}

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

Shoot射击游戏第五天：

1. 将小敌机数组、大敌机数组、小蜜蜂数组组合为FlyingObject数组，并测试
2. 在6个派生类中重写step方法

回顾：

1. 继承：

代码复用，extends，

超类:派生类所共有的属性和行为 派生类:派生类所特有的属性和行为

派生类具有: 派生类的+超类的 超类不能访问派生类的

单一继承、传递性

Java规定: 构造派生类之前必须先构造超类

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

----派生类的构造方法中若调用了超类构造，则不再默认提供

2. super：指代当前对象的超类对象

super.成员变量名---------------访问超类的成员变量

super.方法名()------------------调用超类的方法

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

正课：

1. 向上造型：

1) 超类型的引用指向派生类的对象

2) 能点出来什么，看引用的类型-------这是规定，访问派生类特有的东西后期讲多态的时候讲

1. 方法的重写(Override)：重新写，覆盖

1) 发生在父子类中，方法名相同，参数列表相同，方法体不同

2) 重写方法被调用时，看对象的类型(new谁就调谁的)-----这是规定

3) 遵循”两同两小一大”原则:-----------了解(一般都是一模一样的)

3.1) 两同:

3.1.1) 方法名相同

3.1.2) 参数列表相同

3.2) 两小:

3.2.1) 派生类方法的返回值类型小于或等于超类方法的

1) void时，必须相同

2) 基本类型时，必须相同

3) 引用类型时，小于或等于

3.2.2) 派生类方法抛出的异常小于或等于超类方法的----------异常之后说

3.3) 一大:

3.3.1) 派生类方法的访问权限大于或等于超类方法的----------明天讲完访问控制修饰符讲

1. 重写与重载的区别：---------------常见面试题(面试时只答第一条)

1) 重写(Override):

1.1) 发生在父子类中，方法名相同，参数列表相同，方法体不同

1.2) 遵循”运行期绑定”，看对象类型来调用方法

2) 重载(Overload):

2.1) 发生在一个类中，方法名相同，参数列表不同，方法体不同

2.2) 遵循”编译期绑定”，看参数/引用类型来绑定调用的方法

编译期: .java源文件，经过编译，生成.class字节码文件-----重载发生在此时

运行期: JVM加载.class并运行.class-----------------------重写发生在此时

Eoo o = new Foo(); //向上造型

重写看对象-----------Foo

重载看参数/引用------Eoo

实践出真知

6个对象类分别重写step()，运行World类看输出结果--12分钟------------5:18继续

小敌机y向下移动

大敌机y向下移动

小蜜蜂x左右移动，y向下移动

英雄机切换图片

天空y和y1向下移动

子弹y向上移动

每个飞行物的具体移动方式----都是不一样的

初级程序员---------很少项目经理面试你，一般项目经理看谁闲着就让谁去面

--------------------------下课休息15分钟----------3:15上课

\1. 问: 什么情况需要重写?

答: 当派生类觉得超类的行为不够好的时候才重写

\2. 问: 重写一定要发生吗?

答: 不是一定要发生

\3. 通过super.方法名()可以调用超类的方法----应用率不高

----当派生类的行为是在超类的基础之上再加其他操作的时候使用

继承了一个中餐馆:

\1. 我还是想做中餐----------------------不用重写

\2. 我想改做西餐------------------------要重写

\3. 我想在中餐基础之上加西餐------------要重写(先super中餐再加入西餐)

FlyingObject[ ] enemies;

FlyingObject[ ] enemies = null; //等价的

System.out.println(enemies.length); //空指针异常

FlyingObject[ ] enemies = { };

FlyingObject[ ] enemies = new FlyingObject[0]; //等价的

System.out.println(enemies.length); //0

6:Static

7:抽象

8:

FlyingObject o = new Airplane();

FlyingObject o = new BigAirplane();

FlyingObject o = new Bee();

FlyingObject o = new Hero();

FlyingObject o = new Sky();

FlyingObject o = new Bullet(100,200);

o. 只能访问FlyingObject中的

Person p = new Student();

Person p = new Teacher();

Person p = new Doctor();

p. 只能访问Person中的

Animal o = new Tiger();

Animal o = new Dog();

Animal o = new Lion();

o. 只能访问Animal中的

//动物是动物

动物 o = new 动物();

//老虎是老虎

老虎 o = new 老虎();

//老虎是动物

动物 o = new 老虎();

//动物是老虎---------语义不通

老虎 o = new 动物(); //编译错误

class 动物{

}

class 老虎 extends 动物{

}