复合与继承
双向关联

主讲老师：申雪萍

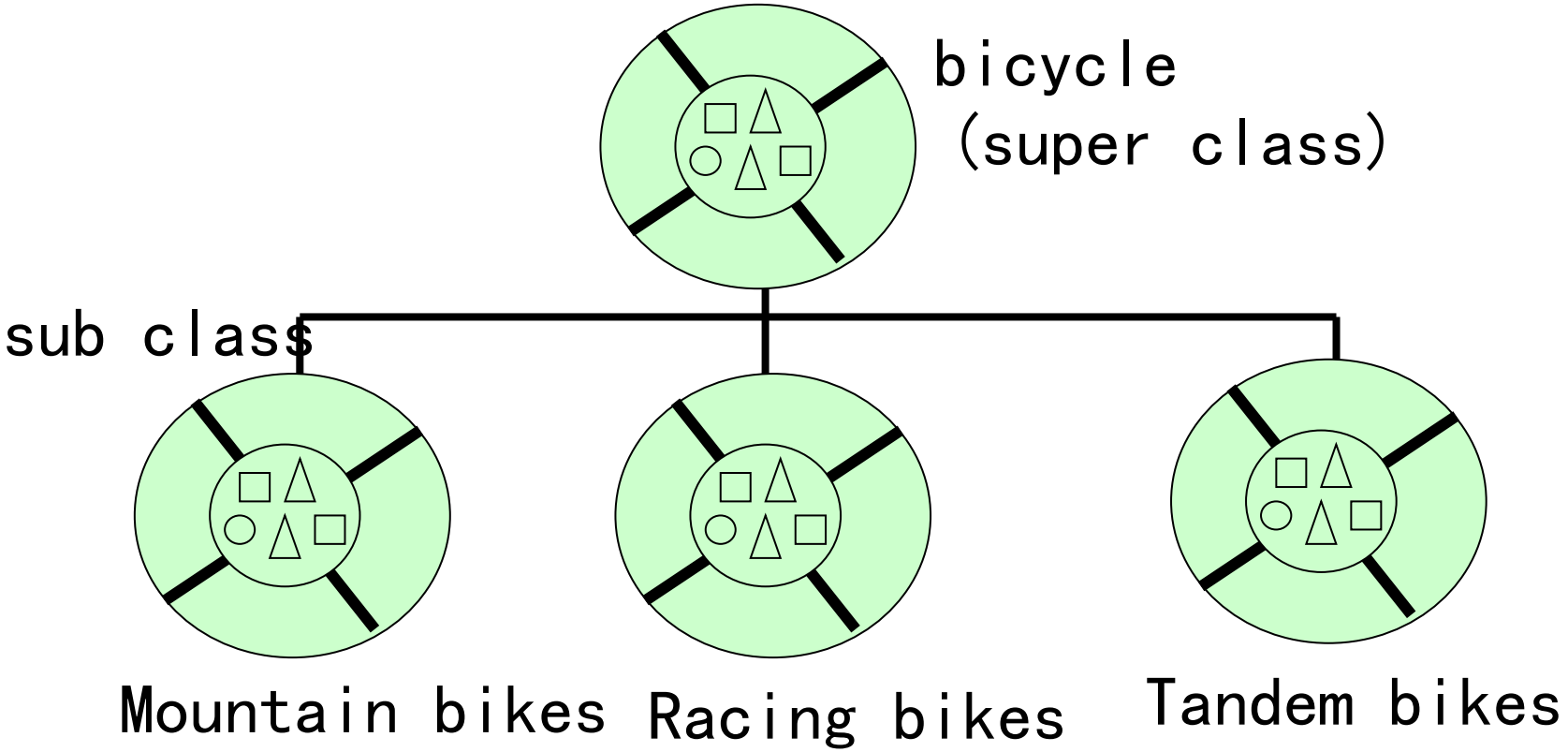
主要内容

- 复合与继承
- 双向关联

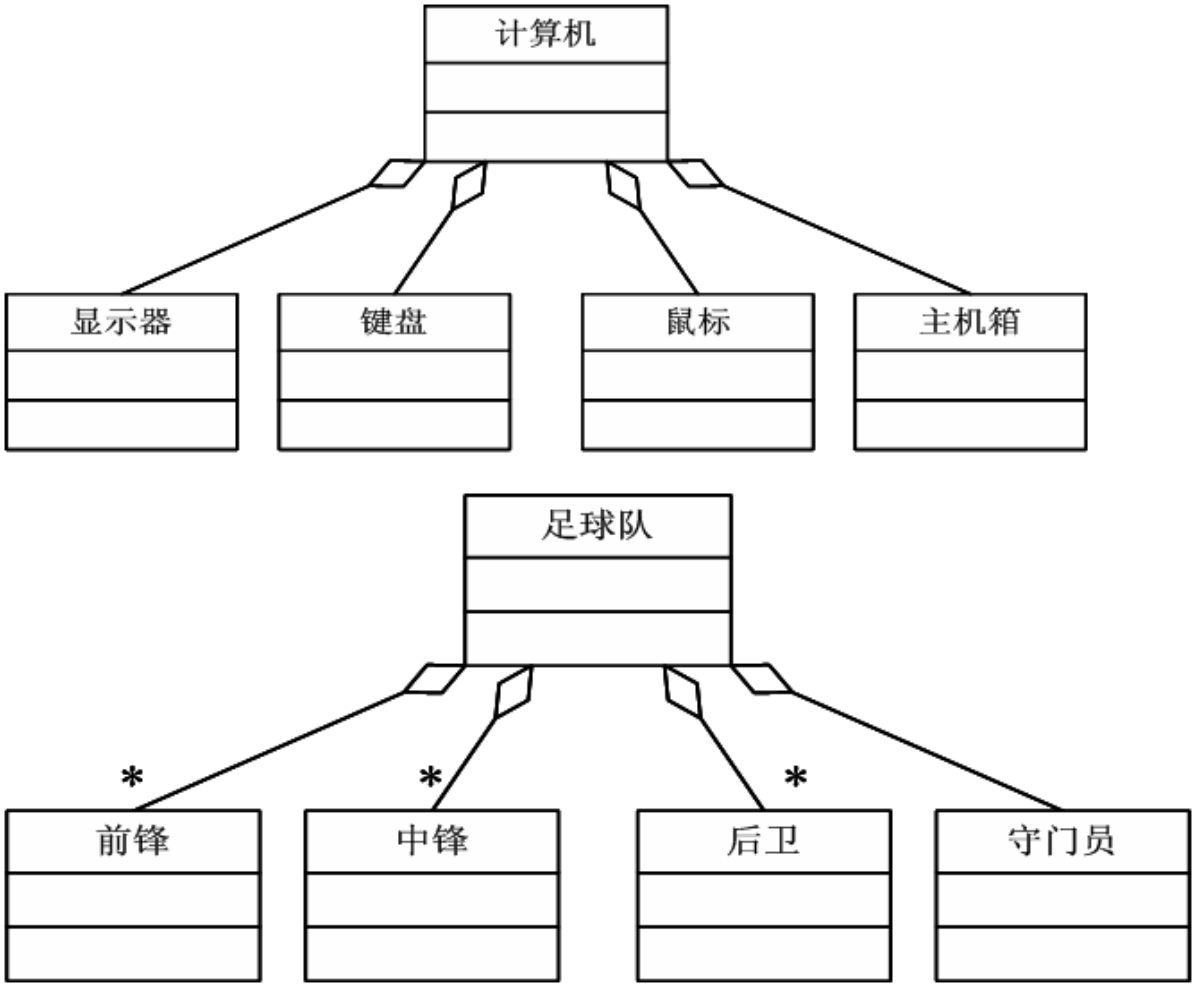
复用其它对象的代码有两种方法：

- ***继承：继承是由子对象继承父对象的数据和操作。“is”的关系。通过继承可以支持多态***
- 包容/复合/包含：包容是指把一个对象作为另一个对象的一部份，从而达到复用的目的。 ***“has”***的关系。
- 通过继承和包容，类之间相互关联，组成树状结构

