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

Shoot射击游戏第六天：

1. 给类中成员添加访问控制修饰符
2. 设计Images工具类给对象准备图片

Shoot射击游戏第七天：

1. 设计窗口的宽和高为常量、适当地方做修改
2. 画对象：

1) 想画对象需要获取对象的图片，每个对象都得获取图片，意味着获取图片行为为共有行为，

所以将获取图片行为设计在超类中，每个对象获取图片的方式都是不一样的，所以设计为抽象方法

----在FlyingObject中设计抽象方法getImage()来获取对象的图片

2) 想获取图片需要考虑对象的状态，因为在不同状态下获取的图片是不一样的，因为每个对象都有状态，

所以将状态设计在超类中，状态一般都设计为常量，同时再设计变量state表示当前状态

----在 FlyingObject中设计LIFE、DEAD、REMOVE常量，state变量

状态设计好了，那么对象在获取图片时需要去判断对象的状态，每个对象都需要判断状态，为共有行为，

所以将判断状态行为设计在超类中，每个对象判断状态的方式都是一样的，所以设计为普通方法

----在FlyingObject中设计isLife()、isDead()、isRemove()来判断对象的状态

3) ----重写超类的getImage()获取图片:

3.1) 天空Sky，直接返回sky图片即可

3.2) 子弹Bullet:

3.2.1) 若活着的，直接返回bullet图片即可

3.2.1) 若死了的，直接删除

3.3) 英雄机Hero:

3.3.1) 若活着的，直接返回两张英雄机图片的来回切换

3.4) 小敌机Airplane:

3.4.1) 若活着的，直接返回第一张图片即可

3.4.2) 若死了的，返回从第二张图到第五张图的切换，5后删除

3.5) 大敌机BigAirplane:

3.5.1) 若活着的，直接返回第一张图片即可

3.5.2) 若死了的，返回从第二张图到第五张图的切换，5后删除

3.6) 小蜜蜂Bee:

3.6.1) 若活着的，直接返回第一张图片即可

3.6.2) 若死了的，返回从第二张图到第五张图的切换，5后删除

4) 图片有了就可以开画了，在World类中重写paint()画方法----此内容不需要掌握

Shoot射击游戏第八天：

1. 敌人入场:

1) 创建敌人为窗口所特有的行为，所以在World中创建nextOne()生成敌人

2) 敌人入场为定时发生的，所以在run()中调用enterAction()实现敌人入场---enterAction()下调用repaint()重画

在enterAction中:

每400毫秒，获取敌人obj，enemies扩容，并将obj装到最后一个元素位置上

1. 子弹入场:

1) 创建子弹为英雄机所特有的行为，所以在Hero中创建shoot()生成子弹

2) 子弹入场为定时发生的，所以在run()中调用shootAction()实现子弹入场

在shootAction中:

每300毫秒，获取子弹对象bs，bullets数组扩容，数组的追加

回顾:

1. final: 最终的，不可改变的

变量不能被改变，方法不能被重写，类不能被继承

1. static final常量:

必须声明同时初始化，类名点来访问、不能改变，大写

编译器在编译时会将常量直接替换为具体的值，效率高

数据永远不变、并且经常使用

1. 抽象方法:

abstract，只有方法的定义，没有具体的实现(连{}都没有)

1. 抽象类:

abstract，包含抽象方法的类必须是抽象类，不能被实例化(new对象)

是需要被继承的，派生类:

1) 重写抽象类中的所有抽象方法----变不完整为完整

2) 也声明为抽象类

意义:

代码复用、向上造型

包含抽象方法，为所有派生类提供统一的入口(向上造型之后通过超类引用能点出来)

正课:

1. 成员内部类：应用率不高，了解即可

1) 类中套类，外面的叫Outer外部类，里面的叫Inner内部类

2) 内部类通常只服务于外部类，对外不具备可见性

3) 内部类对象通常是需要在外部类中创建的

4) 内部类中可以直接访问外部类的成员(包括私有的)

内部类中有个隐式的引用指向了创建它的外部类对象----外部类名.this

1. 匿名内部类：

1) 若想创建一个类(派生类)的对象，并且对象只被创建一次，此时该类不必命名，称为匿名内部类

优点: 代码结构看起来更清晰

2) 小面试题:

问: 内部类有独立的.class吗?

