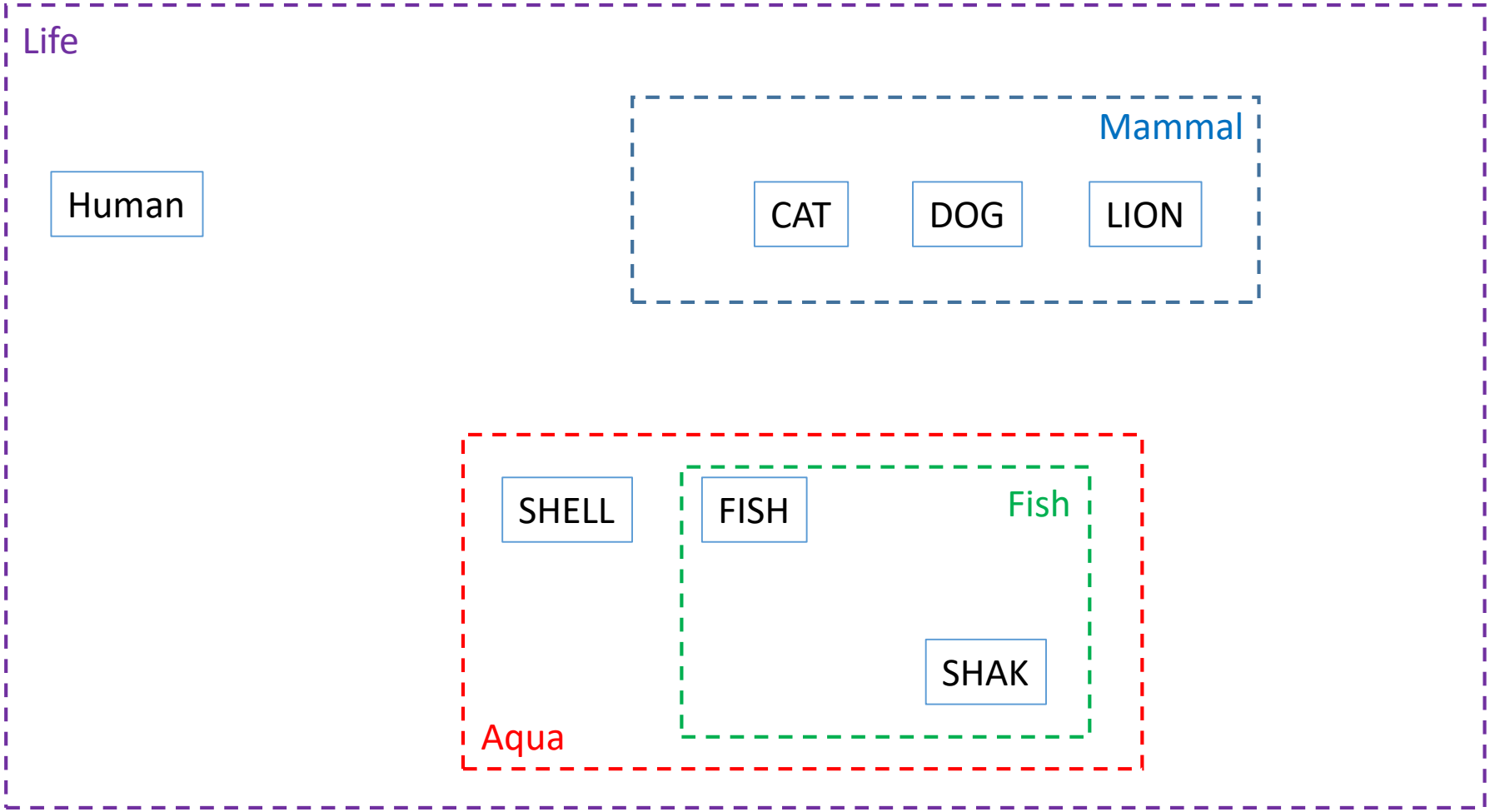
•CS117 OOP

•Chapter 3

•Class relationship

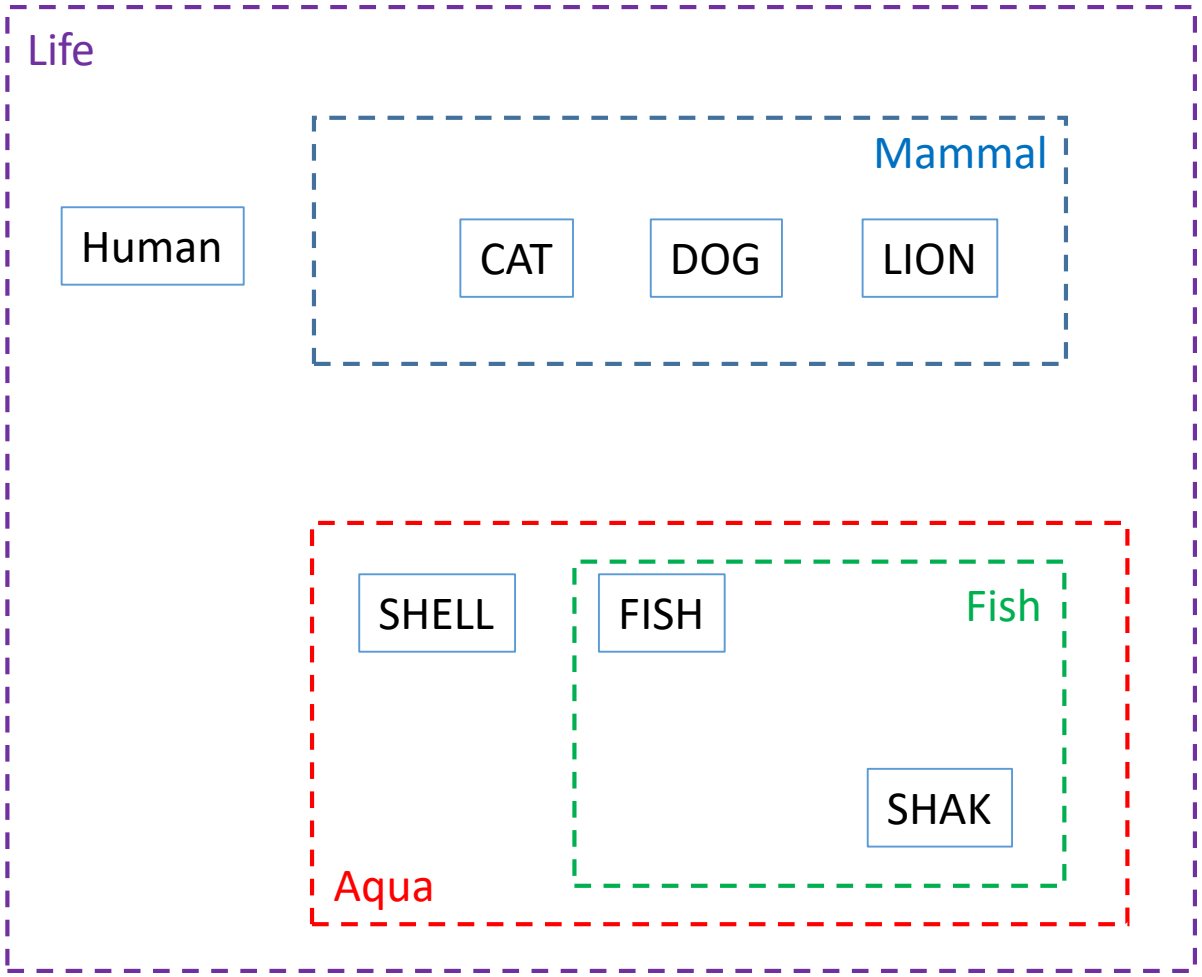
By Dr. Paween Khoenkaw
Computer Science MJU