is a关系

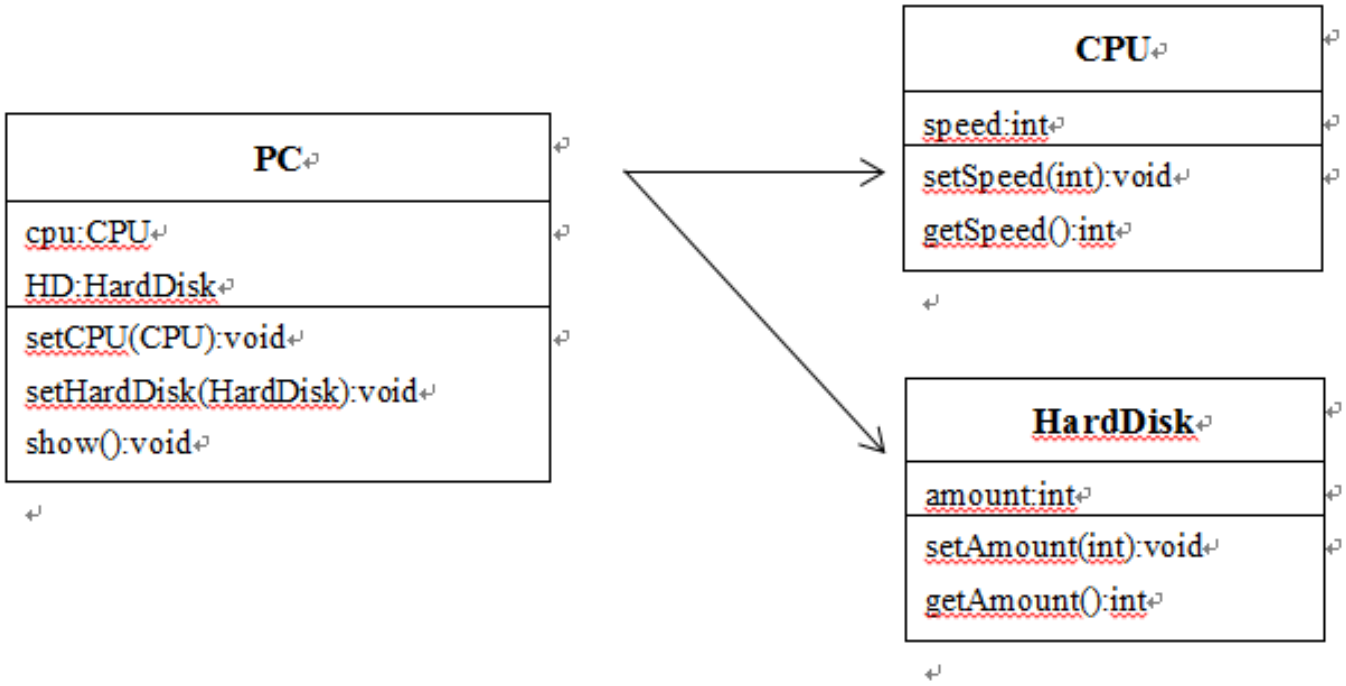

bicycle
(super class)
sub class
Mountain bikes
Racing bikes
Tandem bikes

has a关系



案例分析

- PC类与CPU和HardDisk类关联的UML图如下，请设计各相关类，描述计算机中CPU的速度和硬盘的容量。
-



PC 与 CPU 和 HardDisk 关联 UML 图

示例代码

```
package com.buaa.test;
public class CPU {
    private int speed;
    public int getSpeed() {
        return speed;
    }
    public void setSpeed(int speed) {
        this.speed = speed;
    }
}
```

```java
package com.buaa.test;

public class HardDisk {
    private int amount;

    public int getAmount() {
        return amount;
    }
    public void setAmount(int amount) {
        this.amount = amount;
    }
}
```

```java
package com.buaa.test;
public class PC {
    private CPU cpu;
    private HardDisk HD;
    public CPU getCpu() {
        return cpu;
    }
    public void setCpu(CPU cpu) {
        this.cpu = cpu;
    }
    public HardDisk getHD() {
        return HD;
    }
    public void setHD(HardDisk hD) {
        HD = hD;
    }
    public void show() {
        System.out.println("CPU速度:" + cpu.getSpeed());
        System.out.println("硬盘容量:" + HD.getAmount());
    }
}
```

```
package com.buaa.test;

public class Test {
    public static void main(String args[]) {
        CPU cpu = new CPU();
        HardDisk HD = new HardDisk();
        cpu.setSpeed(2200);
        HD.setAmount(200);
        PC pc = new PC();
        pc.setCpu(cpu);
        pc.setHD(HD);
        pc.show();
    }
}
```

主要内容

- 复合与继承
- 双向关联

案例分析（3）：学生选课（has 关系）

- Encapsulate工程：
  - **package com.buaa.hasEx;**
- 本案例的目的：
  - **类之间双向关联的理解（has a 关系）**
    - **关联属性的设置：**两个类中的成员相互关联的解决方法
    - **特殊方法的编写：**
      1. **关联方法的编写**
      2. **数据合法性验证**

案例分析：学生选课

- 学生类：Student
  - 重点
  - Introduction()：自我介绍的信息，包括姓名，学号，性别，年龄，**所学专业名称**，学制年限
  - setStudent_age()：保证数据的合法性
  - setStudent_sex：保证数据的合法性
- 学科类：Subject
  - 重点
  - setSubjectLife()：设置学科年限，保证数据的合法性
  - info()：返回学科相关信息

第一种处理方案：com.buaa.hasEx

```java
public class Student {
    // 成员属性：学号，姓名，性别，年龄
    private String student_id;
    private String student_name;
    private String student_sex;
    private int student_age;
```

