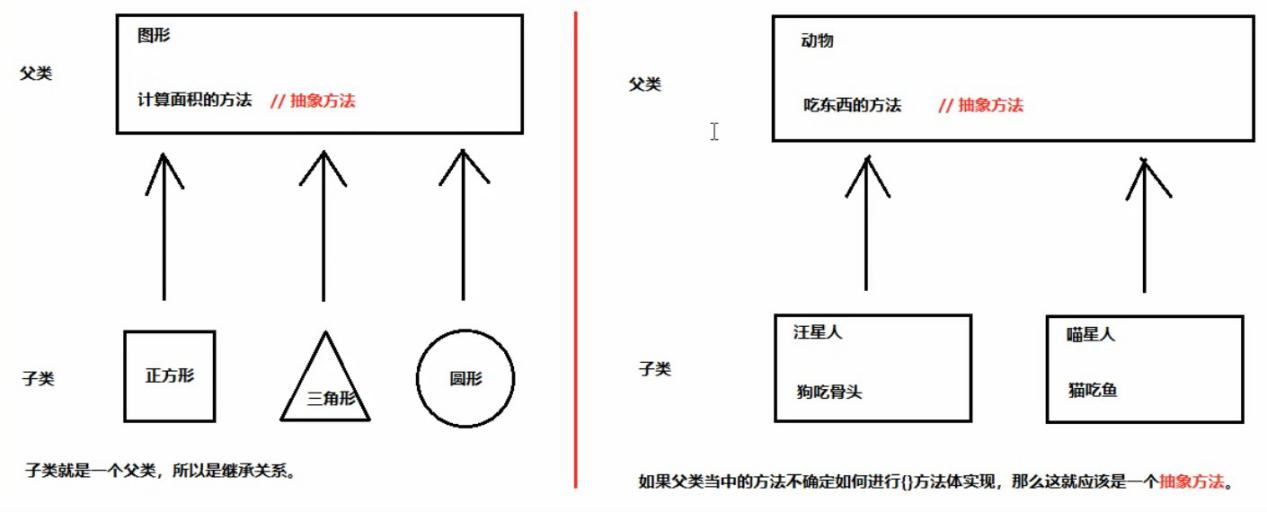
* + - * 1. 抽象的概念



* + - * 1. 抽象方法和抽象的格式

**package** doem01.inherit;  
*/\*  
抽象方法：就是加上abstr关键字，然后去掉大括号，直接分号结束。  
抽象类：抽象方法所在的类，必须是抽象类才行。在class之前写上abstr即可。  
 \*/***public abstract class** Anima {  
 *//这是一个抽象方法，代表吃东西，但具体吃什么东西（大括号的内容）不确定。* **public abstract void** eat();  
   
 *// 这是普通成员方法* **public void** normalMethod(){  
 *// 方法体...* }  
}

* + - * 1. 抽象方法和抽象类的使用

*/\*  
如何使用抽象类和抽象方法：  
1. 不能直接创建new抽象类对象。  
2. 必须用一个子类来进行继承。  
3. 子类必须覆盖重写抽象父类当中所有的抽象方法。  
覆盖重写（实现）：子类去掉抽象方法的abstract关键字，然后补上方法体大括号。  
4. 创建子类对象进行使用。  
 \*/*

抽象方法（父类）：

**public abstract class** Anima {  
 *//这是一个抽象方法，代表吃东西，但具体吃什么东西（大括号的内容）不确定。* **public abstract void** eat();  
  
 *// 这是普通成员方法* **public void** normalMethod(){  
 *// 方法体...* }  
}

子类：

**public class** Cat **extends** Anima {  
  
 @Override  
 **public void** eat(){  
 System.***out***.println(**"猫吃鱼"**); *// 将父类进行覆盖重写。* }  
}

主函数：

**public class** DemoMain {  
 **public static void** main(String[] args) {  
 *// Anima anima = new Anima(); 错误写法* Cat cat = **new** Cat();  
 cat.eat(); *// 可以通过子类定义方法函数* }  
}

* + - * 1. 抽象方法的注意事项



爷爷类：

*// 最高的抽象父类***public abstract class** Animal {  
 **public abstract void** sleep();  
  
 **public abstract void** eat();  
}

父类：

*// 子类也是一个抽象类***public abstract class** Dog **extends** Animal{  
 @Override  
 **public void** eat() {  
 System.***out***.println(**"狗吃骨头"**);  
 }  
 *// public abstract void sleep();*}