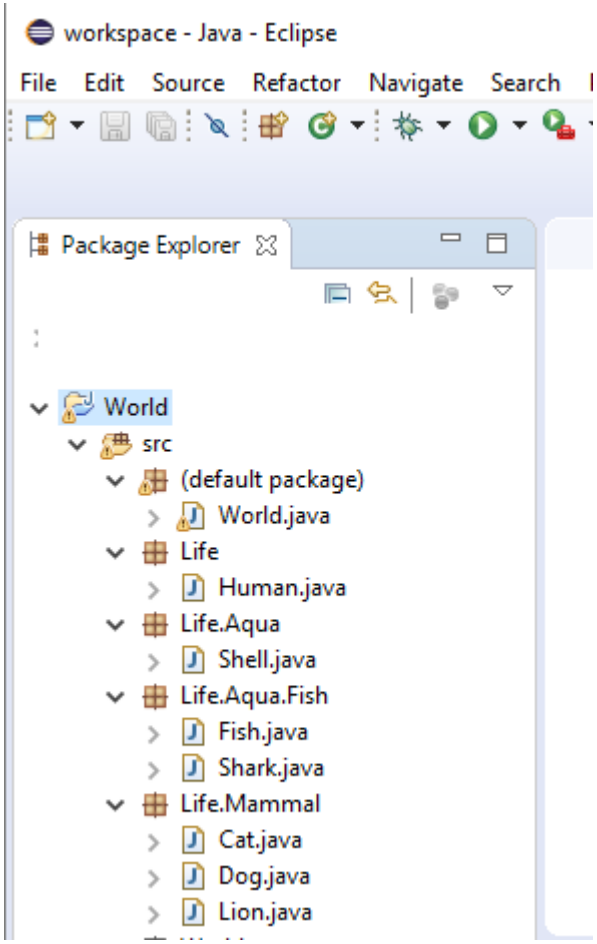
Java Package



Life
Life.Mammal
Life.Aqua
Life.Aqua.Fish

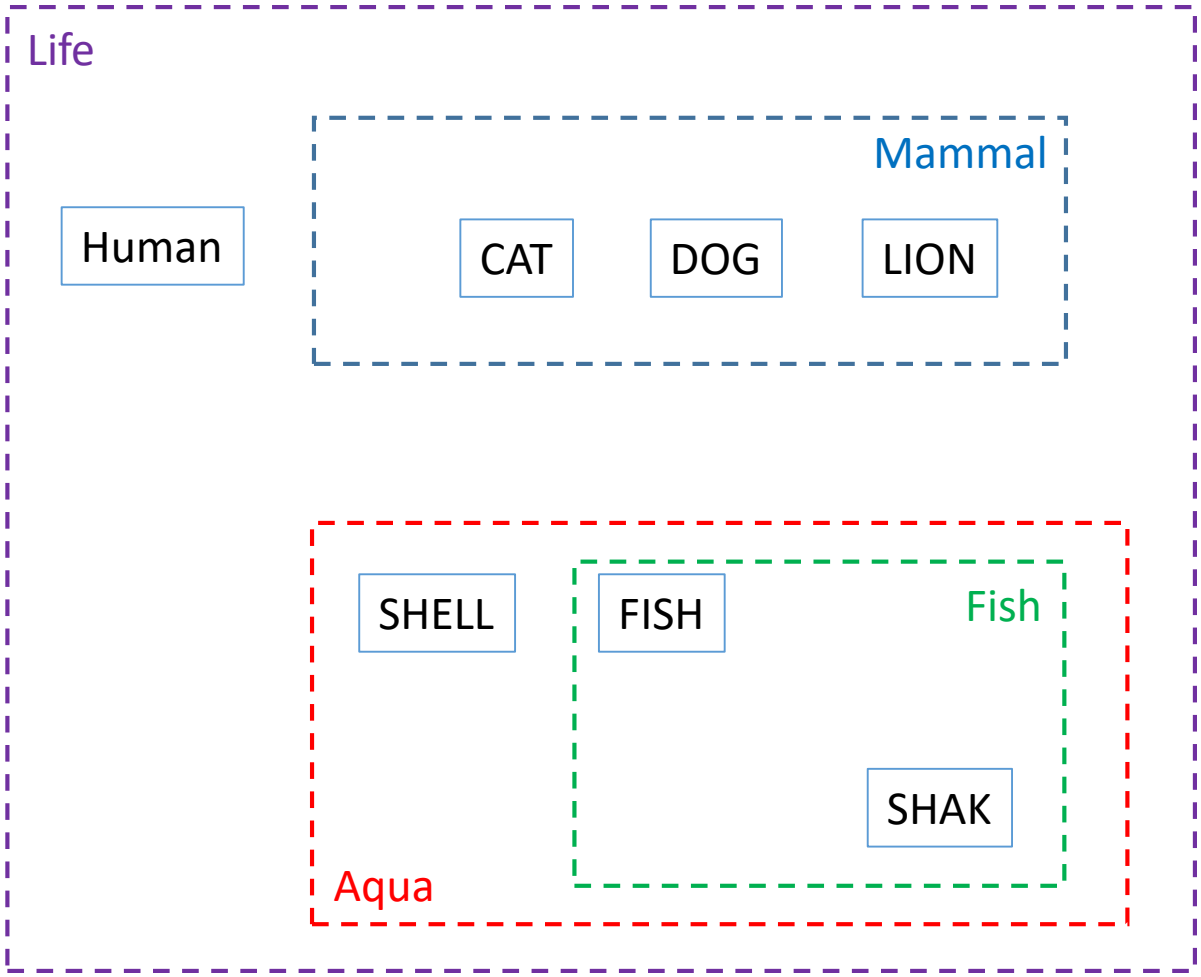
Java Package





Life
Life.Mammal
Life.Aqua
Life.Aqua.Fish

Java Package

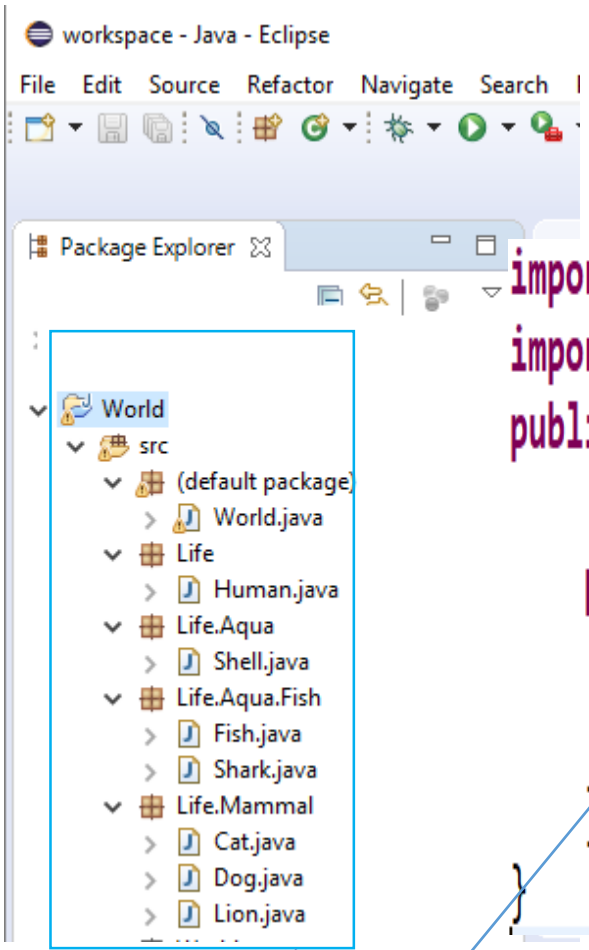


World
- src
  - World.java
  - Life
    - Human.java
    - Aqua
      - Shell.java
      - Fish
        - Fish.java
        - Shark.java
    - Mammal
      - Cat.java
      - Dog.java
      - Lion.java

Life
Life.Mammal
Life.Aqua
Life.Aqua.Fish

Java Package

World
- src
  - World.java
  - Life
    - Human.java
    - Aqua
      - Shell.java
      - Fish
        - Fish.java
        - Shark.java
    - Mammal
      - Cat.java
      - Dog.java
      - Lion.java




```
import Life.Human;
import Life.Aqua.Fish.Shark;
public class World {

    public static void main(String[] args) {
      Human peter=new Human();
      Shark jaws=new Shark();
    }
}
```

Class

Instance

Life
Life.Mammal
Life.Aqua
Life.Aqua.Fish

Java Package

Why do we need package

“C” does not support package

Java Package

Car 

Airplane 

Driver 

Car 

Airplane 

Driver 

Car honda= new Car(); 

Java Package

vehicle

Car 

Airplane 

Driver 

toy

Car 

Airplane 

Driver 

vehicle.Car honda= new vehicle.Car(); 

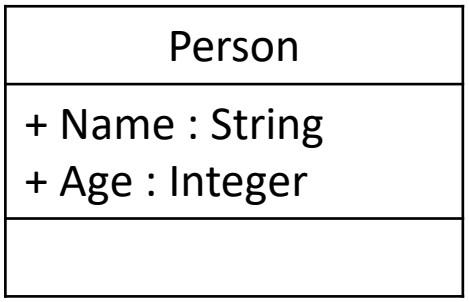
toy.Car honda= new toy.Car(); 

Java Package

Java Development Kits	Codename	Release	Packages	Classes
Java SE 8 with JDK 1.8.0	---	2014	217	4,240
Java SE 7 with JDK 1.7.0	Dolphin	2011	209	4,024
Java SE 6 with JDK 1.6.0	Mustang	2006	203	3,793
Java 2 SE 5.0 with JDK 1.5.0	Tiger	2004	166	3,279
Java 2 SE with SDK 1.4.0	Merlin	2002	135	2,991
Java 2 SE with SDK 1.3	Kestrel	2000	76	1,842
Java 2 with SDK 1.2	Playground	1998	59	1,520
Development Kit 1.1	---	1997	23	504
Development Kit 1.0	Oak	1996	8	212

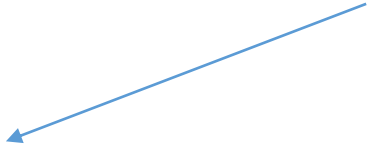
https://docs.oracle.com/javase/8/docs/api/allclasses-noframe.html

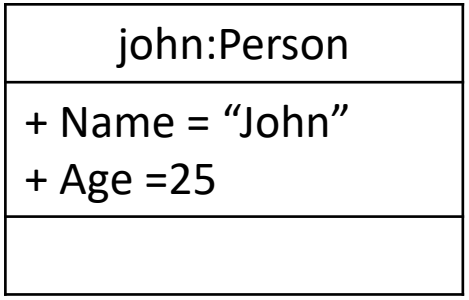
Encapsulation



Class

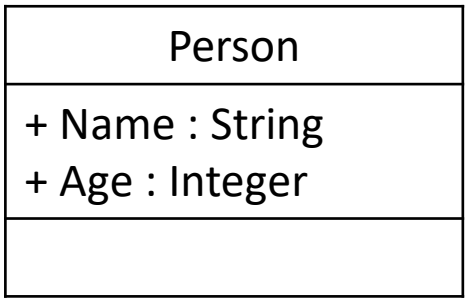
```
Person John = new Person();
John.Name="John";
John.Age=25;
```





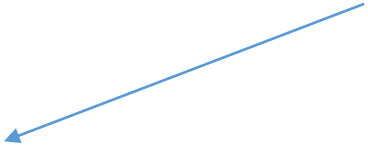
Instance

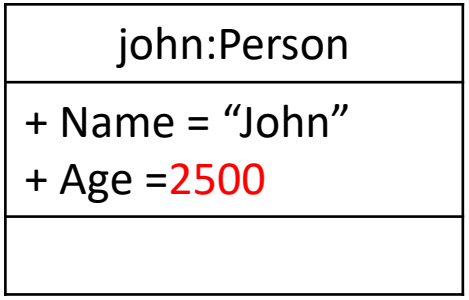
Encapsulation



Class

```
Person John = new Person();
John.Name="John";
John.Age=2500;
```





Instance

Encapsulation (การเข้าห่อ)

ป้องกันการเข้าถึง , แก้ไข , ใช้งาน

Properties และ Methods

ช่วยให้ code มีความปลอดภัยมากขึ้น

Encapsulation

Access Modifier

+ **Public**
- **Private**
Protected

Modifier	Class	Package	Subclass	World
`public`	✔	✔	✔	✔
`protected`	✔	✔	✔	✘
`no modifier*`	✔	✔	✘	✘
`private`	✔	✘	✘	✘

No read-only access modifier in JAVA !

Encapsulation

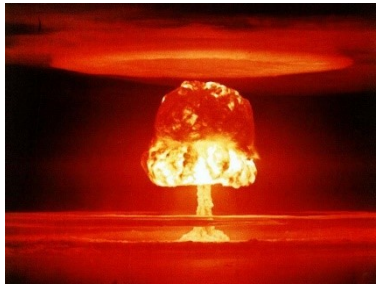
Person
+ Name : String - Age : Integer

Class

```
Person John = new Person();
John.Name="John";
John.Age=25;
```

john:Person
+ Name = "John" + Age =

Instance



+ Public

- Private

 # Protected

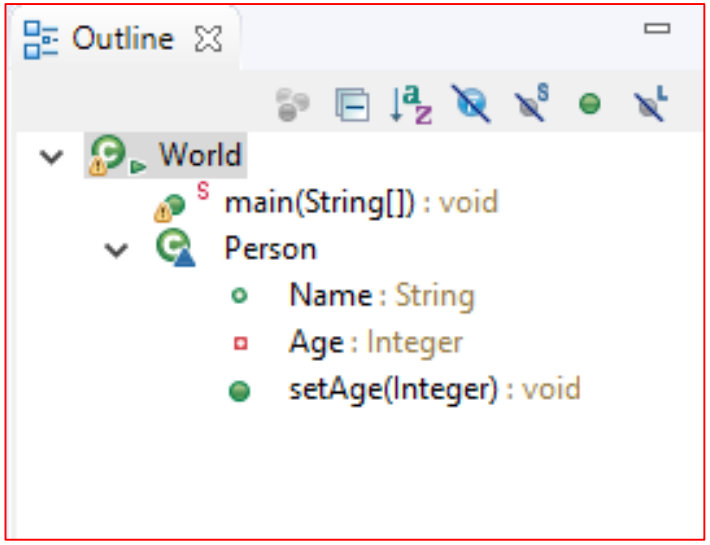
Modifier	Class	Package	Subclass	World
public	✔	✔	✔	✔
protected	✔	✔	✔	✘
*no modifier**	✔	✔	✘	✘
private	✔	✘	✘	✘