第一种处理方案：com.buaa.hasEx

```
public class Subject {
    // 成员属性：学科名称，学科编号，学制年限
    private String subjectName;
    private String subjectNo;
    private int subjectLife;
```

第一种处理方案：com.buaa.hasEx

```java
/**
 * 解决方法1:参数传参 学生自我方法介绍，实现学生学科相关联
 *
 * @param subject_name
 * @param subject_life
 * @return 自我介绍的信息，包括姓名，学号，性别，年龄，所学专业名称，学制年限
 */
public String introduction(String subject_name, int subject_life) {
    String str = "学生信息如下：\n姓名:" + this.getStudent_name() + "\n学号："
            + this.getStudent_id() + "\n性别：" + this.getStudent_sex()
            + "\n年龄：" + this.getStudent_age() + "\n所报专业名称：" + subject_name
            + "\n学制年限：" + subject_life;
    return str;
}
```

第一种处理方案：com.buaa.hasEx

```java
/**
 * 解决方法2:对象传参 学生自我介绍，实现学生学科相关联
 *
 * @param mySubject
 *                (构建了一个对象)
 * @return自我介绍的信息，包括姓名，学号，性别，年龄，所学专业名称，学制年限
 */
public String introduction(Subject mySubject) {
    String str = "学生信息如下：\n姓名:" + this.getStudent_name() + "\n学号："
            + this.getStudent_id() + "\n性别：" + this.getStudent_sex()
            + "\n年龄：" + this.getStudent_age() + "\n所报专业名称："
            + mySubject.getSubjectName() + "\n学制年限："
            + mySubject.getSubjectLife();
    return str;
}
```

第二种处理方案：com.buaa.hasEx2 单项关联

```java
public class Student {
    // 成员属性：学号，姓名，性别，年龄
    private String student_id;
    private String student_name;
    private String student_sex;
    private int student_age;

    //实现关联学生，学科关联的，解决方法3:将专业信息作为成员属性存在(解决方法1，2在后面)
    private Subject student_subject;//注意，类型是一个对象
```

```java
    /**
     * 获取专业对象，如果没有实例化，先实例化再返回
     * @return 专业对象信息
     */
    public Subject getStudent_subject(){
        if(this.student_subject==null)//防止getStudent_subject()是没有实例化的
        {
            this.student_subject=new Subject();//无参构造的重要性体现出来
        }
        return student_subject;
    }
```

第二种处理方案：com.buaa.hasEx2

```
public String introduction(Subject mySubject) {
        mySubject=this.getStudent_subject();
        String str = "学生信息如下：\n姓名:" + this.getStudent_name() + "\n学号："
                + this.getStudent_id() + "\n性别：" + this.getStudent_sex()
                + "\n年龄：" + this.getStudent_age() + "\n所报专业名称："
                + mySubject.getSubjectName() + "\n学制年限："
                + mySubject.getSubjectLife();
        return str;
}

}
```

第三种处理方案：com.buaa.hasEx3（互相拥有、双向关联）

```java
public class Student {
    // 成员属性：学号，姓名，性别，年龄
    private String student_id;
    private String student_name;
    private String student_sex;
    private int student_age;

    //实现关联学生，学科关联的，解决方法3:将专业信息作为成员属性存在(解决方法1，2在后面)
    private Subject student_subject;//注意，类型是一个对象
```

```java
public class Subject {
    // 成员属性：学科名称，学科编号，学制年限,报名选修的学生信息、报名选修的学生个数
    private String subjectName;
    private String subjectNo;
    private int subjectLife;

    private Student[] myStudents;
    private int studentNum;
```

第三种处理方案：com.buaa.hasEx3（互相拥有）

Subject类

```
/**
 * 获取选修专业的学生信息的，如果保存的学生信息未被初始化，则先初始化长度200
 * @return 保存学生信息的数组
 */
public Student[] getMyStudents() {
    if(this.myStudents==null)//如果没有初始化，实例化开辟空间
    {
        this.myStudents=new Student[200];
    }
    return myStudents;
}

public void setMyStudents(Student[] myStudents) {
    this.myStudents = myStudents;
}
```

第三种处理方案：com.buaa.hasEx3（互相拥有）

```
public void addStudent(Student stu)
{
    /*
     * 1,将学生保存在数组中
     * 2，将学生个数保存到studentNum当中
     */
    for(int i=0;i<this.getMyStudents().length;i++)
    {
        if(this.getMyStudents()[i]==null)//找到空地，插入学生
        {
            this.getMyStudents()[i]=stu;
            this.studentNum=i+1;
            return;
        }
    }
}
```

