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

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

// Tips：描述实验前自己为本次实验做了哪些准备，主要包括 C++ **理论知识的复习和巩固，实验所需 C++代码的编辑和调试等。

//分别描述实验 1、2、3、4 的实验准备及方法

1.实验一：对C++理论知识进行了大体的复习，重新回顾了曾经的代码练习与知识点，掌握代码调试的方法。重点复习了 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。再次熟悉了面向过程设计思想与方法。

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。

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。

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。

三.实验设备及软件

// Tips：描述此次实验中所使用的实验设备，以及相关的软件。如：XXX 型号 PC 机，内存 XXX 兆，win10 操作系统；软件有 Visual studio 2013 集成开发环境，Visual C++ 10.0 工具套件。

Legion Y9000P IAH7H,内存 32.0 GB，win11 操作系统；软件有 Visual Studio Code,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)

// Tips：依据实验要求的内容，主要描述自己在实验过程中的实验步骤、主要实验测试数据以及期待结果。

(一).实验任务完成情况

完成了基础实验和综合实验**功能；**程序绘制了流程图/类图/时序图，见图**；得到***结果，见图**；主要实现代码见程序段***

((完成了那些写那些))

//分别描述实验 1、2、3、4 的实验任务完成

（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；得到胜利小孩结果，见图，图；主要实现代码见程序段表 1

Josephus.cpp，表 2 Josephus2.cpp。

（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 到胜利小孩结果，见图，图；主要实现代码见程序段表 3 Josephus.java 表 4 Boy.java，表 5 Jose.java，表 6 Ring.java。

