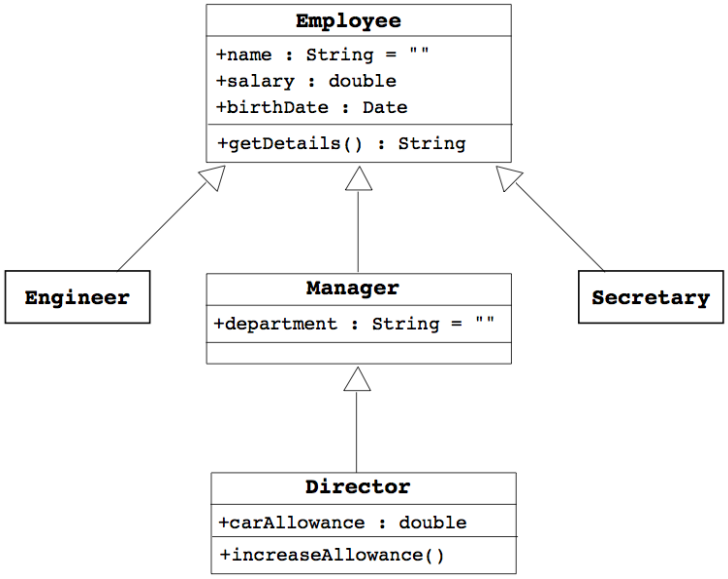
Chapter 7 Inheritance

1. inheritance creates specialized class versions of a another class
2. new class is called: derived/subclass/child class
   original class is called: base/superclass/parent class
3. derived class = (base class attributes + methods) + (new attributes + new methods)
   inheritance list:
   a. public methods (no private methods!)
   b. all public and private instance variables
      i. private instance variables should be accessed by public accessor and mutator
   c. all public and private static variables
4. 繼承的目的：重複再利用相同的資料，share!
5. single inheritance: inherits from only one class

```
<modifier> class <name> [extends <superclass>] {
      <declarations>*
}
```

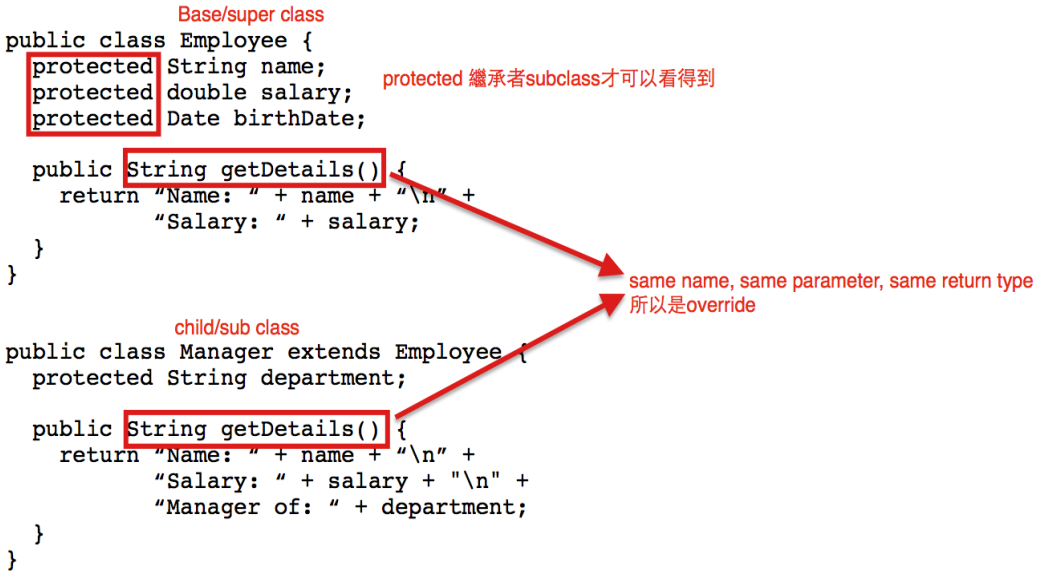
6. 越上層的class的共同屬性越少，所能包含的類別越多。
   越下層的class的共同屬性越多，所能包含的類別越少。