Encapsulation

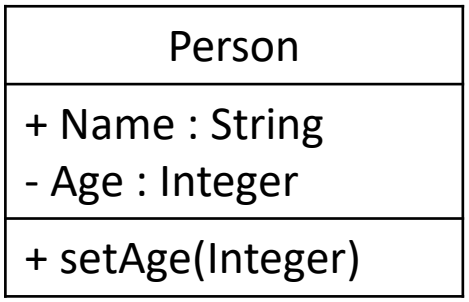
Person
+ Name : String - Age : Integer
+ setAge(Integer)

Class

```
class Person{
    public String Name;
    private Integer Age;
 public void setAge(Integer x)
 {
     if((x > 0)  && (x< 100))
     {
         Age= x;
     }
 }

                 }
```

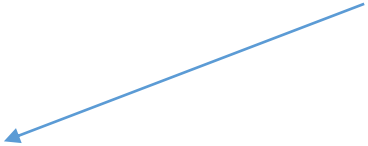


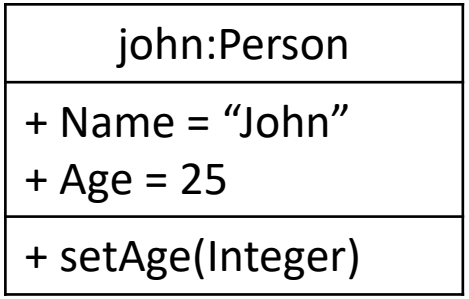
Encapsulation



Class

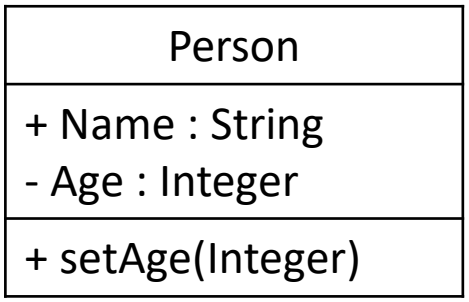
```
Person John = new Person();
John.Name="John";
John.setAge(25);
```





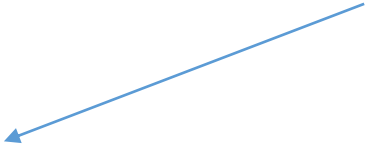
Instance

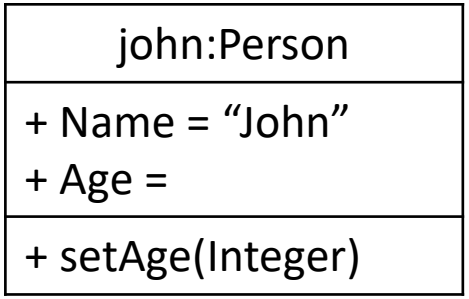
Encapsulation



Class

```
Person John = new Person();
John.Name="John";
John.setAge(2500);
```





Instance

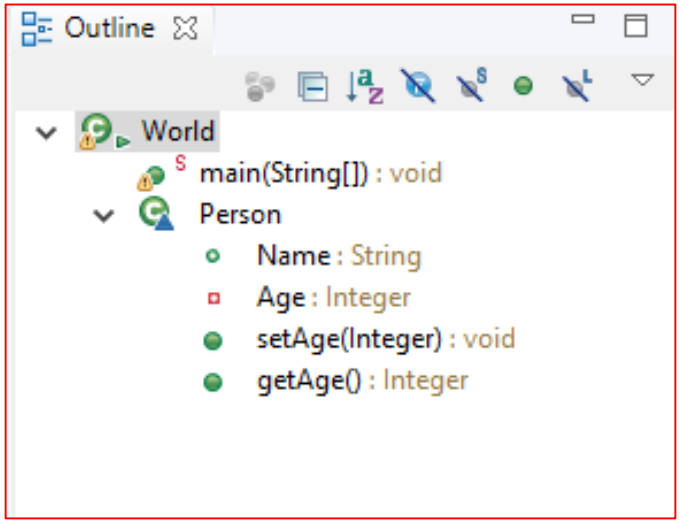
จะเรียกใช้ค่าของ Age มาใช้ได้อย่างไร ?

No read-only access modifier in JAVA !

Encapsulation

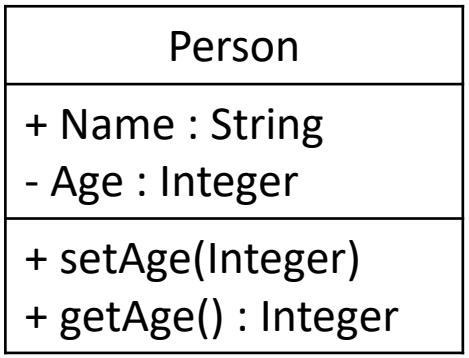
Person
+ Name : String - Age : Integer
+ setAge(Integer) + getAge() : Integer

Class



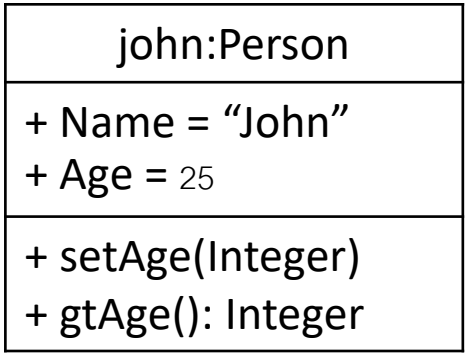

```
class Person{
    public String Name;
    private Integer Age;
 public void setAge(Integer x)
 {
     if((x > 0)  && (x< 100))
     {
         Age= x;
     }
 }
 public Integer getAge(){
    return Age;
                    }


                }
```

Encapsulation



Class

```
Person John = new Person();
John.Name="John";
John.setAge(25);
Integer age= John.getAge();
```

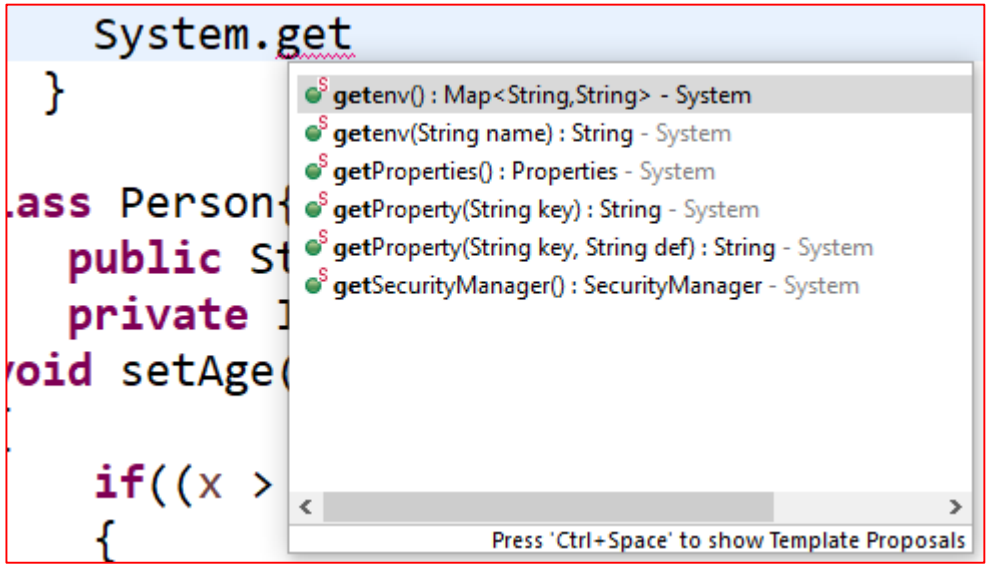


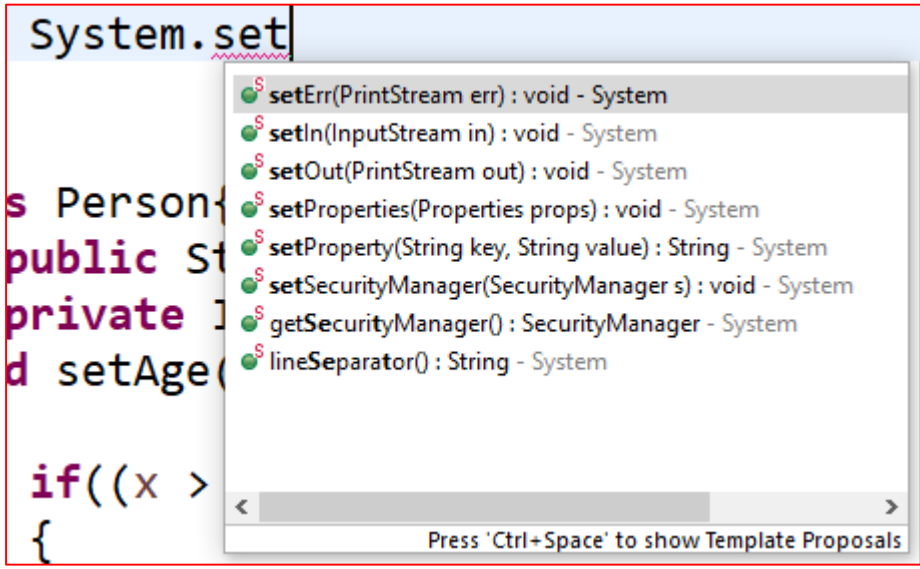
Instance

JAVA get / set Method

getXXX method นิยมใช้ในการเรียกอ่านสถานะของ property

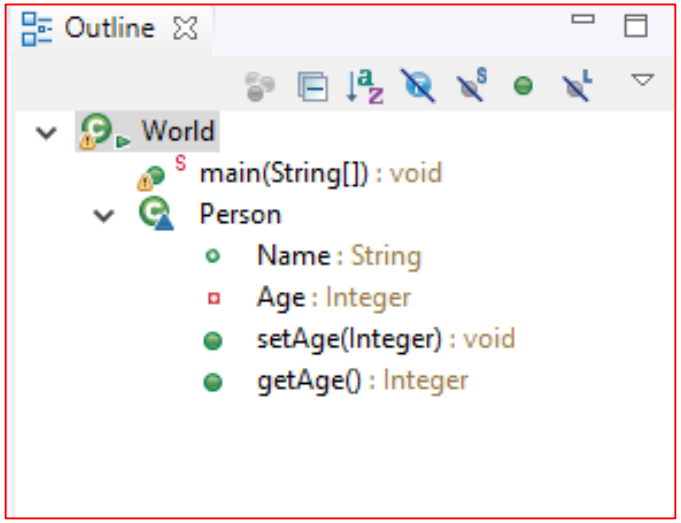
setXXX method นิยมใช้ตั้งค่าสถานะของ property