**子类一：**

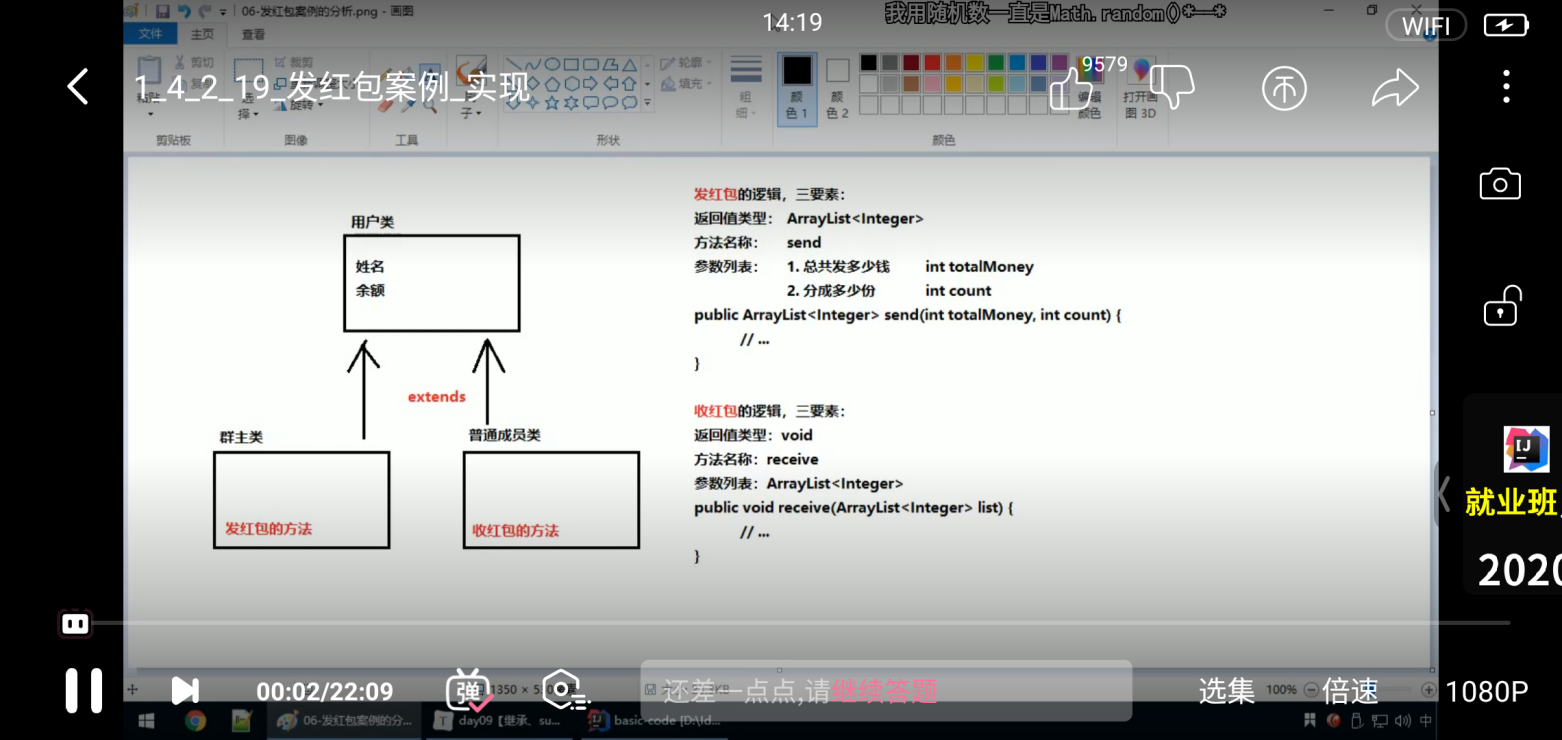
**public class** DodGolden **extends** Dog{  
 @Override  
 **public void** sleep() {  
 System.***out***.println(**"呼呼呼...."**);  
 }  
}

子类二：

**public class** Dog2Ha **extends** Dog{  
 @Override  
 **public void** sleep() {  
 System.***out***.println(**"嘿嘿嘿..."**);  
 }  
}

主函数：  
**public class** DemoMain {  
 **public static void** main(String[] args) {  
 *// Animal animal = new Animal() ; 错误 只要存在abstr就不可以直接new  
 // Dog dog = new Dog(); 错误，这也是抽象类* Dog2Ha ha = **new** Dog2Ha(); *// 这是一个普通类，可以直接new对象。* ha.eat(); *// 狗吃骨头* ha.sleep(); *// 嘿嘿嘿...* System.***out***.println(**"==============="**);  
  
