Shoot射击游戏第一天：

\1. 创建6个对象类，并创建World类并测试

Shoot射击游戏第二天：

\1. 给6个对象类添加构造方法，并测试

Shoot射击游戏第三天：

\1. 设计小敌机数组、大敌机数组、小蜜蜂数组、子弹数组，并测试

Shoot射击游戏第四天：

\1. 设计FlyingObject超类，6个对象类继承超类

\2. 在超类中设计2个构造方法，6个对象类分别调用

\3. 画窗口

回顾：

\1. null: 空，没有指向任何对象，若引用的值为null，则该引用不能再进行任何操作了

若操作则发生NullPointerException空指针异常

\2. 引用类型画等号：指向同一个对象，会影响

基本类型画等号：赋值，不会影响

\3. 引用类型数组：

1) Student[ ] stus = new Student[3]; //null,null,null

stus[0] = new Student(…);

stus[1] = new Student(…);

stus[2] = new Student(…);

for(int i=0;i<stus.length;i++){

stus[i].age = 35;

System.out.println(stus[i].name);

stus[i].sayHi();

}

2) Student[ ] stus = new Student[]{

new Student(…),

new Student(…),

new Student(…)

};

正课：

1. 继承:

1) 作用: 代码复用

2) 通过extends来实现继承

3) 超类/父类: 所有派生类所共有的属性和行为

派生类/子类: 派生类所特有的属性和行为

4) 派生类继承超类后，派生类具有: 派生类的+超类的

( 注: 派生类可以访问超类的，但超类不能访问派生类的 )

5) 单一继承(一个超类可以有多个派生类，一个派生类只能有一个超类)

6) 传递性

7) Java规定：构造派生类之前必须先构造超类

---在派生类的构造方法中若没有调用超类的构造方法，则默认super()调用超类的构造无参构造方法

---在派生类的构造方法中若自己调用了超类的构造方法，则不再默认提供

1. super: 指代当前对象的超类对象

super的用法:

1) super.成员变量名----------------访问超类的成员变量

2) super.方法名()-------------------调用超类的方法-------------明天讲方法的重写时候讲

3) super()-------------------------调用超类的构造方法

说明:

1. 继承一定要符合is a(是一个)的关系，不要乱用
2. 派生类继承的是超类的成员变量和方法

构造方法不是被派生类继承的，而是派生类通过super去调用的

练习:-------------------------oo.day05包中(必须做完)

1. 创建类Person，包含:

1) 成员变量: name,age,address

2) 构造方法: Person( 3个参数 ) { }

3) 方法: void sayHi() { 输出3个参数的值 }

1. 创建类Student，继承Person，包含:

1) 成员变量: stuId学号

2) 构造方法: Student(4个参数) { }

1. 创建类Teacher，继承Person，包含:

3) 成员变量: salary工资

4) 构造方法: Teacher(4个参数) { }

1. 创建类Doctor，继承Person，包含:

5) 成员变量: level职称

6) 构造方法: Doctor(4个参数) { }

1. 创建测试类Test，main中:

1) 创建Student数组stus，包含3个元素，给每个元素分别赋值，遍历数组输出每个学生的名字并问好

2) 创建Teacher数组tes，包含3个元素，给每个元素分别赋值，遍历数组输出每个老师的名字并问好

3) 创建Doctor数组docs，包含2个元素，给每个元素分别赋值，遍历数组输出每个医生的名字并问好

swing相关的(做窗口的):-------不要求掌握，现在已经被淘汰了

窗口程序(少)--------.net(C#)做得比较多

网页程序(多)--------java做得比较多

class Dog{ //狗

String name;

int age;

String address;

}

class FlyingObject{ //飞行物

}

class Airplane extends FlyingObject{

}

class BigAirplane extends FlyingObject{

}

class Bee extends FlyingObject{

}

class Hero extends FlyingObject{

}

class Sky extends FlyingObject{

}

class Bullet extends FlyingObject{

}

class Person{ //人

}

class Student extends Person{ //学生

}

class Teacher extends Person{ //老师

}

class Doctor extends Person{ //医生

}

class Person{ //人类-------------------------------------------------------超类/父类

String name;

int age;

String address;

void eat(){}

void sleep(){}

}. //继承表示-----代码虽然我没有写，但是也属于我，只是没有写在一起

class Student extends Person { Student zs = new Student(); ---------------------派生类/子类

String stuId; zs. stuId/study();

void study(){} zs. name/age/address/eat()/sleep();

}

class Teacher extends Person {. Teacher ls = new Teacher();---------------------派生类/子类

double salary; ls. salary/teach();

void teach(){} ls. name/age/address/eat()/sleep();

}

class Doctor extends Person {-------------------------------------------------派生类/子类

String level;

void cut(){}

}

class Aoo{-------------a

int a;

}

class Boo extends Aoo{---b+a ( a虽然没有写在Boo类中，但因为继承了，所以a也属于Boo，就是没有写在一起 )

int b;

}

class Coo extends Boo{---c+b+a

int c;

}

软件中的继承:

代码不用自己写，自己也能用

生活中的继承:

1. 继承财产:

钱不用自己挣，自己也能花

1. 继承皇位:

江山不用自己打，自己也能坐

1. 继承工作:

工作不用自己找，自己也能干

必须把命名写好

bullets[0] = new Bullet(100，200);--------此处语法错误，都是因为构造方法参数的问题

bullets[1] = new Bullet(170，230);