บทที่ 3 Object Oriented Review

บทเรียนย่อย

3.1 Class Diagrams

3.2 Classes and Objects Implementation

3.3 Encapsulation and Information Hiding

3.4 Overloading and Overriding

วัตถุประสงค์

- นิสิตได้ทบทวนความรู้ และความเข้าใจในการวิเคราห์และออกแบบโปรแกรมเชิงวัตถุ
- นิสิตได้ทบทวนความรู้ และความเข้าใจในการใช้เครื่องมือสำหรับการออกแบบโปรแกรมเชิงวัตถุ
- นิสิตได้ทบทวนความรู้ และความเข้าใจในการใช้งานคำสั่งพื้นฐานในการเขียนโปรแกรมเชิงวัตถุ

บทที่ 3 Object Oriented Review

บทเรียนย่อย

3.1 Class Diagrams

3.2 Classes and Objects Implementation

3.3 Encapsulation and Information Hiding

3.4 Overloading and Overriding

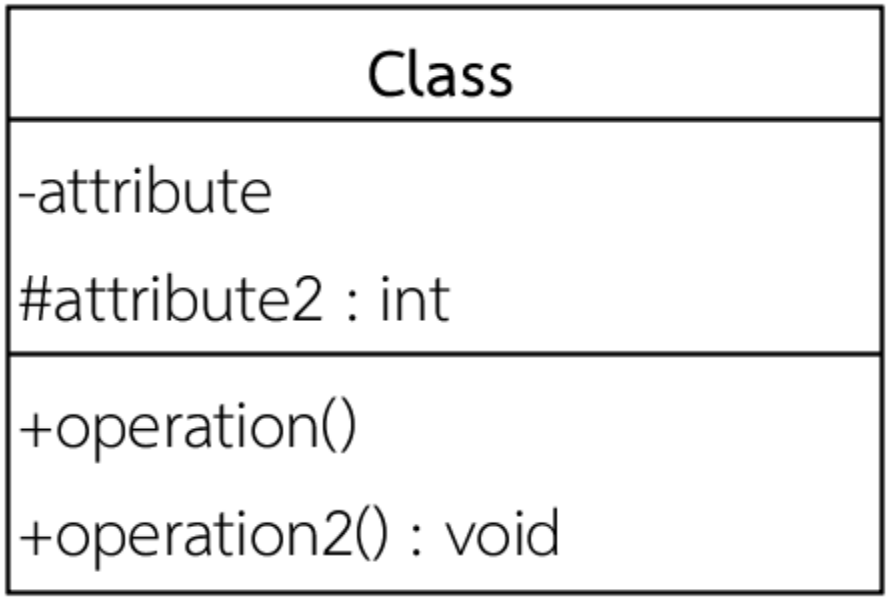
3.1 Class Diagrams

Class Diagrams หรือ แผนภาพคลาส คือ เป็นแผนภาพที่ใช้แสดงมุมมองของระบบที่เน้นโครงสร้างของคลาส คุณสมบัติ การทำงาน รวมไปถึงความสัมพันธ์ของคลาส (Relationship)

