

# **CSS232**

**การแปลงเป็นภาษาโปรแกรม**

# การแปลงเป็นภาษาโปรแกรม

- Class Diagram: Attribute เป็นการบอกคุณลักษณะของ Class ซึ่ง Class หนึ่งๆ จะมีได้หลายAttribute หรือไม่มีเลยก็ได้
- รูปแบบของการเขียน Attribute ใน Class
- visibility name [multiplicity ordering] : type = initial\_value
  - *visibility*: + = public, - = private, # = protected
  - *multiplicity*: lower-bound .. upper-bound
  - *ordering*: unordered, ordered
  - *type*: boolean, integer, real, string เป็นต้น

ถ้าค่าของ Attribute มีค่าเฉพาะสำหรับ Object นั้นๆ เฉพาะ จะเรียกว่า  
เป็น Attribute ที่มีค่าระดับ *object scoped*

แต่ถ้าค่าของ Attribute นั้นมีค่าเดียวกันเสมอไม่ว่าจะสร้างที่ Object  
ใน Class นั้น จะเรียกว่าเป็น Attribute ที่มีค่าระดับ *class scoped*

ซึ่งการเขียน attribute นั้นจะขีดเส้นใต้เพื่อบอกว่าเป็น Class scoped

# ตัวอย่าง Attribute

UserAccount	
+	userID: String
-	name: String
-	gender: char
-	dateOfBirth: Date
-	<u>session: int</u>
-	emailAddress: String [1..5 ordered] = No Email
#	password: String

- EmailAddress

EmailAddress

EmailAddress[1..5]

EmailAddress[1..5 unordered]

EmailAddress[1..5 unordered]:String

EmailAddress[1..5 unordered]:String = “No  
email”

# ตัวอย่างโปรแกรม

```
public class UserAccount {  
    public String userID;  
    private String name;  
    private char gender;  
    private Date dateOfBirth;  
    private static int session;  
    private String emailAddress[1,5] = No Email;  
    protected String password;  
    public UserAccount(){  
    }  
}
```

# Operation

- Operations คือพฤติกรรมที่สามารถกระทำกับ Object ได้โดยที่ Object ที่อยู่ใน Class เดียวกันจะมี Operation ร่วมกัน
- ใน Class หนึ่งๆ อาจจะมี Operation ได้มากกว่าหนึ่งหรือไม่มี Operation ก็ได้
- วิธีการตั้งชื่อ Operation ส่วนใหญ่จะขึ้นต้นด้วยคำกริยา
- โดยปกติการเรียกใช้ Operation ของ Class หนึ่งจะมีผลทำให้ค่า Attribute ของ Object เปลี่ยนไป

# ตัวอย่าง Operation

UserAccount	
<ul style="list-style-type: none"><li>+ userID: String</li><li>- name: String</li><li>- gender: char</li><li>- dateOfBirth: Date</li><li>- <u>session: int</u></li><li>- emailAddress: String [1..5 ordered] = No Email</li><li># password: String</li></ul>	
<ul style="list-style-type: none"><li>+ getName() : String</li><li>+ setName(String) : void</li><li>+ generateUserID() : String</li></ul>	



# ตัวอย่างโปรแกรม

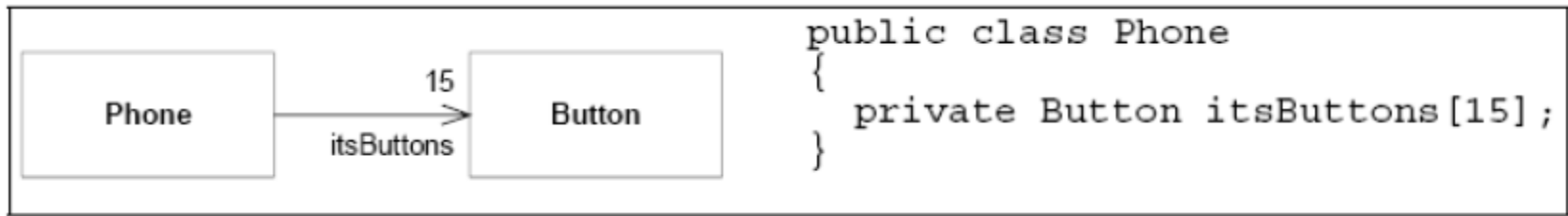
```
public class UserAccount {  
    public String userID;  
    private String name;  
    private char gender;  
    private Date dateOfBirth;  
    private static int session;  
    private String emailAddress = No Email;  
    protected String password;  
    public UserAccount(){ }  
    public String getName(){  
        return "";  
    }  
    public void setName(String name){ }  
    public String generateUserID(){  
        return "";  
    }  
}
```

