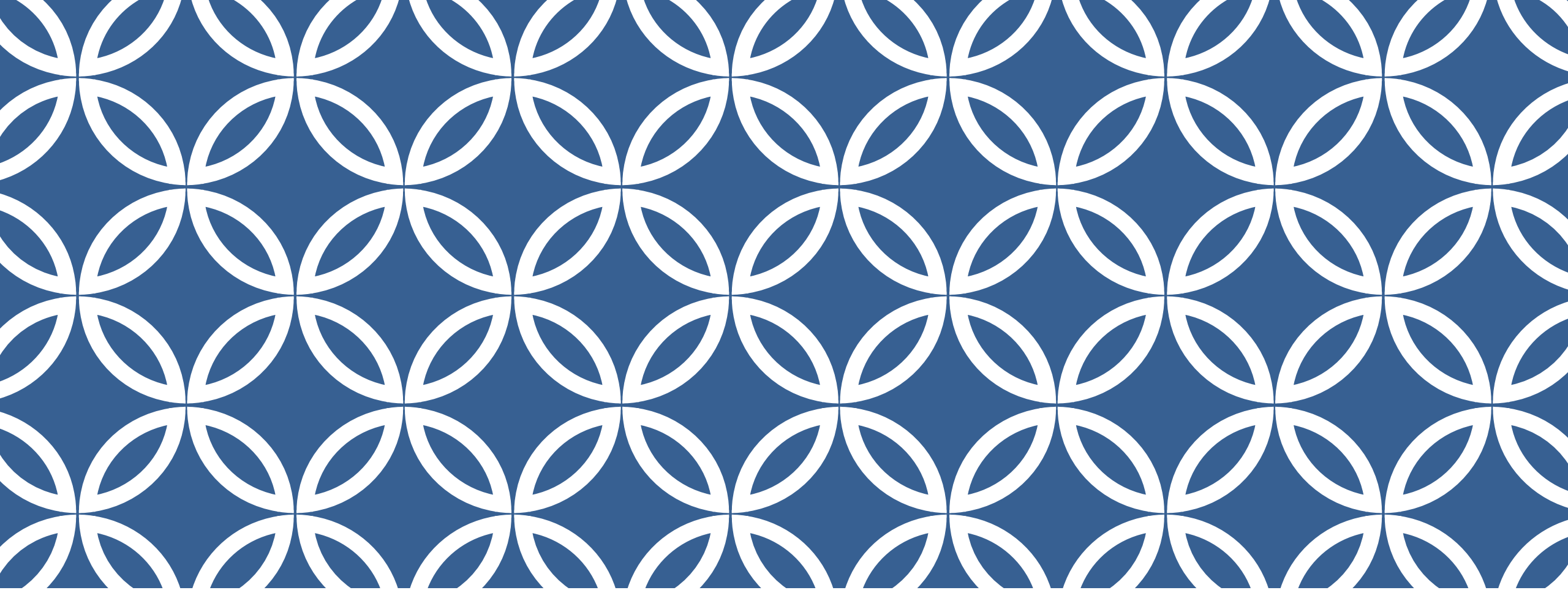


บทที่ 4
INHERITANCE PART 1

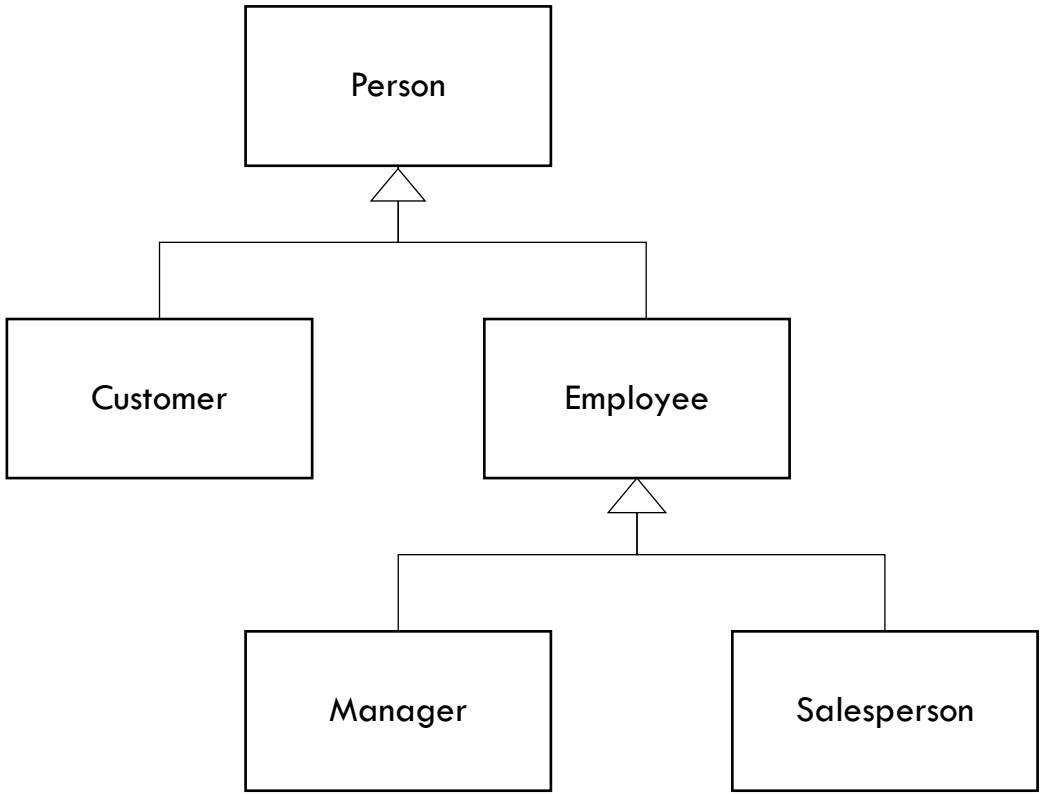
อ.สกรณ์ บุษบง
สาขาวิทยาการคอมพิวเตอร์ คณะวิทยาศาสตร์
มหาวิทยาลัยราชภัฏบุรีรัมย์

