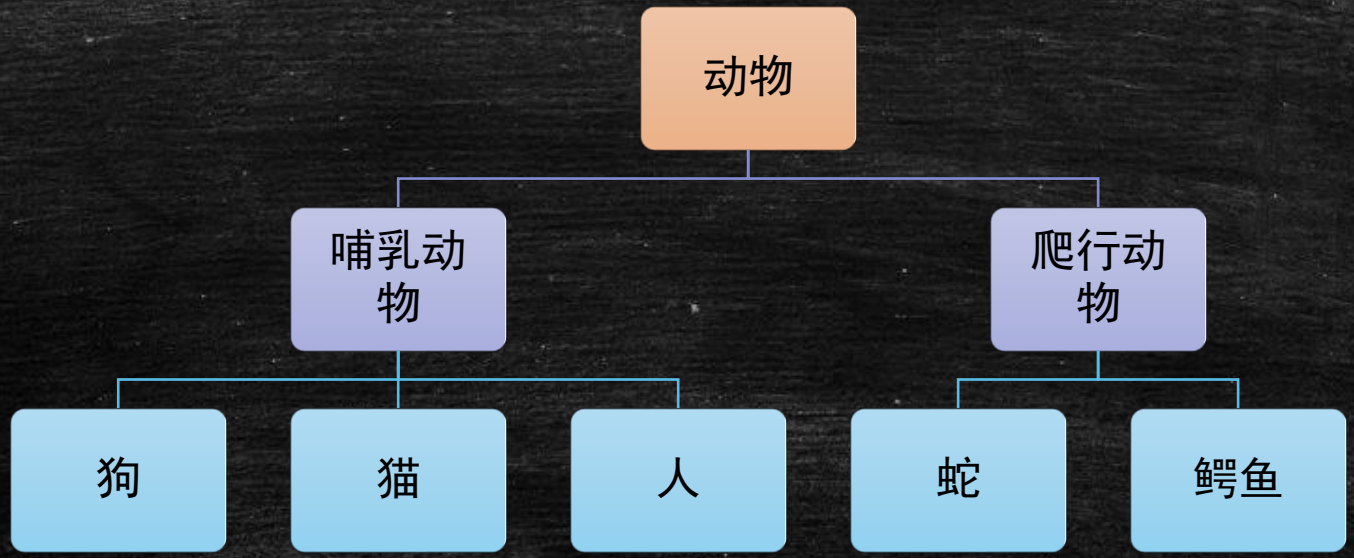
Java面向对象编程3

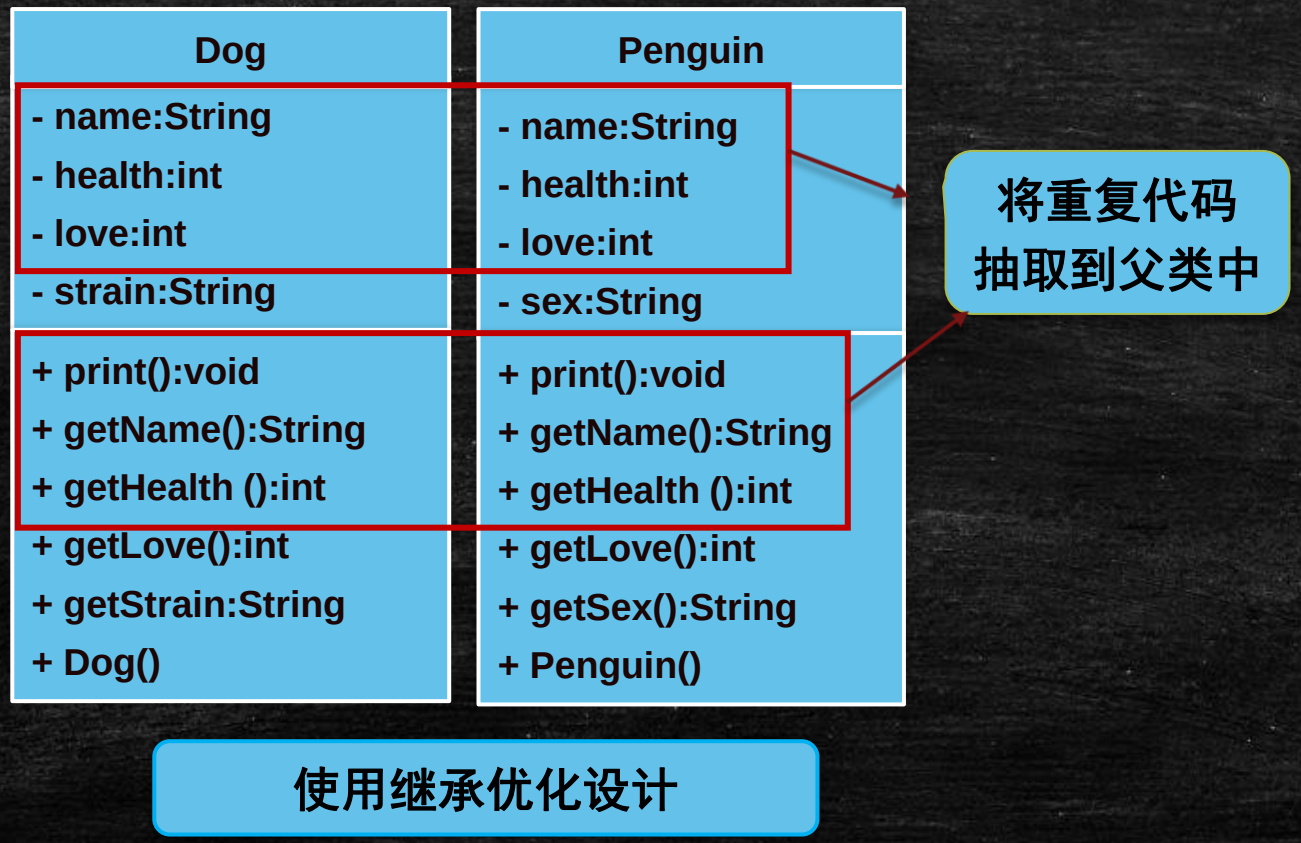
