

**学生实验实习报告册**

|  |  |
| --- | --- |
| 学年学期： | 2022-2023 学年 🗹春**🞎**秋学期 |
| 课程名称： | 程序设计实训 |
| 学生学院： | 国际学院 |
| 专业班级： | 34082202 |
| 学生学号： | 2022214986 |
| 学生姓名： | 周昕阳 |
| 联系电话： | 18297704195 |

**重庆邮电大学教务处制**

目 录

[一.实验任务概述 1](#_Toc135851428)

[二.实验准备及方法 1](#_Toc135851429)

[三.实验设备及软件 2](#_Toc135851430)

[四.实验步骤、过程原始记录(数据、图表、计算等) 2](#_Toc135851431)

[(一).实验任务完成情况 2](#_Toc135851432)

[(二).任务成果展示 3](#_Toc135851433)

[1.流程图/时序图/类图 3](#_Toc135851434)

[2.设计思路必要说明 5](#_Toc135851435)

[3.实验结果 6](#_Toc135851436)

[五.实验体会和收获 7](#_Toc135851437)

[六.程序源代码 7](#_Toc135851438)

|  |  |  |  |
| --- | --- | --- | --- |
| **课程名称** | 程序设计实训 | **课程编号** | **A2131200** |
| **实验地点** | **综合实验大楼B409/410** | **实验时间** | **6月5日至6月9日** |
| **校外指导教师** | **无** | **校内指导教师** | **赵春泽** |
| **实验名称** | **程序设计实训** | | |
| **评阅人签字** |  | **成绩** |  |

# 一.实验任务概述

1.实验一：一群小孩围一圈，任意假定一个数，从第k个小孩起，顺时针方向数，每到第m个小孩时，该小孩便离开。小孩不断离开，圈子不断缩小。最后，剩下的一个小孩便是胜利者。由此条件，解决Josephus问题：最后胜利的是第几个小孩。实验一要求使用数组和链表分别完成代码实现，解决Josephus问题。

2.实验二：与实验一一样，实验二将解决Josephus问题。Josephus游戏从小孩问题中抽象而来，实验二将运用面向对象的程序设计方法，分析Josephus问题中的事物与关系，抽象出对象与联系建立模型，设计出类与关联，完成代码实现。

3.实验三：运用模块化程序设计方法，完成打飞机游戏的代码实现。游戏程序中，用各种符号代表飞机子弹与敌机，显示飞机生命值与得分，飞机撞上敌机生命值减1，减到0游戏结束。玩家可以通过w，a，s，d按键控制玩家飞机上下左右移动，按空格键发射子弹，子弹击中飞机计分。敌机位置随机出现。代码要求有多文件结构，采用面向过程与面向对象设计方法，分别对飞机展开程序框架设计。并作出一定程度上的程序改进。

4.实验四：运用面向对象程序设计方法，抽象出类与类的关系，建立模型，改写实验3中的飞机游戏的代码实现。实现程序的改进，如多子弹多敌机，丰富游戏内容。

# 二.实验准备及方法

1.实验一：对C++理论知识进行大体复习，重新回顾曾经的代码练习与知识点，掌握代码调试的方法。重点复习C++中数组的知识，指针与地址、引用的知识，结构体的知识与链表的知识等等。熟悉面向过程设计思想与方法。需要C++代码的编辑与调试。

2.实验二：复习java的相关理论知识，回顾调试代码的技巧。回顾并运用面向对象的程序设计思想与方法。学会类的抽象的方法以及找到类的关系并学会对其进行代码实现。需要java代码的编辑与调试。

3.实验三：逻辑思维的训练，在面向过程设计的实现中充分理解代码的实现与编写逻辑。继续理解面向过程的设计思想。需要java代码的编辑与调试。

4.实验四：面向对象程序设计思想与方法。深入复习java中类的抽象与方法的编写. 继续理解面向过程的设计思想。能够精准巧妙地进行类的抽象，方法的实现，以及类之间关系的连接。需要java代码的编辑与调试。

# 三.实验设备及软件

Legion Y9000P IAH7H,内存32.0 GB，win11操作系统；软件有Visual Studio Code编辑器，Draw.io Integration作图插件，Violet UML Editor，MinGW-W64 GCC-8.1.0 C++编译器，JDK 8工具包。

# 四.实验步骤、过程原始记录(数据、图表、计算等)

## (一).实验任务完成情况

（1）实验一：完成了基础实验和综合实验Josephus游戏功能面向过程实现；Josephus程序绘制了流程图，见[图1](#图1)；得到胜利小孩结果，见[图4](#图4)，[图5](#图5)；主要实现代码见程序段[表1 Josephus.cpp](#表1),[表2 Josephus2.cpp](#表2)。

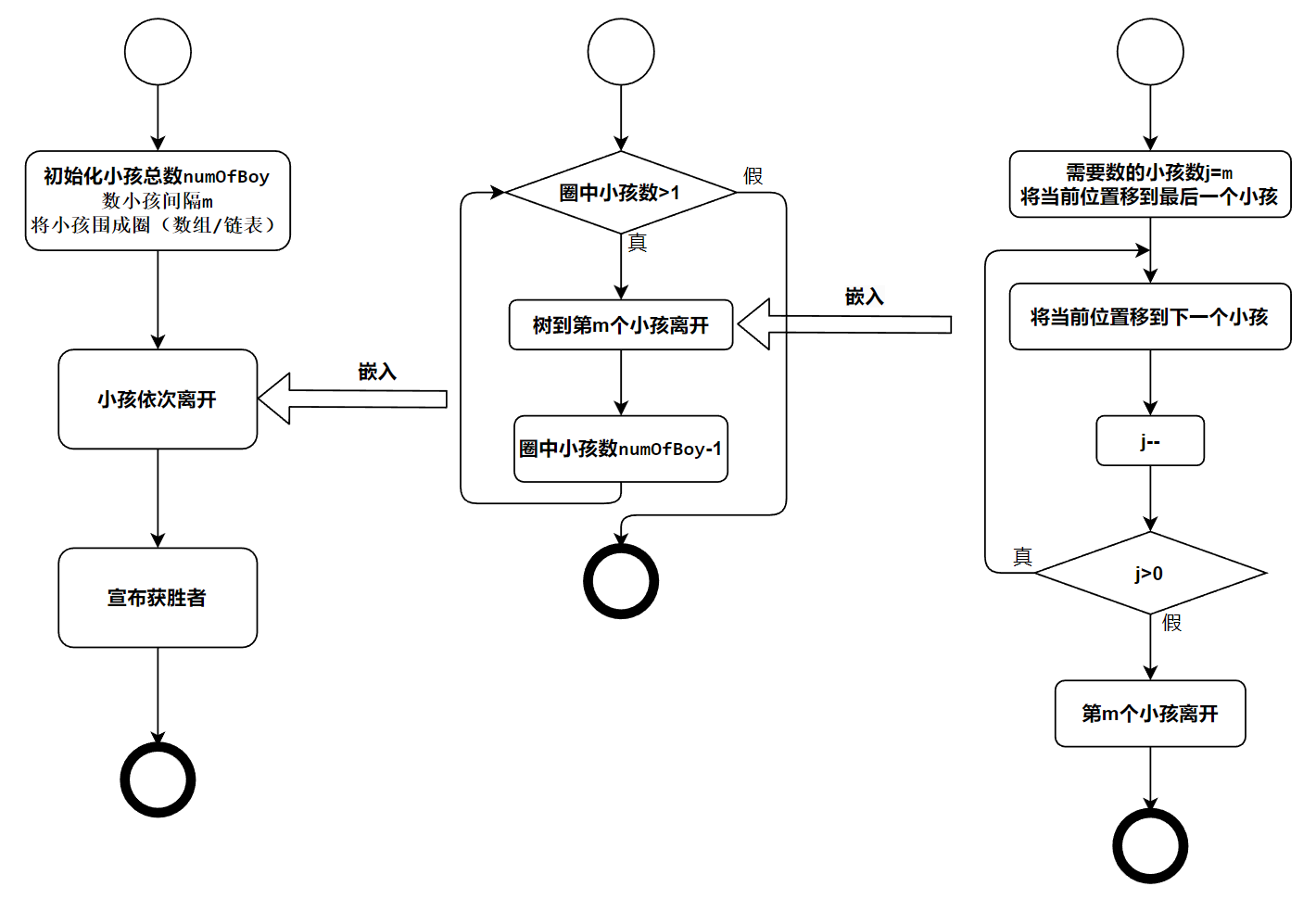
（2）实验二：完成了基础实验和综合实验Josephus游戏功能面向对象实现；Josephus程序绘制了类图，见[图2](#图2)；得到胜利小孩结果，见[图6](#图6)；主要实现代码见程序段[表3 Josephus.java](#表3),[表4 Boy.java](#表4)，[表5 Jose.java](#表5)，[表6 Ring.java](#表6)。

（3）实验三：完成了基础实验和综合实验打飞机游戏功能面向过程实现；得到程序运行结果，见[图7](#图7)；主要实现代码见程序段[表7 Fighter.java](#表7)，[表8 mainFighter.java](#表8)。

（4）实验四：完成了基础实验和综合实验打飞机游戏功能面向对象实现；FlyMe2TheMoon程序绘制了类图，见[图3](#图3)；得到程序运行结果，见[图8](#图8)；主要实现代码见程序段[表9 mainFlyMe2TheMoon.java](#表9)，[表10 Picture.java](#表10)，[表11 Fighter.java](#表11)，[表12 Bullet.java](#表12)，[表13 EnemyFighter.java](#表13)，[表14 Boss.java](#表14)，[表15 Operate.java](#表15)。