OVERVIEW

- ในบทนี้จะเรียนรู้เกี่ยวกับ **inheritance**
- ซึ่งเป็นกลไกการถ่ายทอดคุณสมบัติ **(attributes and method)** จาก **superclass** ไปยัง **subclass**

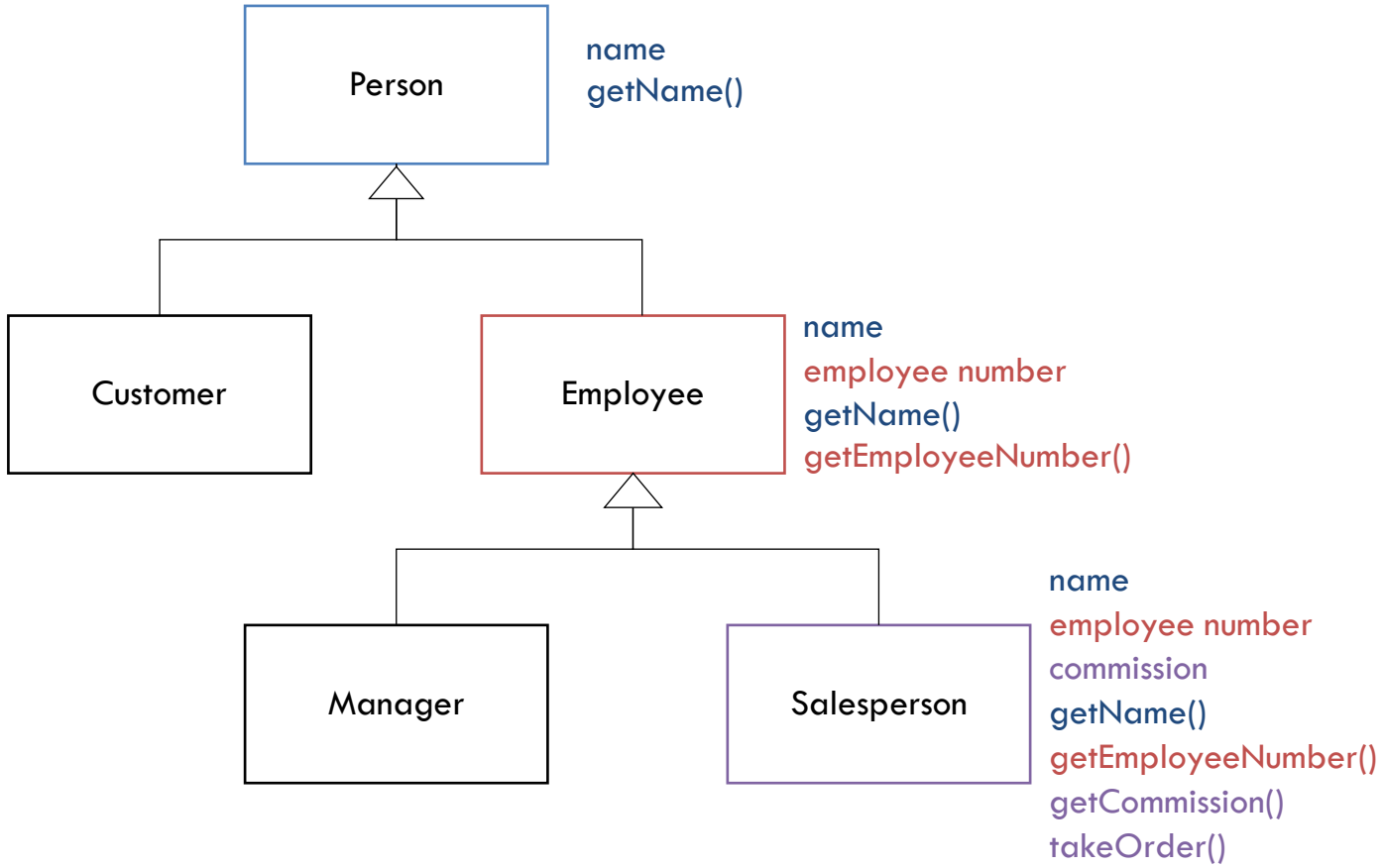
COMMON PROPERTIES

- จาก **class hierarchy** ของบทที่แล้ว



COMMON PROPERTIES

- class hierarchy ที่เพิ่ม attribute และ method



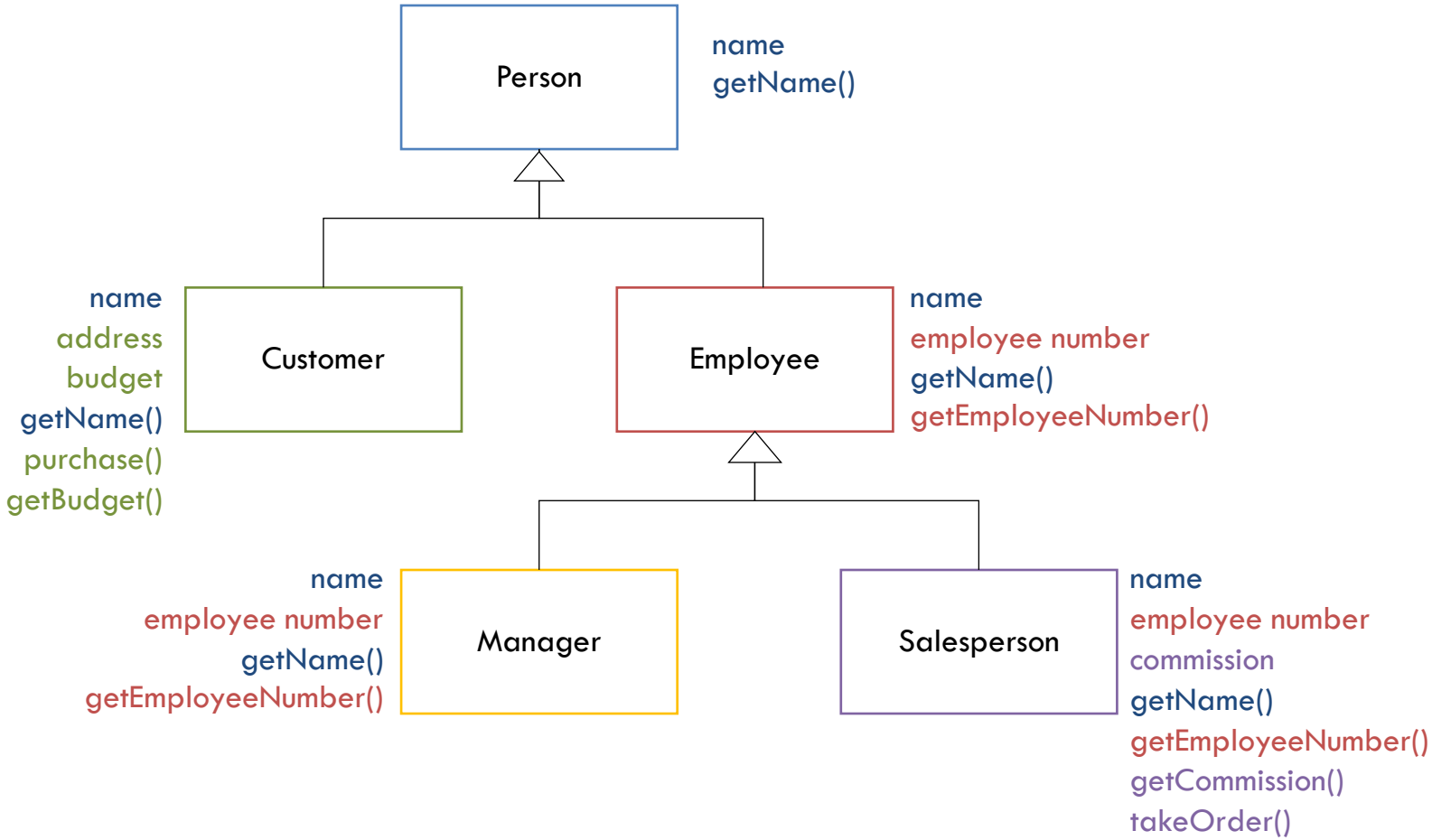
INHERITANCE

- Reusable เป็นคุณสมบัติของการนำกลับมาใช้ใหม่
- คุณสมบัติทั่วไป (generalized properties) จะถูกกำหนดไว้ใน superclass และจะส่งต่อไปยัง subclass โดยที่ subclass ไม่ต้องประกาศ properties หรือ method
