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

Shoot射击游戏第九天：

1. 飞行物移动: ----先要画出天空的第二张图片

1) 飞行物移动为派生类所共有的行为，所以在超类中设计抽象step()方法实现飞行物移动，派生类重写step()

2) 飞行物移动为定时发生的，所以在run()中调用stepAction()实现飞行物的移动

在stepAction()中:

天空动，遍历敌人数组敌人动，遍历子弹数组子弹动

1. 英雄机随着鼠标移动:

1) 英雄机随着鼠标动为英雄机的行为，所以在Hero中设计moveTo()实现英雄机随着鼠标动

2) 英雄机随着鼠标移动为事件触发的，所以在侦听器中重写mouseMoved()鼠标移动事件

在mouseMoved()中:

获取鼠标的x坐标和y坐标，调用moveTo()实现英雄机移动

1. 删除越界的敌人和子弹:

1) 在FlyingObject中设计outOfBounds()检测敌人是否越界，在Bullet中重写outOfBounds()检测子弹是否越界

2) 删除越界敌人和子弹为定时发生的，所以在run中调用outOfBoundsAction()删除越界敌人和子弹

在outOfBoundsAction()中:

声明不越界数组，遍历enemies/bullets数组，

判断若不越界: 将不越界的敌人/子弹装到不越界数组中

最后将不越界数组复制到enemies/bullets中

Shoot射击游戏第十天：

1. 设计Enemy得分接口，Airplane和BigAirplane实现Enemy得分接口

设计Award奖励接口，Bee实现Award奖励接口

从头开始做项目，应该如何入手:

1. 首先要了解业务需求，而后先设计数据结构，再写算法
2. 设计数据结构: 类+成员变量

1) 设计超类FlyingObject、接口Enemy、Award

2) 设计派生类继承超类、实现接口

3) 设计图片工具类Images

4) 设计World窗口类，画窗口

1. 设计算法: 方法 -------今天6:30开始上职业素质课，7:30开始上晚课

1) 画对象

2) 敌人入场、子弹入场

3) 飞行物移动、英雄机随着鼠标移动

4) 删除越界敌人和子弹

正课:----------今天只讲语法，具体接口和多态好处----下次课讲

1. 接口:

1) 是一种引用数据类型

2) 由interface定义

3) 只能包含常量和抽象方法(数据默认都是常量，方法默认都是抽象的---接口中成员的默认访问权限为public)

4) 接口不能被实例化

5) 接口是需要被实现/继承的，实现类/派生类: 必须重写接口中的所有抽象方法

6) 一个类可以实现多个接口，用逗号分隔。若又继承又实现时，应先继承后实现

7) 接口可以继承接口

1. 多态: 多种形态

1) 多态的意义:

1.1) 同一个对象被造型为不同的类型时，有不同的功能-----所有对象都是多态的(周六面向对象总结讲)

----对象的多态：水、我、你…

1.2) 同一类型的引用在指向不同对象时，有不同的实现-----抽象方法都是多态的

----行为的多态：cut()、run()、step()、getImage()…

2) 向上造型:

2.1) 超类型的引用指向派生类的对象

2.2) 能造型成为的类型有: 超类+所实现的接口

2.3) 能点出来什么，看引用的类型

3) …

4) …

今天只讲就接口的语法和什么时候用----至于接口的好处下次课做案例才能看出来

设计规则:

1. 将所有派生类所共有的属性和行为，抽到超类中--------抽共性
2. 派生类的行为都一样，设计为普通方法

派生类的行为都不一样，设计为抽象方法

1. 将部分派生类所共有的行为，抽到接口中

接口是对继承单根型的扩展---------------实现多继承

子弹射击敌人:----------------------业务周五讲

1) 打掉小敌机------玩家得1分

2) 打掉大敌机------玩家得3分

3) 打掉小蜜蜂------英雄机得奖励

4) 打掉大黄蜂------英雄机得奖励、玩家得5分

----得分行为、得奖励行为都是小敌机和大敌机所共有的行为

----1. 不能放在超类中，是因为放在超类中意味着所有派生类都具有了

----2. 不能放在单独的类中，是因为一个类只能继承一个超类

----3. 结论: 只能放在接口中

class FlyingObject{

width,height,x,y

step(), getImage(), isLife(), isDead(), isRemove()

}

interface Enemy{ //得分接口

int getScore();

}

interface Award{ //奖励接口(只是为了以后的扩展使用)

int getAwardType();

}

class Bee extends FlyingObject implement Award {

public int getAwardType(){ … }

}

class BigYellowBee extends FlyingObject implement Award,Enemy {

public int getAwardType(){ … }

public int getScore(){ return 5; }

}

class Airplane extends FlyingObject implements Enemy {

public int getScore() { return 1; }

}

class BigAirplane extends FlyingObject implements Enemy {

public int getScore() { return 3; }

}

class Hero extends FlyingObject{

}

class Sky extends FlyingObject{

}

class Bullet extends FlyingObject{

}

下课休息15分钟-------------4:10上课

动物 o1 = new 老虎();

动物 o2 = new 鸟();

动物 o3 = new 鱼();

o1.run(); //地上

o2.run(); //天上

o3.run(); //水里

abstract class 动物{

abstract void run();

}

class 老虎 extends 动物{

void run{ 在地上跑 }

}

class 鸟extends 动物{

void run{ 在天上飞 }

}

class 鱼extends 动物{

void run{ 在水里游 }

}

人 p1 = new 理发师();

人 p2 = new 外科医生();

人 p3 = new 演员();

p1.cut(); //剪发

p2.cut(); //开刀

p3.cut(); //停止表演

abstract class 人{

abstract void cut();

}

class 理发师 extends 人{

void cut() { 剪发 }

}

class 外科医生extends 人{

void cut() { 开刀 }

}

class 演员extends 人{

void cut() { 停止表演 }

}

cut()

我 me = new 我();

讲师 o1 = me; //向上造型

孩子他妈 o2 = me;

老公的老婆 o3 = me;

o1.授课();

o2.买玩具();

o3.收工资();

me.授课();

me.买玩具();

me.收工资();

interface 讲师{

void 授课();

}

interface 孩子他妈{

void 买玩具();

}

interface 老公的老婆{

void 收工资();

}

class 我 implement 讲师,孩子他妈,老公的老婆{

public void 授课() { … }

public void 买玩具() { … }

public void 收工资() { … }

}

水：固态、液态、气态

我：讲师、孩子他妈、老婆、女儿、儿媳妇、路人甲

你：学生、儿子/女儿、路人甲

我的QQ: 1945019382

类和类---------继承extends

接口和接口-----继承extends

类和接口-------实现implements

interface Inter{

public static final int NUM = 1;

public abstract void show();

int COUNT = 2; //默认public static final

void say(); //默认public abstract

//int num; //编译错误，常量必须声明同时初始化

//void test() { } //编译错误，抽象方法没有方法体

}

class Aoo{

int num;

}

//下课休息15分钟------------10:02上课

问题分三种:

1. 编译错误: 直接在错误处显示红色线，都是语法错误
2. 异常: 运行时发生的错误，一般表示不正常的现象
3. 程序的运行结果与你所预期的结果不同:

-----打桩: System.out.println( 数据 );-----帮助咱们找到问题所在，而不帮我们解决问题

数组下标越界(ArrayIndexOutOfBoundsException)

空指针异常(NullPointerException)

类型转换异常(ClassCastException)

API时专门的一堂课讲异常

子弹射击敌人