第三种处理方案：com.buaa.hasEx3（互相拥有）

```
//测试指定专业的中到底有多少学生报名
sub1.addStudent(stu1);
sub1.addStudent(stu2);
sub1.addStudent(stu3);
sub1.addStudent(stu4);
System.out.println("软件工程的专业中已有："+sub1.getStudentNum()+"学生进行报名");
```

第四种处理方案：com.buaa.hasEx4（互相拥有、双向关联）

```
public class Subject {
    // 成员属性：学科名称，学科编号，学制年限,报名选修的学生信息、报名选修的学生个数
    private String subjectName;
    private String subjectNo;
    private int subjectLife;

    private Student[] myStudents;
    private int studentNum;
```

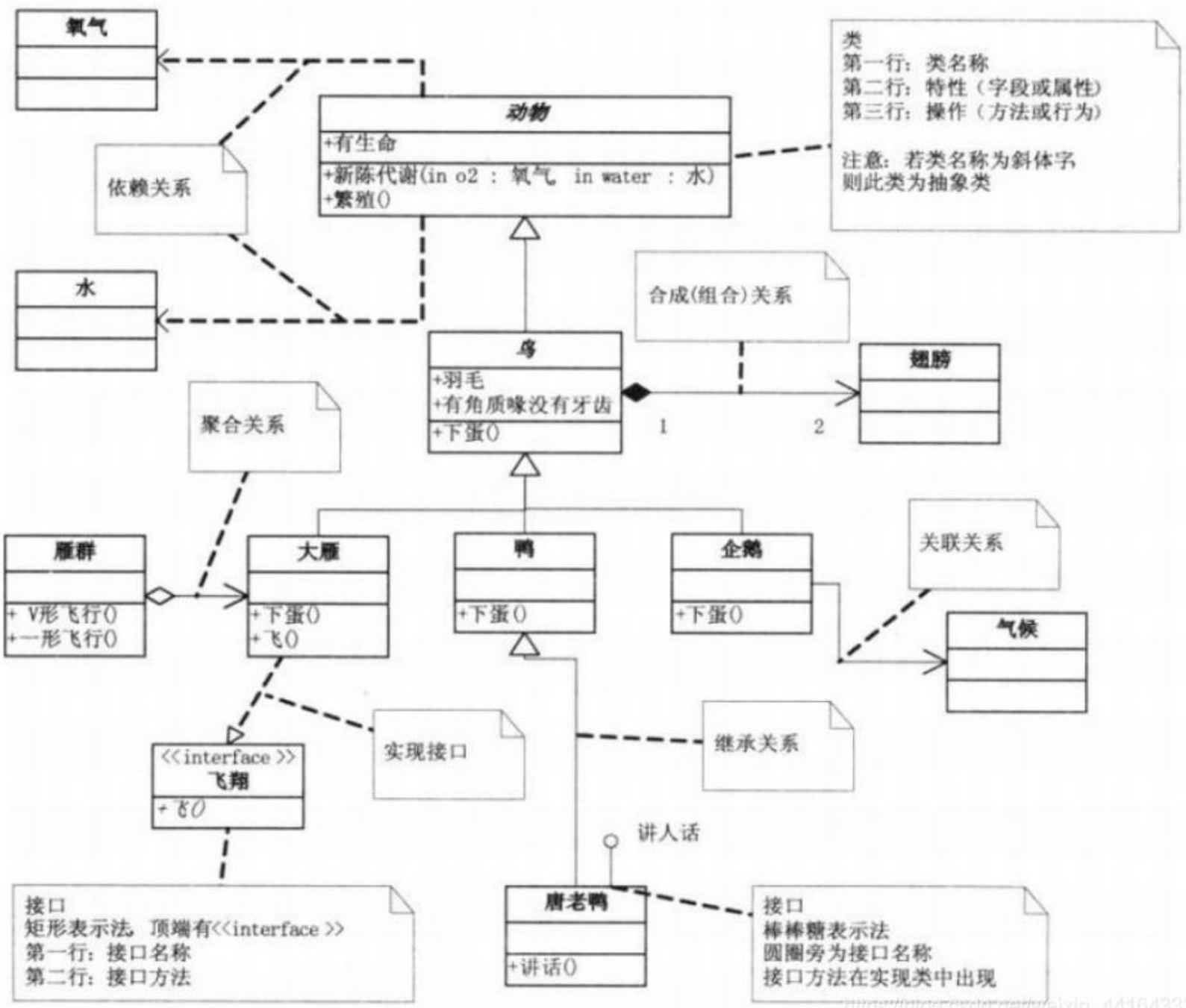
第四种处理方案：com.buaa.hasEx4（互相拥有、双向关联）

```
public void addStudent(Student stu) {
    /*
     * 1,将学生保存在数组中 2，将学生个数保存到studentNum当中
     */

    for (int i = 0; i < this.getMyStudents().length; i++) {
        if (this.getMyStudents()[i] == null)// 找到空地，插入学生
        {// 可以通过这里直接双关联，相当于将sub1当前对象传给了student类
            stu.setStudent_subject(this);
            this.getMyStudents()[i] = stu;
            // 2.
            this.studentNum = i + 1;
            return;
        }
    }

}
```