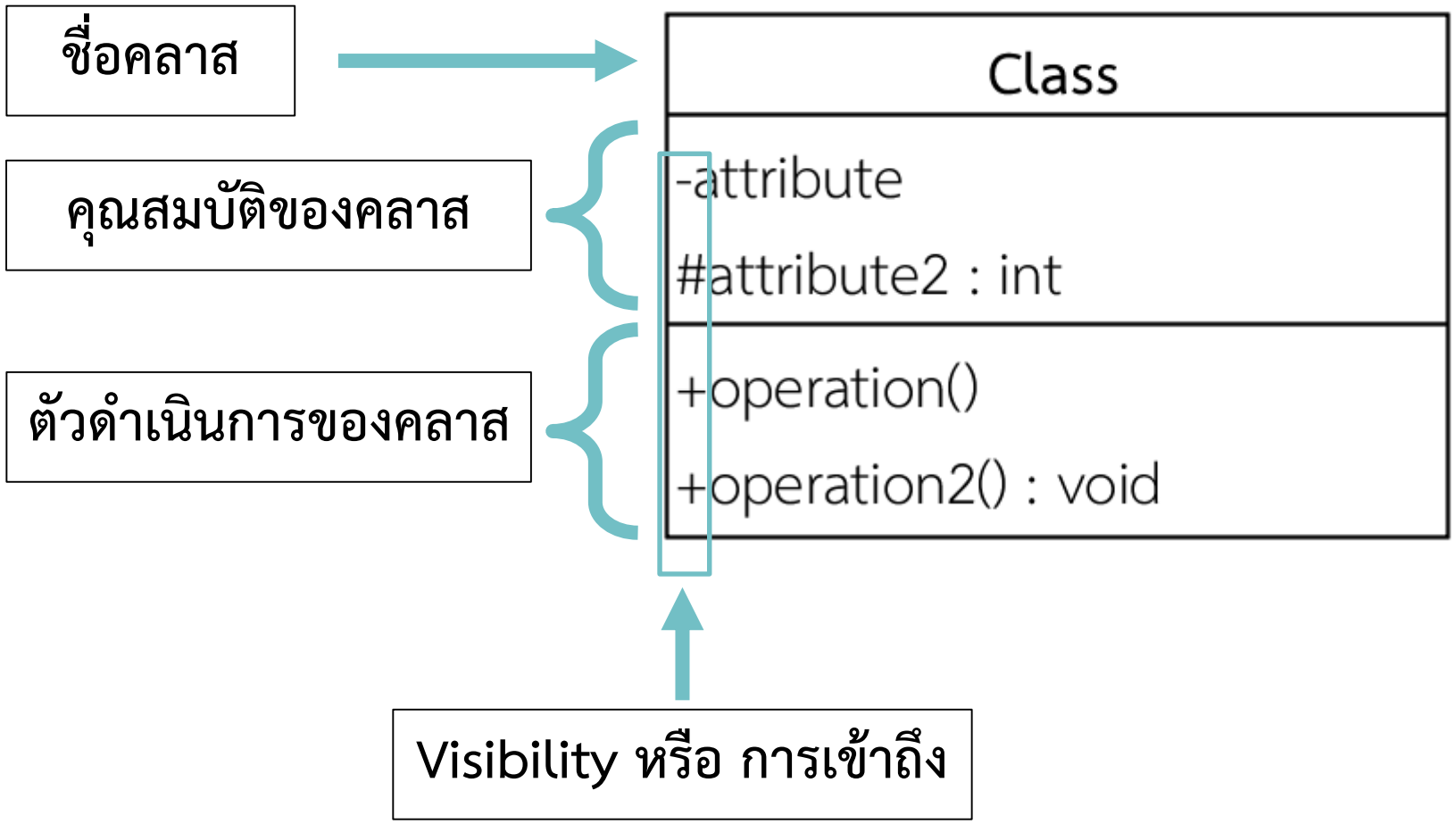
โดยที่ในหนึ่งคลาส จะประกอบไปด้วย

- ชื่อคลาส
- Attribute หรือคุณสมบัติ
- Operation/Method หรือการดำเนินการ ตัวดำเนินการ

ส่วนประกอบของคลาส ในแผนภาพคลาส



ส่วนประกอบของคลาส ในแผนภาพคลาส [2]



ระดับการเข้าถึง (Visibility)

ประกอบไปด้วย

เครื่องหมาย - แสดงถึงการเข้าถึงแบบส่วนตัว (Private)

เครื่องหมาย + แสดงถึงการเข้าถึงแบบสาธารณะ (Public)

เครื่องหมาย # แสดงถึงความสามารถในการเข้าถึงของคลาส และคลาสที่สืบทอด (Protected)