- คุณสมบัตินี้เรียกว่า Inheritance

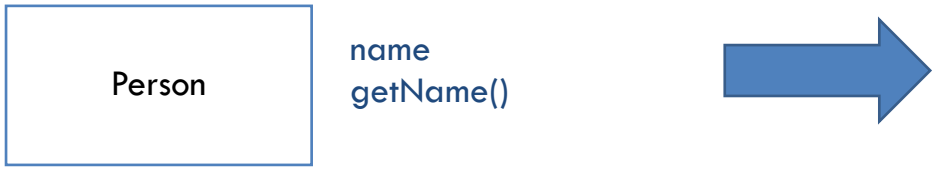
INHERITANCE

- **Inheritance** คือความสามารถของ **subclass** ที่ได้รับ คุณสมบัติทั่วไป (generalized properties) จาก **superclass** ใน **inheritance chain**
- คุณสมบัติทั่วไป ที่ได้รับมานั้นเปรียบเสมือน **subclass** ประกาศเอง
- subclass *inherited* from superclass

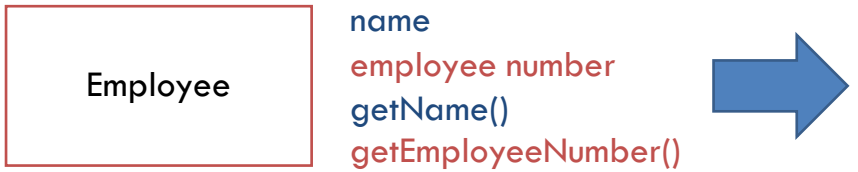
INHERITANCE



INHERITANCE

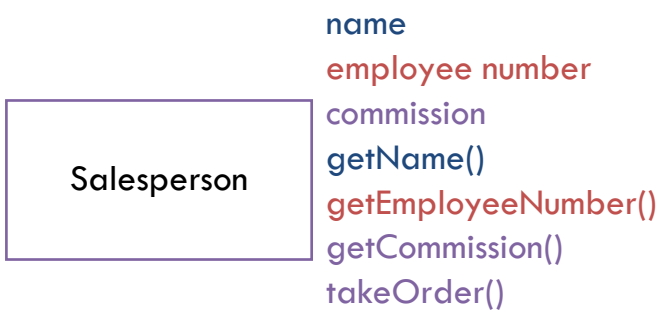



```
Class Person {
   Attributes :
      name
   Methods :
      getName()    {return name}
}
```