其他範例：數系（上層→下層）：實數→有理數→整數
圓錐曲線→橢圓、雙曲線、拋物線→圓（越下層限制條件越多）

7. overriding methods：subclass non-static method和superclass non-static method的method name、參數類型、回傳類型完全一樣(因為static不care有沒有物件)
   a. overloading methods：同一class下，有多個相同method name，但method的傳入參數類型或數量不同
   b. access control 權限控制

| Modifier | Same Class | Same Package | Subclass | Universe |
|---|---|---|---|---|
| private | Yes | | | |
| default | Yes | Yes | | |
| protected | Yes | Yes | Yes | |
| public | Yes | Yes | Yes | Yes |




   c. changing access permission of an overridden method
      (only becoming more permissive，只能越來越公開)
      i. base class: `private void doSomething()`
         derived class: `public void doSomething()`
         This is OK
      ii. base class: `public void doSomething()`
         derived class: `private void doSomething()`
         This is not OK
   d. covariant return type：
      i. 若return type是一種class，那這個return type可以為這個class或是這個class的繼承class。

ii. if there is a return class type, then the returned type may be changed to that of any descendent class of the returned type

iii. for example

```
public class BaseClass
{
      public Employee getSomeone(int someKey);
}
public class DerivedClass extends BaseClass
{
      public Manager getSomeone(int someKey);
}
```

Manager繼承Employee，所以overriding時，可以使用subclass回傳

8. `final`
   a. final method: method cannot be overrided by derived class
   b. final class: class cannot be inheritted to derive other classes
9. `super` constructor
   a. derived class calls base class constructor to initialize all the data inherited from the base class by using only `super`
   b. syntax

```
public derivedClass(int p1, int p2, double p3)//constructor
{
      super(p1, p2);
      instanceVariable  = p3;
}
```

   c. instance variable cannot be used as an argument to `super`
   d. super must always be the first action in the constructor
   e. if derived class constructor does not include invocation of super, then the base class no-argument constructor will automatically be invoked
      i. this can result in an error if the base class has not defined a no-argument constructor
10. `super` keyword
    a. refered to its superclass
    b. for example

```
public class BaseClass{
      private String name;
      public String getName(){
            return name;
      }
}
public class DerivedClass{
      public String getName(){
            return super.getName(); //call superclass method
      }
}
```

11. `this` constructor
    a. to invoke another constructor in the same class
    b. if both `super` and `this` are necessay,
        i. call `this` first
        ii. the constructor called by `this` must call `super` first
    c. for example

```
public ClassName()
{
      this(argument1, argument2);
}
public ClassName(type1 param1, type2 param2)
{
      ...
}
```

12. polymorphism
    a. definition: the ability to have many different forms
        i. for example, `DerivedClass` has access to methods from `BaseClass`
    b. derived class object can be assigned to a variable of any ancestor type