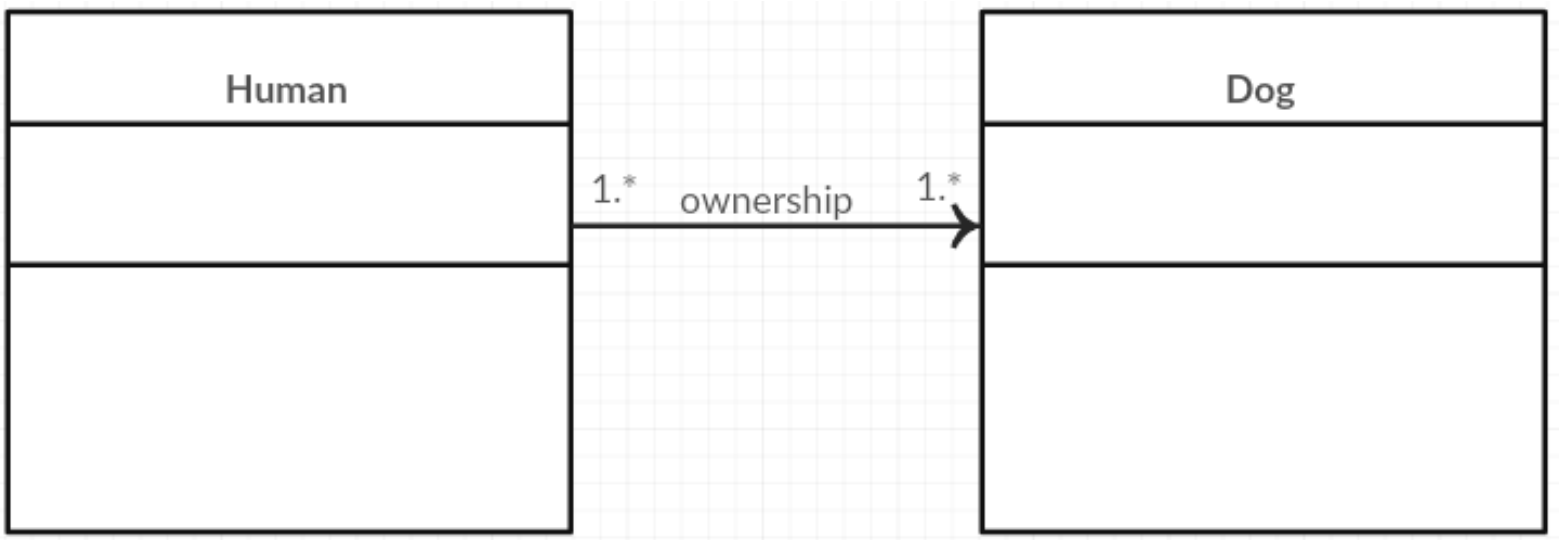
ความสัมพันธ์ระหว่างคลาส

ความสัมพันธ์ระหว่างคลาส จะแสดงโดยใช้เส้นลากเชื่อมระหว่างคลาส ซึ่งแบ่งได้เป็น

- Associations
- Generalization หรือ Inheritance
- Aggregation และ Composition

Associations

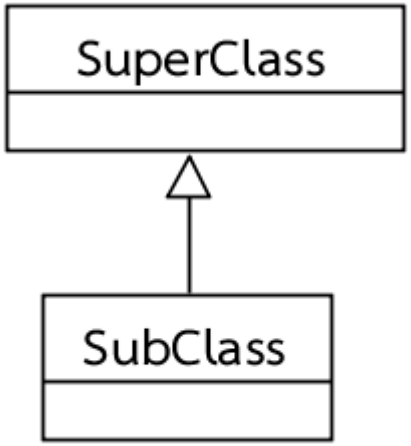
Associations คือ ความสัมพันธ์ของวัตถุที่สามารถระบุถึงบทบาท ทิศทาง และสัดส่วนได้