- What?Why?How?

继承



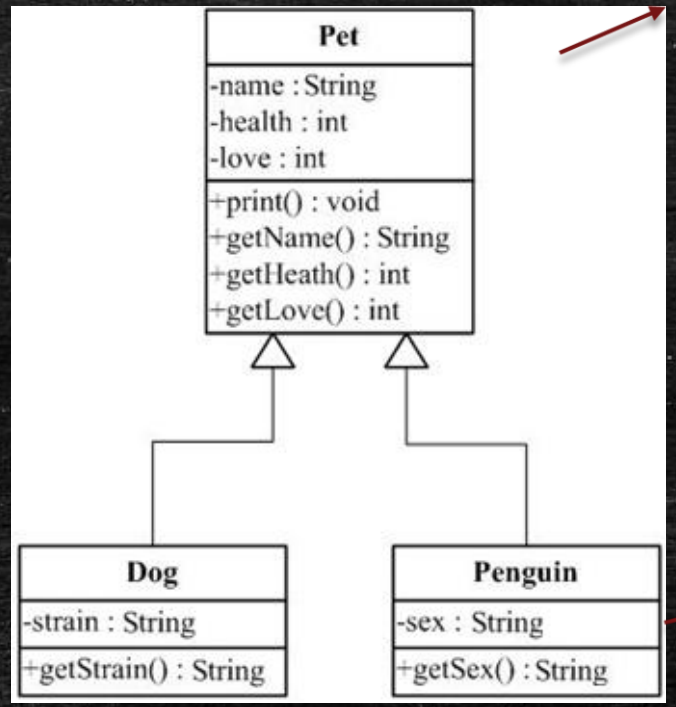
为什么使用继承

- 这两个类图有什么问题?