## (二).任务成果展示

### 1.流程图/时序图/类图

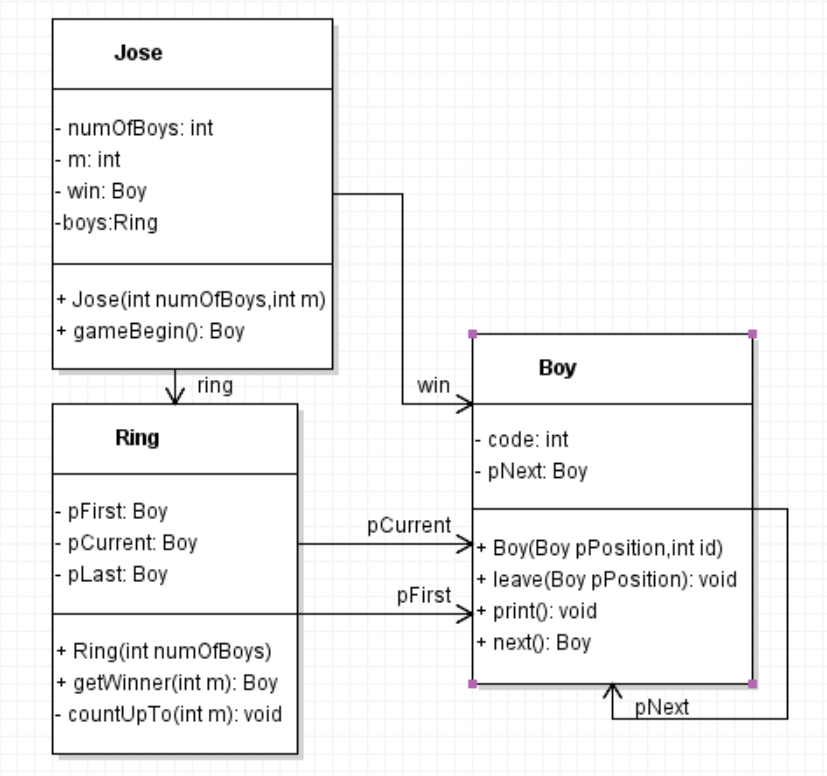
（1）实验一：面向过程实现Josephus问题

Josephus问题面向过程流程图如下。

图1 Josephus问题面向过程流程图

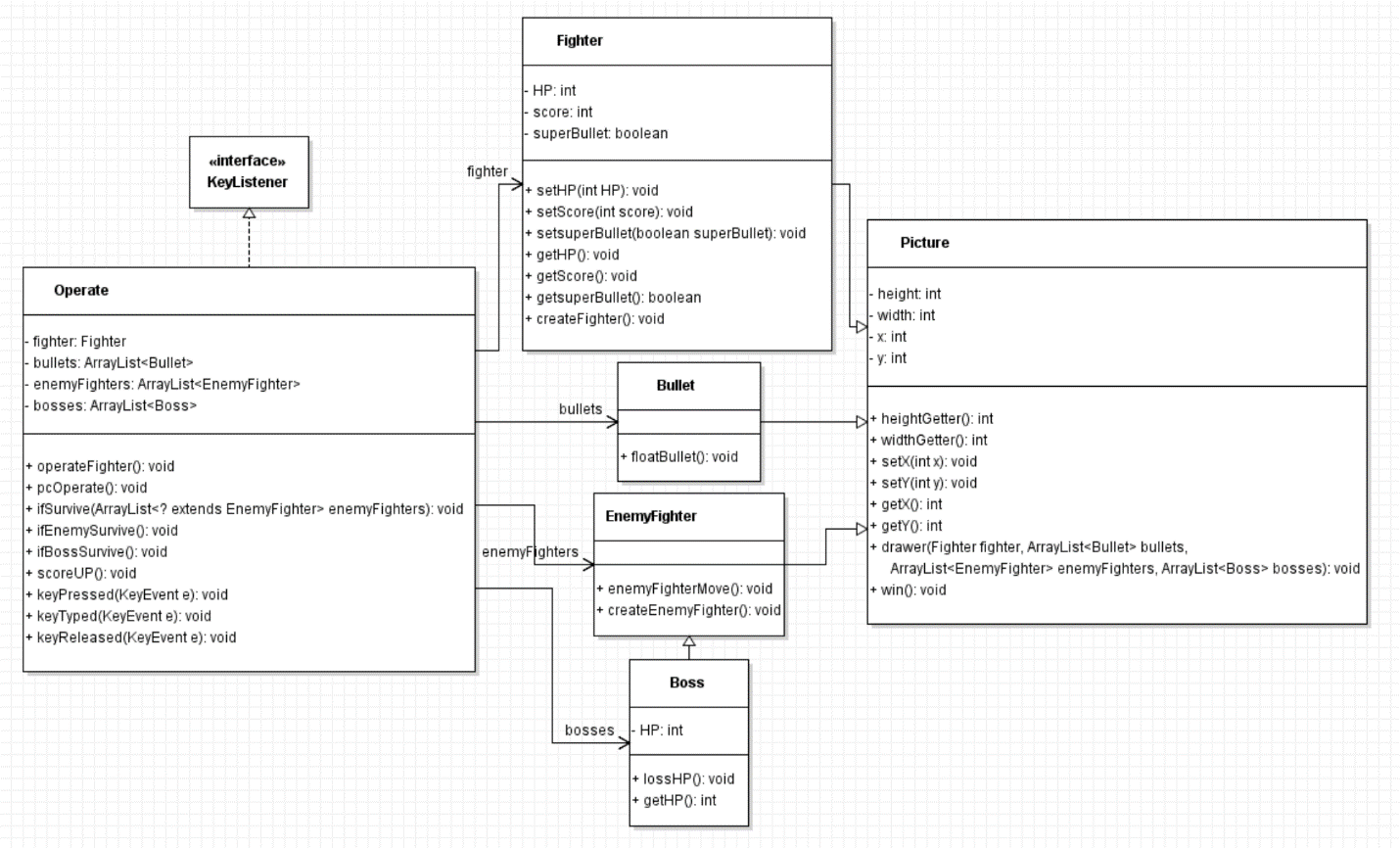
（2）实验二：面向对象实现Josephus问题

Josephus问题面向对象类图如下。

图2 Josephus问题面向对象类图

（4）实验四：面向对象实现打飞机小游戏

打飞机游戏类图如下。

图3 打飞机游戏面向对象类图

### 2.设计思路必要说明

1. 实验一：面向过程实现Josephus问题

初始化全局变量小孩个数为numOfBoys，数小孩间隔为m。如[图1](#图1)将小孩围成圈，可以通过数组或者链表进行实现，然后用一个大循环，判断圈中是否只剩一个小孩，如果只有一个小孩宣布小孩胜利，如果小孩不止1个，则进入小孩离开的循环区块。小孩离开中，先记录要数的小孩个数为j=m;创建小孩指示器指向最后一个小孩（为了第一次数数数到第一个小孩），进入循环体用j计数，循环3遍，每一遍将指示器移动到下一个小孩。循坏结束后，让小孩离开圆圈。记录小孩数-1，再进行大循环的判断，最后小孩只剩一个。

数组实现中，要实现数到最后一个小孩，下一个是第一个小孩，形成闭环，需要小孩指示器i，运行 i = (i + 1) % numOfBoy;取余语句来使指示器回到i=0。用数组存小孩编号和是否在圈内。用boys[i]=i+1来表示 第i+1个小孩，用boys[i]=0来表示第i+1个小孩离开。

链表实现中，要实现小孩围成圈，可以将每个小孩抽象为一个结构体，结构体中存两个属性，小孩编号和下一个小孩的指针。每个小孩依次指向下一个小孩，最后一个小孩指向第一个小孩，实现围成圈。小孩离开，只需要对链表中元素进行重新连接与删除。比如要删除编号为i的小孩，只需要将上一个小孩指向下一个小孩，然后再删除这个小孩的数据就能实现。

1. 实验二：面向对象实现Josephus问题

本实验程序使用java语言编写。运用面向对象的程序设计思想，对问题进行抽象，抽象出Boy、Jose、Ring三个类，如[图2](#图2)。Josephus类实现main方法进入程序读入数据，实例并调用Jose类的gameBegin()开始游戏。Jose类还包含小孩个数，间隔，圆圈类，胜利小孩类。Jose类的实例会在其中实例Ring来连接小孩操作。Ring类用于实例小孩并将他们连接，连接方式类似c++中的链表，在Boy类中包含一个Boy类的属性用于存下一个小孩的地址。Jose类的gameBegin()方法调用Ring类中的getWinner(int m)方法获取胜利小孩。getWinner(int m)传入小孩间隔，利用小孩指示器数小孩，每数m个小孩，将指示器上一个小孩指向下一个小孩，就能实现小孩离开。

1. 实验三：面向过程实现打飞机游戏

本实验采用java语言编写，运用面向过程程序设计思想。游戏命名**FlyMe2TheMoon**。由main方法进入，实例Fighter类并调用方法开始游戏并画出初始图像。

类Fighter中有以下属性，整型height，width表示游戏尺寸；整型fighter\_x, fighter\_y表示飞机位置；ArrayList<Integer> bullets\_x数组表示子弹x坐标；ArrayList<Integer> bullets\_y表示子弹y坐标；ArrayList<Integer> enemys\_x表示敌机x坐标；ArrayList<Integer> enemys\_y表示敌机y坐标；整型score表示分数；整型HP表示血量。实验指导上只提供颗子弹和敌机的坐标信息，于是对其改进扩充，使用二维数组难以进行更自由的增删操作，于是用ArrayList来存储平行数组表示子弹与敌机的坐标。构造器初始化数据score = 0分，HP = 5点生命值，游戏界面尺寸height = 24; width = 30，飞机位置fighter\_x = height - 5; fighter\_y = width / 2。将子弹与敌机数组实例化。

Fighter类中drawPicture方法用于画出游戏界面图像，使用双层for循环遍历x与y坐标循环内部判断这个坐标应当是边框，飞机，子弹，还是敌机，是，则输出对应图像，都不是则输出空格。添加boolean型printed用于判断该位置是否已经输出来取代else if语句从而实现遍历判断多子弹多敌机的输出。循坏之外输出分数和HP。

