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

2. 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:

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

//删除越界敌人和子弹

public void outOfBoundsAction(){

int index = 0; //1.下标 2.不越界敌人个数

FlyingObject[ ] enemyLives = new FlyingObject[enemies.length]; //不越界敌人数组

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

FlyingObject f = enemies[i];

if( !f.outOfBounds() ){ //不越界

enemyLives[index] = f;

index++;

}

}

enemies = Arrays.copyOf( enemyLives, index );

}

假设有50个敌人，其中有30个越界了-------缩容30次(效率低)

//删除越界的敌人和子弹

**public** **void** outOfBoundsAction(){ //每10毫秒走一次

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

FlyingObject f = enemies[i];

if(f.outOfBounds()){ //越界了

缩容(本身是数组复制)

}

}

}

class FlyingObject{

public boolean outOfBounds(){ //敌人的越界检查

return this.y>=World.HEIGHT;

}

}

class Airplane extends FlyingObject{

}

class BigAirplane extends FlyingObject {

}

class Bee extends FlyingObject {

}

class Bullet extends FlyingObject {

public boolean outOfBounds(){ //重写

return this.y<=-this.height;

}

}

现在的是两辈-----超类+派生类

更好的是三辈-----超超类+超类+派生类

class FlyingObject{ //所有飞行物的超类----------------超超类

width,height,x,y

}

class Enemy extends FlyingObject{ //敌人的超类---------超类

所有敌人所共有的属性和行为

}

class Airplane extends Enemy{

}

class BigAirplane extends Enemy{

}

class Bee extends Enemy{

}

class Sky extends FlyingObject{

}

class Hero extends FlyingObject{

}

class Bullet extends FlyingObject{

}

想删除越界的对象，首先得知道对象是否越界

敌人最多10个、子弹最多16个-----共26个

运行时间长一些，会比较慢、卡-----因为对象太多了

每秒-----2.5个敌人+3.3个子弹-----少说每秒钟5对象-----1分钟300个对象—10分钟3000个对象

每10毫秒遍历stepAction()3000个+paint()3000个，共6000个

每1秒600000，1分钟36000000

Swing窗口相关的------不要求掌握

1) 事件: 发生了一个事

2) 事件处理: 发生事之后所做的操作

3) 侦听器:

3.1) 有一个侦听器对象

3.2) 把侦听器装到面板去

MouseAdapter l = new MouseAdapter(){

重写mouseMoved()事件

};

装到面板上…

class Aoo extends MouseAdapter{

}

鼠标移动: 英雄机随着动

事件 事件处理

1) 鼠标点击 启动状态变为运行状态

2) 鼠标移动 英雄机随着动

3) 鼠标移出 运行状态变为暂停状态

4) 鼠标移入 暂停状态变为运行状态

程序的状态:

1) 启动状态

2) 运行状态

3) 暂停状态

4) 游戏结束状态

API

5+1--------加1

5+(-1)------减1

行为+调用

天空、子弹、小敌机、大敌机、小蜜蜂

回顾:

1. 成员内部类:----了解

类中套类，内部类对外不具备可见性，内部类对象需要在外部类中创建，

内部类中可以直接访问外部类成员(包括私有的) 外部类名.this指向了创建它的外部类对象

1. 匿名内部类:

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

结构清晰、操作更方便

正课:

业务action()方法 与 画paint()方法

多线程并发的-----乍一看是同时进行，但是实际上还是一个一个的执行