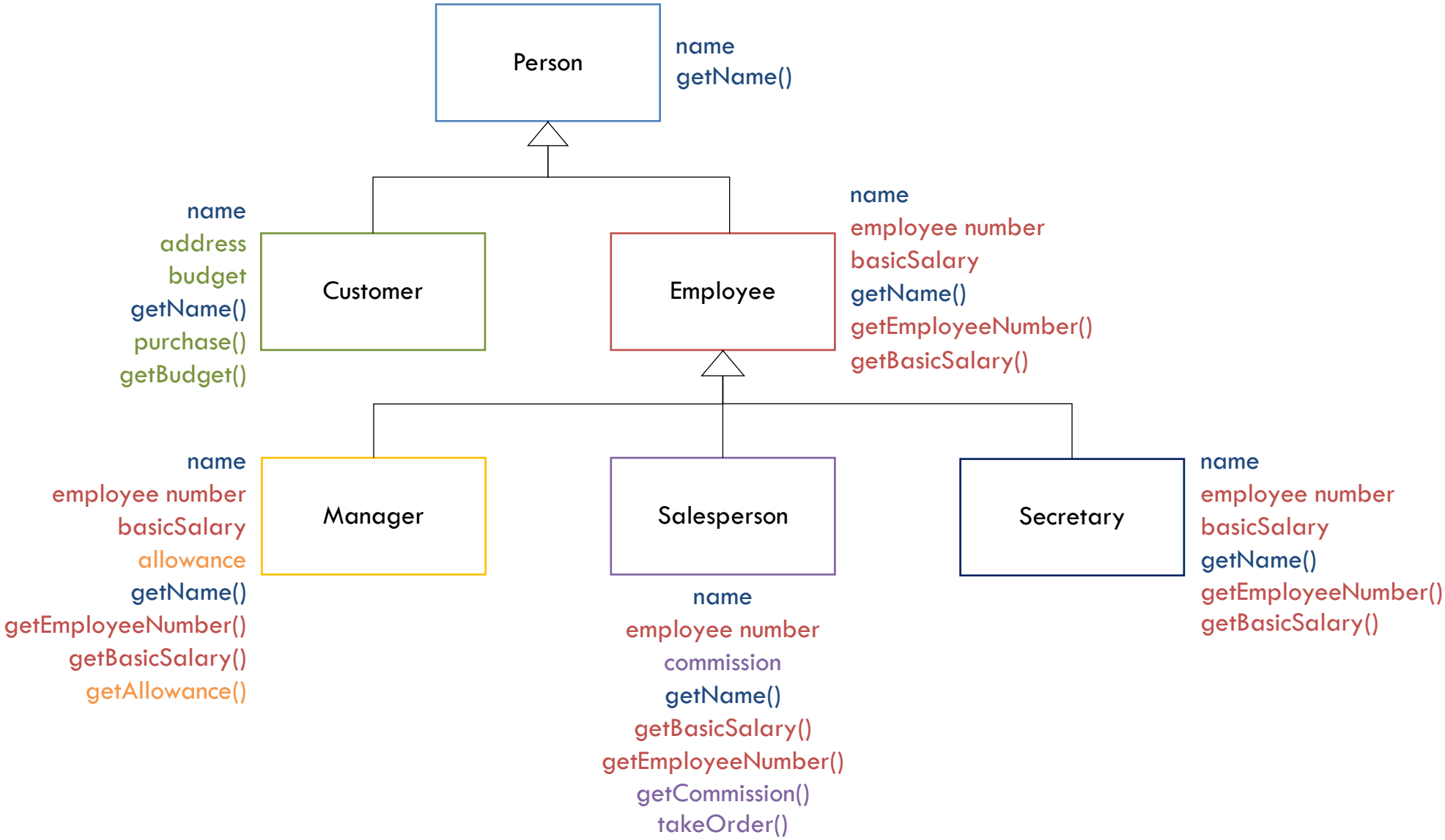

```
Class Employee {
   Attributes :
      name                                  (inherited from Person class)
      employee number
   Methods :
      getName()    {return name}     (inherited from Person class)
      getEmployeeNumber(){return employee number}
}
```

INHERITANCE




```
Class SalesPerson {
   Attributes :
      name                                                    (inherited from Person Class)
      employee number                                         (inherited from Employee class)
      commission
   Methods :
      getName()    {return name}                              (inherited from Person class)
      getEmployeeNumber(){return employee number}             (inherited from Employee class)
      getCommission() {return commission}
      takeOrder(who, stock, address, date)   {
         check with warehouse on stock availability
         check with warehouse on delivery schedule
         if ok
         then {  instruct warehouse to deliver stock
                  to address on date
                  return ok }
         else return not ok
         }
}
```