Fighter类中operateFighter方法用于操控飞机移动。实验指导指出通过键盘输入实现控制飞机且不需要键入回车，java中可通过窗口和键盘监听实现。通过实例JFrame类获得飞机操纵台窗口frame，并进行窗口的一些基本设置。实例JLabel类提供提示“上下左右箭头移动，空格发射子弹”。设置焦点，并将KeyListener键盘监听添加到frame来通过键盘输入来控制飞机。

将键盘操作通过实现接口KeyListener中的keyPressed方法来进行按压键盘按键控制飞机，用keyCode记录按键信息，switch语句判断按下的是上下左右空格或是ESC键，分别执行x、y坐标的变化实现飞机移动；用后面写的createBullet方法给子弹坐标数组“add”数据添加子弹以及退出程序。每次键盘输入后用后面的ifSurvive方法判断飞机是否存活，用drawPicture方法更新图像。keyTyped与keyReleased方法重写但不进行操作。

pcOperate方法用于执行子弹移动于飞机移动操作。指导书中给出添加speed变量记录玩家控制飞机行动次数来更新子弹与敌机移动。这里对这种回合制做出改进，通过实例Timer类timer来更灵活控制游戏更新频率。重写run方法实现每500ms运行代码块：createEnemy();createBullet();floatBullet();ifEnemySurvive();enemyFighterMove();ifSurvive();ifEnemySurvive();来实现创造一个敌机，自动发射子弹，子弹流动，判断敌机存活，敌机移动，判断飞机与敌机是否存活。重写run方法实现每2s调用drawPicture方法更新图像。

floatBullet方法遍历子弹坐标数组使每个子弹x坐标减1实现子弹上移，并判断x坐标是否超出边界，若是，则用remove删除，经过尝试与调查，在带有“:”的遍历语句中remove数组元素会出现错误于是使用普通for循环，且删除元素且索引不为0将索引前移来遍历每一个元素。enemyFighterMove方法类似子弹移动方法实现敌机下移。

ifSurvive方法遍历每一个敌机，判断坐标是否与飞机坐标重合，如果是，删除敌机，将飞机重置到初始位置，生命值HP减1。再判断HP是否为0，如果是则结束游戏。

ifEnemySurvive使用双层循环遍历每个子弹与敌机，判断它们是否重合，如果是，得分score加1；删除对应子弹与敌机。

createEnemy方法实例Random类random。实现在数组中添加一个x坐标为零，y坐标为游戏界面内可击中范围内的随机值的一个敌机。createBullet方法实现在数组中飞机前的位置添加一个子弹。

1. 实验四：面向对象实现打飞机游戏

该实验在实验三基础上进行面向对象的改写与改进。如[图3](#图3)，抽象出Picture, Fighter,Bullet,EnemyFighter,Boss,Operate 6个类。所有类都有其private属性的set与get方法来进行取值和修改。从mainFlyMe2TheMoon类中的main方法进入，实例Operate类初始化游戏数据，调用operateFighter与pcOperate方法启动游戏。

实验四中为改进程序，加入了superBullet得到更大的子弹；Boss类更难破坏的敌机。

Picture类定义整型height, width属性记录游戏尺寸；整型x, y继承给子类记录坐标。构造器为尺寸赋值。drawer方法进行画图操作，与实验三中drawPicture方法类似，加入super子弹装填以及Boss的输出。win方法用于在游戏胜利时输出胜利画面，在Operate中判定并调用。Fighter,Bullet,EnemyFighter,Boss类均为Picture的子类，用于继承x,y坐标的set与get方法。通过Fighter继承的方法进行图像输出。

Fighter类有整型HP,score,superBullet来表示血量，分数与是否装填超级子弹。构造器调用类中createFighter方法调整x,y坐标到界面下方中间位置并设置超级子弹装填；赋值HP=5，score=0。

Bullet类中构造器传入飞机x,y坐标 作为参数，将自己坐标设置在飞机正前方，及发射一个子弹。floatBullet方法控制该子弹x坐标减1及上升一格。

EnemyFighter类中构造器调用自己的createEnemyFighter方法创造一个x坐标为零，y坐标为游戏界面内可击中范围内的随机值的一个敌机。enemyFighterMove方法实现x坐标加1及下降一格。

Boss类继承EnemyFighter类并有自己的整型HP属性。构造器调用父类super()并赋值HP = 10。lossHP方法实现Boss扣一点血。

Operate是打飞机程序的控制程序。属性有Fighter型的fighter飞机；ArrayList数组的bullets子弹，enemyFighters敌机，bosses，首领敌人。构造器中实例飞机类、敌机类数组、Boss类数组与弹类数组并调用fighter.drawer打印初始图像。

operateFighter方法与实验三中大体相同。实例JFrame类获得飞机操纵台窗口。实例JLabel类提供提示“WASD移动，空格发射超级子弹”。将KeyListener键盘监听添加到frame来通过键盘输入来控制飞机。

实现接口KeyListener中的keyPressed方法来实现按压键盘按键控制飞机。用keyCode记录按键信息，switch语句判断按下的是WASD，空格，R或是ESC键，分别执行x、y坐标的变化实现飞机移动；bullets.add发射子弹；以及退出程序。其中按下空格“add”更大更多的子弹；按下R键则发射隐藏的清屏子弹。发射super子弹运行fighter.setsuperBullet(false);冷却超级子弹。手动发射子弹会调用ifEnemySurvive();ifBossSurvive();判断敌机是否存活。每次键入方向按键，调用ifSurvive(enemyFighters); ifSurvive(bosses);判断飞机是否存活。每次键入按键使用fighter.drawer(fighter, bullets, enemyFighters, bosses);更新图像。

pcOperate方法与实验三中类似，用于执行子弹移动于飞机移动操作。实例Timer类控制游戏更新频率。重写run方法实现每200ms更新一次floatBullet子弹移动，bullets.add自动发射子弹，enemyFighters.add添加敌机； 每1s调用enemyFighterMove敌机移动，调用drawPicture方法更新图像；每4s使用fighter.setsuperBullet(true);更新超级子弹，执行bosses.add添加Boss。每次更新子弹移动调用ifEnemySurvive();ifBossSurvive();判断敌机是否存活。每次更新敌机运动都调用ifEnemySurvive();ifSurvive(enemyFighters); ifSurvive(bosses);判断飞机与敌机是否存活。

ifSurvive方法遍历每一个敌机，判断坐标是否与飞机坐标重合，如果是，删除敌机，将飞机重置到初始位置，生命值HP减1。再判断HP是否为0，如果是则结束游戏。

ifEnemySurvive使用双层循环遍历每个子弹与敌机，判断它们是否重合，如果是，得分score加1；删除对应子弹与敌机。

ifBossSurvive使用双层循环遍历每个子弹与Boss，判断是否击中，如果是，得分score加1；删除对应子弹，Boss的HP减1，到0时删除该boss。

### 3.实验结果

1. 实验一：面向过程实现Josephus问题。

使用数组实现：

按[图1](#图1)所示流程，该程序初始化小孩数为7，数小孩间隔为3，输出小孩编号后，循坏进行小孩离开操作，小孩离开中循坏数小孩，每数3次，离开一个小孩，3，6，2，7，5，1号小孩依次离开，最后4号小孩获胜。

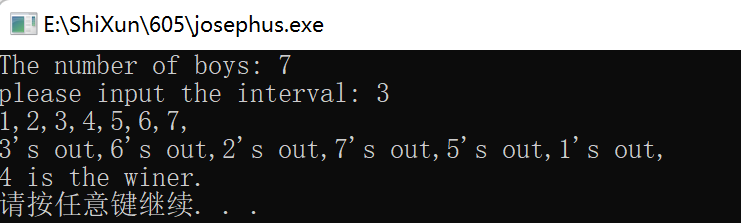
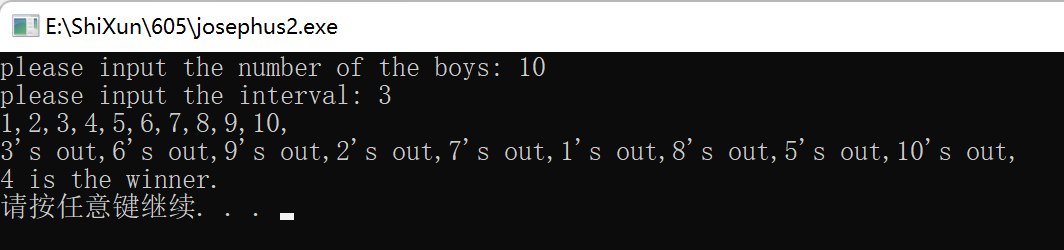
使用数组实现运行结果如下。

图4 Josephus问题数组实现

使用链表实现：

按[图1](#图1)所示流程，该程序初始化小孩数为10，数小孩间隔为3，输出小孩编号后，循坏进行小孩离开操作，小孩离开中循坏数小孩，每数3次，离开一个小孩，3，6，2，7，1，8，5，10号小孩依次离开，最后4号小孩获胜。

使用链表实现运行结果如下。

图5 Josephus问题链表实现

1. 实验二：面向过程实现Josephus问题。

按[图2](#图2)示流程，该程序初始化小孩数为10，实例Ring类来实例10个小孩类围城圈，使用Ring中getWinner()方法获得获胜小孩，方法中循环使用**countUpTo(int m)**方法利用小孩指示器数小孩进行小孩离开操作，3，6，2，7，1，8，5，10号小孩依次离开，最后4号小孩获胜。