Eclipse JAVA outline icon

Person
+ Name : String - Age : Integer
+ setAge(Integer) + getAge() : Integer



	class (public) ✓
	interface (public)
	enum type (public)
	annotation type (public)
	package visible class ✓
	private class
	protected class
	default field (package visible)
	private field ✓
	protected field
	public field ✓
	default method (package visible)
	private method
	protected method
	public method ✓

Encapsulation

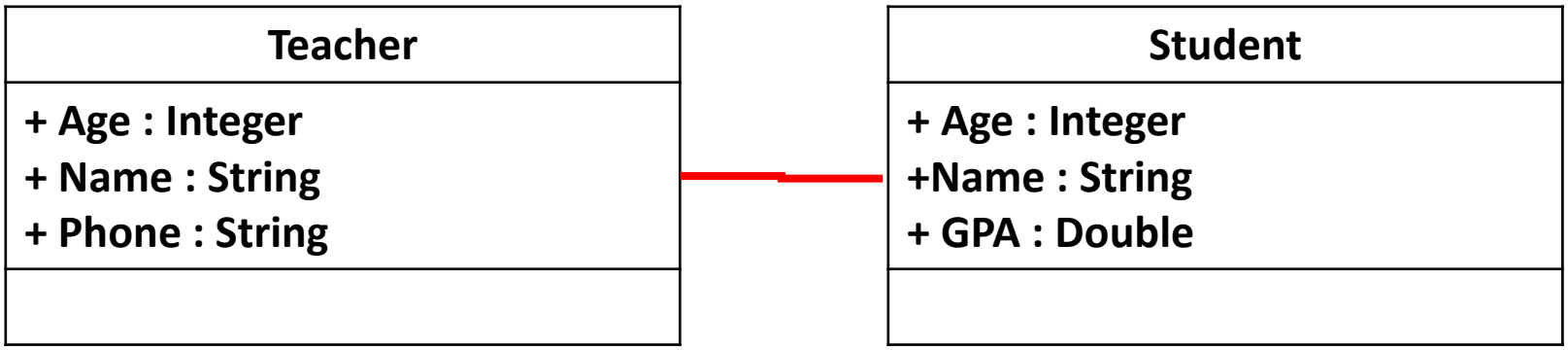
Circle
+ PI : Double + Radius : Double
+ getArea() : Double + getRadius() : Double + getCircumference() : Double - setRadius(Double)

Circle
- PI : Double - Radius : Double
+ getArea() : Double + getRadius() : Double + getCircumference() : Double + setRadius(Double)