# ตัวอย่างโปรแกรม

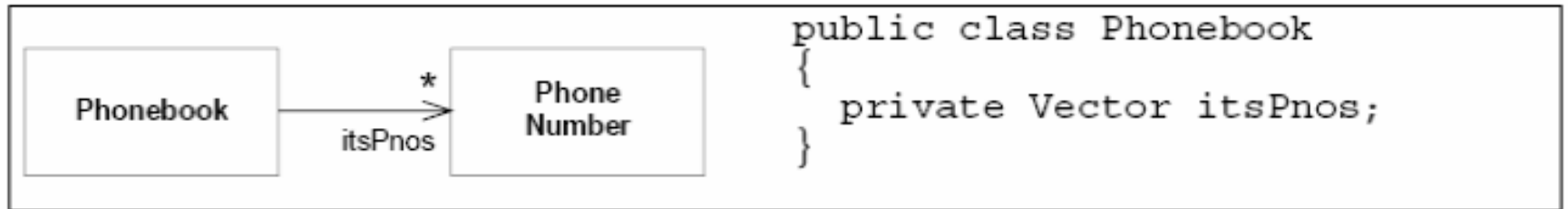
Dialler
- digits : Vector - nDigits : int
+ digit(n : int) # recordDigit(n : int) : boolean

```
public class Dialler
{
    private Vector digits;
    int nDigits;
    public void digit(int n);
    protected boolean recordDigit(int n);
}
```

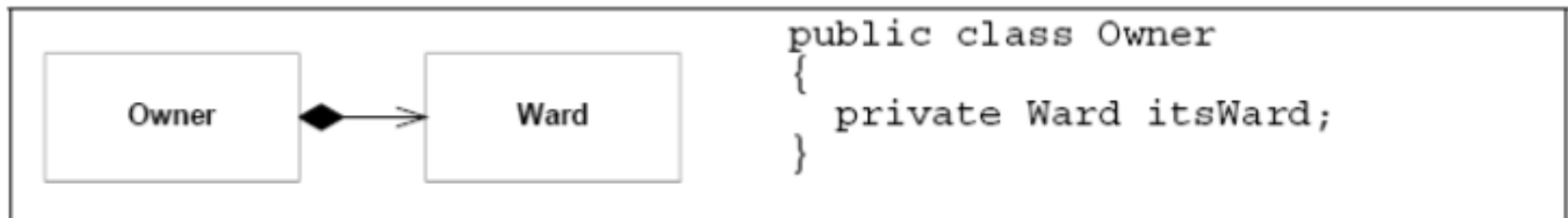
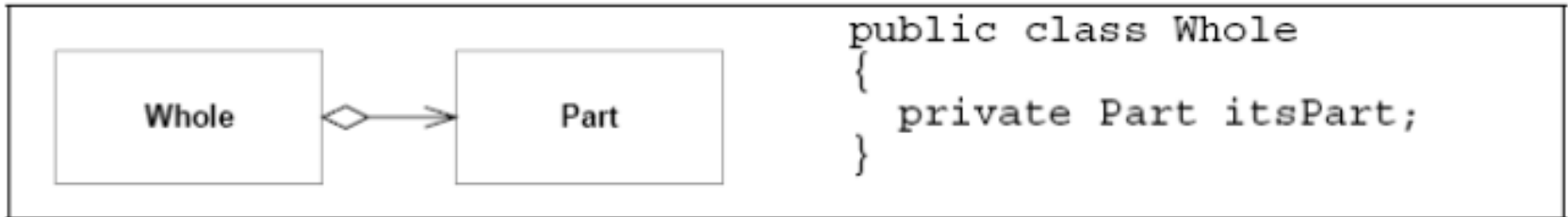
# Class diagram: Association