IMPLEMENTING INHERITANCE

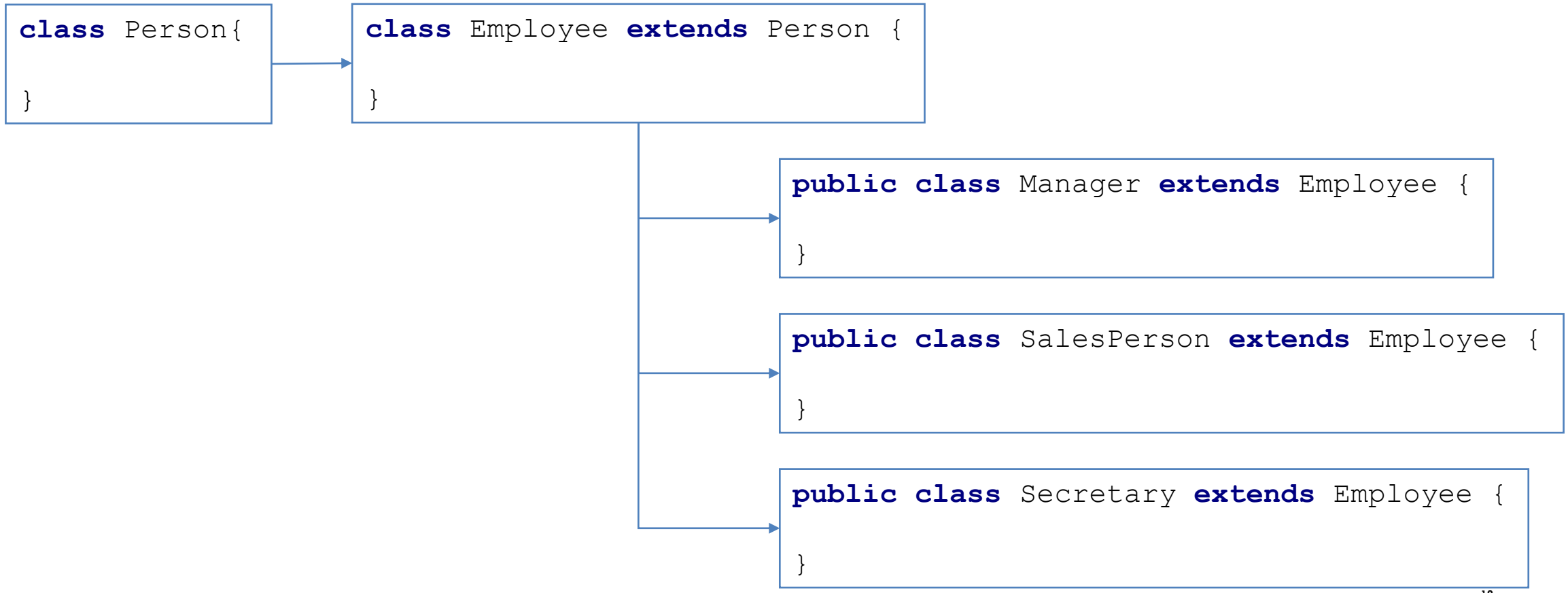


IMPLEMENTING INHERITANCE

- ความสัมพันธ์ระหว่าง **superclass – subclass** เกิดขึ้นจาก **keyword extends**
- จากตัวอย่าง **class Employee** เป็นส่วนขยาย **(extension)** ของ **class Person**
- ในขณะเดียวกัน **class Manager SalesPerson** และ **Secretary** ก็เป็นส่วนขยาย **(extension)** ของ **class Employee**

IMPLEMENTING INHERITANCE

- การเขียนโปรแกรมจากโครงสร้าง **class hierarchy diagram**



IMPLEMENTING INHERITANCE

- จาก **slide** ที่แล้ว จะเห็นว่าไม่มี **class** ที่มี **method main**
- เนื่องจาก **method main** มีหน้าที่ **run** จึงไม่มีความจำเป็นใน **class hierarchy diagram**

IMPLEMENTING INHERITANCE

- สมมุติ ถ้า **method main** รันแล้ว ได้ผลลัพธ์ดังนี้

```
The Manager Simon (employee number 01234M) has a salary of 9000
The Secretary Selene (employee number 98765S) has a salary of 2500
The Manager Simon also has an allowance of 2000
```

- จาก **output** จะเห็นว่า มีข้อมูลบางอย่างที่สืบทอดมาจาก **superclass**
- **Object** จาก **class Manager** หรือ **Secretary** มีการใช้งาน **basicSalary** ซึ่งเป็น **Attribute** ของ **class Employee**

IMPLEMENTING INHERITANCE

- เราพิจารณา Main() ซึ่งเป็น ซึ่งจะเริ่มด้วยการสร้างและกำหนดค่าเริ่มต้น (instantiate) ของ 2 object จาก class Manager : m และ Secretary : s

```
Manager m = new Manager("Simon", "01234M", 9000.0f, 2000.0f);
Secretary s = new Secretary("Selene", "98765S", 2500.0f);
```

IMPLEMENTING INHERITANCE

- ดังนั้นโครงสร้างของ **class** ทั้งหมดจึงเป็นดังนี้

```
class Employee extends Person {
    private float basicSalary;
    private String employeeNumber;
    Employee(String aName, String aEmployeeNumber, float aBasicSalary) {
        super(aName);
        employeeNumber = aEmployeeNumber;
        basicSalary = aBasicSalary;
    }
    public String getEmployeeNumber() {
        return employeeNumber;
    }
    public float getBasicSalary() {
        return basicSalary;
    }
}
```

```
class Person {
    private String name;
    public Person(String aName) {
        name = aName;
    }
    public String getName() {
        return name;
    }
}
```

```
public class Manager extends Employee {
    private float allowance;
    public Manager(String aName, String aEmployeeNumber, float aBasicSalary, float aAllowanceAmt) {
        super(aName, aEmployeeNumber, aBasicSalary);
        allowance = aAllowanceAmt;
    }
    public float getAllowance() {
        return allowance;
    }
}
```

IMPLEMENTING INHERITANCE

```
public class SalesPerson extends Employee {
    private float commission ;
    public SalesPerson (String aName, String aEmployeeNumber, float aBasicSalary) {
        super(aName, aEmployeeNumber, aBasicSalary);
    }
    public void takeOrder() {
    }
    public float getCommission(){
        return commission;
    }
}
```

```
public class Secretary extends Employee {
    public Secretary (String aName, String aEmployeeNumber, float aBasicSalary) {
        super(aName, aEmployeeNumber, aBasicSalary);
    }
}
```

IMPLEMENTING INHERITANCE

Manager : m
Attributes
- name : Simon
- employeeNumber : 01234M
- basicSalary : 9000
- allowance : 2000

Operations
- getName()
- getEmployeeNumber()
- getBasicSalary()
- getAllowance()

Secretary : s
Attributes
- name : Simon
- employeeNumber : 98765S
- basicSalary : 2500

Operations
- getName()
- getEmployeeNumber()
- getBasicSalary()

CODE REUSE

- จากการส่งต่อคุณสมบัติจาก **superclass** ไปยัง **subclass**
- อีกนัยหนึ่งก็คือ **subclass** สามารถใช้คุณสมบัติของ **superclass** ได้ ซึ่งก็คือ การนำกลับมาใช้ใหม่ **(Reuse)**