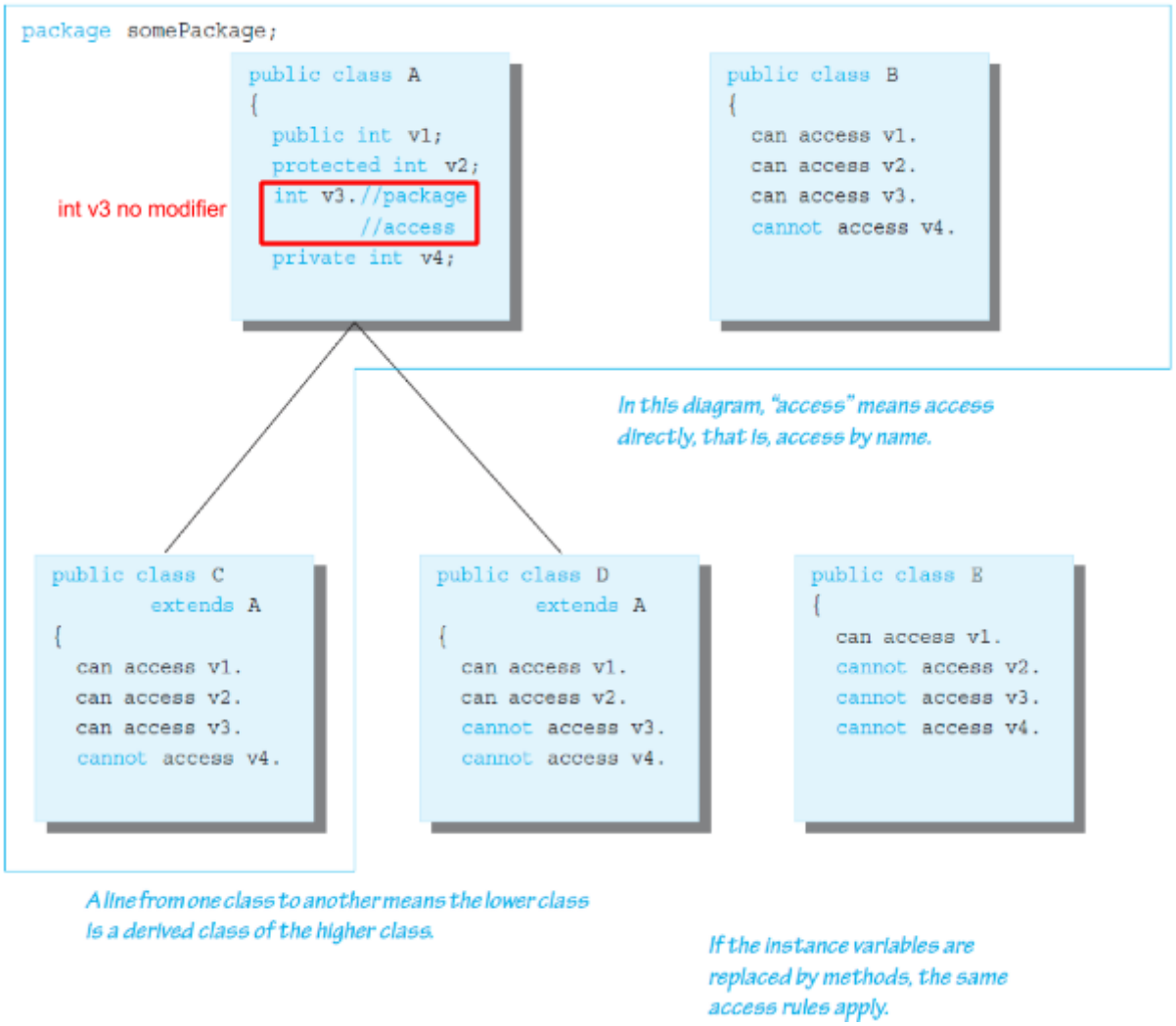
```
//DerivedClass是一種BaseClass; Manager是一種Employee
BaseClass A = new DerivedClass();      //legal
Employee employee = new Manager();     //legal

//BaseClass不是一種DerivedClass; Employee不是一種Manager
DerivedClass B = new BaseClass();      //illegal
Manager manager = new Employee();      //illegal
```

    c. derived class object can be plugged in as a parameter in place of any of its ancestor classes
    d. virtual method invocation

```
Employee e = new Manager();
e.getDetails();  //這會執行Manager的getDetails
                 //若Manager沒有getDetails method，就會執行
                 //Employee的getDetails
```

13. Package Access/default access/friendly access
    a. an instance variable or method definition not preceded with a modifier has package access and can be accessed by name for any class in the same package
    b. cannot be accessed outside the package
    c. more restricted than `protected`

| Modifier | Same Class | Same Package | Subclass | Universe |
|---|---|---|---|---|
| private | Yes | | | |
| default | Yes | Yes | | |
| protected | Yes | Yes | Yes | |
| public | Yes | Yes | Yes | Yes |




14. cannot use multiple `super`s

15. `Object` class
    a. in Java, every class is a descendent of `Object`
    b. if a class not explicitly derived from another class, it is automatically derived from class `Object`
    c. in the `java.lang` package
    d. methods inherited
        i. `equals`, `toString`, `getClass`
        ii. should be overridden, for example:

**//must use getClass()!!!! Not instanceof**

```
public boolean equals(Object otherObject){
      if(otherObject == null)
            return false;
      else if(getClass() != otherObject.getClass())
            return false;
      else{
            Employee otherEmployee = (Employee) otherObject;
            return (name.equals(otherEmployee.name &&
                  hireDate.equals(otherEmployee.hireDate));
      }
}
```

16. `instanceof` Operator
    a. checks if second argument is the type given，檢查指標變數本身的類型
    b. syntax:
       `Object instanceof ClassName`
    c. return true, if…
        i. `Object` is of type `ClassName`
        ii. `Object` is a type of any descendent class of `ClassName`
    d. otherwise, return false
17. `getClass()` method
    a. every object inherits `getClass()` from `Object`，檢查指標變數指到的內容的類型
    b. `final` method (cannot be overridden)
    c. returns representation only of the class that was used with `new` to create the object (to check if exact same class)
    d. can compare using `==` or `!=`
18. `Class Class<T>`：此種類型，有reflex的功能，可以知道物件內有什麼、物件是什麼
19. composition: class contains instance variable of a class type