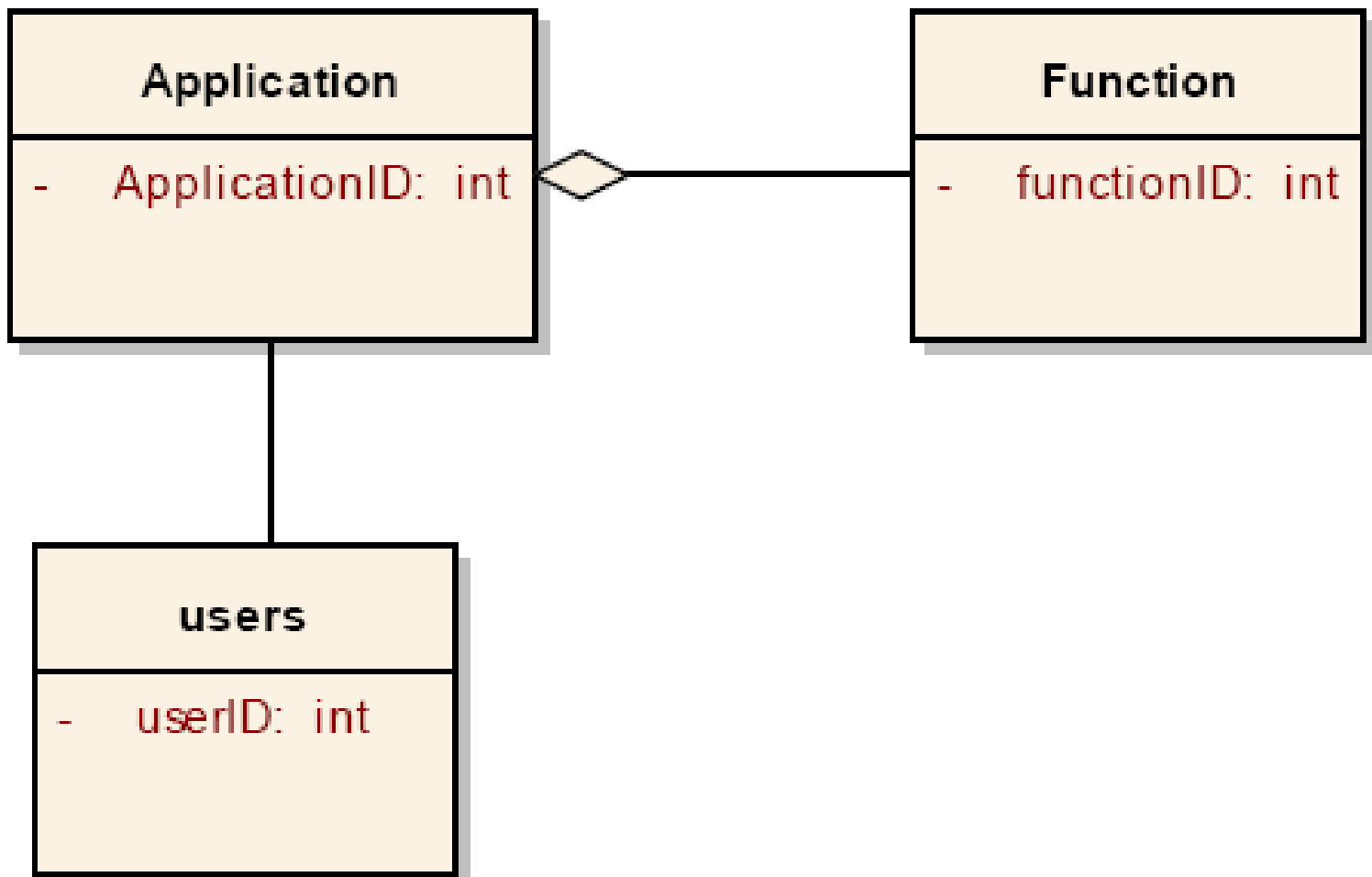
# Class diagram: Multiplicity



# Class diagram: Aggregation



# ตัวอย่าง



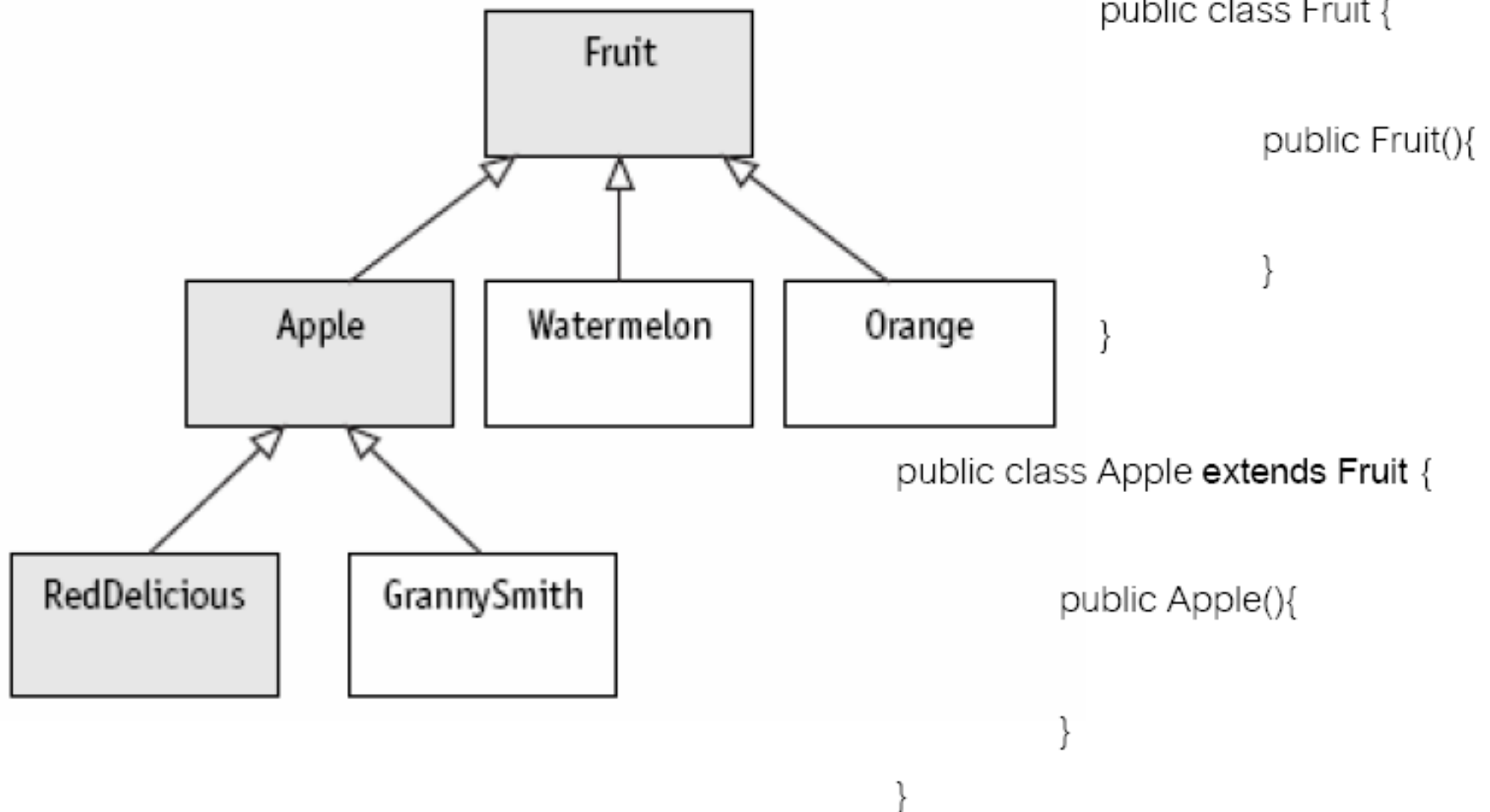
# ตัวอย่างโปรแกรม

```
public class Function {  
    private int functionID;  
    public Function(){  
    }  
}
```

```
public class Application {  
    private int ApplicationID;  
  
    public Function m_Function;  
    public Application(){  
    }  
}
```

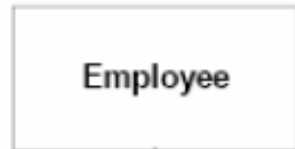
```
public class users {  
    private int userID;  
    public Application m_Application;  
    public users(){  
    }  
}
```