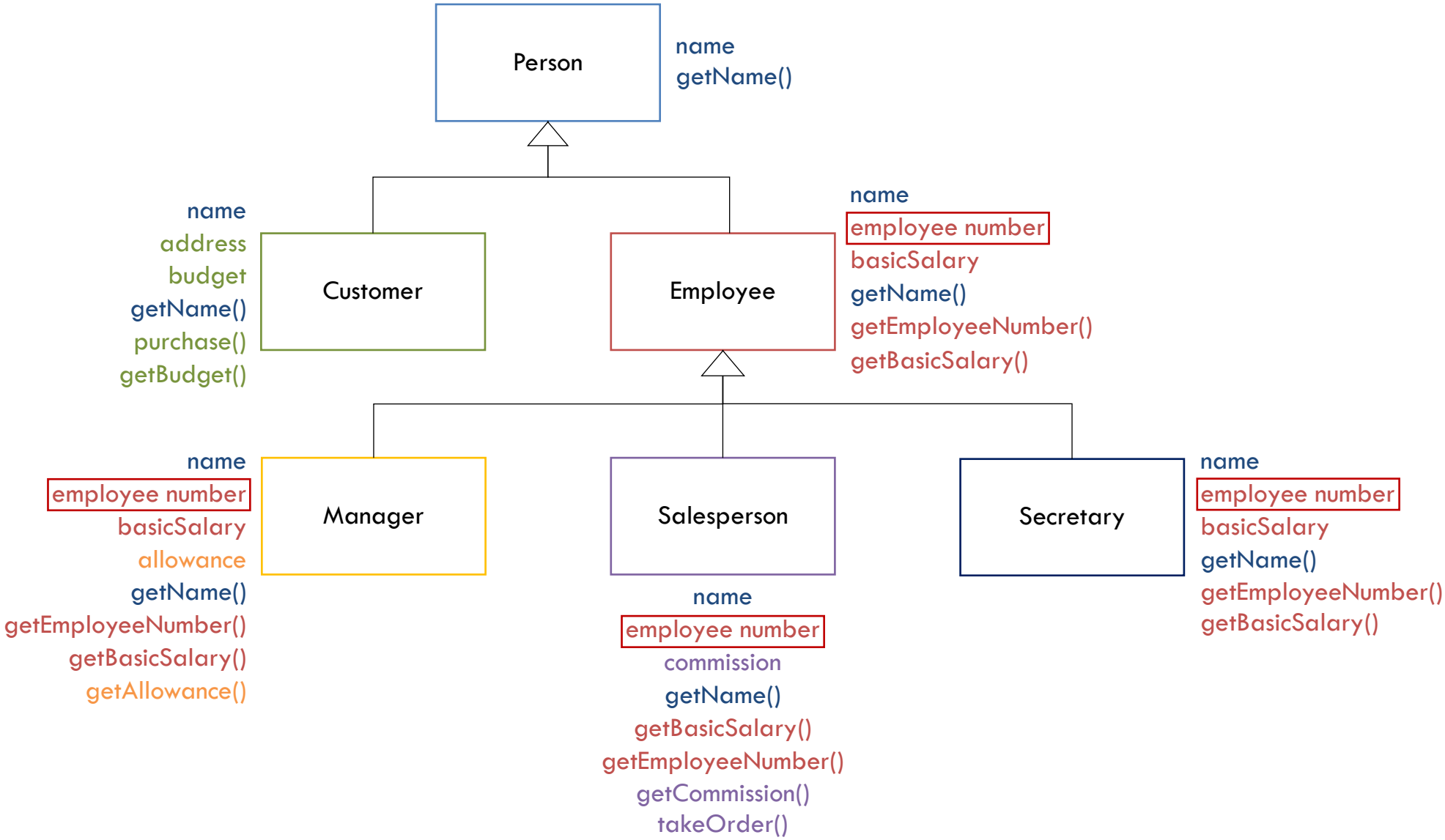
MAKING CHANGES IN CLASS HIERARCHY

- การเปลี่ยนแปลง **software specification** เป็นสิ่งที่หลีกเลี่ยงไม่ได้
- การเปลี่ยนแปลงที่เกิดขึ้น จะกระทบกับ **class hierarchy** ทั้งหมด
  - Change in property definition for all subclasses.
  - Change in property definition for some subclasses.
  - Adding/deleting a class.

MAKING CHANGES IN CLASS HIERARCHY >CHANGE IN PROPERTY DEFINITION FOR ALL SUBCLASSES

- ถ้าเราเปลี่ยนแปลง **employeeNumber** ใน **class Employee**
- การเปลี่ยนแปลงที่เกิดขึ้นจะไม่ใช่เฉพาะ **attribute employeeNumber**
- การเปลี่ยนแปลงจะเกิดกับที่ **class** ที่ **Inherit** และ **method** ที่ทำงานกับ **employeeNumber** ด้วย

MAKING CHANGES IN CLASS HIERARCHY >CHANGE IN PROPERTY DEFINITION FOR ALL SUBCLASSES



MAKING CHANGES IN CLASS HIERARCHY >CHANGE IN PROPERTY DEFINITION FOR ALL SUBCLASSES

```
class Employee extends Person {
    private float basicSalary;
    private String employeeNumber;
    Employee(String aName, String aEmployeeNumber, float aBasicSalary) {
        super(aName);
        employeeNumber = aEmployeeNumber;
        basicSalary = aBasicSalary;
    }
    public String getEmployeeNumber() {
        return employeeNumber;
    }
    public float getBasicSalary() {
        return basicSalary;
    }
}
```

MAKING CHANGES IN CLASS HIERARCHY >CHANGE IN PROPERTY DEFINITION FOR SOME SUBCLASSES

- ในบางครั้งการเปลี่ยนแปลงคุณสมบัติของ **superclass** อาจจะไม่ส่งผลต่อ **subclass** ทั้งหมด
- ยกตัวอย่างโดยเพิ่ม 2 **subclass** ของ **Employee** เข้าไป คือ
  - Technician
  - Clerk

MAKING CHANGES IN CLASS HIERARCHY >CHANGE IN PROPERTY DEFINITION FOR SOME SUBCLASSES

```
public class Technician extends Employee {
    Technician (String aName, String aEmployeeNumber, float aBasicSalary) {
        super(aName, aEmployeeNumber, aBasicSalary);
    }
}
```

```
public class Clerk extends Employee {
    Clerk (String aName, String aEmployeeNumber, float aBasicSalary) {
        super(aName, aEmployeeNumber, aBasicSalary);
    }
}
```

MAKING CHANGES IN CLASS HIERARCHY >CHANGE IN PROPERTY DEFINITION FOR SOME SUBCLASSES

- a manager—basic salary plus allowance;
- a salesperson—basic salary plus commission;
- a secretary—basic salary;
- a technician—basic salary;
- a clerk—basic salary.