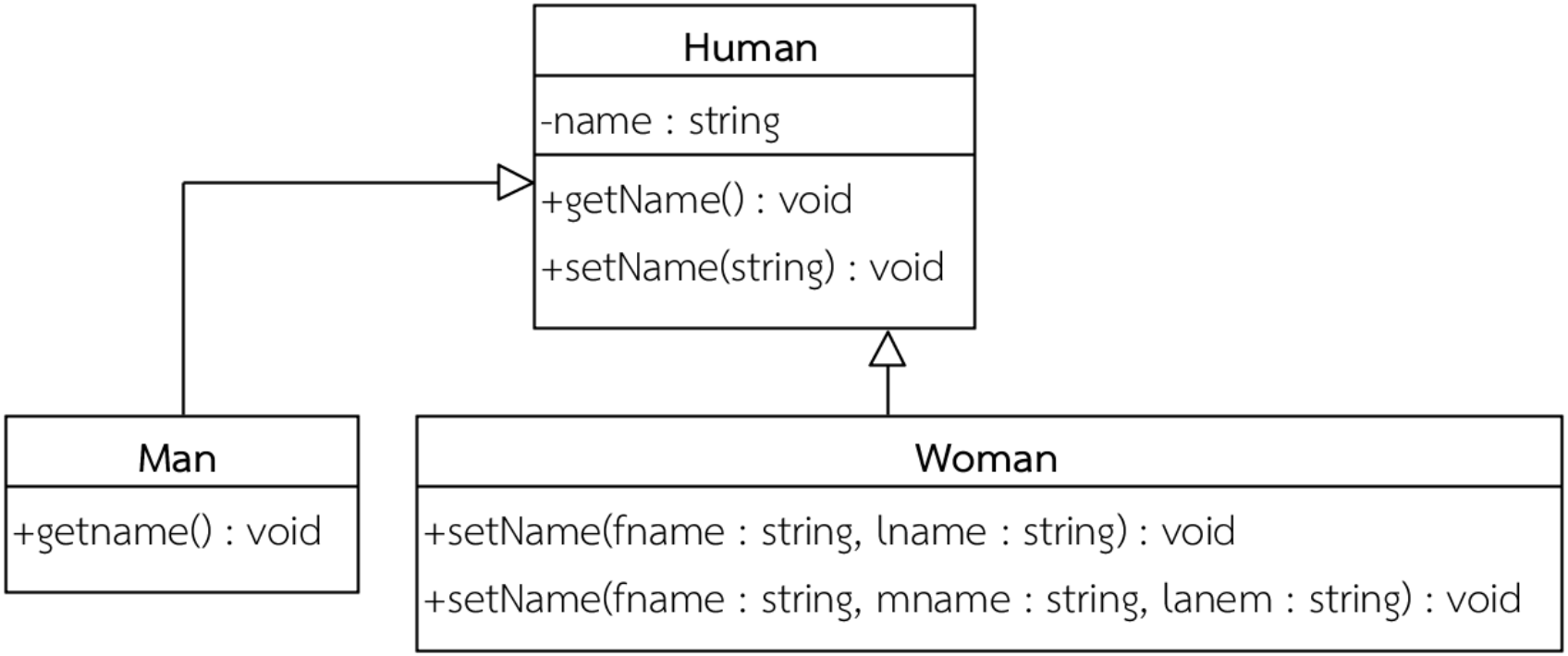
Generalization หรือ Inheritance

Generalization หรือ Inheritance คือคุณสมบัติในการสืบทอดโดยที่จะรับคุณสมบัติจาก Super Class มา โดยคุณสมบัติที่รับมาสามารถ Overload ค่าได้

สัญลักษณ์ของ Generalization หรือ Inheritance ประกอบไปด้วย



ตัวอย่างคลาส Generalization หรือ Inheritance



Aggregation และ Composition

Aggregation เป็นความสัมพันธ์แบบที่เป็นการพึ่งพากันแต่สามารถทำงานแยกจากกันได้

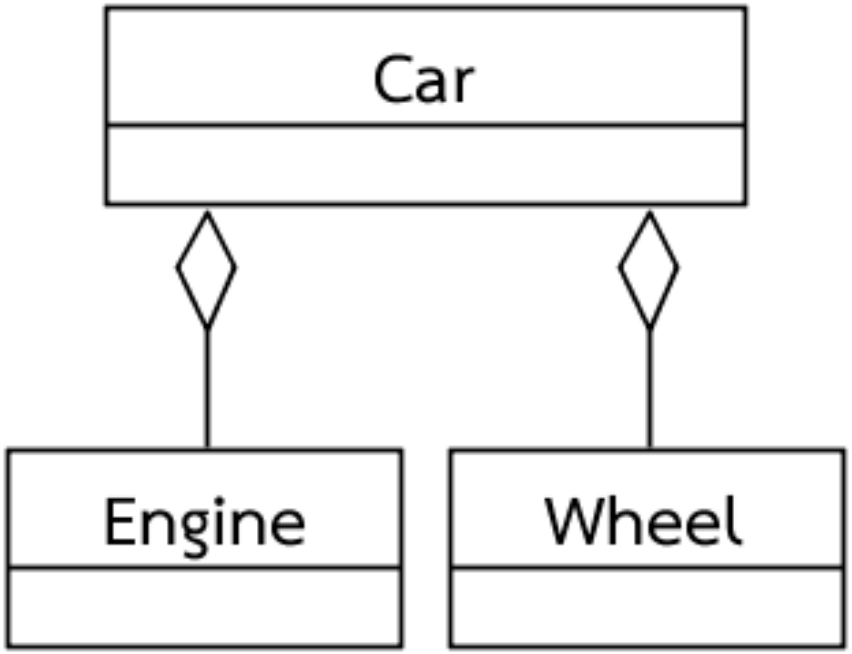
Composition เป็นความสัมพันธ์แบบ Aggregation แต่เข้มงวดกว่านั้นคือไม่สามารถขาดตัวใดตัวหนึ่งไปได้