# ตัวอย่างโปรแกรม





# ตัวอย่างโปรแกรม



```
public class Employee
{
    ...
}
```

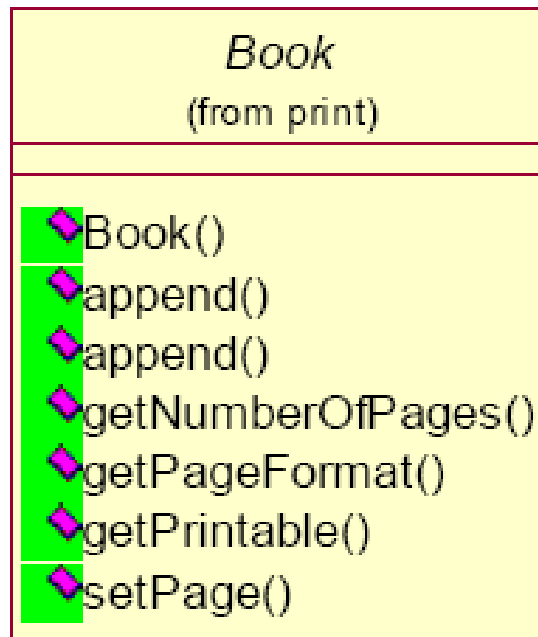
```
public class SalariedEmployee extends Employee
{
    ...
}
```

# Refinement class diagram

- คือการปรับปรุง Diagram ต่างๆ ที่ได้จากการวิเคราะห์ เชิงวัตถุ ให้มีรายละเอียดมากขึ้น สื่อความหมายได้ดีขึ้น มีส่วนเกี่ยวข้องกับการแก้ปัญหาด้วย Computer เพิ่มขึ้น และสามารถนำไปเป็นต้นแบบในการพัฒนาระบบงานให้เกิดขึ้นจริงๆ ได้

# Class diagram: Abstract class

- Abstract class เป็น Class ที่ไม่สามารถเป็น Object ได้
- Parent Class เท่านั้นที่จะเป็น Abstract Class ได้
- จะเขียนชื่อ Class เป็นตัวเอน