พักสมองแป๊บนึง

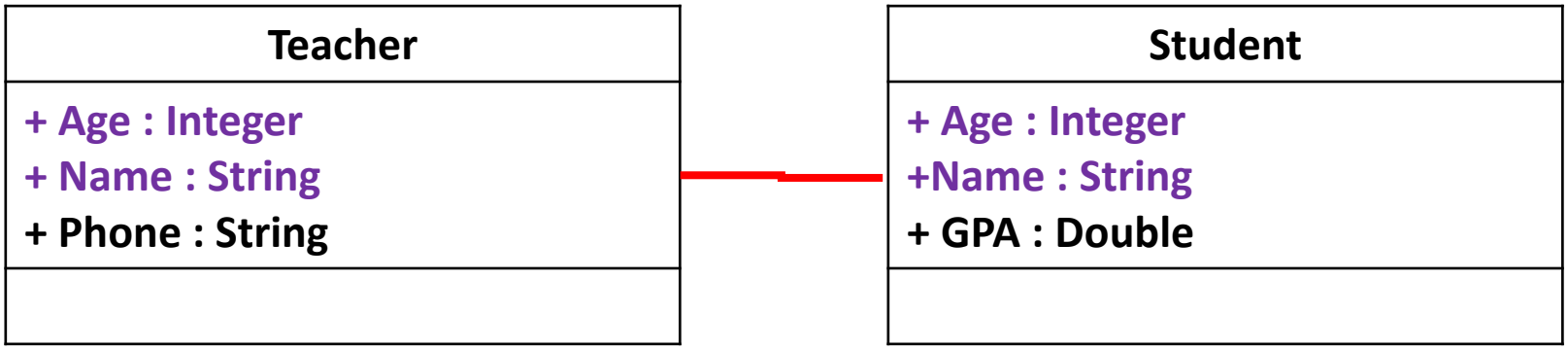
Class Relation

The class room

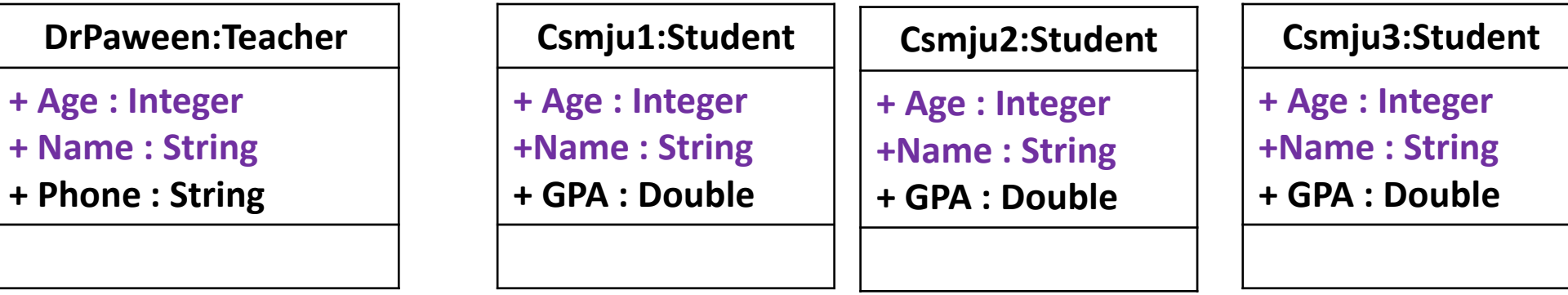


Class Relation

The class room

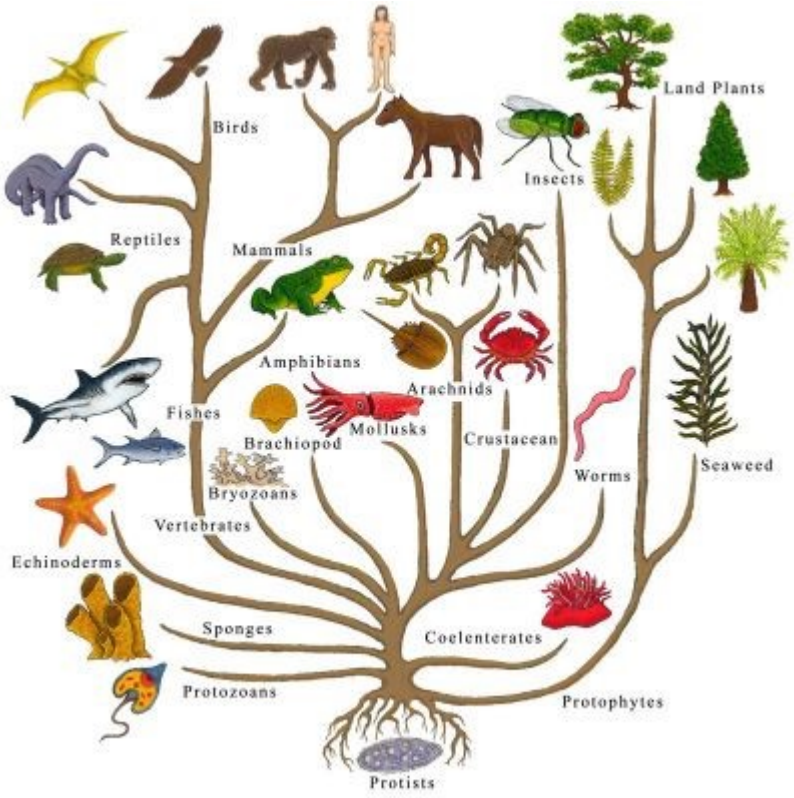


ซ้ำกัน

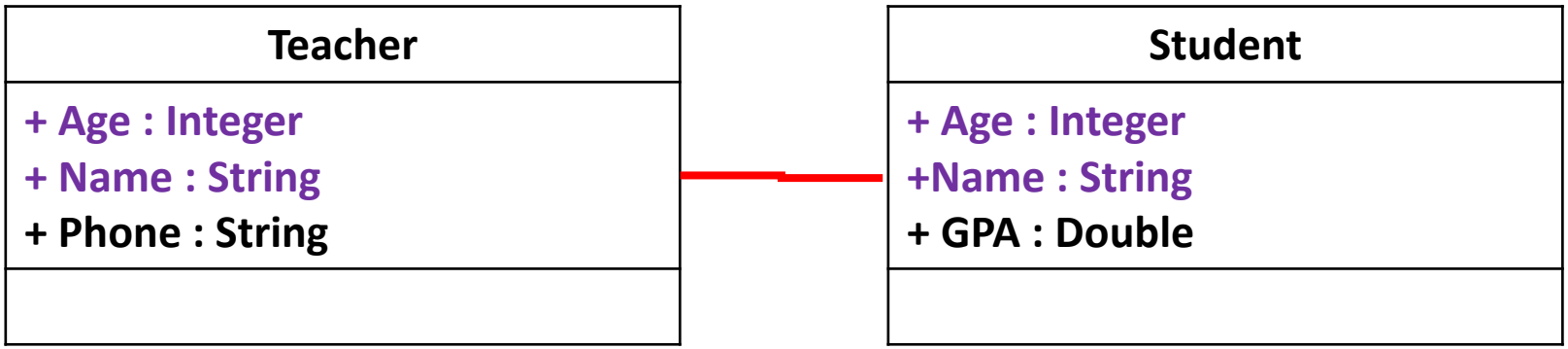


Class Relation

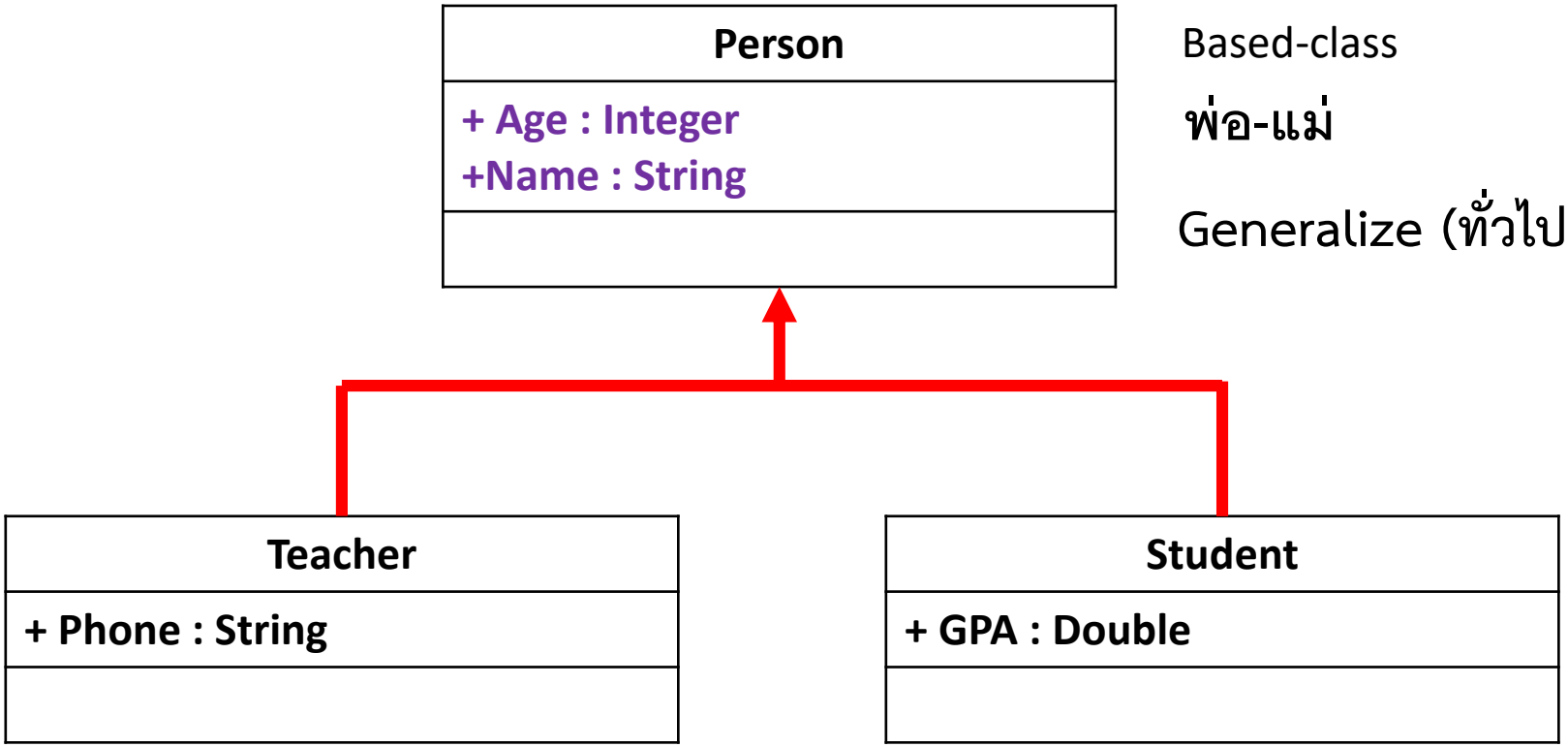
Inheritances (การสืบทอด)