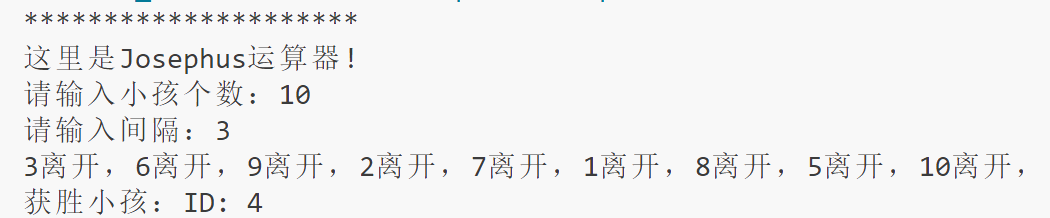
面向对象实现运行结果如下。

图6 Josephus问题面向对象实现

1. 实验三：面向过程实现打飞机游戏

由mainFighter类中的main方法进入，实例Fighter类并调用其中drawPicture方法画出初始图像，调用operateFighter与pcOperate开始游戏。operateFighter实例窗口加入键盘监听，在监听按下按键的重写方法中实现上下左右键对飞机坐标的操控，空格键发射子弹。pcOperate实例Timer并在重写方法run中调用createEnemy，createBullet，floatBullet，enemyFighterMove，ifEnemySurvive，ifSurvive，drawPicture定时实现子弹敌机的创造，移动；得分、生命值的判定；图像刷新。

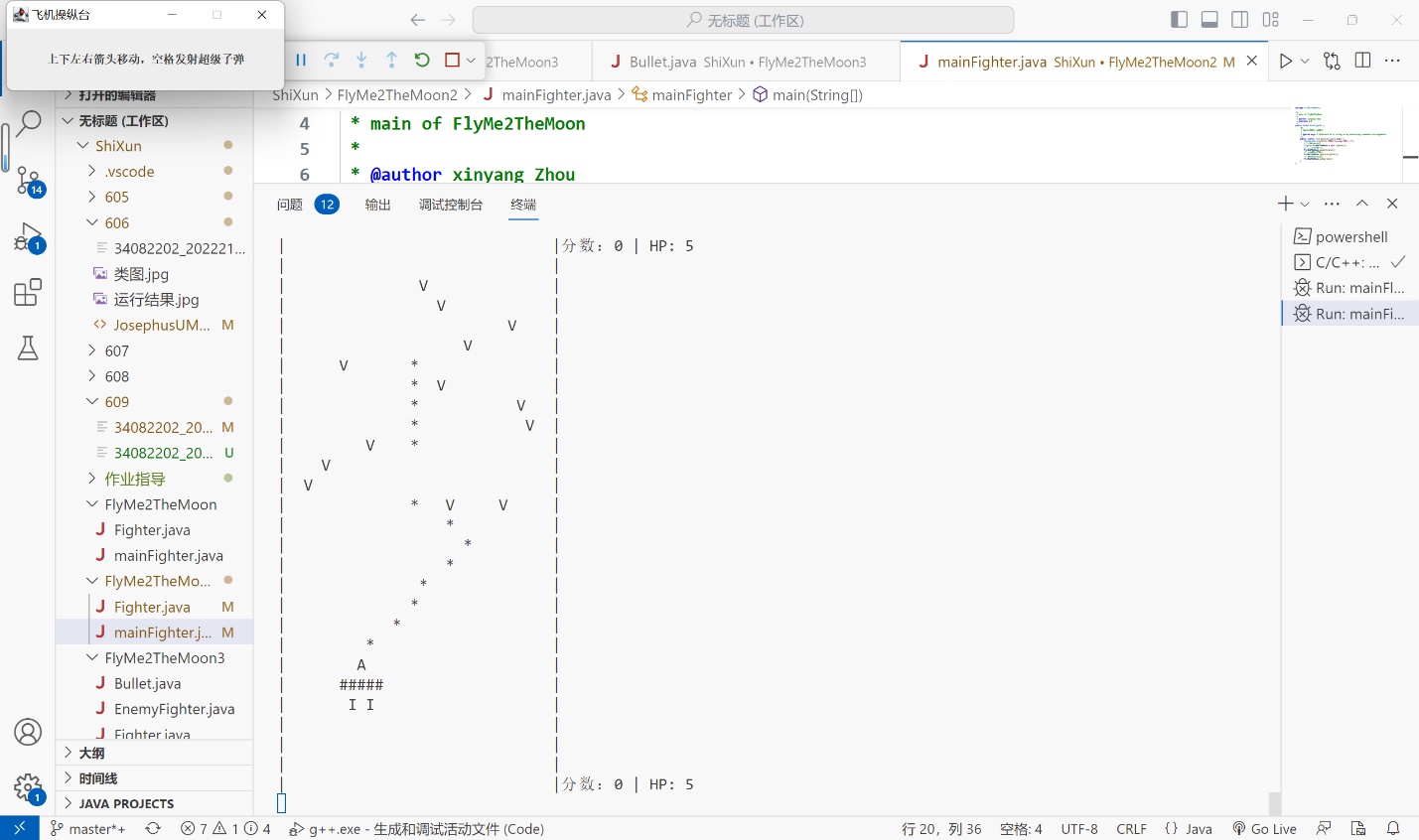
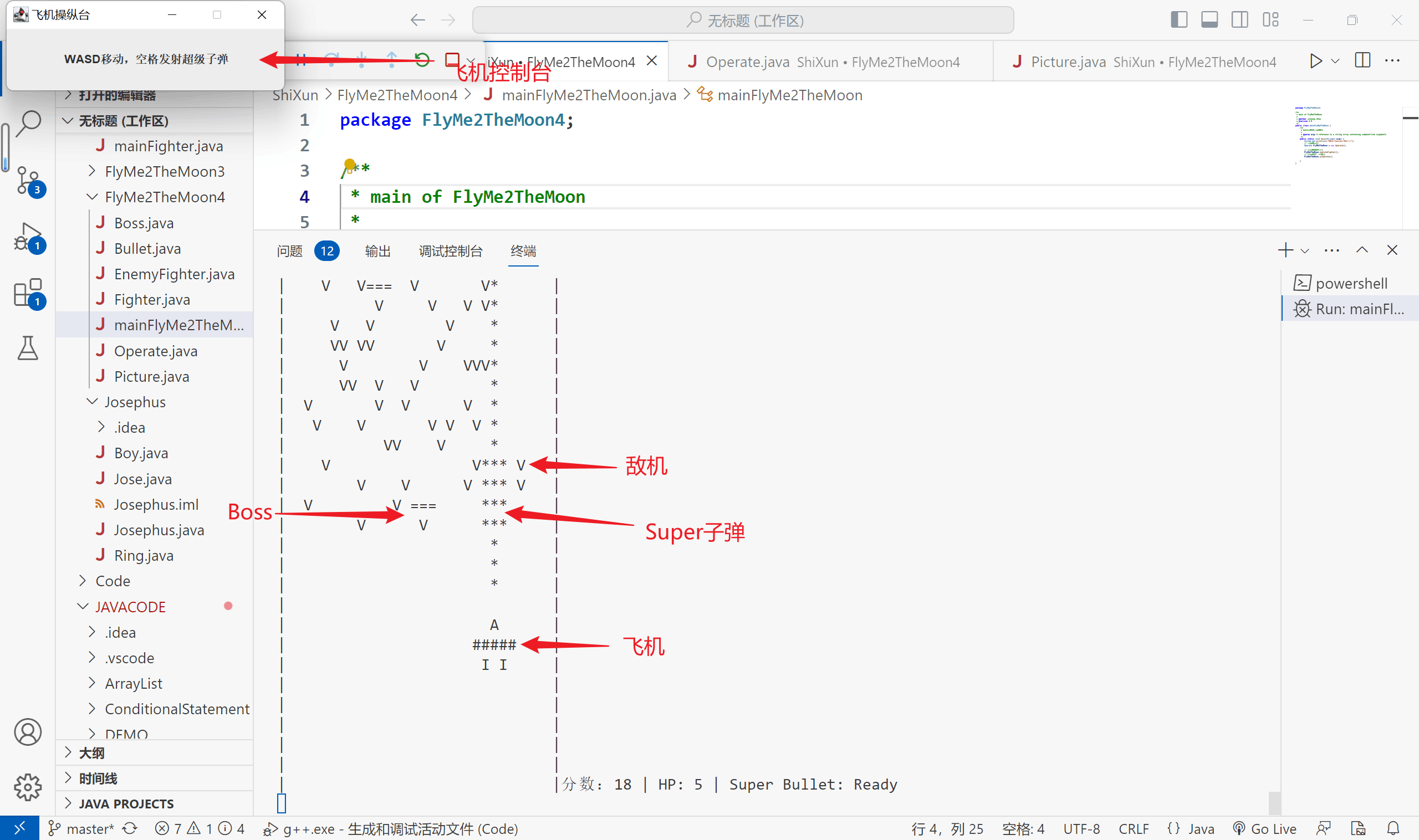
面向过程实现运行结果如下。

图7 面向过程实现打飞机游戏

1. 实验四：面向对象实现打飞机游戏

由mainFlyMe2TheMoon类中的main方法进入程序，实例Operate类初始游戏数据，调用operateFighter与pcOperate方法启动游戏。operateFighter实例窗口加入键盘监听，在监听按下按键的重写方法中实现上下左右键对飞机坐标的操控，空格键发射大子弹。pcOperate实例Timer并在重写方法run中定时实现子弹敌机的创造，移动；得分、生命值的判定；图像刷新。

面向对象实现运行结果如下。

图8 面向对象实现打飞机游戏

# 五.实验体会和收获

在这次的程序设计实训中，我使用了C++和Java语言来分别实现面向过程和面向对象的解决Josephus问题，以及打飞机游戏。

首先，对于C++和Java知识的学习方面，这次实训给了我很好的机会来巩固和应用所学的编程语言知识。通过实际动手编写代码，我深入了解了C++和Java的语法和特性，如变量、函数等。在实现面向过程和面向对象的解决方案时，我学会了如何使用不同的编程风格和设计模式来解决同一个问题。这让我更好地理解了面向过程和面向对象的思维方式，并能够在实际应用中选择合适的方法。