为什么使用继承

- 使用继承优化后：



方便修改代码

减少代码量

子类与父类是is-a关系

继承

- 类是对对象的抽象，继承是对某一批类的抽象，从而实现对现实世界更好的建模。
- 提高代码的复用性！
- extands的意思是“扩展”。子类是父类的扩展
- 不同的叫法：超类、父类、基类、子类、派生类

如何使用继承

- 使用继承
  - 编写父类

```
class Pet {
    //公共的属性和方法
}
```

```
class Dog extends Pet {
    //子类特有的属性和方法
}
```

- 编写子类，继承父类

```
class Penguin extends Pet {
}
```

只能继承一个父类

继承关键字

super关键字

- super是直接父类对象的引用。
- 可以通过super来访问父类中被子类覆盖的方法或属性。
- 普通方法：
  - 没有顺序限制。可以随便调用。
- 构造函数中：
  - 任何类的构造函数中，若是构造函数的第一行代码没有显式的调用super(...);那么Java默认都会调用super();作为父类的初始化函数。 所以你这里的super();加不加都无所谓。

super 示例代码

```
public class Test {
          public static void main(String[] args) {
                    new ChildClass().f();
           }
}
class FatherClass {
    public int value;
    public void f(){
         value = 100;
         System.out.println
         ("FatherClass.value="+value);
     }
}
class ChildClass extends FatherClass {
    public int value;
    public void f() {
         super.f();
         value = 200;
         System.out.println
              ("ChildClass.value="+value);
         System.out.println(value);
         System.out.println(super.value);
     }
}
```

理解继承

- 子类访问父类成员
  - 访问父类构造方法

```
super();
super(name);
```

使用super关键字，super代表父类对象

在子类构造方法中调用且必须是第一句

  - 访问父类属性

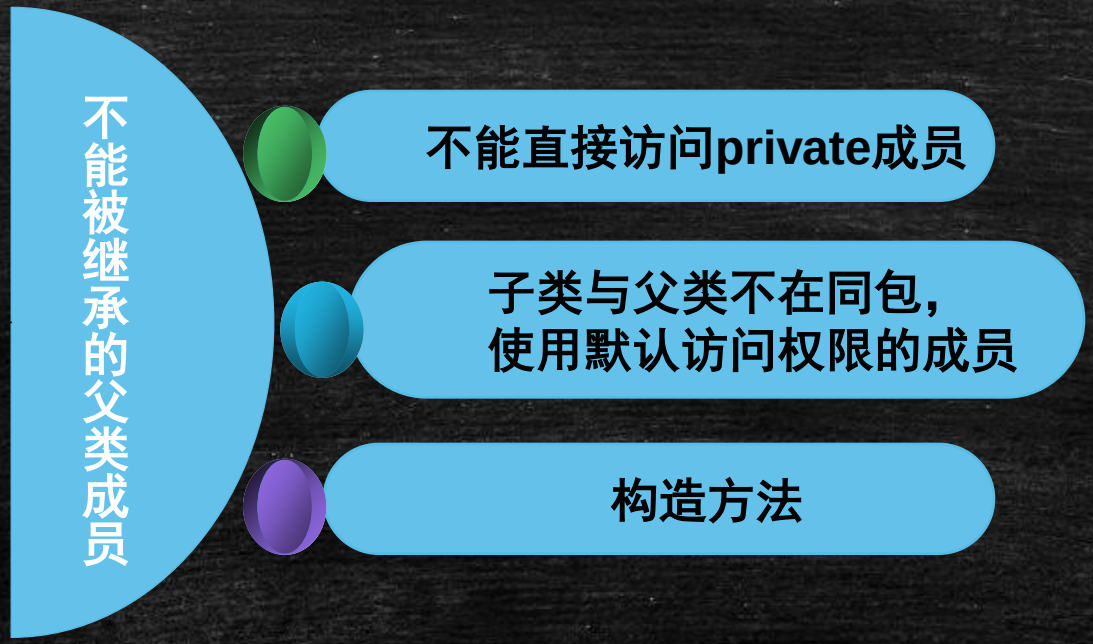
```
super.name;
```

  - 访问父类方法

```
super.print();
```

理解继承

- 子类可以继承父类的所有资源吗?