第四种处理方案：com.buaa.hasEx4（互相拥有）

```
public static void main(String[] args) {
    //测试subject类
    Subject sub1=new Subject("计算机科学与应用","J0001",4);
    //测试学生
    Student stu1=new Student("cs6518822","Mary","女",20);
    //测试指定专业的中到底有多少学生报名
    sub1.addStudent(stu1);
    System.out.println("计算机科学与应用的专业中已有："+sub1.getStudentNum()+"学生进行报名");
}
```

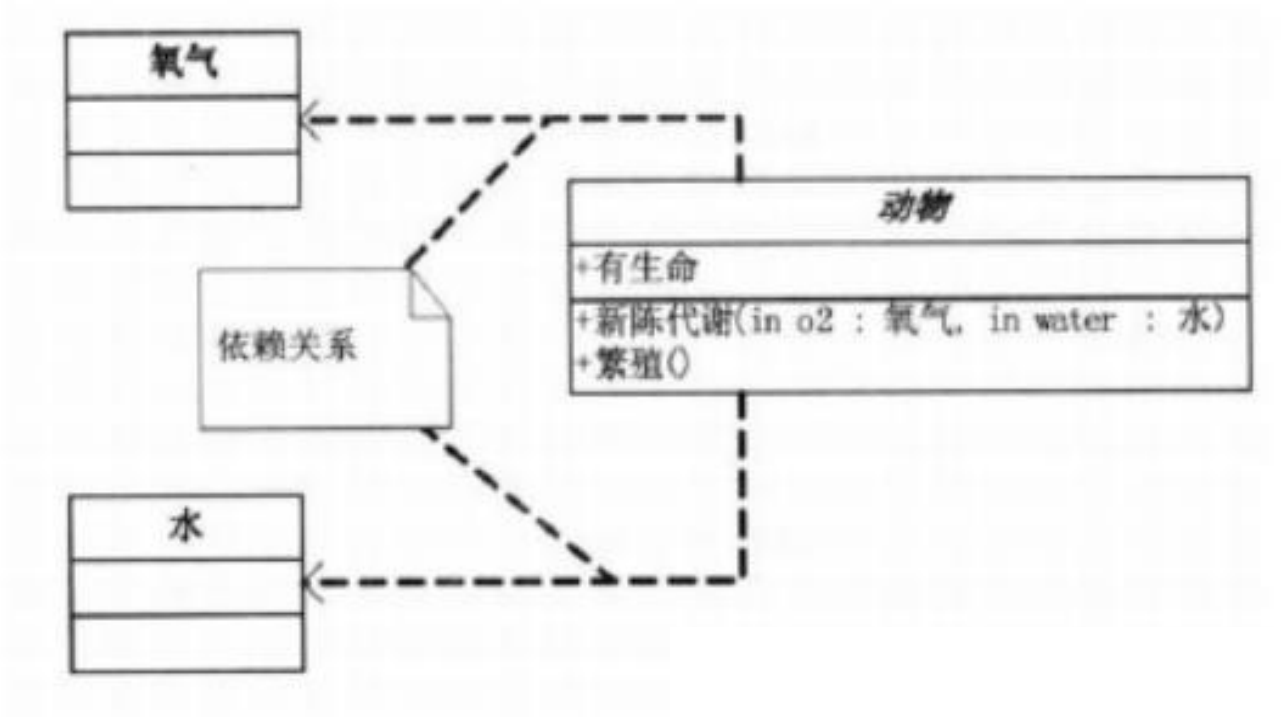
UML类图图示



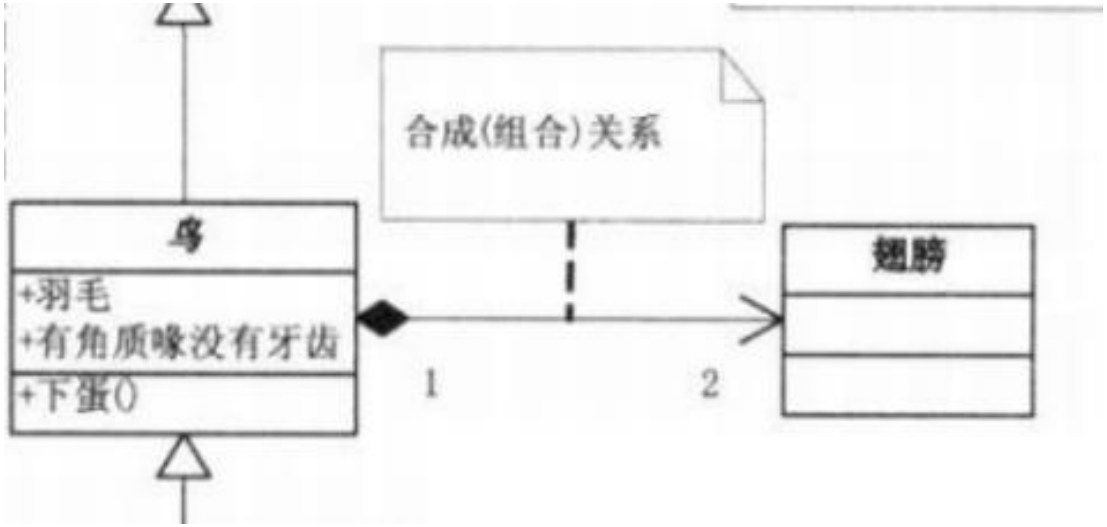
wild goose
penguin
Don Donald
reproduce
metabolise
layeggs
climate

```java
package com.buaa.test;
interface Fly{
    public abstract void fly();
}
```

```java
package com.buaa.test;
public interface Speak {