其次，程序调试是在这次实训中非常重要的部分。在编写代码的过程中，我遇到了各种各样的错误，包括语法错误、逻辑错误和算法错误。通过调试这些错误，我学会了如何使用调试工具和技巧来逐步定位和解决问题。这个过程不仅帮助我找到了代码中的错误，还提高了我对程序运行过程的理解。

总的来说，这次程序设计实训让我获益良多。我不仅学到了C++和Java的知识，还提升了编程能力和解决问题的能力。通过实际动手编写代码和调试错误，我深入理解了编程语言和算法的运作原理。这些都是我在未来进行软件开发和程序设计时非常宝贵的经验。

# 六.程序源代码

1. 实验一：Josephus问题

用数组实现源代码：

表1 **Josephus.cpp**

|  |
| --- |
| **package Josephus;**  **import java.util.Scanner;**  **/\*\***  **\* Josephus game**  **\***  **\* @author xinyang Zhou**  **\* @version 1.0**  **\*/**  **public class Josephus {**  **/\*\***  **\* 运行Josephus问题**  **\***  **\* @param args A reference to a string array containing command-line arguments**  **\*/**  **public static void main(String[] args) {**  **Scanner input = new Scanner(System.in);**  **int numOfBoys;// 小孩总数**  **int interval;// 小孩间隔**  **Boy winnerBoy;// 获胜小孩**  **System.out.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");**  **System.out.println("这里是Josephus运算器!");**  **System.out.print("请输入小孩个数：");**  **numOfBoys = input.nextInt();**  **System.out.print("请输入间隔：");**  **interval = input.nextInt();**  **winnerBoy = new Jose(numOfBoys, interval).gameBegin();// 开始运算**  **System.out.print("获胜小孩：");**  **winnerBoy.print();**  **input.close();**  **}**  **}** |

用链表实现：

表2 **Josephus2.cpp**

|  |
| --- |
| **#include <iostream>**  **using namespace std;**  **struct Boy //小孩节点**  **{**  **int code;**  **Boy \*pNext;**  **};**  **Boy \*pFirt = 0;    //第一个小孩指针**  **Boy \*pCurrent = 0; //当前小孩指针**  **Boy \*pivot = 0;    //上一个小孩指针**  **int main()**  **{**  **//输入小孩总数**  **int numOfBoy;**  **cout << "please input the number of the boys: ";**  **cin >> numOfBoy;**  **//输入小孩间隔**  **int m; // Josephus问题中的间隔m**  **cout << "please input the interval: ";**  **cin >> m;**  **// n个小孩围成圈**  **//添加第一个小孩**  **pFirt = new Boy;**  **pFirt->code = 1;**  **pCurrent = pFirt;**  **//添加其他小孩**  **for (int i = 1; i < numOfBoy; i++)**  **{**  **pivot = pCurrent;        //前一个小孩改为"当前的小孩"**  **pCurrent = new Boy;      //在当前的小孩添加一个小孩**  **pCurrent->code = i + 1;  //为加入的小孩编号**  **pivot->pNext = pCurrent; //前一个小孩指向当前的小孩**  **}**  **pCurrent->pNext = pFirt; //最后一个小孩指向第一个，此时pCurrent还在最后一个小孩**  **//输出开始时的小孩编号**  **for (int i = 0; i < numOfBoy; i++)**  **{**  **pCurrent = pCurrent->pNext;**  **cout << pCurrent->code << ",";**  **}**  **cout << endl;**  **//小孩依次离开,，此时pCurrent还在最后一个小孩**  **while (pCurrent->pNext != pCurrent)**  **{**  **int j;**  **j = m; //第m个小孩离开**  **do**  **{**  **//将位置移到下一个小孩**  **pivot = pCurrent;**  **pCurrent = pCurrent->pNext;**  **j--;         //数一个小孩**  **} while (j > 0); //循环m次表示数m个小孩**  **cout << pCurrent->code << "'s out,"; //输出离开小孩编号**  **pivot->pNext = pCurrent->pNext;      //表示该位置小孩离开**  **delete pCurrent;                     //删除这个小孩**  **pCurrent = pivot;                    // pCurrent退回到上一小孩**  **}**  **cout << endl;**  **//宣布获胜者**  **cout << pCurrent->code << " is the winner.";**  **cout << endl;**  **system("pause");**  **}** |

1. 实验二：Josephus游戏

使用面向对象思想实现：

表3 **Josephus.java**

|  |
| --- |
| **package Josephus;**  **import java.util.Scanner;**  **/\*\***  **\* Josephus game**  **\***  **\* @author xinyang Zhou**  **\* @version 1.0**  **\*/**  **public class Josephus {**  **/\*\***  **\* 运行Josephus问题**  **\***  **\* @param args A reference to a string array containing command-line arguments**  **\*/**  **public static void main(String[] args) {**  **Scanner input = new Scanner(System.in);**  **int numOfBoys;// 小孩总数**  **int interval;// 小孩间隔**  **Boy winnerBoy;// 获胜小孩**  **System.out.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");**  **System.out.println("这里是Josephus运算器!");**  **System.out.print("请输入小孩个数：");**  **numOfBoys = input.nextInt();**  **System.out.print("请输入间隔：");**  **interval = input.nextInt();**  **winnerBoy = new Jose(numOfBoys, interval).gameBegin();// 开始运算**  **System.out.print("获胜小孩：");**  **winnerBoy.print();**  **input.close();**  **}**  **}** |

表4 **Boy.java**

|  |
| --- |
| **package Josephus;**  **/\*\***  **\* The Boy class of the Josephus.**  **\***  **\* @author Xinyang Zhou**  **\* @version 1.0**  **\*/**  **public class Boy {**  **// 小孩编号**  **private int code;**  **// 下一个小孩**  **private Boy pNext;**  **/\*\***  **\* 构造小孩**  **\***  **\* @param pPosition 用于连接为上一个小孩**  **\* @param id        小孩编号**  **\*/**  **public Boy(Boy pPosition, int id) {**  **code = id;**  **if (pPosition == null) {**  **this.pNext = this;// 只有一个小孩时，自己指向自己**  **} else {**  **this.pNext = pPosition.pNext;// 插入到小孩pPosition的后面**  **pPosition.pNext = this;**  **}**  **}**  **/\*\***  **\* 小孩离开**  **\***  **\* @param pPosition 离开小孩的上一个**  **\*/**  **public void leave(Boy pPosition) {**  **pPosition.pNext = this.pNext;// 上一个小孩的下一个接为离开小孩的下一个**  **System.out.print(code + "离开，");**  **}**  **/\*\***  **\* 打印小孩ID**  **\*/**  **public void print() {**  **System.out.print("ID: " + code);**  **}**  **/\*\***  **\* 下一个小孩**  **\***  **\* @return 下一个小孩**  **\*/**  **public Boy next() {**  **return pNext;**  **}**  **}** |

表5 **Jose.java**

|  |
| --- |
| **package Josephus;**  **/\*\***  **\* The Jose class of the Josephus.**  **\***  **\* @author Xinyang Zhou**  **\* @version 1.0**  **\*/**  **public class Jose {**  **// 小孩总数**  **private int numOfBoys;**  **// 小孩间隔**  **private int m;**  **// 胜利的小孩**  **private Boy win;**  **// 小孩圆圈**  **private Ring boys;**  **/\*\***  **\* josephus问题的数据**  **\***  **\* @param numOfBoys 小孩总数**  **\* @param interval  间隔数**  **\*/**  **public Jose(int numOfBoys, int interval) {**  **this.numOfBoys = numOfBoys;**  **this.m = interval;**  **this.boys = new Ring(this.numOfBoys);// 小孩围一圈**  **}**  **/\*\***  **\* 游戏开始**  **\***  **\* @return 获胜小孩**  **\*/**  **public Boy gameBegin() {**  **win = boys.getWinner(m);**  **return win;**  **}**  **}** |

表6 **Ring.java**

|  |
| --- |
| **package Josephus;**  **/\*\***  **\* The Ring class of the Josephus.完成围小孩，数小孩，得到获胜小孩。**  **\***  **\* @author XinYang Zhou**  **\* @version 1.0**  **\*/**  **public class Ring {**  **// 第一个小孩**  **private Boy pFirst;**  **// 现在的小孩**  **private Boy pCurrent;**  **// 上一个小孩**  **private Boy pLast;**  **/\*\***  **\* 小孩围城圈**  **\***  **\* @param numOfBoys 小孩总数**  **\*/**  **public Ring(int numOfBoys) {**  **pFirst = new Boy(null, 1);// 第一个小孩**  **// 依次围上其他小孩**  **Boy pNewBoy = pFirst;**  **for (int i = 2; i <= numOfBoys; i++) {**  **pNewBoy = new Boy(pNewBoy, i);**  **}**  **// 将pCurrent移至最后一个小孩使得第一次数数pCurrent移到第一个**  **pCurrent = pNewBoy;**  **}**  **/\*\***  **\* 得到获胜小孩**  **\***  **\* @param m 间隔数**  **\* @return 获胜小孩**  **\*/**  **public Boy getWinner(int m) {**  **// 数小孩, 直到只剩1个小孩**  **while (pCurrent != pCurrent.next()) {**  **// 每次数小孩的间隔为m。**  **countUpTo(m);**  **}**  **System.out.println();**  **// 返回获胜者**  **return pCurrent;**  **}**  **/\*\***  **\* 往下数m个小孩，数到的小孩离开。**  **\***  **\* @param m 间隔数**  **\*/**  **private void countUpTo(int m) {**  **// 往下数m个小孩**  **for (int i = 0; i < m; i++) {**  **pLast = pCurrent;**  **pCurrent = pCurrent.next();**  **}**  **// 数到的小孩离开，同时pCurrent回到上一个小孩**  **pCurrent.leave(pLast);**  **pCurrent = pLast;**  **}**  **}** |