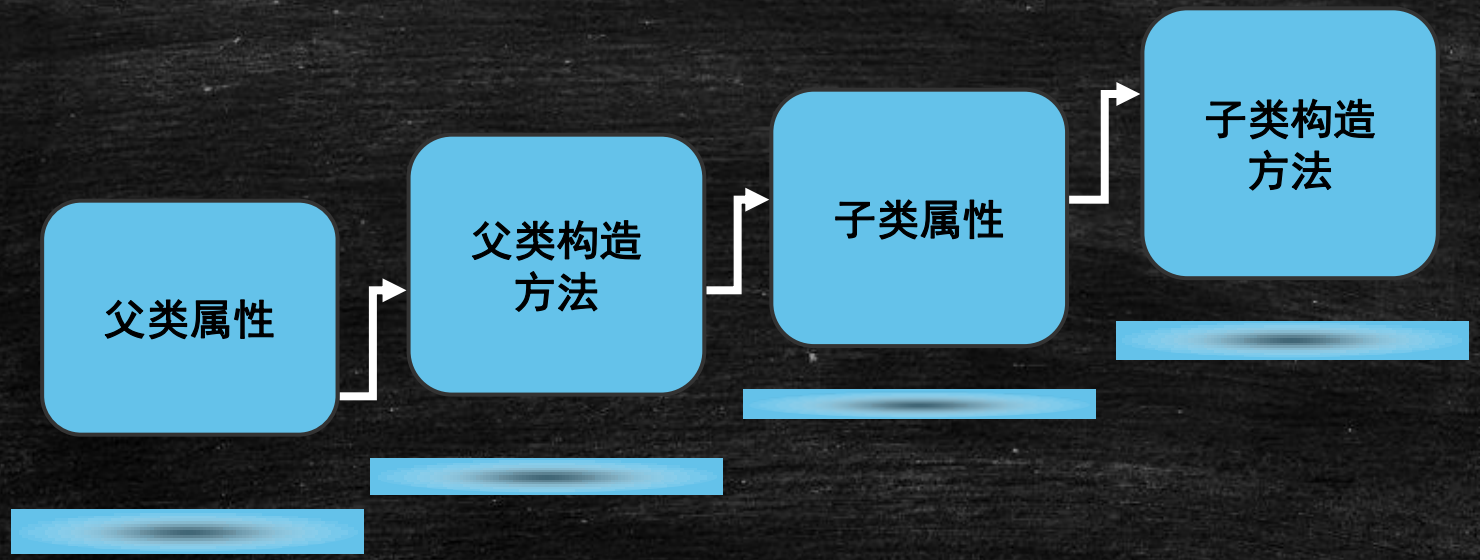
理解继承

- 访问修饰符protected
  - 可以修饰属性和方法
  - 本类、同包、子类可以访问
- 访问修饰符总结

访问修饰符	本类	同包	子类	其他
private	√			
默认(friendly)	√	√		
protected	√	√	√	
public	√	√	√	√

理解继承

- 多重继承关系的初始化顺序是怎样的?



继承

```
public class TestExtends {
        public static void main(String[] args) {
                Mammal m1 = new Mammal();
                m1.puru();
                m1.eat();
        }
}
class Animal {
        String eyes="眼睛";
        String name="无名";

        public void eat(){
                System.out.println("动物吃东西！");
        }
}
class Mammal extends Animal {
        //哺乳
        public void puru(){
                eyes="嘴巴";
                System.out.println("小动物吃奶！");
        }
}
```

在何处使用继承

- 何时使用继承?
  - 继承与真实世界类似
    - 只要说“猫是哺乳动物”，猫的很多属性、行为就不言自明了
    - 藏獒是一种狗

符合is-a关系的设计使用继承

  - 继承是代码重用的一种方式

将子类共有的属性和行为放到父类中

代码阅读

```
class Car {
    private int site = 4;  //座位数
    Car(){
        System.out.println ("载客量是"+site+"人);
    }
    public void setSite(int site){
        this.site = site;
    }
    void print(){
        System.out.print("载客量是"+sit
    }
}
```

```
class Bus extends Car {
    Bus(int site){
        setSite(site);
    }
}
```

```
public static void main(String[] args) {
    Bus bus = new Bus(20);
    bus.print();
}
```