    public abstract void speak();
}
```

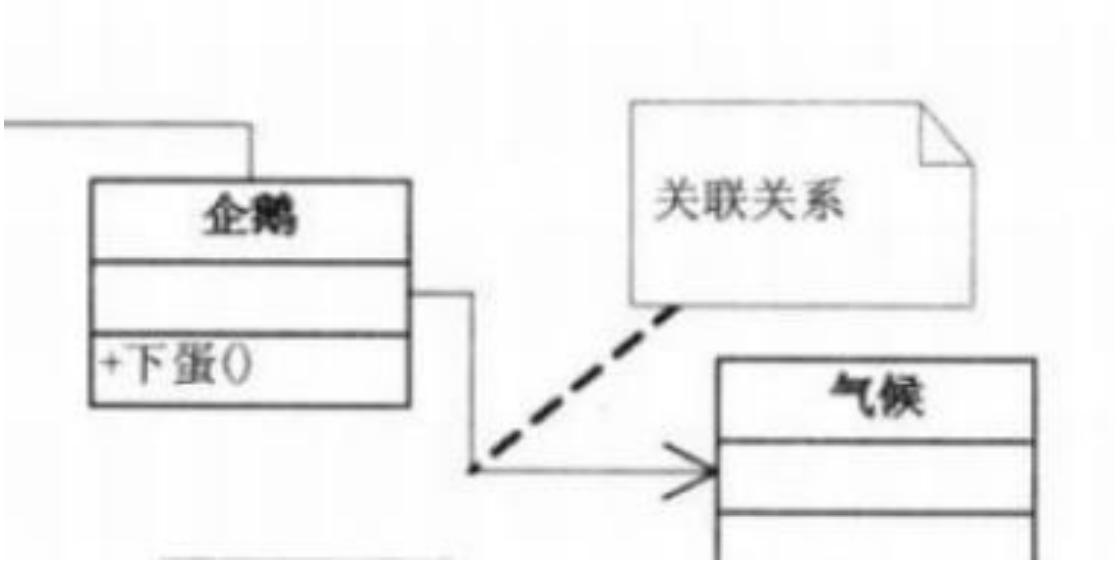

氧气
动物
+有生命
+新陈代谢(in o2 : 氧气, in water : 水)
+繁殖()
依赖关系
水

```java
package com.buaa.test;
public class Animal {
    public char living; // 是否有生命
    // 新陈代谢
    public void metabolism(Oxygen oxygen, Water water) {
        System.out.print("进行新陈代谢 ");
        oxygen.needOxygen();
        water.needWater();
    }
    public void reproduce() {
        System.out.print("我是可以繁殖的! "); // 繁殖
    }
}
class Oxygen{ //依赖关系空气
    public void needOxygen() {
        System.out.print("被依赖类氧气 ");
    }
}
class Water{ //依赖关系水
    public void needWater() {
        System.out.print("被依赖类水 ");
    }
}
```