Inheritances

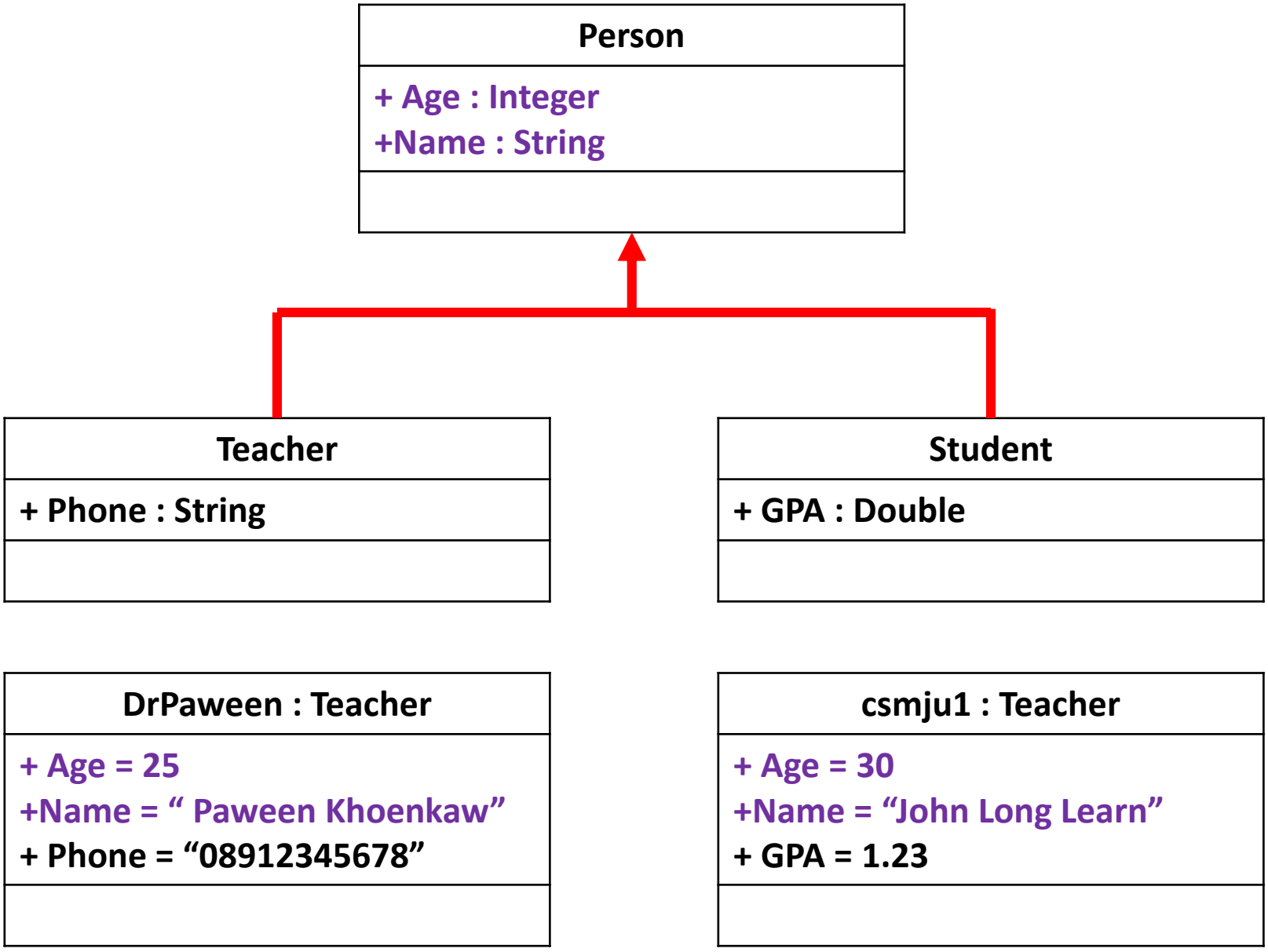


Inheritances

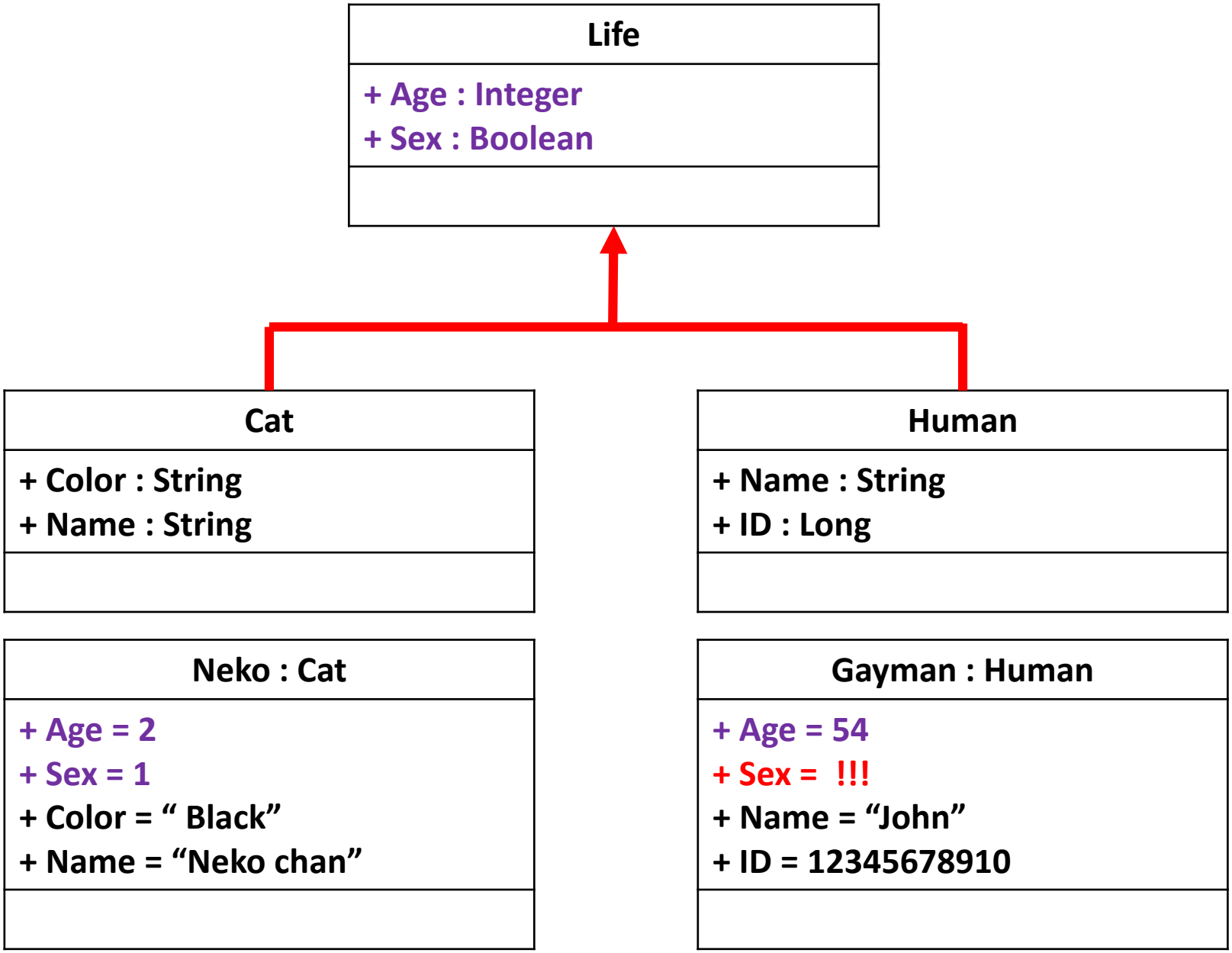


Java เรียก Person ว่าเป็น super class ของ Teacher และ Student

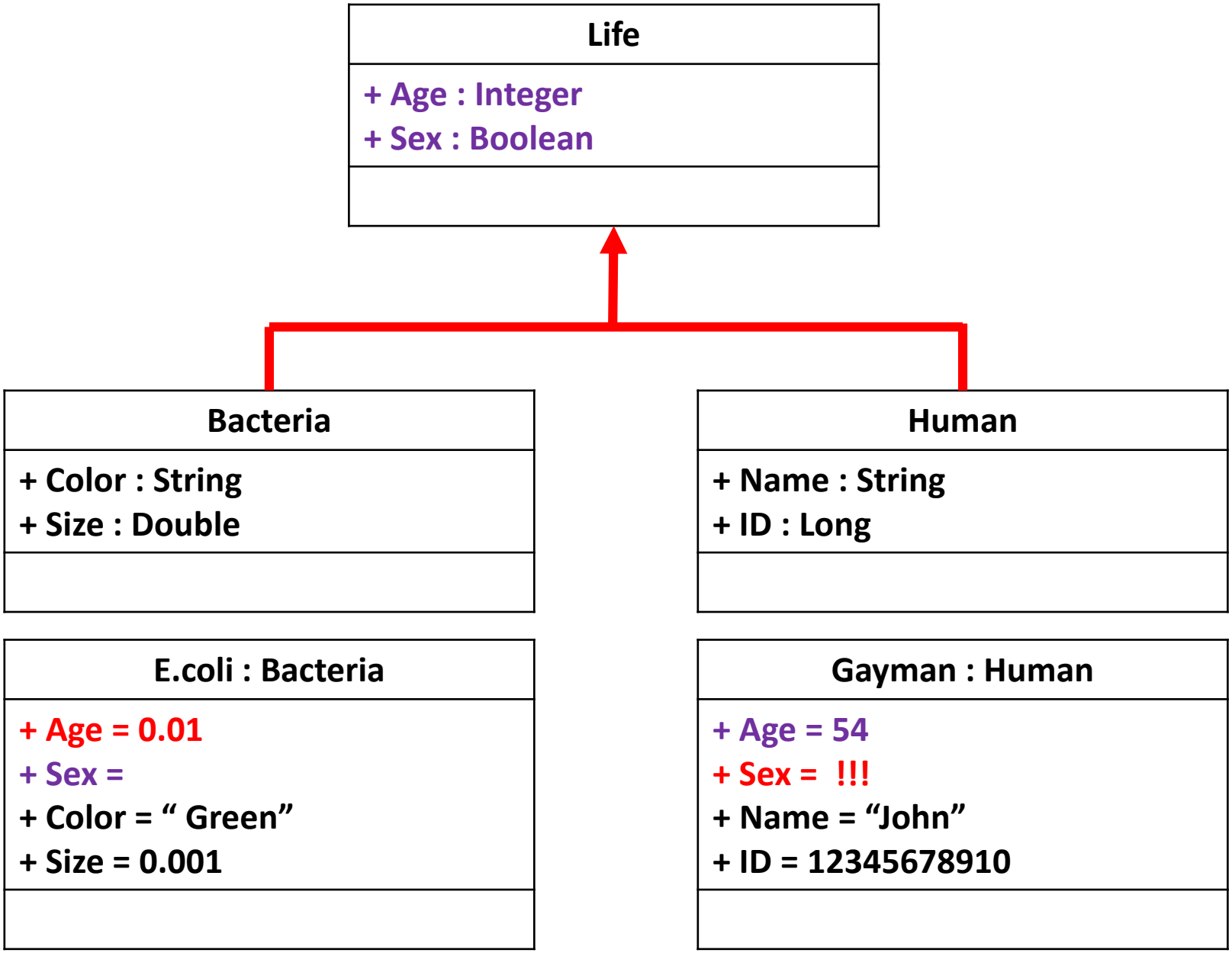
Inheritances



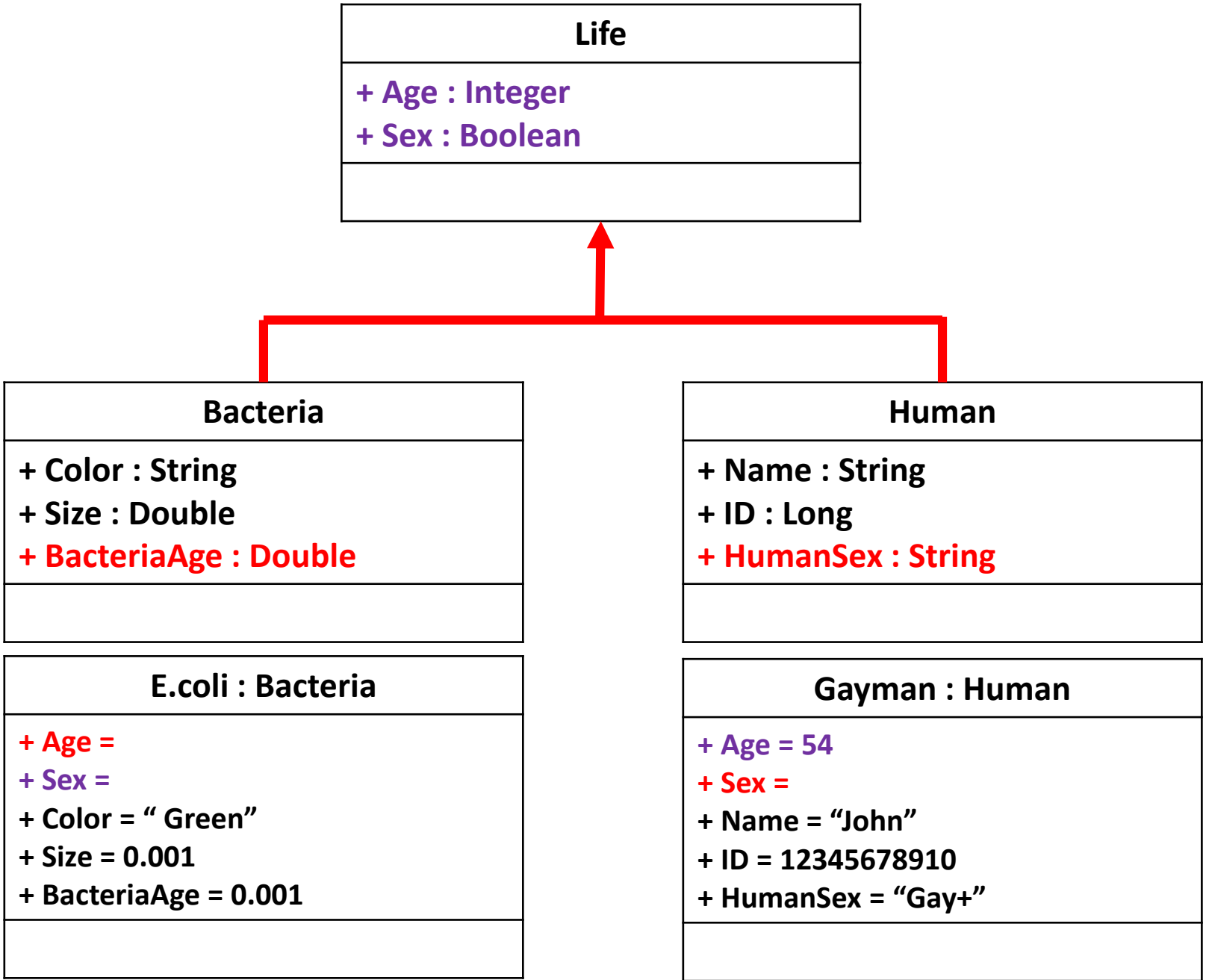
Inheritances



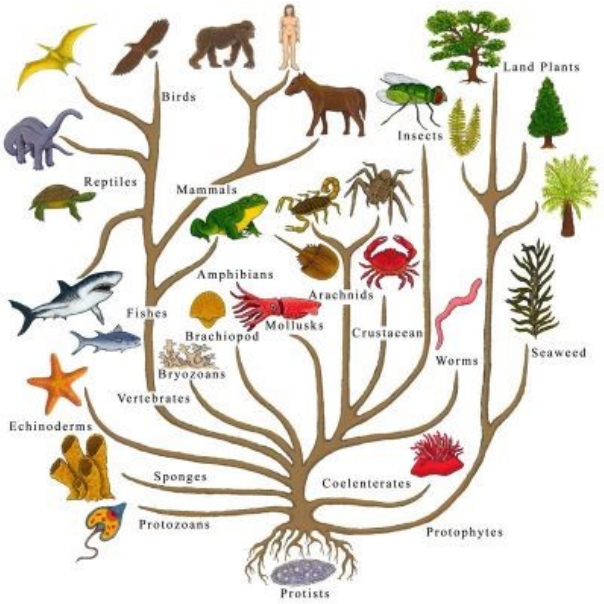
Inheritances



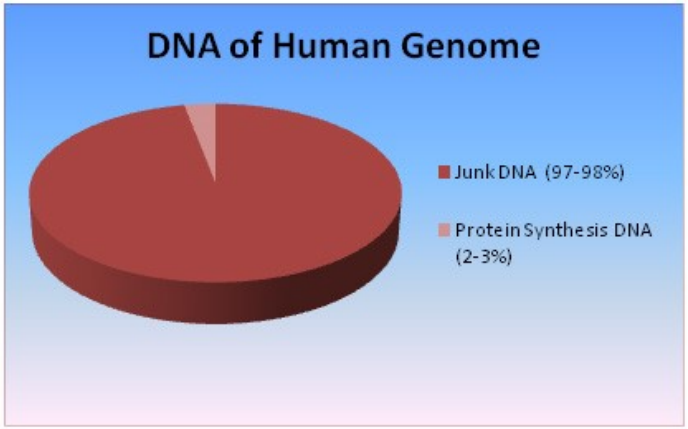
Inheritances



Inheritances



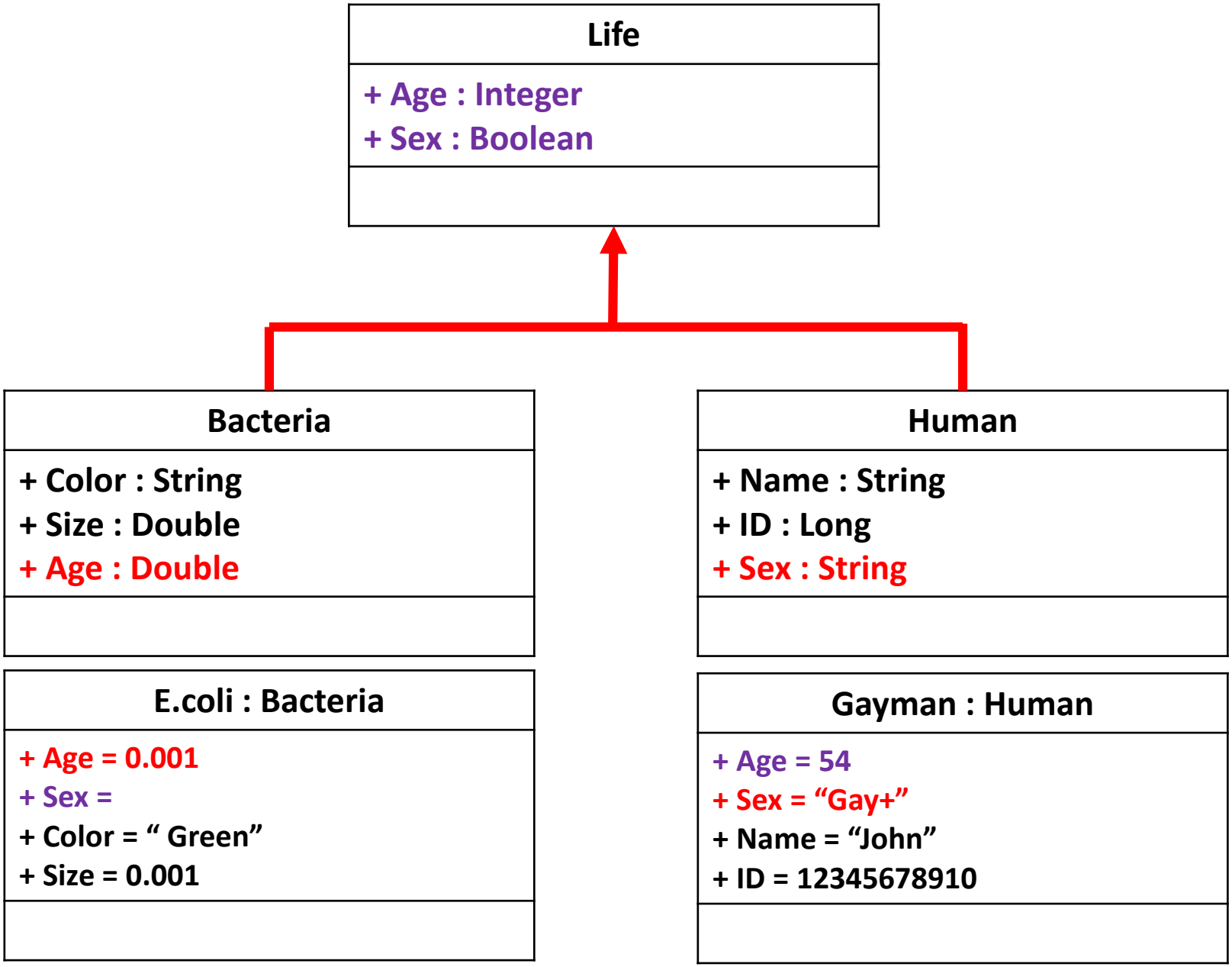
การเพิ่ม Fields ทำให้เกิดขยะ เมื่อมีการสืบทอดคลาสต่อ



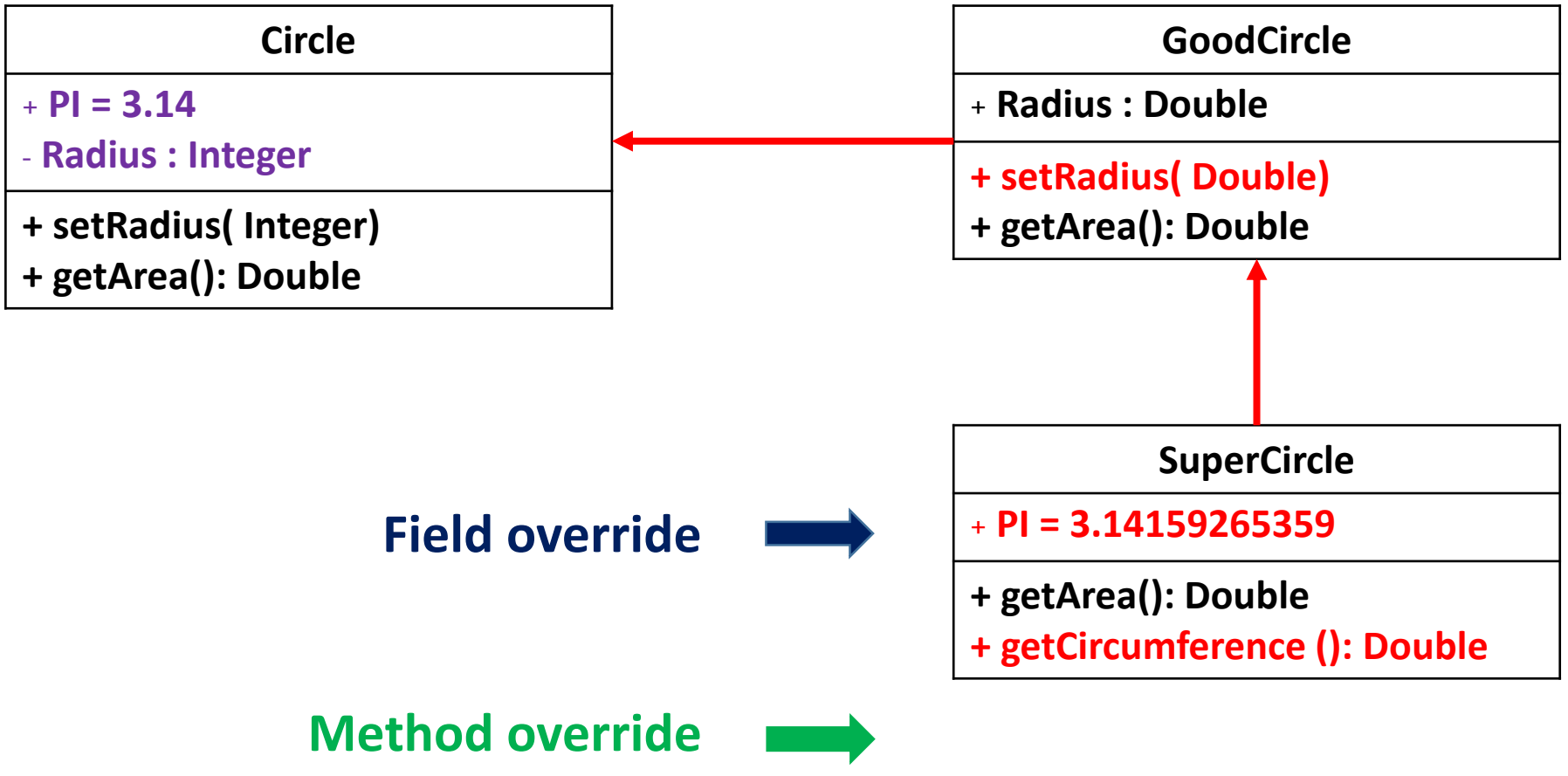
Junk DNA is the non-coding DNA in scientific terms.

Overriding

Inheritances : Override



Inheritances : Override



ควรทำ Method override มากว่า Field override

ด้วยเหตุผลทางด้านความปลอดภัย

Inheritances : Overloading

IDCARD
- Age: Integer - Name : String - ID : Long
+ setAge(Integer) + setName(String) + setID (Long) + getName(): String + getAge(): Integer + getID() : Long

SuperIDCARD
- Age: Integer - Name : String - ID : Long