载客量是4人
载客量是20人

继承

- 小结：
  - 通过继承可以简化类的定义，实现代码的重用
  - 子类继承父类的成员变量和成员方法，但不继承父类的构造方法
  - java中只有单继承，没有像c++那样的多继承。多继承会引起混乱，使得继承链过于复杂，系统难于维护。就像我们现实中，如果你有多个父母亲，那是一个多么混乱的世界啊。多继承，就是为了实现代码的复用性，却引入了复杂性，使得系统类之间的关系混乱。
  - java中的多继承，可以通过接口来实现
  - 如果定义一个类时，没有调用extends，则它的父类是：java.lang.Object。

方法重写

- 使用继承后效果

```
宠物的自白：
我的名字叫欧欧，我的健康值是100，我和主人的亲密程度是0。
宠物的自白：
我的名字叫楠楠，我的健康值是100，我和主人的亲密程度是0。
```

→ 调用父类的print()方法，不能显示Dog的strain信息和Peguin的sex信息

- **如何实现如下效果呢？**

```
宠物的自白：
我的名字叫欧欧，我的健康值是100，我和主人的亲密程度是0。
我是一只雪娜瑞犬。
宠物的自白：
我的名字叫楠楠，我的健康值是100，我和主人的亲密程度是0。
我的性别是Q妹。
```

→ 子类重写父类方法

方法的重写（override）

- 在子类中可以根据需要对从基类中继承来的方法进行重写。
- 重写方法必须和被重写方法具**有相同方法名称、参数列表**和返回类型。
- 重写方法不能使用比被重写方法更严格的访问权限。（由于多态）

重写（override）举例代码

```
public class TestOverride {
        public static void main(String[] args) {
                Animal animal = new Animal();
                animal.shout();
                Dog dog = new Dog();
                dog.shout();
        }
}
class Animal{
    void shout(){
        System.out.println("发出声音！");
    }
}
class Dog extends Animal {
    void shout(){
        System.out.println("旺旺旺！");
    }
}
```

小结

- 构造方法也会被重写吗?
- 方法重写的规则
  - 方法名相同
  - 参数列表相同
  - 返回值类型相同或者是其子类;
  - 访问权限不能严于父类
- 方法重载与方法重写

不能被继承，
因此不能重写

	位置	方法名	参数表	返回值	访问修饰符
方法重写	子类	相同	相同	相同或是其子类	不能比父类更严格
方法重载	同类	相同	不相同	无关	无关

小结

- super关键字来访问父类的成员
  - super只能出现在子类的方法和构造方法中
  - super调用构造方法时，只能是第一句
  - super和this不能同时出现在构造方法中
  - super不能访问父类的private成员
  - super和this都不能再static方法中

抽象类

- 以下代码有什么问题?

```
Pet pet = new Pet ("贝贝",20,40);
pet.print();
```

实例化Pet没有意义

- Java中使用抽象类，限制实例化

```
public abstract class Pet {
}
```

抽象方法

- 以下代码有什么问题？

```
public abstract class Pet {
    public void print() {
        //…
    }
}
```

每个子类的实现不同

- abstract也可用于方法——抽象方法
- 抽象方法没有方法体
  - 抽象方法必须在抽象类里
  - 抽象方法必须在子类中被实现，除非子类是抽象类

```
public abstract void print();
```

没有方法体

上机练习2——抽象Pet类2-1

- 需求说明：
  - 修改Pet类为抽象类
  - 修改Pet类的print()方法为抽象方法
  - 输出Dog信息

final用法

- Penguin类不希望再被其他类继承?
  - 使用final类

```
public final class Penguin extends Pet {
    //...
}
```