1. 实验三：面向过程实现打飞机游戏

下面是程序代码：

表7 **Fighter.java**

|  |
| --- |
| **package FlyMe2TheMoon2;**  **import java.util.ArrayList;**  **import java.util.Random;**  **import javax.swing.JFrame;**  **import javax.swing.JLabel;**  **import java.awt.event.KeyEvent;**  **import java.awt.event.KeyListener;**  **import java.util.Timer;**  **import java.util.TimerTask;**  **/\*\***  **\* The Fighter class of the FlyMe2TheMoon.**  **\***  **\* @author Xinyang Zhou**  **\* @version 2.0**  **\*/**  **public class Fighter implements KeyListener {**  **private int height, width;// 游戏界面尺寸**  **private int fighter\_x, fighter\_y;// 飞机位置**  **private ArrayList<Integer> bullets\_x;**  **private ArrayList<Integer> bullets\_y;**  **private ArrayList<Integer> enemys\_x;**  **private ArrayList<Integer> enemys\_y;**  **private int score;// 分数**  **private int HP;// 生命值**  **/\*\***  **\* 构造打飞机游戏类**  **\*/**  **public Fighter() {**  **score = 0;// 0分**  **HP = 5;// 5点生命值**  **// 游戏界面尺寸**  **height = 24;**  **width = 30;**  **// 飞机位置**  **fighter\_x = height - 5;**  **fighter\_y = width / 2;**  **// 实例子弹数组类**  **bullets\_x = new ArrayList<Integer>();**  **bullets\_y = new ArrayList<Integer>();**  **bullets\_x.add(fighter\_x - 2);**  **bullets\_y.add(fighter\_y);**  **// 创建一个敌机**  **enemys\_x = new ArrayList<Integer>();**  **enemys\_y = new ArrayList<Integer>();**  **createEnemy();**  **}**  **/\*\***  **\* 画游戏图像**  **\*/**  **public void drawPicture() {**  **for (int x = 0; x < height + 2; x++) {**  **for (int y = 0; y < width + 2; y++) {**  **boolean printed = false;**  **// 输出边框**  **if (y == 0 || y == width + 1) {**  **System.out.print("|");**  **printed = true;**  **}**  **// 画出飞机**  **if (!printed && x == fighter\_x**  **&& (y >= fighter\_y - 2 && y <= fighter\_y + 2)) {**  **System.out.print("#");**  **printed = true;**  **} else if (!printed && x == fighter\_x - 1 && y == fighter\_y) {**  **System.out.print("A");**  **printed = true;**  **} else if (!printed && x == fighter\_x + 1**  **&& (y == fighter\_y - 1 || y == fighter\_y + 1)) {**  **System.out.print("I");**  **printed = true;**  **}**  **// 画出子弹**  **for (int i = 0; i < bullets\_x.size(); i++) {**  **if (!printed && x == bullets\_x.get(i) && y == bullets\_y.get(i)) {**  **System.out.print("\*");**  **printed = true;**  **break;**  **}**  **}**  **// 画出敌机**  **for (int i = 0; i < enemys\_x.size(); i++) {**  **if (!printed && x == enemys\_x.get(i) && y == enemys\_y.get(i)) {**  **System.out.print("V");**  **printed = true;**  **break;**  **}**  **}**  **// 输出空格**  **if (!printed) {**  **System.out.print(" ");**  **}**  **}**  **System.out.println();**  **}**  **System.out.println(**  **"|                              |分数：" + score + " | HP: " + HP);**  **}**  **/\*\***  **\* 控制飞机移动**  **\*/**  **public void operateFighter() {**  **// 创建一个 飞机操纵台**  **JFrame frame = new JFrame("飞机操纵台");**  **frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);**  **frame.setSize(300, 100);// 大小**  **frame.setVisible(true);// 可见**  **frame.setResizable(false);// 不可改变大小**  **JLabel tips = new JLabel("上下左右箭头移动，空格发射子弹", null, 0);// 提示**  **frame.add(tips);**  **frame.setFocusable(true);**  **frame.requestFocusInWindow();// 将焦点设置到 frame 上**  **frame.addKeyListener(this);// 添加 KeyListener 到 frame,keyPressed实现飞机运动**  **}**  **@Override**  **public void keyPressed(KeyEvent e) {**  **// 当按键被按下时触发**  **// 处理按键事件**  **int keyCode = e.getKeyCode();**  **switch (keyCode) {**  **case 38: {// 上键**  **if (fighter\_x > 2) {**  **fighter\_x--;**  **}**  **break;**  **}**  **case 37: {// 左键**  **if (fighter\_y > 3) {**  **fighter\_y--;**  **}**  **break;**  **}**  **case 40: {// 下键**  **if (fighter\_x < height - 1) {**  **fighter\_x++;**  **}**  **break;**  **}**  **case 39: {// 右键**  **if (fighter\_y < width - 2) {**  **fighter\_y++;**  **}**  **break;**  **}**  **case 32: {// " "**  **bullets\_x.add(fighter\_x - 1);**  **bullets\_y.add(fighter\_y);**  **break;**  **}**  **case 27: {**  **System.out.println("游戏结束！");**  **System.exit(0); // 按下 ESC 键退出程序**  **break;**  **}**  **}**  **ifSurvive();// 飞机是否存活**  **drawPicture();**  **}**  **/\*\***  **\* 子弹与飞机移动**  **\*/**  **public void pcOperate() {**  **Timer timer = new Timer();**  **timer.scheduleAtFixedRate(new TimerTask() {**  **@Override**  **public void run() {**  **createEnemy();// 创造敌机**  **createBullet();// 创造子弹**  **floatBullet();// 子弹移动**  **ifEnemySurvive();// 敌机是否存活**  **enemyFighterMove();// 敌机移动**  **ifSurvive();// 飞机是否存活**  **ifEnemySurvive();// 敌机是否存活**  **}**  **}, 0, 500);**  **timer.scheduleAtFixedRate(new TimerTask() {// 更新图像**  **@Override**  **public void run() {**  **drawPicture();**  **}**  **}, 0, 2000);**  **}**  **/\*\***  **\* 子弹移动**  **\*/**  **public void floatBullet() {**  **for (int i = 0; i < bullets\_x.size(); i++) {**  **bullets\_x.set(i, bullets\_x.get(i) - 1);**  **if (bullets\_x.get(i) < 0) {**  **bullets\_x.remove(i);// 移除子弹**  **bullets\_y.remove(i);**  **if (i != 0) {**  **i--;**  **}**  **}**  **}**  **}**  **/\*\***  **\* 敌机移动**  **\*/**  **public void enemyFighterMove() {**  **for (int i = 0; i < enemys\_x.size(); i++) {**  **enemys\_x.set(i, enemys\_x.get(i) + 1);**  **if (enemys\_x.get(i) > height) {**  **enemys\_x.remove(i);// 移除敌机**  **enemys\_y.remove(i);**  **if (i != 0) {**  **i--;**  **}**  **}**  **}**  **}**  **/\*\***  **\* 判断飞机是否存活**  **\*/**  **public void ifSurvive() {**  **// 飞机被击中**  **for (int i = 0; i < enemys\_x.size(); i++) {**  **if ((enemys\_x.get(i) == fighter\_x - 1 && enemys\_y.get(i) == fighter\_y)**  **|| (enemys\_x.get(i) == fighter\_x && enemys\_x.get(i) > fighter\_y - 3**  **&& enemys\_x.get(i) < fighter\_y + 3)) {**  **enemys\_x.remove(i);// 移除飞机**  **enemys\_y.remove(i);**  **if (i != 0) {**  **i--;**  **}**  **// 刷新飞机位置**  **fighter\_x = height - 5;**  **fighter\_y = width / 2;**  **HP--;// 减少生命值**  **if (HP == 0) {**  **System.out.println("\*\*\*飞机被摧毁，游戏结束！\*\*\*");**  **System.exit(0);**  **}**  **}**  **}**  **}**  **/\*\***  **\* 判断敌机是否存活**  **\*/**  **public void ifEnemySurvive() {**  **for (int i = 0; i < enemys\_x.size(); i++) {**  **for (int j = 0; j < bullets\_x.size(); j++) {**  **if (bullets\_x.get(j) == enemys\_x.get(i) && bullets\_y.get(j) == enemys\_y.get(i)) {**  **score++;// 加分**  **bullets\_x.remove(j);// 移除子弹**  **bullets\_y.remove(j);**  **if (j != 0) {**  **j--;**  **}**  **enemys\_x.remove(i);// 移除飞机**  **enemys\_y.remove(i);**  **if (i != 0) {**  **i--;**  **}**  **}**  **}**  **}**  **}**  **/\*\***  **\* 创造敌机**  **\*/**  **public void createEnemy() {**  **Random random = new Random();**  **enemys\_x.add(0);**  **enemys\_y.add(random.nextInt(width - 4) + 3);**  **}**  **/\*\***  **\* 创造子弹**  **\*/**  **public void createBullet() {**  **bullets\_x.add(fighter\_x - 1);**  **bullets\_y.add(fighter\_y);**  **}**  **@Override**  **public void keyTyped(KeyEvent e) {**  **}**  **@Override**  **public void keyReleased(KeyEvent e) {**  **}**  **}** |

表8 **mainFighter.java**

|  |
| --- |
| **package FlyMe2TheMoon2;**  **/\*\***  **\* main of FlyMe2TheMoon**  **\***  **\* @author xinyang Zhou**  **\* @version 2.0**  **\*/**  **public class mainFighter {**  **/\*\***  **\* main函数（主模块）**  **\***  **\* @param args A reference to a string array containing command-line arguments**  **\*/**  **public static void main(String[] args) {**  **System.out.println("这里是打飞机小游戏~~");**  **// 实例打飞机类**  **Fighter FlyMe2TheMoon = new Fighter();**  **// 画出初始的图像**  **FlyMe2TheMoon.drawPicture();**  **// 飞机操作台运作**  **FlyMe2TheMoon.operateFighter();**  **// 子弹与飞机移动**  **FlyMe2TheMoon.pcOperate();**  **}**  **}** |