+ setAge(Integer) + setName(String) + setID (Long) + getName(): String + getAge(): Integer + getID() : Long + setAge(Double) + setAge(String) + setAge(Date) + setAge(Date , Integer)

```
IDCARD myID = new IDCARD();
setAge(18);
setAge(18.0);
setAge("18");
setAge("15-04-1998");
```

Inheritances : Overloading

IDCARD
- Age: Integer - Name : String - ID : Long
+ setAge(Integer) + setName(String) + setID (Long) + getName(): String + getAge(): Integer + getID() : Long

SuperIDCARD
- Age: Integer - Name : String - ID : Long
+ setAge(Integer) + setName(String) + setID (Long) + getName(): String + getAge(): Integer + getID() : Long + setAge(Double) + setAge(String) + setAge(Date) + setAge(Date , Integer)

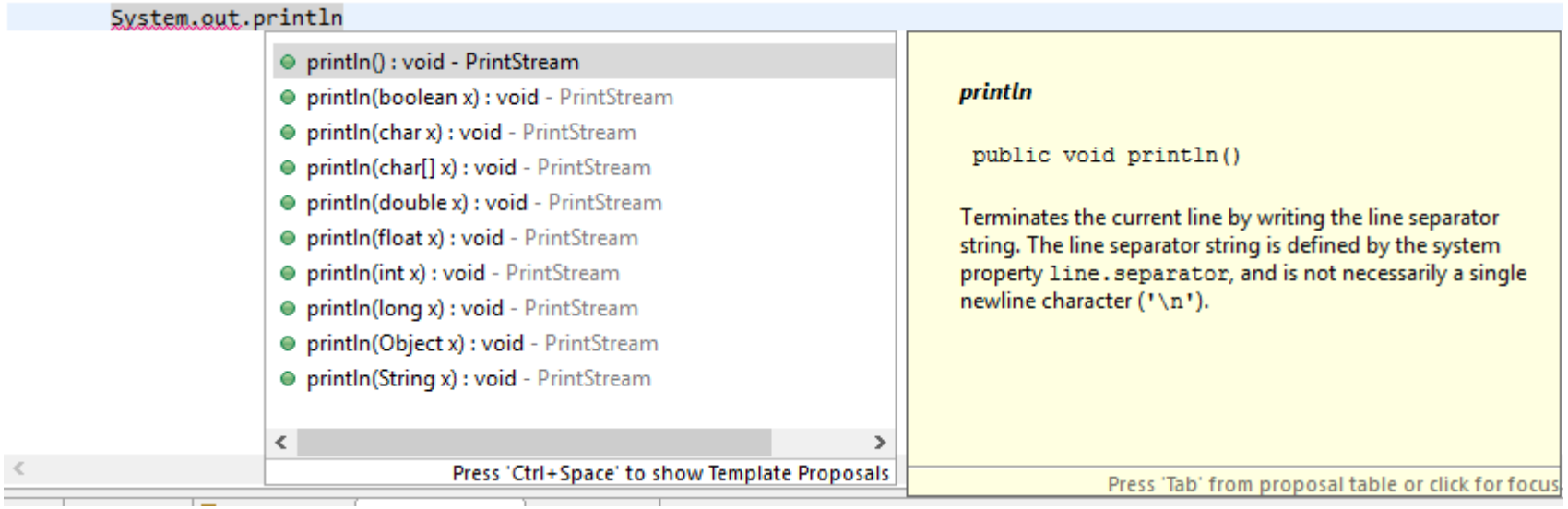
ชื่อ Method เหมือนกัน พารามิเตอร์เหมือนกัน = Override

ชื่อ Method เหมือนกัน พารามิเตอร์ต่างกัน = Overload

ทั้งสองทำงานได้แบบเดียวกัน แต่เรียกชื่อต่างกัน

C ไม่รองรับ Overload แต่ Java รองรับทั้ง Override และ Overload

Inheritances : Overloading



ตัวอย่าง Overloading ใน Method println()

หากไม่ใช้ Overloading จะต้องสร้าง method เพิ่มอีกหลายตัว เช่น .