มีสัญลักษณ์

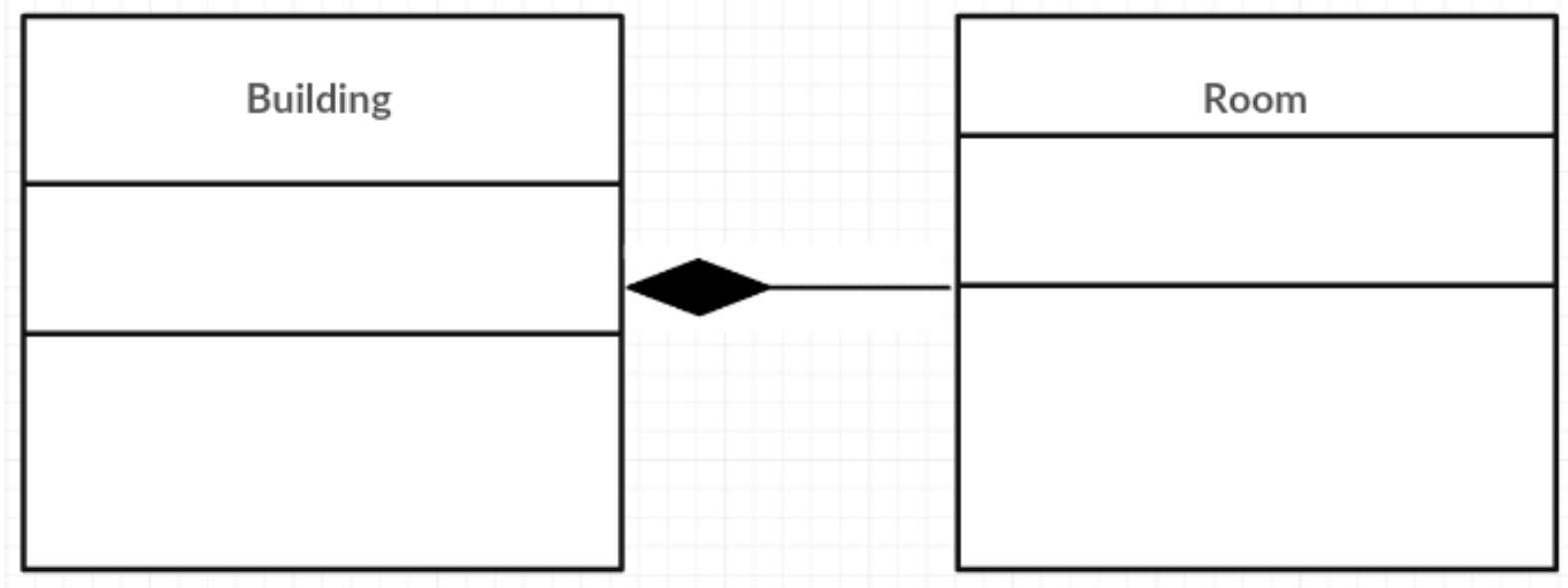
Aggregation

Composition

ตัวอย่างคลาสแบบ Aggregation



ตัวอย่างคลาสแบบ Composition



บทที่ 3 Object Oriented Review

บทเรียนย่อย

3.1 Class Diagrams

3.2 Classes and Objects Implementation

3.3 Encapsulation and Information Hiding

3.4 Overloading and Overriding

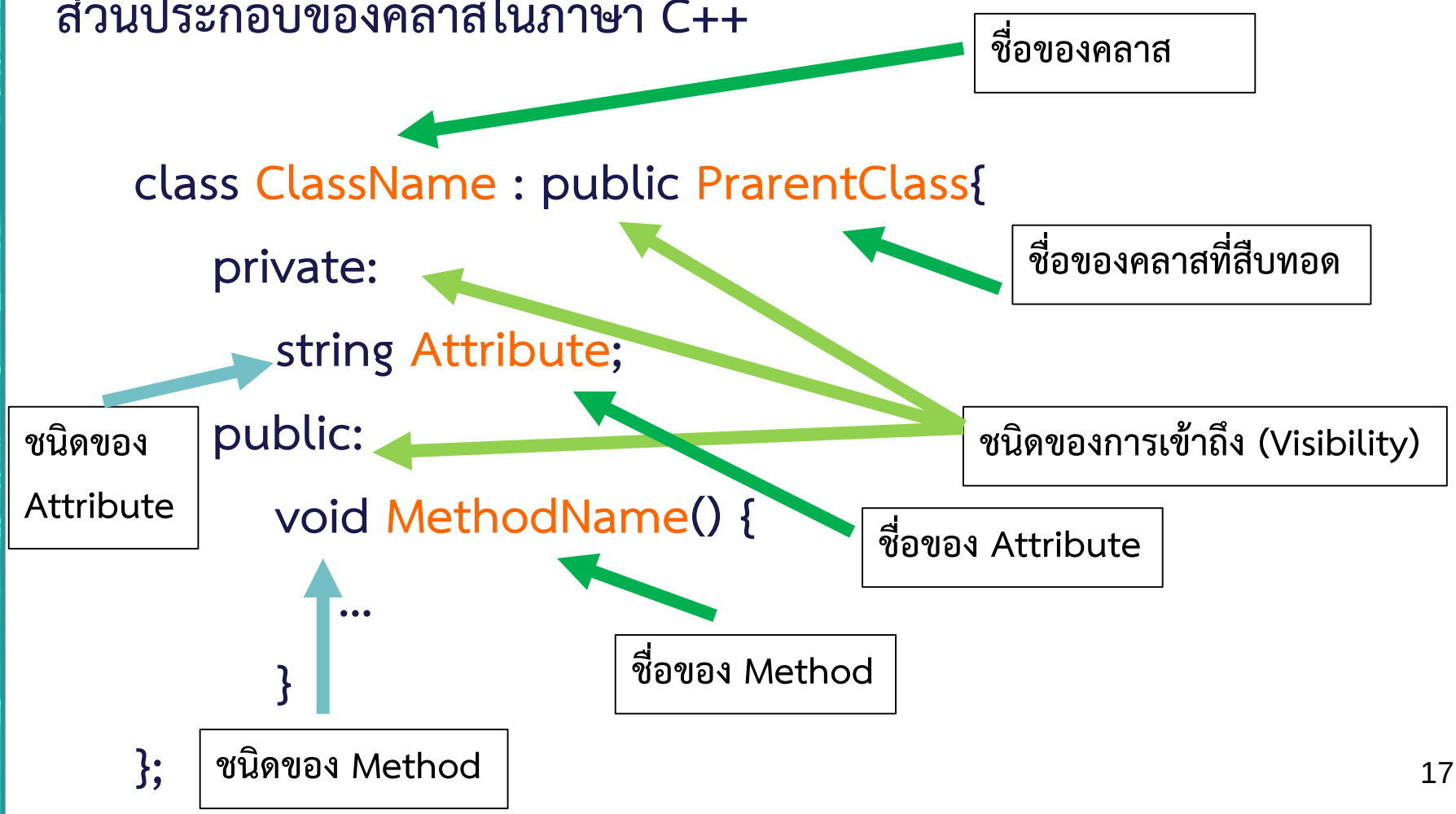
3.2 Classes and Objects Implementation

ส่วนประกอบของคลาสในภาษา C++

```
class ClassName : public PrarentClass{
    private:
        string Attribute;
    public:
        void MethodName() {
            ...
        }
};
```

3.2 Classes and Objects Implementation [2]

ส่วนประกอบของคลาสในภาษา C++



3.2 Classes and Objects Implementation [3]

ส่วนประกอบของคลาสในภาษา C++

```
class ClassName : public PrarentClass{
    private:
        string Attribute;
    public:
        void MethodName() {
            ...
        }
};
```