最终版的类

- 方法不希望被重写?
  - 使用final方法

```
public final void print () {
    //...
}
```

最终版的方法

- 属性值不希望被修改?
  - 使用常量

```
public class Penguin {
    final String  home ="南极";// 居住地
    public void setHome(String name){
    this.home=home; //错误，不可再赋值
  }
}
```

最终版的属性值

常见错误

```
class Dog {
    String name;
    public Dog(String name) {
        this.name = name;
    }
}
class Test {
    public static void main(String[] args) {
        final Dog dog = new Dog("欧欧");
        dog.name = "美美";
        dog = new Dog("亚亚");
    }
}
```

使用final修饰引用型变量，
变量不可以再指向另外的对象

使用final修饰引用型变量，变量的值是固定不变的，
而变量所指向的对象的属性值是可变的

Object类A

- **Object类是所有类的父类**
- 一个类如果没有使用extends显性的继承另外一个类，
- 那么这个类就继承自Object类。

```
public class Person{

}
等 同于
public class Person extends Object{

}
```

Object类B

- **Object类的主要方法**
- (1)toString()方法

```
public static void main(String [] args){
    Person p=new Person();//创建Person的对象p;
    System.out.println("不加toString()的输出:"+p);
    System.out.println("加上toString()的输
出:"+p.toString());
}
```

加与不加toString的效果相同，都会调用父类Object中的toString方法

Object类C

- **Object类的主要方法**
- (2)equals()方法

```
public class TestPerson//声明类
{
      public static void main(String [] args){
          Person p1=new Person("王一",33,"220283...");//创建Person
的对象p1;
          Person p2=new Person("王一",33,"220283....");//创建
Person的对象p2;
//调用父类Object的equals方法
System.out.println(p1.equals(p2)?"是同一个人":"不是同一个人");
      }
```

对象的比较——==和equals()

- ==：
  - 比较两基本类型变量的值是否相等
  - 比较两个引用类型的值即内存地址是否相等，即是否指向同一对象。
- equals()：
  - 两对象的内容是否一致
- 示例
  - object1.equals(object2)　　如：p1.equals(p2)
    - 比较所指对象的内容是否一样
    - 是比较两个对象，而非两个基本数据类型的变量
  - object1 == object2　　如：p1==p2
    - 比较p1和p2的值即内存地址是否相等，即是否是指向同一对象。
- 自定义类须重写equals()，否则其对象比较结果总是false。

上机练习--综合案例

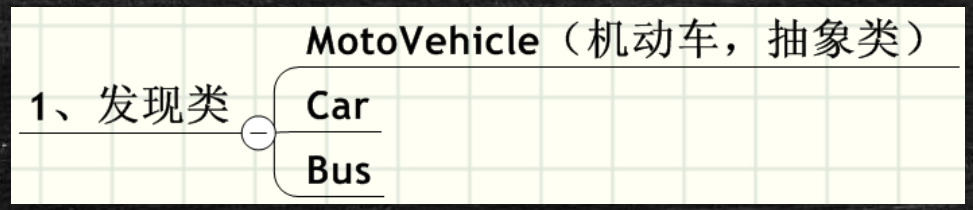
- 某汽车租赁公司出租多种车辆，车型及租金情况如下：

	轿车			客车（金杯、金龙）	
车型	别克商务舱GL8	宝马550i	别克林荫大道	<=16座	>16座
日租费(元/天)	600	500	300	800	1500