1. 实验四：面向对象实现打飞机游戏

下面是程序代码：

表9 **mainFlyMe2TheMoon.java**

|  |
| --- |
| **package FlyMe2TheMoon4;**  **/\*\***  **\* main of FlyMe2TheMoon**  **\***  **\* @author xinyang Zhou**  **\* @version 4.0**  **\*/**  **public class mainFlyMe2TheMoon {**  **/\*\***  **\* main函数（主模块）**  **\***  **\* @param args A reference to a string array containing command-line arguments**  **\*/**  **public static void main(String[] args) {**  **System.out.println("这里是打飞机小游戏~~");**  **// 实例操作类**  **Operate FlyMe2TheMoon = new Operate();**  **// 飞机操纵台启动**  **FlyMe2TheMoon.operateFighter();**  **// 敌机子弹开始运动**  **FlyMe2TheMoon.pcOperate();**  **}**  **}** |

表10 **Picture.java**

|  |
| --- |
| **package FlyMe2TheMoon4;**  **import java.util.ArrayList;**  **/\*\***  **\* The Picture class of the FlyMe2TheMoon.**  **\***  **\* @author Xinyang Zhou**  **\* @version 4.0**  **\*/**  **public class Picture {**  **private int height, width;// 游戏界面尺寸**  **private int x, y;// 界面坐标，继承给子类用于定位图像**  **/\*\***  **\* 构造游戏图像框架数据**  **\*/**  **public Picture() {**  **// 游戏界面尺寸**  **height = 24;**  **width = 30;**  **}**  **/\*\***  **\* @return 游戏高**  **\*/**  **public int heightGetter() {**  **return height;**  **}**  **/\*\***  **\* @return 游戏宽**  **\*/**  **public int widthGetter() {**  **return width;**  **}**  **/\*\***  **\* 设置x坐标**  **\***  **\* @param x x坐标**  **\*/**  **public void setX(int x) {**  **this.x = x;**  **}**  **/\*\***  **\* 设置y坐标**  **\***  **\* @param y y坐标**  **\*/**  **public void setY(int y) {**  **this.y = y;**  **}**  **/\*\***  **\* @return x坐标**  **\*/**  **public int getX() {**  **return x;**  **}**  **/\*\***  **\* @return y坐标**  **\*/**  **public int getY() {**  **return y;**  **}**  **/\*\***  **\* 画图像**  **\***  **\* @param fighter       飞机**  **\* @param bullets       子弹**  **\* @param enemyFighters 敌机**  **\* @param score         分数**  **\*/**  **public void drawer(Fighter fighter, ArrayList<Bullet> bullets,**  **ArrayList<EnemyFighter> enemyFighters, ArrayList<Boss> bosses) {**  **String SuperBullet;**  **if (fighter.getsuperBullet()) {**  **SuperBullet = "Ready";// 超级子弹已装填**  **} else {**  **SuperBullet = "Not Ready";**  **}**  **for (int x = 0; x < height + 2; x++) {**  **for (int y = 0; y < width + 2; y++) {**  **boolean printed = false;**  **// 输出边框**  **if (y == 0 || y == width + 1) {**  **System.out.print("|");**  **printed = true;**  **}**  **// 画出飞机**  **if (!printed && x == fighter.getX()**  **&& (y >= fighter.getY() - 2 && y <= fighter.getY() + 2)) {**  **System.out.print("#");**  **printed = true;**  **} else if (!printed && x == fighter.getX() - 1 && y == fighter.getY()) {**  **System.out.print("A");**  **printed = true;**  **} else if (!printed && x == fighter.getX() + 1**  **&& (y == fighter.getY() - 1 || y == fighter.getY() + 1)) {**  **System.out.print("I");**  **printed = true;**  **}**  **// 画出Boss**  **for (Boss boss : bosses) {**  **if (!printed && x == boss.getX() && y == boss.getY()) {**  **System.out.print("V");**  **printed = true;**  **break;**  **} else if (!printed && x == boss.getX() - 1 && y > boss.getY() - 2 && y < boss.getY() + 2) {**  **System.out.print("=");**  **printed = true;**  **break;**  **}**  **}**  **// 画出子弹**  **for (Bullet bullet : bullets) {**  **if (!printed && x == bullet.getX() && y == bullet.getY()) {**  **System.out.print("\*");**  **printed = true;**  **break;**  **}**  **}**  **// 画出敌机**  **for (EnemyFighter enemyFighter : enemyFighters) {**  **if (!printed && x == enemyFighter.getX() && y == enemyFighter.getY()) {**  **System.out.print("V");**  **printed = true;**  **break;**  **}**  **}**  **// 输出空格**  **if (!printed) {**  **System.out.print(" ");**  **}**  **}**  **System.out.println();**  **}**  **System.out.println(**  **"|                              |分数：" + fighter.getScore() + " | HP: " + fighter.getHP()**  **+ " | Super Bullet: " + SuperBullet);**  **}**  **public void win() {**  **System.out.println("            \*\*\*\*\*\*\*            ");**  **System.out.println("         \*\*\*\*\*\*\*\*\*\*\*\*\*         ");**  **System.out.println("      \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*      ");**  **System.out.println("     \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*     ");**  **System.out.println("    \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*    ");**  **System.out.println("    Fly Me 2 The Moon ~~~~~    ");**  **System.out.println("    \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*    ");**  **System.out.println("     \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*     ");**  **System.out.println("      \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*      ");**  **System.out.println("         \*\*\*\*\*\*\*\*\*\*\*\*\*         ");**  **System.out.println("            \*\*\*\*\*\*\*            ");**  **System.exit(0);**  **}**  **}** |

表11 **Fighter.java**

|  |
| --- |
| **package FlyMe2TheMoon4;**  **/\*\***  **\* The Fighter class of the FlyMe2TheMoon.**  **\***  **\* @author Xinyang Zhou**  **\* @version 4.0**  **\*/**  **public class Fighter extends Picture {**  **private int HP;// 飞机生命值**  **private int score;// 分数**  **private boolean superBullet;// 超级子弹是否装填**  **/\*\***  **\* 构造Fighter**  **\*/**  **public Fighter() {**  **createFighter();**  **HP = 5;// 初始生命值为5**  **score = 0;// 初始分数为0**  **}**  **/\*\***  **\* 设置飞机HP**  **\***  **\* @param HP 飞机HP**  **\*/**  **public void setHP(int HP) {**  **this.HP = HP;**  **}**  **/\*\***  **\* 设置分数**  **\***  **\* @param score 分数**  **\*/**  **public void setScore(int score) {**  **this.score = score;**  **}**  **/\*\***  **\* 设置超级子弹**  **\***  **\* @param superBullet 导弹是否装填**  **\*/**  **public void setsuperBullet(boolean superBullet) {**  **this.superBullet = superBullet;**  **}**  **/\*\***  **\* @return 飞机HP**  **\*/**  **public int getHP() {**  **return HP;**  **}**  **/\*\***  **\* @return 得分**  **\*/**  **public int getScore() {**  **return score;**  **}**  **/\*\***  **\* @return 是否装填super子弹**  **\*/**  **public boolean getsuperBullet() {**  **return superBullet;**  **}**  **/\*\***  **\* 创造飞机（刷新飞机位置）**  **\*/**  **public void createFighter() {**  **// 飞机位置**  **super.setX(super.heightGetter() - 5);**  **super.setY(super.widthGetter() / 2);**  **superBullet = true;// 装填超级子弹**  **}**  **}** |

表12 **Bullet.java**

|  |
| --- |
| **package FlyMe2TheMoon4;**  **/\*\***  **\* The Bullet class of the FlyMe2TheMoon.**  **\***  **\* @author Xinyang Zhou**  **\* @version 4.0**  **\*/**  **public class Bullet extends Picture {**  **/\*\***  **\* 子弹构造器**  **\*/**  **public Bullet(int fighter\_x, int fighter\_y) {**  **// 创造一个子弹**  **setX(fighter\_x - 2);**  **setY(fighter\_y);**  **}**  **/\*\***  **\* 子弹移动**  **\*/**  **public void floatBullet() {**  **setX(getX() - 1);// 子弹移动**  **}**  **}** |

表13 **EnemyFighter.java**

|  |
| --- |
| **package FlyMe2TheMoon4;**  **import java.util.Random;**  **/\*\***  **\* The EnemyFighter class of the FlyMe2TheMoon.**  **\***  **\* @author Xinyang Zhou**  **\* @version 4.0**  **\*/**  **public class EnemyFighter extends Picture {**  **/\*\***  **\* 敌机构造器**  **\*/**  **public EnemyFighter() {**  **createEnemyFighter();**  **}**  **public void enemyFighterMove() {**  **setX(getX() + 1); // 敌机移动**  **}**  **/\*\***  **\* 创造敌机（刷新敌机位置）**  **\*/**  **public void createEnemyFighter() {**  **Random random = new Random();**  **// 产生新敌机**  **setX(0);**  **setY((random.nextInt(super.widthGetter() - 4) + 3));// 宽度限制敌机出现位置**  **}**  **}** |

表14 **Boss.java**

|  |
| --- |
| **package FlyMe2TheMoon4;**  **/\*\***  **\* The Boss class of the FlyMe2TheMoon.**  **\***  **\* @author Xinyang Zhou**  **\* @version 4.0**  **\*/**  **public class Boss extends EnemyFighter {**  **private int HP;**  **/\*\***  **\* 创造boss（刷新敌机位置）**  **\*/**  **public Boss() {**  **super();**  **HP = 10;**  **}**  **/\*\***  **\* 扣血**  **\*/**  **public void lossHP() {**  **HP--;**  **}**  **/\*\***  **\* @return BossHP**  **\*/**  **public int getHP() {**  **return HP;**  **}**  **}** |