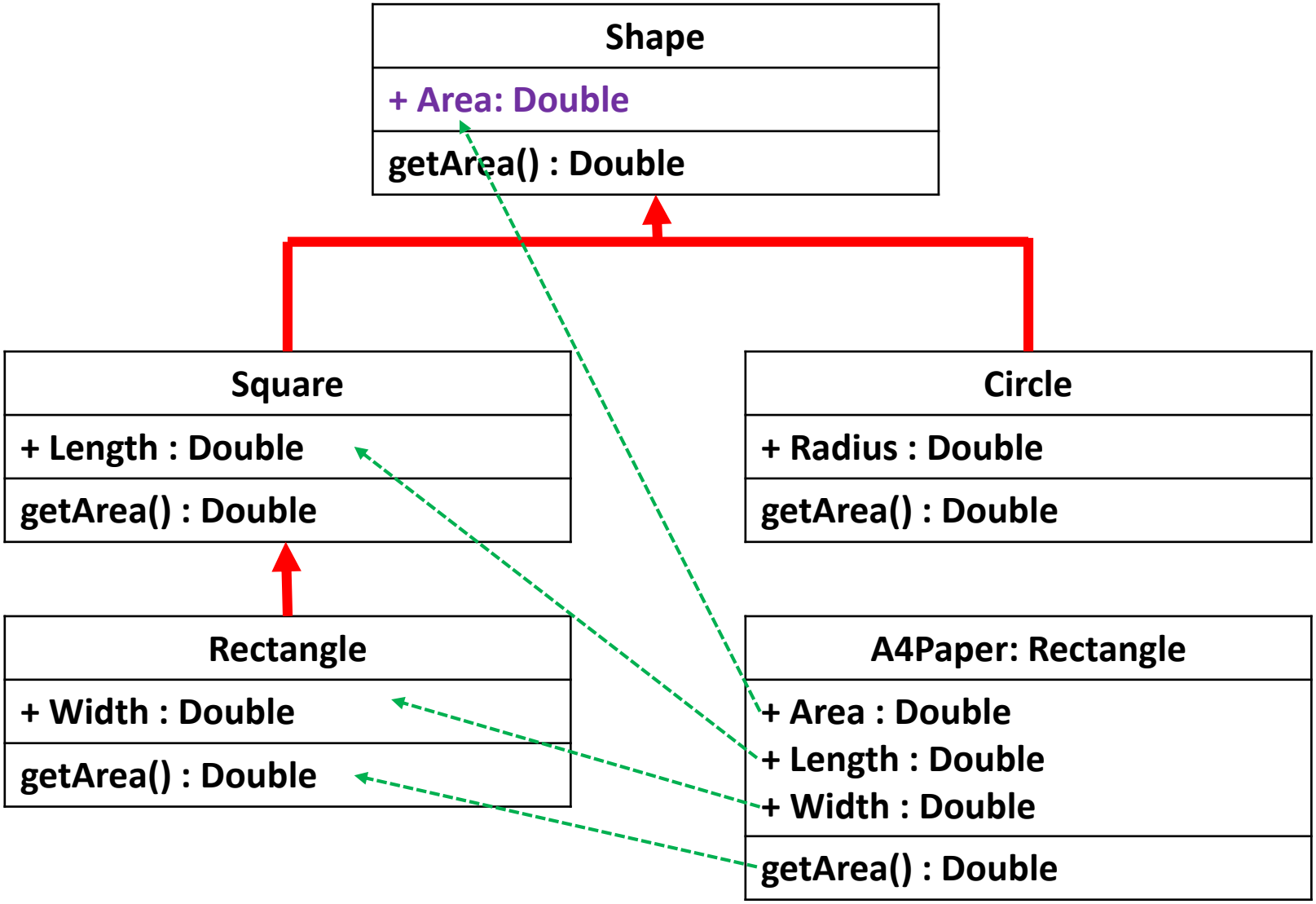
println_Boolean(Boolean x)
println_Char(Char x)
println_Char[](Char[] x)
println_double(double x)
println_float(float x)
.....

Inheritances : Overloading

Identifier	Description	
Format conversions	asctime	converts a struct tm object to a textual representation (deprecated)
	ctime	converts a time_t value to a textual representation
	strftime	converts a struct tm object to custom textual representation
	wcsftime	converts a struct tm object to custom wide string textual representation
	gmtime	converts a time_t value to calendar time expressed as Coordinated Universal Time[2]
	localtime	converts a time_t value to calendar time expressed as local time
	mktime	converts calendar time to a time_t value.

ตัวอย่าง Method แปลงรูปแบบเวลาใน C / C++

Inheritances



Constructor & Destructor

ผู้สร้างและผู้ทำลาย

Constructor & Destructor

Constructor คือ Method แรกที่ทำงานโดยอัตโนมัติ ในขั้นตอนของการ Instantiation มีหน้าที่เตรียมคลาสให้พร้อมใช้งาน เช่น จองหน่วยความจำ เป็นต้น

ภาษา Java กำหนดให้ Method ชื่อเดียวกับชื่อคลาส คือ Constructor

Computer
- **Model: String** - **Ram: Integer** - **Harddrive: Integer** - **CPU: String**
+ Computer(Model:String, Ram:Integer,Hdd:Integer,CPU:Integer) **+ Computer()** **+ setModel(String)** **+ setRam(Integer)** **+ setHarddrive(Integer)** **+ setCPU(String)** **+ turnOn()** **+ turnOff()**

Computer Mac=new Computer(“Macbook”,8000,500, “Core I7”);

Mac : Computer
- **Model=“Macbook”** - **Ram= 8000** - **Harddrive= 500** - **CPU= “Core I7”**
+ Computer(Model:String, Ram:Integer,Hdd:Integer,CPU:Integer) **+ Computer()** **+ setModel(String)** **+ setRam(Integer)** **+ setHarddrive(Integer)** **+ setCPU(String)** **+ turnOn()** **+ turnOff()**

Constructor & Destructor

Constructor คือ Method แรกที่ทำงานโดยอัตโนมัติ ในขั้นตอนของการ Instantiation มีหน้าที่เตรียมคลาสให้พร้อมใช้งาน เช่น จองหน่วยความจำ เป็นต้น

ภาษา Java กำหนดให้ Method ชื่อเดียวกับชื่อคลาส คือ Constructor

Computer
- **Model: String** - **Ram: Integer** - **Harddrive: Integer** - **CPU: String**
+ Computer(Model:String, Ram:Integer,Hdd:Integer,CPU:Integer) **+ Computer()** **+ setModel(String)** **+ setRam(Integer)** **+ setHarddrive(Integer)** **+ setCPU(String)** **+ turnOn()** **+ turnOff()**

Computer Mac=new Computer();

Mac : Computer
- **Model=** - **Ram=** - **Harddrive=** - **CPU=**
+ Computer(Model:String, Ram:Integer,Hdd:Integer,CPU:Integer) **+ Computer()** **+ setModel(String)** **+ setRam(Integer)** **+ setHarddrive(Integer)** **+ setCPU(String)** **+ turnOn()** **+ turnOff()**

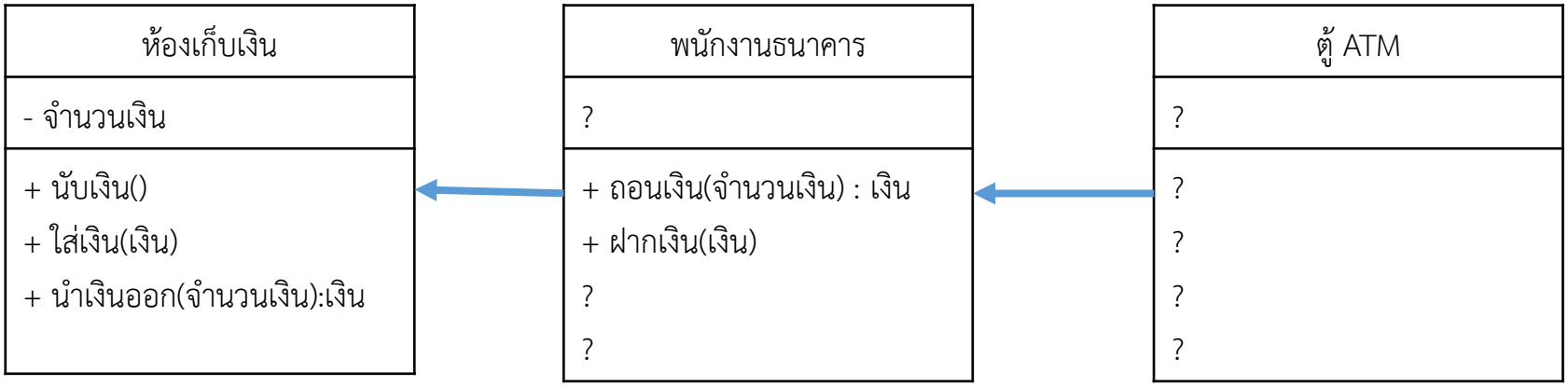
Constructor & Destructor

Destructor คือ Method สุดท้ายที่ถูกเรียกใช้งานเมื่อ class นั้นสิ้นอายุขัย หรือเมื่อไม่ต้องการใช้งานแล้ว มีหน้าที่ทำงายตัวเอง เช่น คืนหน่วยความจำ ปิดการใช้งานแฟ้มข้อมูลจองหน่วยความจำ Method ที่ขึ้นต้นด้วย ~ และตามด้วยชื่อคลาส คือ Destructor

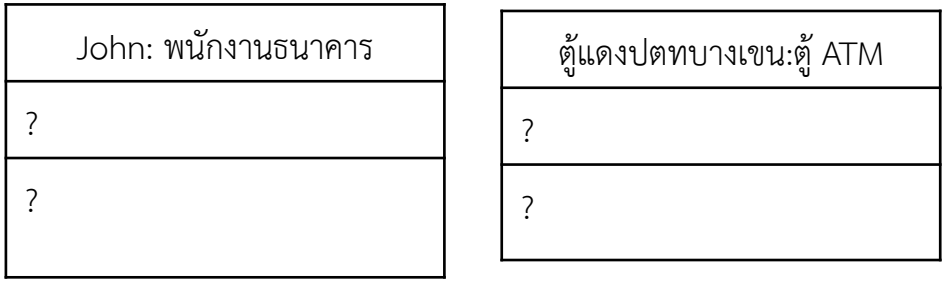
Computer
- **Model: String** - **Ram: Integer** - **Harddrive: Integer** - **CPU: String**
+ Computer(Model:String, Ram:Integer,Hdd:Integer,CPU:Integer) **+ Computer()** **+ ~Computer()** **+ setModel(String)** **+ setRam(Integer)** **+ setHarddrive(Integer)** **+ setCPU(String)** **+ turnOn()** **+ turnOff()**

ภาษา Java ไม่นิยาม Destructor เนื่องจาก Instance ที่ไม่จำเป็น จะถูกกำจัดออก โดยอัตโนมัติ ขณะที่โปรแกรมทำงาน ด้วยระบบ Garbage collector

การบ้าน 3



จงเติม class พนักงานธนาคาร และ class ATM ให้สมบูรณ์ จำนวน properties และ method ให้ใส่ไปตามความเหมาะสม พร้อมทั้งสร้าง Instance ของทั้งคลาส พนักงานธนาคาร และ ตู้ ATM ให้ดูอย่างละ 1 instance ด้วย ตัวอย่างเช่น



เขียนคำตอบใส่<u>กระดาษ A4 จำนวน 1 แผ่น</u> ส่งทางกล่องรับการบ้าน หน้าแลป 2

เขียนชื่อ รหัส กลุ่มเรียน ที่<u>มุมบนขวา</u>

ภายในวันพฤหัสที่ 27 กรกฎาคม 2560