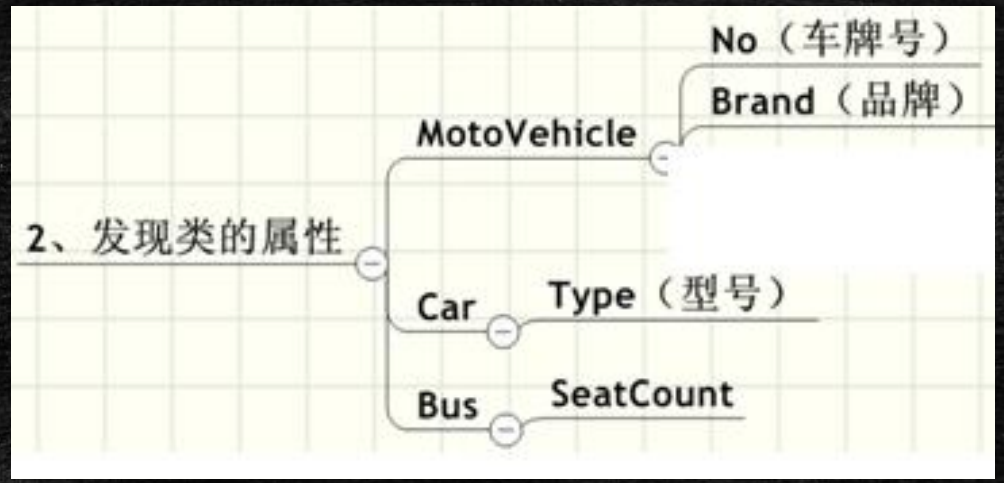
- 编写程序实现计算租赁价

上机练习--综合案例分析2-1

- 发现类

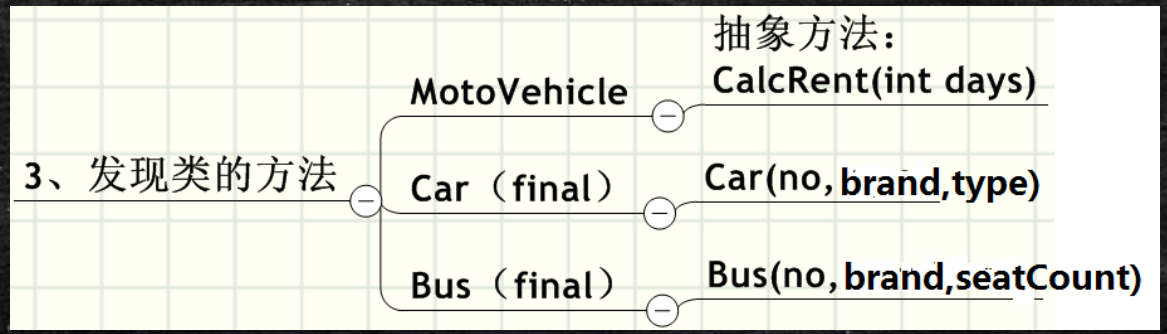


- 发现类的属性

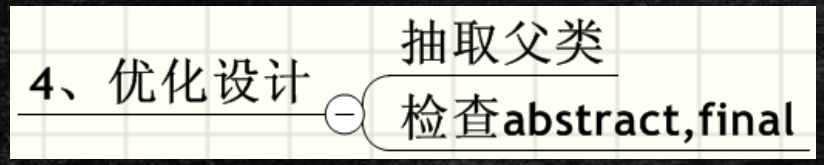


上机练习--综合案例分析2-2

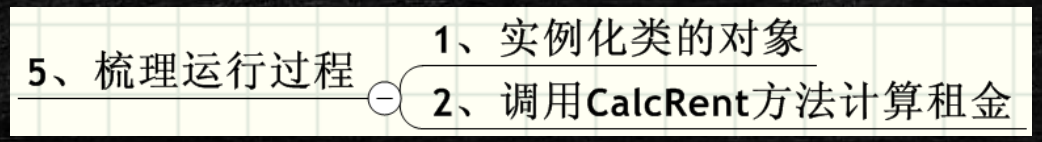
- 发现类的方法



- 优化设计

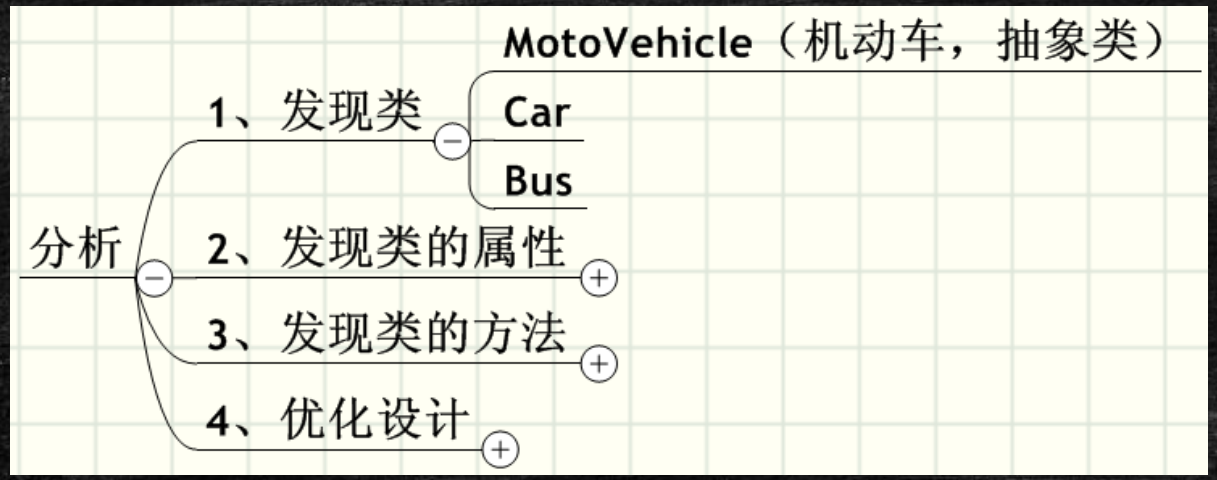


- 编写程序入口



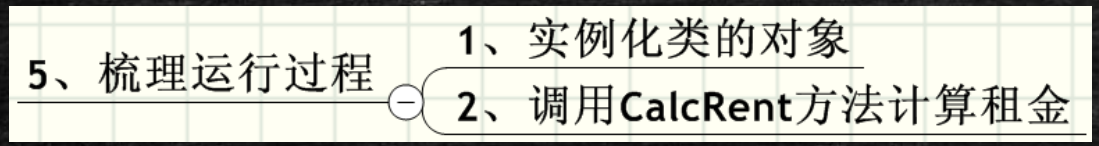
上机练习——编写MotoVehicle、Car、Bus类

- 需求说明：
  - 根据分析编写MotoVehicle、Car、Bus类



上机练习3——编写测试代码运行

- 需求说明：
  - 编写测试代码运行



Object类

Object类是所有Java类的根基类

如果在类的声明中未使用extends关键字指明其基类，则默认基类为Object类

```
public class Person {
        ...
}
```



```
public class Person extends Object {