表15 **Operate.java**

|  |
| --- |
| **package FlyMe2TheMoon4;**  **import java.util.ArrayList;**  **import javax.swing.JFrame;**  **import javax.swing.JLabel;**  **import java.awt.event.KeyEvent;**  **import java.awt.event.KeyListener;**  **import java.util.Timer;**  **import java.util.TimerTask;**  **/\*\***  **\* The Operate class of the FlyMe2TheMoon.（控制程序）**  **\***  **\* @author Xinyang Zhou**  **\* @version 4.0**  **\*/**  **public class Operate implements KeyListener {**  **private Fighter fighter;// 飞机**  **private ArrayList<Bullet> bullets;// 子弹**  **private ArrayList<EnemyFighter> enemyFighters;// 敌机**  **private ArrayList<Boss> bosses;// Boss**  **public Operate() {**  **fighter = new Fighter();// 实例飞机类**  **enemyFighters = new ArrayList<EnemyFighter>();// 实例敌机类数组**  **bosses = new ArrayList<Boss>();// 实例Boss类数组**  **bullets = new ArrayList<Bullet>();// 实例子弹类数组**  **fighter.drawer(fighter, bullets, enemyFighters, bosses); // 打印初始图像**  **}**  **/\*\***  **\* 控制飞机移动**  **\*/**  **public void operateFighter() {**  **// 创建一个 飞机操纵台**  **JFrame frame = new JFrame("飞机操纵台");**  **frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);**  **frame.setSize(300, 100);// 大小**  **frame.setVisible(true);// 可见**  **frame.setResizable(false);// 不可改变大小**  **JLabel tips = new JLabel("WASD移动，空格发射超级子弹", null, 0);// 提示**  **frame.add(tips);**  **frame.setFocusable(true);**  **frame.requestFocusInWindow();// 将焦点设置到 frame 上**  **frame.addKeyListener(this);// 添加 KeyListener 到 frame,keyPressed实现飞机运动**  **}**  **@Override**  **public void keyPressed(KeyEvent e) {**  **// 当按键被按下时触发, 处理按键事件,操纵飞机**  **int keyCode = e.getKeyCode();**  **switch (keyCode) {**  **case 87: {// W 上**  **if (fighter.getX() > 2) {**  **fighter.setX(fighter.getX() - 1);**  **ifSurvive(enemyFighters);// 飞机是否存活**  **ifSurvive(bosses);**  **}**  **break;**  **}**  **case 65: {// A 左**  **if (fighter.getY() > 3) {**  **fighter.setY(fighter.getY() - 1);**  **ifSurvive(enemyFighters);// 飞机是否存活**  **ifSurvive(bosses);**  **}**  **break;**  **}**  **case 83: {// S 下**  **if (fighter.getX() < fighter.heightGetter() - 1) {**  **fighter.setX(fighter.getX() + 1);**  **ifSurvive(enemyFighters);// 飞机是否存活**  **ifSurvive(bosses);**  **}**  **break;**  **}**  **case 68: {// D 右**  **if (fighter.getY() < fighter.widthGetter() - 2) {**  **fighter.setY(fighter.getY() + 1);**  **ifSurvive(enemyFighters);// 飞机是否存活**  **ifSurvive(bosses);**  **}**  **break;**  **}**  **case 32: {// " " 空格发射super子弹**  **if (fighter.getsuperBullet()) {**  **for (int x = 0; x < 4; x++) {**  **for (int y = 0; y < 3; y++) {**  **bullets.add(new Bullet(fighter.getX() - x, fighter.getY() + y - 1));**  **}**  **}**  **fighter.setsuperBullet(false);// super子弹进入冷却**  **ifEnemySurvive();// 敌机是否存活**  **ifBossSurvive();// Boss是否存活**  **}**  **break;**  **}**  **case 82: {// R 发射清屏子弹**  **for (int x = fighter.getX(); x > fighter.getX() - 4; x--) {**  **for (int y = 1; y < fighter.widthGetter() + 1; y++) {**  **bullets.add(new Bullet(x, y));**  **}**  **}**  **ifEnemySurvive();// 敌机是否存活**  **ifBossSurvive();// Boss是否存活**  **break;**  **}**  **case 27: {// ESC 退出程序**  **System.out.println("游戏结束！");**  **System.exit(0);**  **break;**  **}**  **}**  **fighter.drawer(fighter, bullets, enemyFighters, bosses); // 更新图像**  **}**  **public void pcOperate() {**  **Timer timer = new Timer();**  **timer.scheduleAtFixedRate(new TimerTask() {// 每200ms更新一次子弹移动并自动发射子弹，添加敌机**  **@Override**  **public void run() {**  **bullets.add(new Bullet(fighter.getX(), fighter.getY())); // 自动发射子弹**  **enemyFighters.add(new EnemyFighter());// 自动添加敌机**  **for (int i = 0; i < bullets.size(); i++) {**  **bullets.get(i).floatBullet(); // 子弹移动**  **if (bullets.get(i).getX() == -1) {**  **bullets.remove(bullets.get(i));// 移除出界子弹**  **if (i != 0)// 回正数组索引**  **i--;**  **}**  **}**  **ifEnemySurvive();// 敌机是否存活**  **ifBossSurvive();// Boss是否存活**  **}**  **}, 0, 200);**  **timer.scheduleAtFixedRate(new TimerTask() {// 每1s更新一次敌机运动**  **@Override**  **public void run() {**  **for (int i = 0; i < enemyFighters.size(); i++) {**  **enemyFighters.get(i).enemyFighterMove(); // 敌机移动**  **if (enemyFighters.get(i).getX() > fighter.heightGetter()) {**  **enemyFighters.remove(enemyFighters.get(i));// 移除出界飞机**  **if (i != 0)// 回正数组索引**  **i--;**  **}**  **}**  **for (int i = 0; i < bosses.size(); i++) {**  **bosses.get(i).enemyFighterMove(); // 敌机移动**  **if (bosses.get(i).getX() > fighter.heightGetter()) {**  **bosses.remove(bosses.get(i));// 移除出界飞机**  **if (i != 0)// 回正数组索引**  **i--;**  **}**  **}**  **ifEnemySurvive();// 敌机是否存活**  **ifSurvive(enemyFighters);// 飞机是否存活**  **ifSurvive(bosses);// Boss是否存活**  **fighter.drawer(fighter, bullets, enemyFighters, bosses);// 更新图像**  **}**  **}, 0, 1000);**  **timer.scheduleAtFixedRate(new TimerTask() {// 每4s刷新super子弹并添加敌机**  **@Override**  **public void run() {**  **fighter.setsuperBullet(true);**  **bosses.add(new Boss());**  **}**  **}, 0, 4000);**  **}**  **/\*\***  **\* 判断飞机是否存活**  **\*/**  **public void ifSurvive(ArrayList<? extends EnemyFighter> enemyFighters) {**  **for (int i = 0; i < enemyFighters.size(); i++) {**  **if ((enemyFighters.get(i).getX() == fighter.getX() - 1**  **&& enemyFighters.get(i).getY() == fighter.getY())**  **|| (enemyFighters.get(i).getX() == fighter.getX()**  **&& enemyFighters.get(i).getY() > fighter.getY() - 3**  **&& enemyFighters.get(i).getY() < fighter.getY() + 3)) { // 飞机撞上敌机**  **enemyFighters.remove(enemyFighters.get(i)); // 删除敌机**  **fighter.createFighter(); // 刷新飞机位置**  **fighter.setHP(fighter.getHP() - 1);// 减少生命值**  **if (fighter.getHP() == 0) {**  **System.out.println("\*\*\*飞机被摧毁，游戏结束！\*\*\*");**  **System.exit(0);**  **}**  **break;**  **}**  **}**  **}**  **/\*\***  **\* 判断敌机是否存活**  **\*/**  **public void ifEnemySurvive() {**  **for (int i = 0; i < bullets.size(); i++) {**  **for (int j = 0; j < enemyFighters.size(); j++) {**  **if (bullets.get(i).getX() == enemyFighters.get(j).getX()**  **&& bullets.get(i).getY() == enemyFighters.get(j).getY()) {// 击中敌机**  **scoreUP();// 加分**  **bullets.remove(i);// 删除子弹**  **enemyFighters.remove(enemyFighters.get(j)); // 删除敌机**  **// 回正数组索引**  **if (i != 0)**  **i--;**  **if (j != 0)**  **j--;**  **}**  **}**  **}**  **}**  **/\*\***  **\* 判断Boss是否存活**  **\*/**  **public void ifBossSurvive() {**  **for (int i = 0; i < bullets.size(); i++) {**  **for (int j = 0; j < bosses.size(); j++) {**  **if (bullets.get(i).getX() == bosses.get(j).getX() && (bullets.get(i).getY() > bosses.get(j).getY() - 2**  **&& bullets.get(i).getY() < bosses.get(j).getY() + 2)) {// 击中Boss**  **scoreUP();// 加分**  **bosses.get(j).lossHP();// Boss扣血**  **if (bosses.get(j).getHP() == 0) {**  **bosses.remove(j); // 删除Boss**  **if (j != 0) // 回正数组索引**  **j--;**  **}**  **bullets.remove(i);// 删除子弹**  **if (i != 0) // 回正数组索引**  **i--;**  **}**  **}**  **}**  **}**  **public void scoreUP() {**  **fighter.setScore(fighter.getScore() + 1);// 加分**  **if (fighter.getScore() > 1995)**  **fighter.win();// 胜利**  **}**  **@Override**  **public void keyTyped(KeyEvent e) {**  **}**  **@Override**  **public void keyReleased(KeyEvent e) {**  **}**  **}** |