MAKING CHANGES IN CLASS HIERARCHY >CHANGE IN PROPERTY DEFINITION FOR SOME SUBCLASSES

- ภายใน **class Employee** เพิ่ม **method getpay()** ซึ่งจะ **return** เงินเดือนของพนักงาน
- **getpay()** เป็น **method** ของ **class Employee** จะถูกสืบทอดไปยัง **subclass** ต่างๆ

MAKING CHANGES IN CLASS HIERARCHY >CHANGE IN PROPERTY DEFINITION FOR SOME SUBCLASSES

```
class Employee extends Person {
    private float basicSalary;
    private String employeeNumber;
    public Employee(String aName, String aEmployeeNumber, float aBasicSalary) {
        super(aName);
        employeeNumber = aEmployeeNumber;
        basicSalary = aBasicSalary;
    }
    public String getEmployeeNumber() {
        return employeeNumber;
    }
    public float getBasicSalary() {
        return basicSalary;
    }
    public float getPay() { return basicSalary; }
}
```

MAKING CHANGES IN CLASS HIERARCHY >CHANGE IN PROPERTY DEFINITION FOR SOME SUBCLASSES

```
public class Main {

    public static void main(String[] args) {
        Manager m = new Manager("Simon", "01234M", 9000.0f, 2000.0f);
        Secretary s = new Secretary("Selene", "98765S", 2500.0f);
        Technician t = new Technician("Terrence", "42356T", 2000.0f);
        Clerk c = new Clerk("Charmaine", "68329C", 1200.0f);
        System.out.print("The Manager " +m.getName()+ " (employee number "+m.getEmployeeNumber()+")");
        System.out.println(" has a pay of "+m.getPay());
        System.out.print("The Secretary "+s.getName()+ " (employee number "+s.getEmployeeNumber()+")");
        System.out.println(" has a pay of "+s.getPay());
        System.out.print("The Technician "+t.getName()+ " (employee number "+t.getEmployeeNumber()+")");
        System.out.println(" has a pay of "+t.getPay());
        System.out.print("The Clerk "+c.getName()+ " (employee number "+c.getEmployeeNumber()+")");
        System.out.println(" has a pay of "+c.getPay());
    }
}
```

MAKING CHANGES IN CLASS HIERARCHY >CHANGE IN PROPERTY DEFINITION FOR SOME SUBCLASSES

- Output

```
The Manager Simon (employee number 01234M) has a pay of 9000.0
The Secretary Selene (employee number 98765S) has a pay of 2500.0
The Technician Terrence (employee number 42356T) has a pay of 2000.0
The Clerk Charmaine (employee number 68329C) has a pay of 1200.0
```

MAKING CHANGES IN CLASS HIERARCHY >CHANGE IN PROPERTY DEFINITION FOR SOME SUBCLASSES

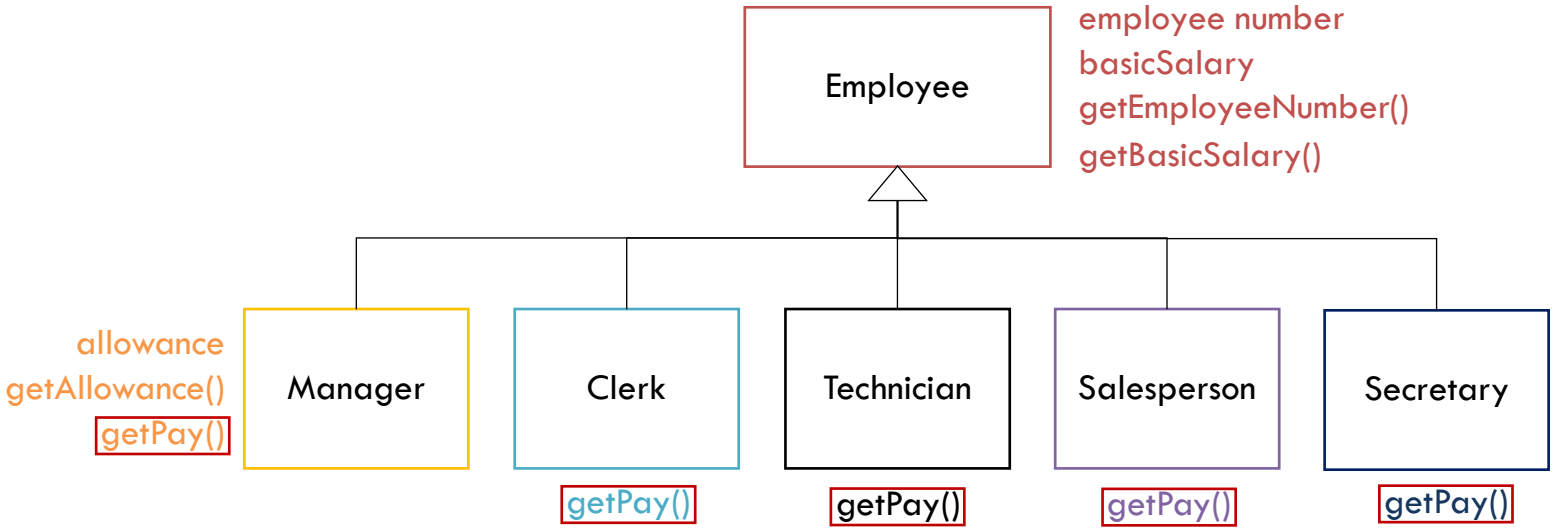
- สังเกตที่ m จาก class Manager เงินเดือนนั้นจะต้องเป็น
  - a manager—basic salary plus allowance;
- แต่สิ่งที่แสดงคือ basic salary ยังไม่ได้รวม allowance
- ค่าของ allowance คือ 2000
- ทำอย่างไร ?????

MAKING CHANGES IN CLASS HIERARCHY >CHANGE IN PROPERTY DEFINITION FOR SOME SUBCLASSES

- วิธีที่ 1
  - Remove the getPay() method from the Employee class and define it individually in the subclasses (Secretary, Technician, Clerk, and Manager).
- วิธีที่ 2
  - Maintain the definition of getPay() method in Employee class and *redefine* it in the Manager class.