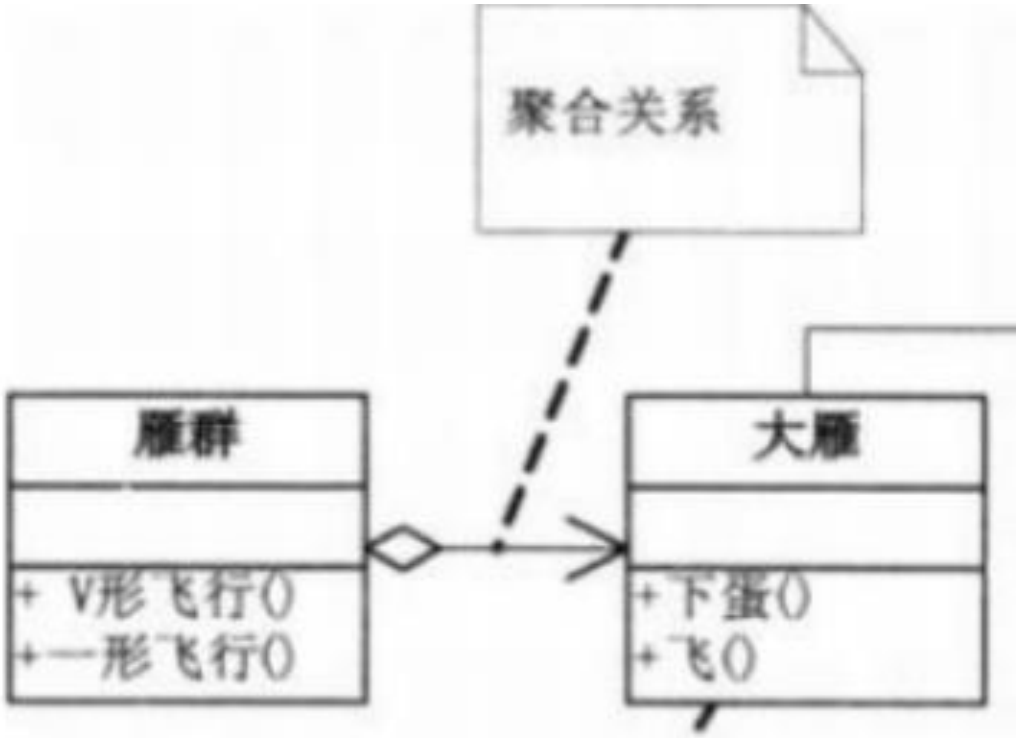

合成(组合)关系
鸟
+羽毛
+有角质喙没有牙齿
+下蛋()
翅膀
1
2

```java
package com.buaa.test;

public class Wing {
    public void wing() {
        System.out.print("合成关系“有翅膀”");
    }
}
```

```java
package com.buaa.test;

public class Bird extends Animal {
    public int feather; // 是否有羽毛
    public int mouse; // 嘴巴是否是有角质喙而没有牙齿
    private Wing wing;
    public Bird() {
        wing = new Wing();
        wing.wing();
    }
    public void layEggs() { // 下蛋
        System.out.print("不同种类的鸟下蛋种类不同！");
    }
}
```


企鹅
+下蛋()
关联关系
气候

```java
package com.buaa.test;
public class Penguin extends Bird{
    private Climate climate;                    //关联关系

    public Penguin(Climate climate) {
        this.climate=climate;
    }
    public void layEggs() {
        System.out.print("我是企鹅所以我生企鹅蛋  ");
        climate.changeClimate();
    }
}
class Climate {
    public void changeClimate() {
        System.out.print("关联关系“天气”");
    }
}
```


聚合关系
雁群
+ V形飞行()
+一形飞行()
大雁
+下蛋()
+飞()

```java
package com.buaa.test;
public class WideGooseAggregate {
    private WideGoose[] arrayWideGoose;
    public void horizontalFly() {
        System.out.println("雁群一字型飞行队列");
    }
    public void vFiy() {
        System.out.println("雁群一字型飞行队列");
    }
}
class WideGoose extends Bird implements Fly{
    public void layEggs() {
        System.out.print("我是大雁所以我生大雁蛋");
    }
    public void fly() {
        System.out.print("我能够飞翔");
    }
}
```

```
class Duck extends Bird{
    public void layEggs()
    {
        System.out.print("我是鸭子所以我生鸭蛋");
    }
}
public class DonaldDuck extends Duck implements Speak{
    public void speak() {
        System.out.print("大家好，我是唐老鸭，我和其他鸭子不一样的是我继承了讲人话接口，会说话");
    }
    public static void main(String args[]) {
        Penguin penguin=new Penguin(new Climate());
        System.out.print("  ");
        penguin.layEggs();
        System.out.print("  ");
        penguin.metabolism(new Oxygen(), new Water());
        System.out.print("  ");
        penguin.reproduce();
        System.out.println();
        DonaldDuck donaldduck=new DonaldDuck();
```