答: 有

做项目功能的套路:

1. 先写行为/方法:

1) 若为某个对象所特有的行为，就将方法设计在对应的类中

2) 若为所有对象所共有的行为，就将方法设计在超类中

1. 窗口中调用:

1) 定时触发的，就在定时器中调用

2) 事件触发的，就在侦听器中调用-----明天讲

敌人入场: 创建敌人

敌人是由窗口产生的，所以将创建敌人的行为设计在窗口中

子弹入场: 创建子弹

子弹是由英雄机发射出来的，所以将创建子弹行为设计在英雄机中

飞行物移动: 移动

飞行物移动是所有对象所共有的行为，所以将飞行物移动设计在超类中

Bullet[] bs = hero.shoot();

bullets = Arrays.copyOf(bullets,bullets.length+bs.length);

System.arraycopy( bs, 0, bullets, ?, bs.length );

if(bs.length>2){ //三

bullets[bullets.length-1] = bs[0];

bullets[bullets.length-2] = bs[1];

bullets[bullets.length-3] = bs[2];

}else if(bs.length>1){ //双

bullets[bullets.length-1] = bs[0];

bullets[bullets.length-2] = bs[1];

}else{ //单

bullets[bullets.length-1] = bs[0];

}

FlyingObject---------正确

FlyingObject[ ]-------错误(因为一次只生成一个敌人)

//生成敌人(小敌机、大敌机、小蜜蜂)

public FlyingObject nextOne(){

}

?处为什么类型:

既能接收小敌机对象

也能接收大敌机对象

也能接收小蜜蜂对象

假设有5个敌人---扩容(变成6个元素)

来一个敌人，就将enemies数组扩一个容，将敌人装到 enemies的最后一个元素位置上

**public** FlyingObject nextOne(){

Random rand = **new** Random(); //随机数对象

**int** type = rand.nextInt(3); //0到2之间

**if**(type==0){

**return** **new** Bee();

}**else** **if**(type==1){

**return** **new** Airplane();

}**else** {

**return** **new** BigAirplane();

}

}

项目功能:

1. 敌人入场
2. 子弹入场

TimerTask, long, long

timer.schedule( new Aoo(), 10, 10 );

class Aoo extends TimerTask{

}

第一个10: 从程序启动到第一次触发的时间间隔

第二个10: 从第一次触发到第二次触发的时间间隔

从第二次触发到第三次触发的时间间隔

从第三次触发到第四次触发的时间间隔

…

定时器-------Timer

自动进行的:-------------------定时器来实现

1. 敌人入场
2. 子弹入场
3. 飞行物移动(除了英雄机)
4. 子弹射击敌人
5. 敌人与英雄机撞

API (多线程)

休息15分钟----------11:24继续

Mama$Baby.class-----内部类Baby的字节码文件

NstInnerClassDemo$1.class

NstInnerClassDemo$2.class

NstInnerClassDemo$3.class

字母、

数字、\_和$

射击游戏中------有2个地方特别适合

听得懂，但写不出来------不用重听

听不懂了----------------下个月见

知识会反复迭代

讲得时候，听得挺明白的---------我讲明白了+你认真听了

做的时候，脑子一片空白---------非常正常的

面向对象-----不是一个语法，而是一种设计的思想

向上造型----------上午讲的

中午没有问关于向上造型的问题的

下午没有问关于向上造型的问题的

晚上有个别同学问问关于向上造型的问题的----有什么好处

第二天、第三天-----很多问我向上造型有什么好处

原因-----没有认真的去想

项目功能:

1. 敌人入场:
2. 子弹入场:

---------------18页

回复-----给我截图(17次)

英雄机出不来------World、Hero、FlyingObject、Images