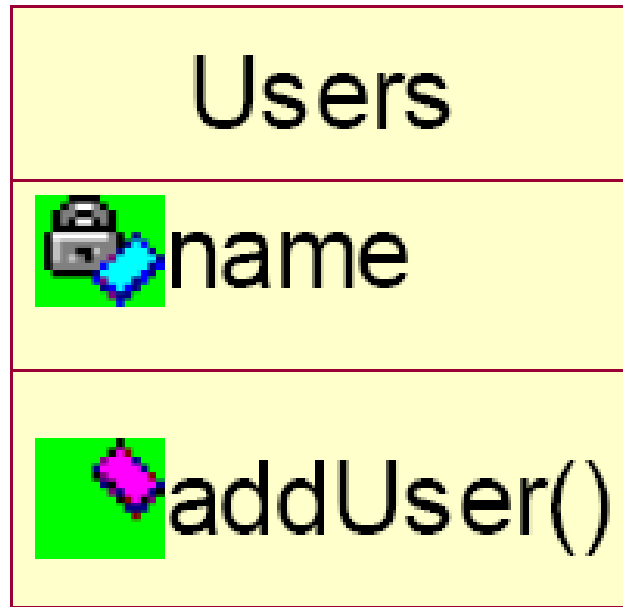


## ตัวอย่างโปรแกรม

```
public abstract class Book {  
    public Book(){  
    }  
    public void append(){  
    }  
    public void getNumberOfPage(){  
    }  
}
```

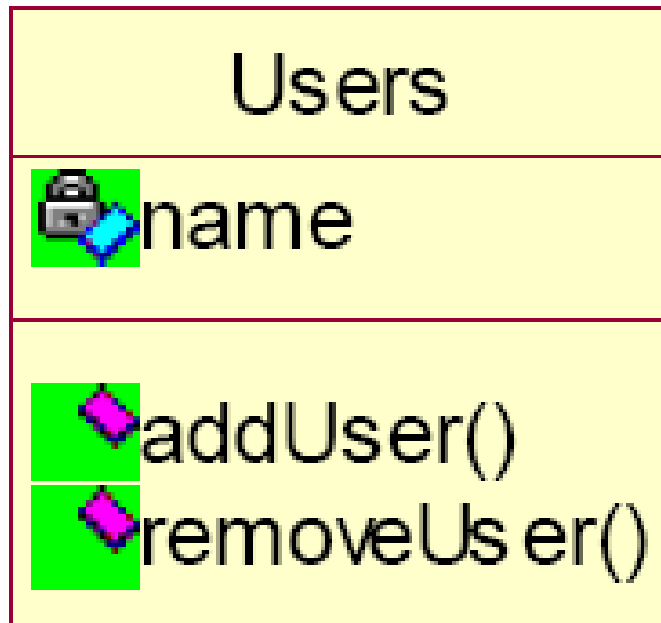
# Refine class diagram: ตรวจสอบความสมบูรณ์ของ class diagram

- การออกแบบที่ดีจะต้องมี Operation การทำงานที่ครบถ้วน
- Class นี้มีความสมบูรณ์หรือไม่



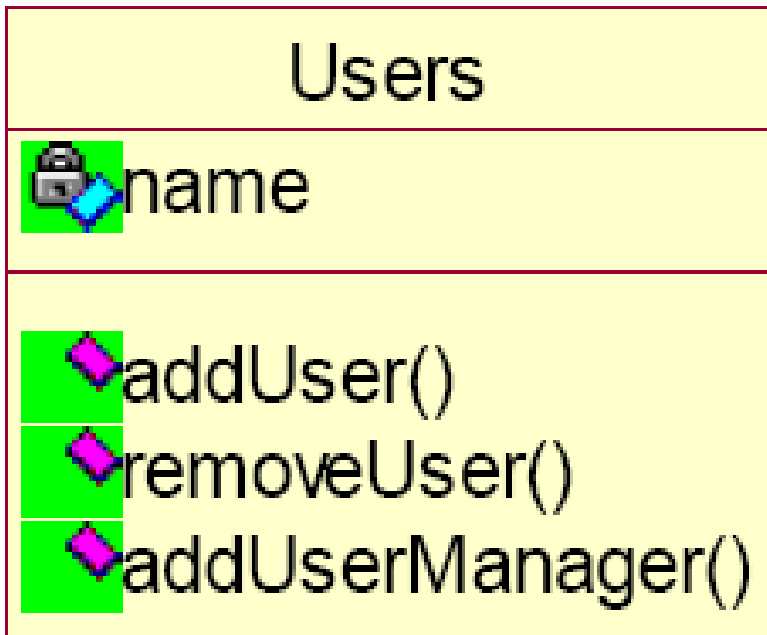
# ตรวจสอบความสมบูรณ์

- Class ดังกล่าวไม่สมบูรณ์เนื่องจากความเป็นไปได้ของข้อมูล เมื่อมีการเพิ่มข้อมูล ควรจะมีการลบข้อมูล