 DodGolden golden = **new** DodGolden();  
 golden.sleep(); *// 呼呼呼...* golden.eat(); *// 狗吃骨头* }  
}

* + - * 1. 发红包案例-分析



* + - * 1. 发红包案例-实现

用户类：

**package** demo03;  
  
**public class** User {  
 **private** String **name**; *// 姓名* **private int money**; *// 钱数* **public** User() {  
 }  
  
 **public** User(String name, **int** money) {  
 **this**.**name** = name;  
 **this**.**money** = money;  
 }  
 *// 展示一下当前用户有多少钱* **public void** show(){  
 System.***out***.println(**"我叫："** + **name** + **",我有多少钱："** + **money**);  
 }  
 **public** String getName() {  
 **return name**;  
 }  
  
 **public void** setName(String name) {  
 **this**.**name** = name;  
 }  
  
 **public int** getMoney() {  
 **return money**;  
 }  
  
 **public void** setMoney(**int** money) {  
 **this**.**money** = money;  
 }  
}

群主类：

**package** demo03;  
  
**import** java.util.ArrayList;  
  
**public class** Manager **extends** User {  
  
 **public** Manager(){  
 }  
  
 **public** Manager(String name, **int** money){  
 **super**(name, money);  
 }  
  
 **public** ArrayList<Integer> send(**int** totalMoney, **int** count){  
 *// 首先需要一个集合，用来存储若干个红包的金额* ArrayList<Integer> redList = **new** ArrayList<>();  
  
 *// 首先看一下群主自己有多少钱* **int** leftMoney = **super**.getMoney(); *// 群主当前余额* **if**(totalMoney > leftMoney) {  
 System.***out***.println(**"余额不足"**);  
 **return** redList; *// 返回空集合* }  
  
 *// 扣钱，其实就是重新设置余额* **super**.setMoney(leftMoney - totalMoney);  
  
 *// 发红包需要平均拆分成count份* **int** avg = totalMoney / count; *// 这个为整除的红包* **int** mod = totalMoney % count; *// 这是是余数，也就是用下的零头  
  
 // 除不开的零头，包在最后一个红包当中  
 // 下面把红包一个个放在集合中* **for** (**int** i = count - 1; i > 0; i--) { *// 这里需要拿出去一个，用来存放余额* redList.add(avg);  
 }  
  
 *// 最后一个红包* **int** last = avg + mod;  
 redList.add(last);  
  
 **return** redList;  
 }  
}

普通类：

**package** demo03;  
  
**import** java.util.ArrayList;  
**import** java.util.Random;  
  
*// 普通成员***public class** Member **extends** User{  
 **public** Member(){  
 }  
  
 **public** Member(String name, **int** money ){  
 **super**(name, money);  
 }  
  
 **public void** receive(ArrayList<Integer> list){  
 *// 从多个红包当中随机抽取一个，给我自己。  
 // 随机获取一个集合当中的索引编号* **int** index = **new** Random().nextInt(list.size());  
 *// 根据索引，从集合当中删除，并且得到被删除的红包，给我自己。* **int** delta = list.remove(index);  
 *// 当前成员自己本来有多少钱。* **int** money = **super**.getMoney();  
 *// 加法，并且重新设置回去* **super**.setMoney(money + delta);  
 }  
  
}

主函数：

**package** demo03;  
  
**import** java.util.ArrayList;  
  
**public class** MainRedPacket {  
 **public static void** main(String[] args) {  
 *// 定义一个群主对象* Manager manager = **new** Manager(**"群主"**,100);  
  
 *// 定义几个成员对象* Member one = **new** Member(**"成员A"** ,0);  
 Member two = **new** Member(**"成员B"** ,0);  
 Member three = **new** Member(**"成员C"** ,0);  
  
 manager.show(); *// 100;* one.show(); *// 0* two.show(); *// 0* three.show(); *// 0* System.***out***.println(**"==============="**);  
  
 *// 群主总共发20的红包，分成3个红包* ArrayList<Integer> redList = manager.send(20,3);  
 *// 三个普通成员收红包* one.receive(redList);  
 two.receive(redList);  
 three.receive(redList);  
  
 manager.show(); *// 100 - 20 = 80  
 // 6 6 8 随机分给三个人* one.show();  
 two.show();  
 three.show();  
 }  
}

结果：

我叫：群主,我有多少钱：100

我叫：成员A,我有多少钱：0

我叫：成员B,我有多少钱：0

我叫：成员C,我有多少钱：0

===============

我叫：群主,我有多少钱：80

我叫：成员A,我有多少钱：6

我叫：成员B,我有多少钱：8

我叫：成员C,我有多少钱：6