（3）实验三：XXXXX

（4）实验四：XXXXX

(二). 任务成果展示

1.流程图/时序图/类图

//分别描述实验 1、2、3、4 的实验流程图/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。

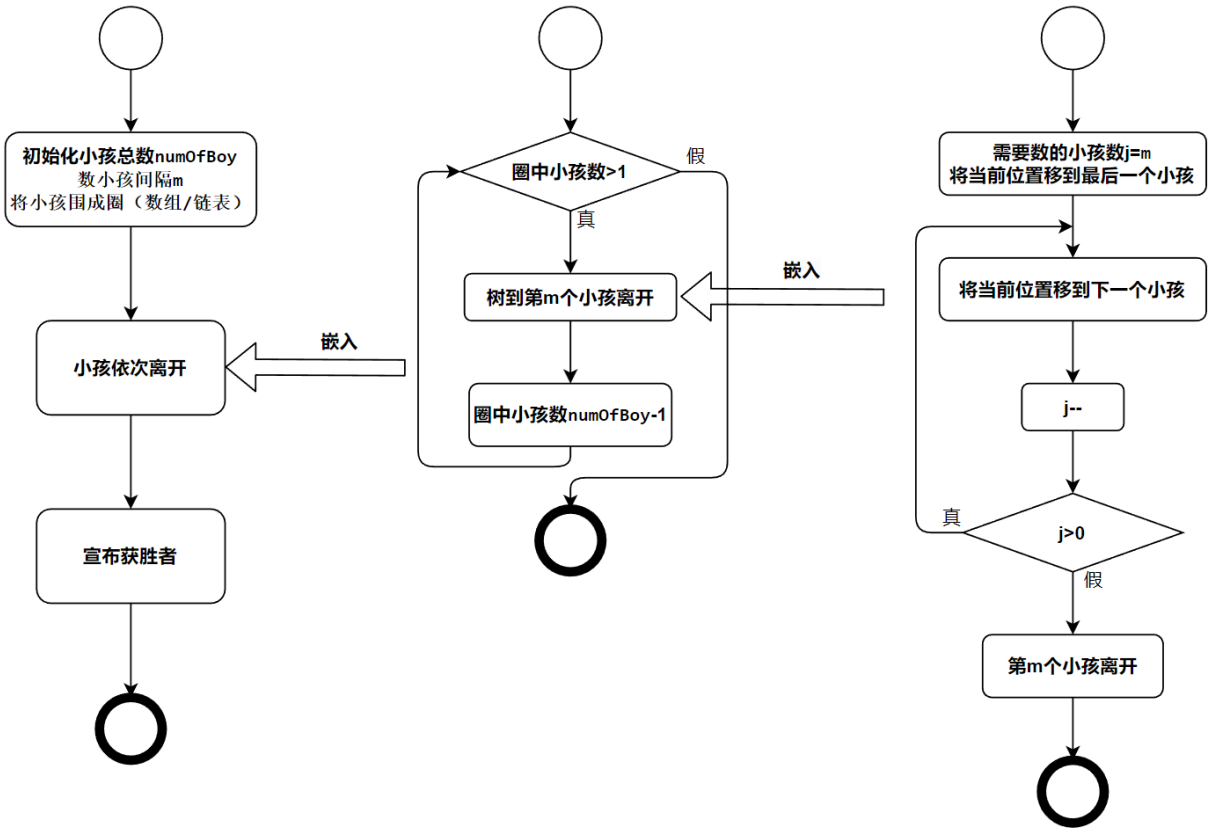

图 1 Josephus 问题面向过程流程图

（2）实验二：面向对象实现 Josephus 问题

Josephus 问题面向对象类图如下。

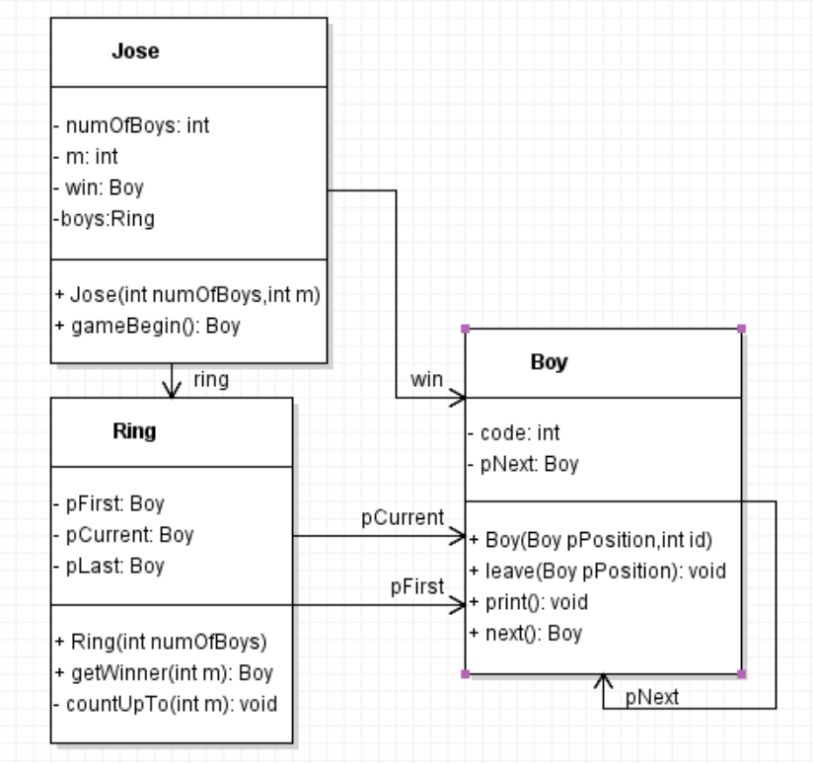

图 2 Josephus 问题面向对象类图

（3）实验三：XXXXX

（4）实验四：XXXXX

2.设计思路必要说明

//分别描述实验 1、2、3、4 的实验设计思路

（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 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。

（2） 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。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。

（3） 实验三：XXXXX

（4）

3.实验结果

// Tips：描述此次实验的结果，注意同实验要求进行比较，检查是否完全完成了实验要求的所有内容。可以将实验结果通过贴图的方式进行表述

如：

说明：按图 1 所示流程，该程序先初始化累加合 sum 的值为 0，循环计算第 n 项的值保存在 v 中，并加到累加和 sum 中，直到第 n 项的绝对值() <0.00000001 跳出循环，最后求出 4 与 sum 的乘积及得到π的值。