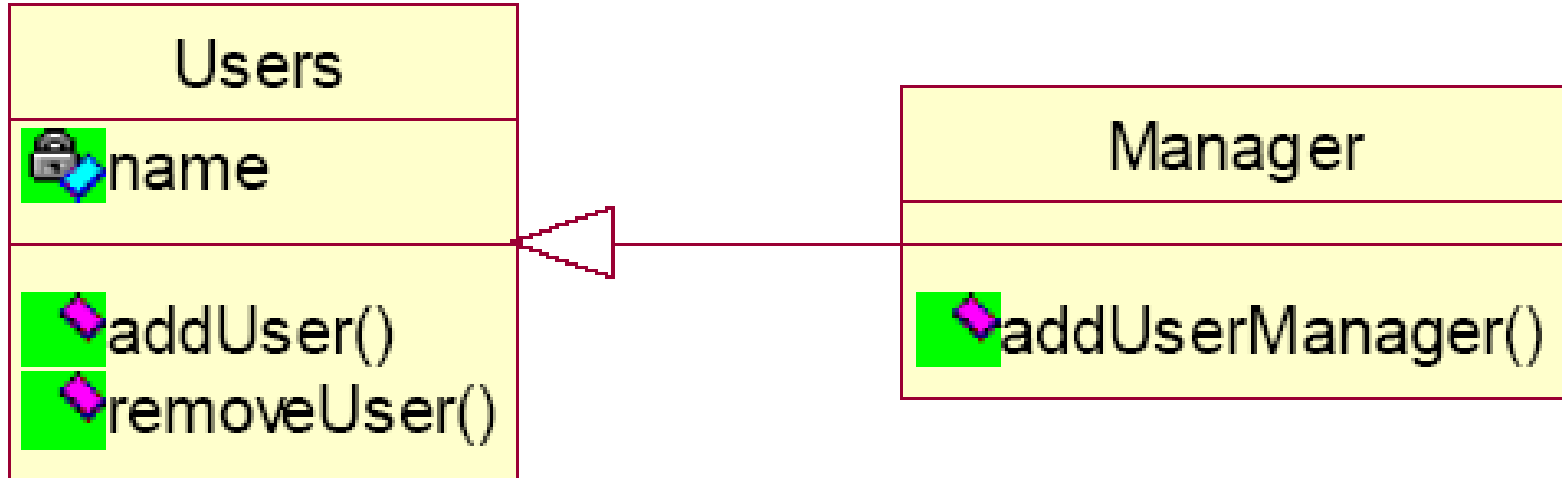
# ตรวจสอบความพอดี

- การออกแบบที่ดีควรจะทำจัด operation ที่ไม่จำเป็นออกไปควรมี Operation ที่เหมาะสมเท่านั้น
- Class นี้มีความพอดีหรือไม่



# ตรวจสอบความพอดี

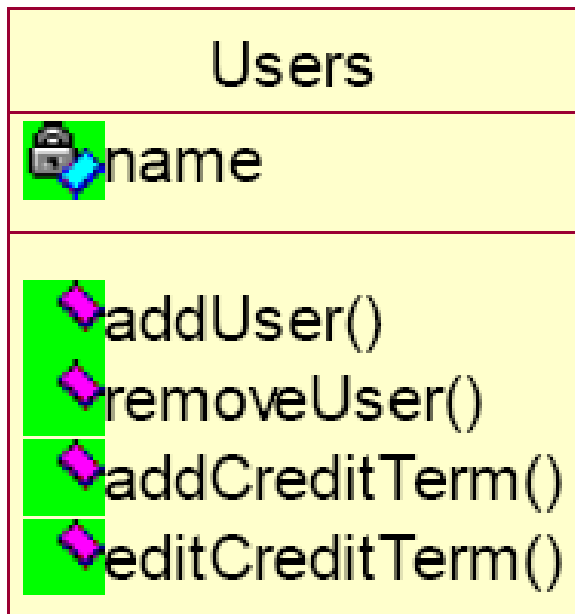
- Class ดังกล่าวไม่สมบูรณ์เนื่องจากมี Operation ที่เกินความพอดีของ Class วิธีแก้ไขควรจะแยก Class ในการจำแนก Operation นั้นออกไป