        ...
}
```

重写：toString方法：

默认返回：包名+类名+@+哈希码

可以重写！

打开API文档，开始熟悉！

根据对象内存位置生成，唯一不重复！

课堂练习(10分钟)

- 熟悉方法重写
- 熟悉Object
- 重写toString方法
- 打开API文档，开始学着看看

继承深化

- 父类方法的重写：
  - “==”：方法名、形参列表相同。
  - “≤≤”：返回值类型和异常类型，子类小于等于父类。
  - “≥”：访问权限，子类大于等于父类
- **构造方法调用顺序**：
  - 根据super的说明，构造方法第一句 总是：super(…)来调用父类对应的构造方法。
  - 先向上追溯到Object，然后再依次向下执行类的初始化块和构造方法，直到当前子类为止。

示例

- 需求
  - 1.鱼类
    - 属性： 年龄 重量
    - 方法：自我介绍 游泳
  - 2.鸟类
    - 属性：年龄 颜色
    - 方法：自我介绍 飞
- 分析
  - 使用继承：
    - 抽取出动物类：属性（年龄） 方法（自我介绍）
    - 鱼类继承动物类，提供特有属性 重量和特有方法 游泳
    - 鸟类继承动物类，提供特有属性 颜色和特有方法 飞
    - 开发测试类，进行测试
  - 使用封装
    - 属性私有 方法public 提供对应的构造方法