เป็นการสร้างคลาสในแบบ Inheritance

ตัวอย่างการสร้างวัตถุ (Object) ใน C++

ฟังก์ชัน main ในภาษา C++

```
int main() {
    Human * ObjHuman01 = new Human;
    ObjHuman01->setName(“Podsaporn Saelim");
    ObjHuman01->getName();
    cout << endl;
    Human ObjHuman02;
    ObjHuman02.setName("Jirayus Arbking");
    ObjHuman02.getName();
    cout << endl;
    return 0;
}
```

ตัวอย่างการสร้างวัตถุ (Object) ใน C++ [2]

ฟังก์ชัน main ในภาษา C++

```
int main() {
    Human * ObjHuman03 = new Human[10];
    ObjHuman03[0].setName("Podsaporn Saelim");
    ObjHuman03[0].getName();
    cout << endl;
    Human ObjHuman04[10];
    ObjHuman04[0].setName("Jirayus Arbking");
    ObjHuman04[0].getName();
    cout << endl;
    return 0;
}
```

การ Implement คลาสแบบ Aggregation

```
class ClassB{
    private:
        ClassA Attribute;
    public:
        void MethodName() {
            ...
        }
};
```

การ Implement คลาสแบบ Aggregation [2]

```
class ClassB{
    private:
        ClassA Attribute;
    public:
        void MethodName() {
            ...
        }
};
```


มีการเรียกใช้ ClassA เป็น Attribute

การ Implement คลาสแบบ Composition

```
class ClassA{
    class ClassB{};
    private:
        ClassB Attribute;
    public:
        void MethodName() {
            ...
        }
};
```

การ Implement คลาสแบบ Composition [2]

```
class ClassA{
    class ClassB{};
    private:
        ClassB Attribute;
    public:
        void MethodName() {
            ...
        }
};
```

มีการสร้าง ClassB อยู่ภายใต้ ClassA

มีการเรียกใช้ ClassB เป็น Attribute

บทที่ 3 Object Oriented Review

บทเรียนย่อย

3.1 Class Diagrams

3.2 Classes and Objects Implementation

3.3 Encapsulation and Information Hiding

3.4 Overloading and Overriding

3.3 Encapsulation and information Hiding

Encapsulation คือ การซ่อนข้อมูล โดยการกำหนดการเข้าถึงข้อมูล ไม่ว่าจะเป็น Attribute หรือ Method

Information Hiding คือ การซ่อนการทำงานทั้งหมด และรวมไปถึงปิดข้อมูลในการรู้ข้อมูลที่แท้จริง

ตัวอย่างโปรแกรมของ Encapsulation

```
class Classname {
        private:
            int Attribute;
        public:
            int GetAttribute();
            void SetAttribute(int SetValue);
};
```

ตัวอย่างโปรแกรมของ information Hiding

```
class Classname {
        private:
            int Attribute;
            int GetAttribute();
        public:
            void SetAttribute(int SetValue);
            int  requestAttribute(){
                return GetAttribute();
            }
};
```