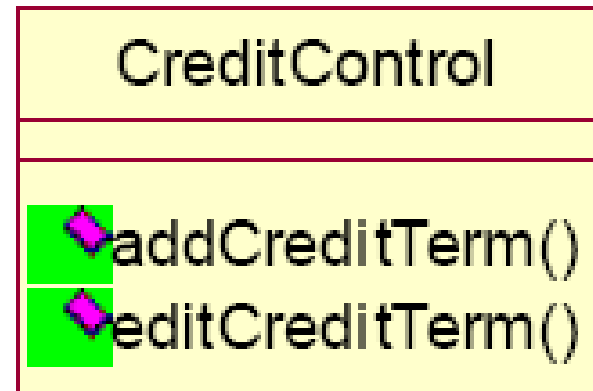
# ตรวจสอบความเป็นพื้นฐาน

- การออกแบบที่ดีควรจะใช้ Operation ที่เป็นพื้นฐานมากที่สุดของ Class นั้น เพื่อลดความซับซ้อน
- Operation ใน Class นี้มีความเป็นพื้นฐานหรือไม่



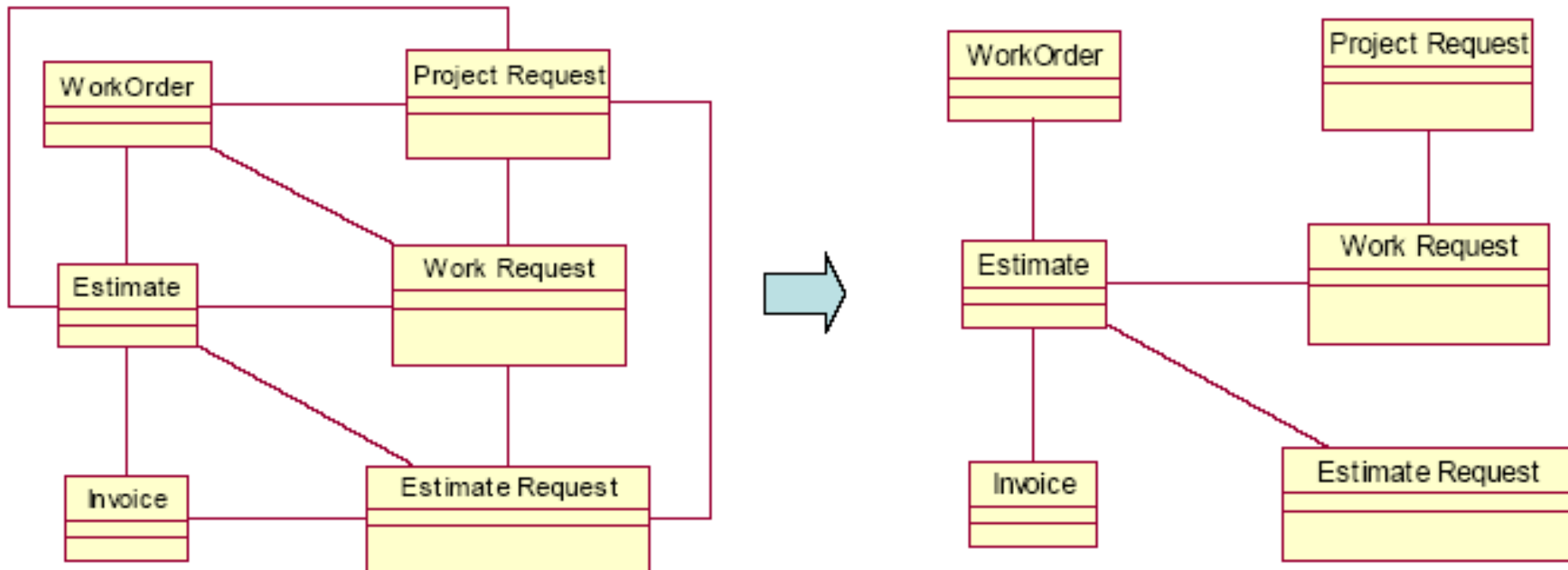
# ตรวจสอบความเป็นพื้นฐาน

- การแบ่ง Class เป็นวิธีการแก้ไขความเป็นพื้นฐานได้ โดยการแบ่ง Class จะยังคงทำงานได้ดังเดิม