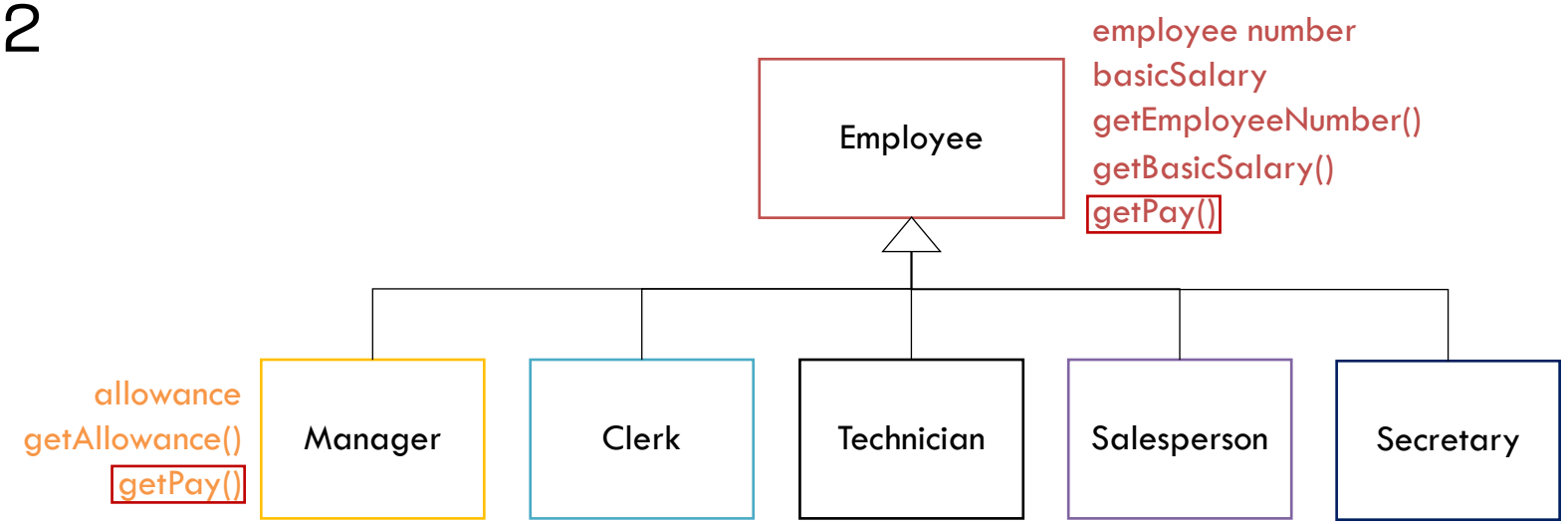
MAKING CHANGES IN CLASS HIERARCHY >CHANGE IN PROPERTY DEFINITION FOR SOME SUBCLASSES

วิธีที่ 1



MAKING CHANGES IN CLASS HIERARCHY >CHANGE IN PROPERTY DEFINITION FOR SOME SUBCLASSES

วิธีที่ 2



MAKING CHANGES IN CLASS HIERARCHY >CHANGE IN PROPERTY DEFINITION FOR SOME SUBCLASSES

- จากวิธีที่ 2 การสร้าง **method getPay()** ใน **class Manager** เป็นการสร้าง **method** ใหม่ที่ใช้ชื่อเหมือนกันกับ **getPay()** ที่สืบทอดมาจาก **Employee**

MAKING CHANGES IN CLASS HIERARCHY >CHANGE IN PROPERTY DEFINITION FOR SOME SUBCLASSES

```
public class Manager extends Employee {
    private float allowance;
    public Manager(String aName, String aEmployeeNumber, float aBasicSalary, float aAllowanceAmt)
{
        super(aName, aEmployeeNumber, aBasicSalary);
        allowance = aAllowanceAmt;
    }
    public float getAllowance() {
        return allowance;
    }
    public float getPay(){
        return (getBasicSalary()+allowance);
    }
}
```

MAKING CHANGES IN CLASS HIERARCHY >CHANGE IN PROPERTY DEFINITION FOR SOME SUBCLASSES

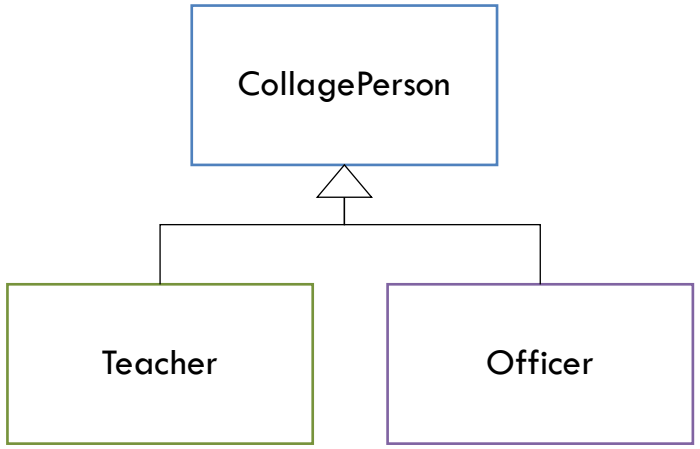
- Run

```
The Manager Simon (employee number 01234M) has a pay of 11000.0
The Secretary Selene (employee number 98765S) has a pay of 2500.0
The Technician Terrence (employee number 42356T) has a pay of 2000.0
The Clerk Charmaine (employee number 68329C) has a pay of 1200.0
```

MAKING CHANGES IN CLASS HIERARCHY >CHANGE IN PROPERTY DEFINITION FOR SOME SUBCLASSES

- จากการทดสอบ วิธีที่ 2 ดีกว่า วิธีที่ 1
- เพราะเป็นการใช้คุณสมบัติ **reuse** ทำให้การเขียน **code** นั้นสั้นลง และแก้ไขได้ง่าย
- การประกาศ **method** ใหม่โดยใช้ชื่อเดียวกันกับ **method** ที่สืบทอดมานั้น ในเชิงวัตถุนั้นเรียกว่า Overloading

EXERCISES



- เขียนโปรแกรมส่ง
  - ใช้ Constructer ได้

```
Class CollagePerson {
   Attributes :
      name
      gender
      salary
   Methods :
      getName()      {return name}
      getGender()    {return gender}
      getSalary()    {return salary}
}
```

```
Class Teacher {
   Attributes :
      overload
      hour
   Methods :
      getSalary()    {return salary + overload}
      getHour()    {return hour}
}
```

```
Class Officer {
   Attributes :
      position
   Methods :
      getPosition() {return position}
}
```

สรุป