บทที่ 3 Object Oriented Review

บทเรียนย่อย

3.1 Class Diagrams

3.2 Classes and Objects Implementation

3.3 Encapsulation and Information Hiding

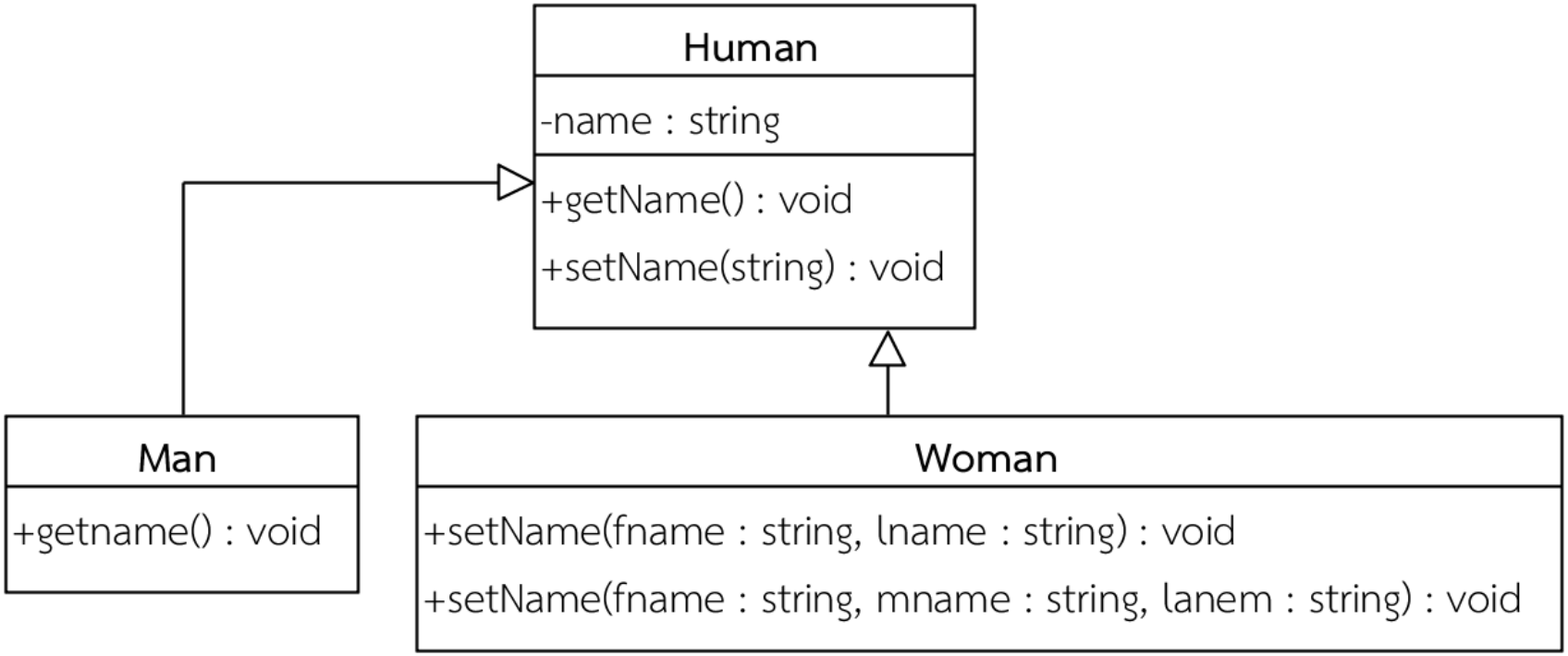
3.4 Overloading and Overriding

3.2 Overloading and Overriding

Overloading คือ การ Implement Method ใหม่โดยที่มีเปลี่ยนชื่อ Method เท่านั้นที่เหมือนเดิม ซึ่งสามารถเปลี่ยนได้ทั้ง Parameter(s) หรือขั้นตอนในการทำงานของ Method นั้น

Overriding คือ การ Implement Method ใหม่โดยที่ชื่อ Method รวมทั้ง Parameter(s) จะเหมือนเดิม แต่จะเปลี่ยนขั้นตอนในการทำงานของ Method นั้นไป

คลาสตัวอย่างการ Overloading และ Overriding



การ Implement คลาส Human

```
class Human {
    protected:
        string name;
    public:
        void getName() {
            cout << "I'm Human Name:" + this->name;
        }
        void setName(string setname) {
            this->name = setname;
        }
};
```

การ Implement Overriding

```
/**
 * Overriding Method getName
 */
class Man : public Human {
public:
    void getName() {
        cout << "I'm Man Name:" + this->name;
    }
};
```

การ Implement Overloading

```
/**
 * Overloading Method setName
 */
class Woman : public Human {
public:
    void setName(string fname, string lname) {
        this->name = fname + " " + lname;
    }
    void setName(string fname, string mname, string lname) {
        this->name = fname + " " + mname + " " + lname;
    }
};
```

แบบฝึกหัด

สร้าง คลาส Student โดยมีเงื่อนไขดังนี้

1. Attribute
   - id
   - prefixName
   - fname
   - lname
   - nickName
2. Method
   - มี get และ set ของ Attribute ข้างต้นทั้งหมด
3. นำคลาส Student ไป Implement ด้วย Array เพื่อเก็บข้อมูลนักศึกษา จำนวน 10 คน

Class Diagram ของแบบฝึกหัด