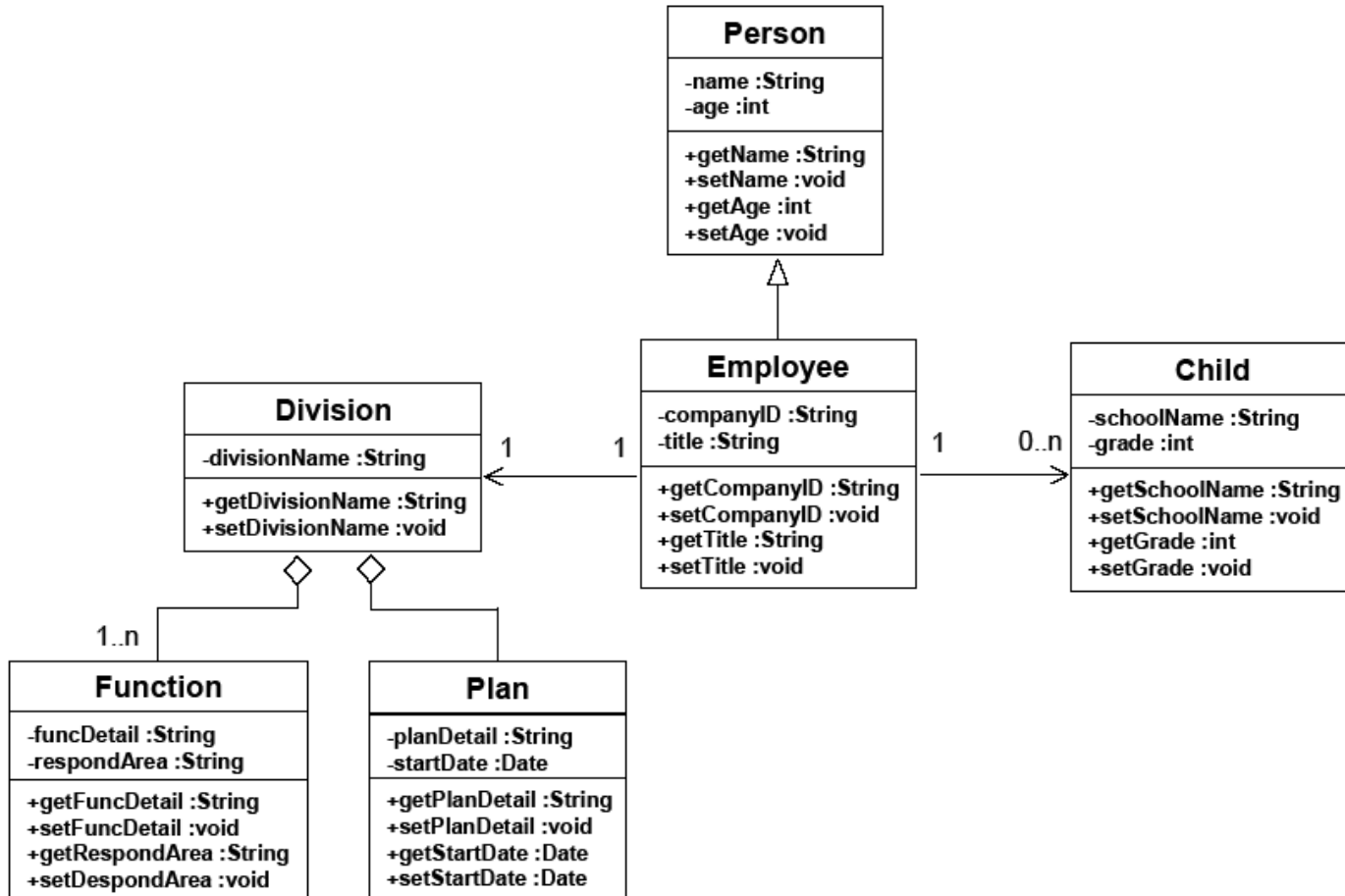


# ตรวจสอบความสัมพันธ์

- Class ใดๆ Class จะต้องมีความสัมพันธ์กับ Class อื่นๆ
- การที่ มีความสัมพันธ์ของ Class น้อยที่สุด จะช่วยลดความซับซ้อนของงาน และบำรุงรักษาหรือพัฒนาโปรแกรมได้ง่าย



# การบ้าน จงแปลลงเป็นภาษาจาวา



# Software แปลงจาก UML Diagram → โค้ดโปรแกรม

- Visual Paradigm: generate: โค้ดจาก UML class model (รวมทั้ง Java, C#, C++, Python, PHP, Ruby ฯลฯ) รวมถึงรองรับ reverse-engineering จากโค้ดกลับเป็น UML ได้ด้วย.
- Software Ideas Modeler: มีฟีเจอร์ “Code Generation from Diagrams” ที่ช่วยแปลงจาก UML → โค้ดในหลายภาษา เช่น Java, C#, C++, Python, Ruby, PHP, TypeScript ฯลฯ.
- Altova Umodel: เป็นโปรแกรม modeling ที่รองรับ UML ครบทุกชนิด + SysML และมีฟีเจอร์ code engineering (generate โค้ด) ใน Java, C#, VB เป็นต้น.

- UML-RSDS: เป็นเครื่องมือ transformation/model-driven engineering ที่รองรับ UML class diagram และสามารถ generate code ในหลายภาษา เช่น Java, C#, C++, Python, Go, Swift, ANSI C.
- BOUML: Freeware for UML ที่รองรับ code generation & reverse engineering เช่น C++, Java, PHP, MySQL.