实现效果如图 2 所示。

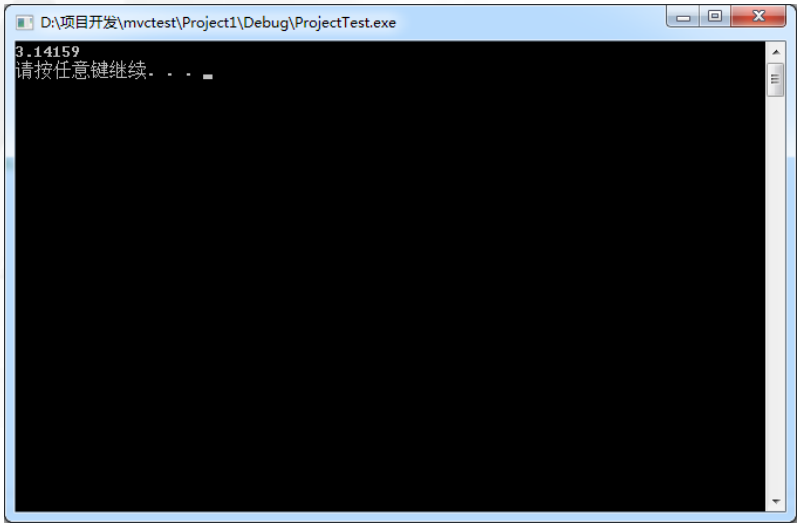

图 2 求π效果图

//分别描述实验 1、2、3、4 的实验结果

（1） 实验一：面向过程实现 Josephus 问题

使用数组实现：

按图 1 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。

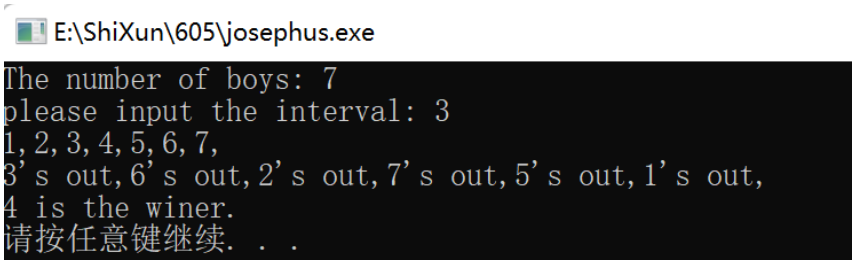

图 Josephus 问题数组实现

使用链表实现：

按图 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。

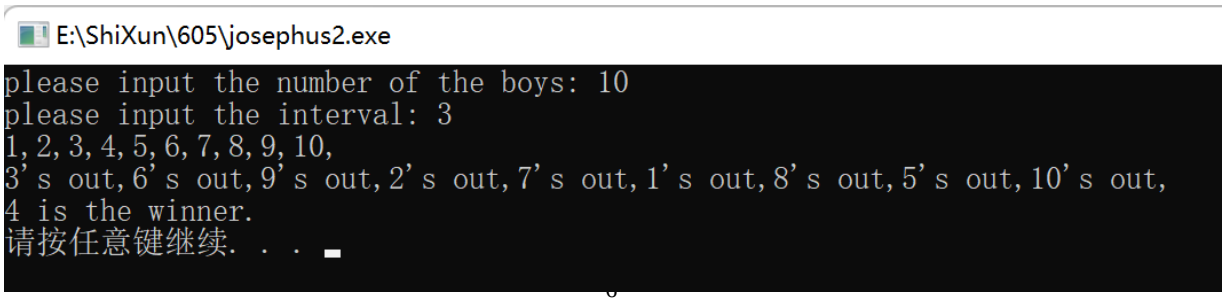

图 Josephus 问题链表实现

（2） 实验二：面向过程实现 Josephus 问题

按图 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。

```
**********
这里是Josephus运算器!
请输入小孩个数：10
请输入间隔：3
3离开，6离开，9离开，2离开，7离开，1离开，8离开，5离开，10离开，
获胜小孩：ID：4
```

图 Josephus 问题面向对象实现

（3） 实验三：XXXXX

（4） 实验四：XXXXX

五.实验体会和收获

//Tips：描述此次实验的体会、感受和收获，如对于 C++知识的学习、程序调试、测试数据构造等方面的感受等。

六.程序源代码

//Tips：将本次实验的所有程序源代码粘贴在此处。（应有必要注释）

//分别描述实验 1、2、3、4 的实验源代码

1. 实验一：Josephus 问题

用数组实现源代码：

表 1 Josephus.cpp

```
package Josephus;
```

```java
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

```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");
}
```

2. 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;
    }
}
```