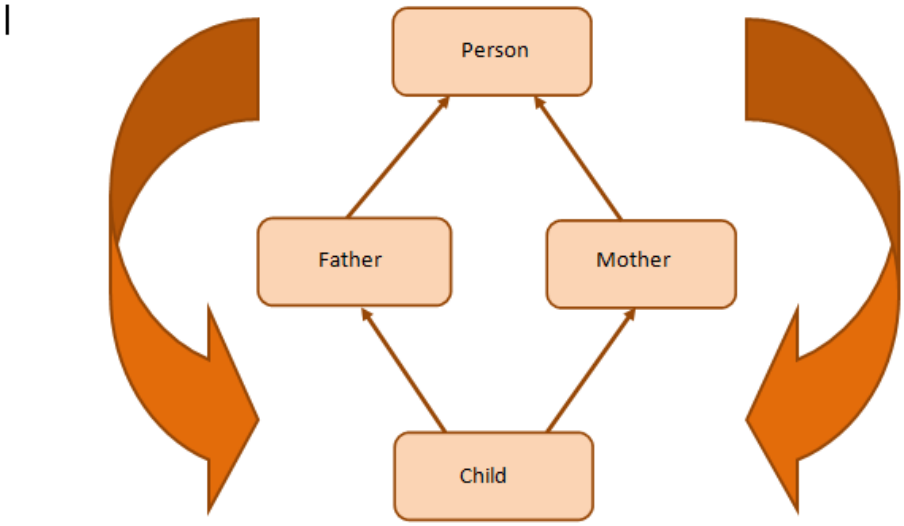
Ilham Nur Romdoni M0520038

Diamond Problem dan Multiple Inheritance

1. Jelaskan apa yang dimaksud dengan *diamond problem*, bagaimana masalah tersebut bisa muncul pada *multiple inheritance*?
   Diamond problem adalah masalah yang mengacu pada *class inheritance graph* di mana *class* diturunkan dari dua basis berbeda, yang, pada gilirannya, berasal dari basis yang sama. Masalah ini dinamakan *diamond* karena bentuknya seperti *diamond*. Masalah tersebut bisa muncul ketika *child class* mewarisi dari dua *parent class* yang keduanya berbagi *grandparent class* yang sama. *Class* turunan dari dua *class* berbeda yang keduanya meminta sesuatu tentang turunannya, dan *compiler* mungkin tidak tahu bagaimana menyelesaikan permintaan yang saling bertentangan. Tanpa menggunakan warisan virtual, *child class* akan mewarisi properti *grandparent class* dua kali, yang mengarah ke ambiguitas. Hal ini digambarkan dalam diagram di bawah ini:



   Referensi:
   How can diamond problems be overcome in multiple inheritance? - Quora
   What Is the Diamond Problem in C++? How to Spot It and How to Fix It (makeuseof.com)

2. Jelaskan cara menangani *diamond problem* di C++!
   Cara menangani *diamond problem* di C++ adalah dengan menggunakan virtual *keyword*. Virtual *keyword* digunakan ketika *parent class* mewarisi dari *grandparent class* bersama. Dengan demikian, hanya satu salinan *grandparent class* yang dibuat,

dan *object construction grandparent class* dilakukan oleh *child class*. Berikut contoh yang merepresentasikan contoh dari menangani *diamond problem*.

```
#include<iostream>
using namespace std;
class Person { //class Person
public:
    Person() { cout << "Person::Person() called" << endl; } //Base constructor
    Person(int x) { cout << "Person::Person(int) called" << endl; }
};

class Father : virtual public Person { //class Father inherits Person
public:
    Father(int x):Person(x)   {
       cout << "Father::Father(int) called" << endl;
    }
};

class Mother : virtual public Person { //class Mother inherits Person
public:
    Mother(int x):Person(x) {
        cout << "Mother::Mother(int) called" << endl;
    }
};

class Child : public Father, public Mother  { //class Child inherits Father
and Mother
public:
    Child(int x):Mother(x), Father(x) {
        cout << "Child::Child(int) called" << endl;
    }
};

int main()  {
    Child child(30);
}
```

Output:

```
Person::Person() called
Father::Father(int) called
Mother::Mother(int) called
Child::Child(int) called
```

Referensi; What Is the Diamond Problem in C++? How to Spot It and How to Fix It (makeuseof.com)

3. Jelaskan bagaimana *multiple inheritance* diimplementasikan pada bahasa pemrograman yang lain misalnya pada bahasa pemrograman Java!
   Pada Java, *multiple inheritance* diimplementasikan menggunakan *interface*. *Interfeace* berisi *method* dan konstanta. *Method* ini abstrak dan tidak memiliki badan *method*. *Method* ini diimplementasikan melalui *class*. Berikut contoh program untuk lebih memahami konsepnya.

```
interface StudentRead{
    void read();
}

interface StudentWrite{
    void write();
}

class Student implements StudentRead, StudentWrite{
    public void read(){
        System.out.println("The students read.");
    }
    public void write(){
        System.out.println("The students write.");
    }
}

public class Main
{
  public static void main(String[] args) {
    Student s = new Student();
    s.read();
    s.write();
  }
}
```

Output:

```
The students read.
The students write.
```

Referensi: Multiple inheritance using interfaces in Java - CodeSpeedy