《Java and DotNet》

实习报告二

学　　号：20161000327

班级序号：114161-03

姓　　名：范鑫

指导教师：杨之江

中国地质大学信息工程学院空间信息系

2018 年 11 月 3 日

题目一　　约瑟夫环

1．需求规格说明

【问题描述】

求解约瑟夫环，用数组实现。

【基本要求】

每个 player 包括姓名，序号等信息。

2．总体分析与设计

（1）设计思想：

约瑟夫环（约瑟夫问题）是一个数学的应用问题：已知 n 个人（以编号 1，2，3...n 分别表示）围坐在一张圆桌周围。从编号为 k 的人开始报数，数到 m 的那个人出列；他的下一个人又从 1 开始报数，数到 m 的那个人又出列；依此规律重复下去，直到圆桌周围的人全部出列。通常解决这类问题时我们把编号从 0~n-1，最后结果+1 即为原问题的解。

（2）算法原理：

a. 一群人（如：N）围在一起坐成环状；

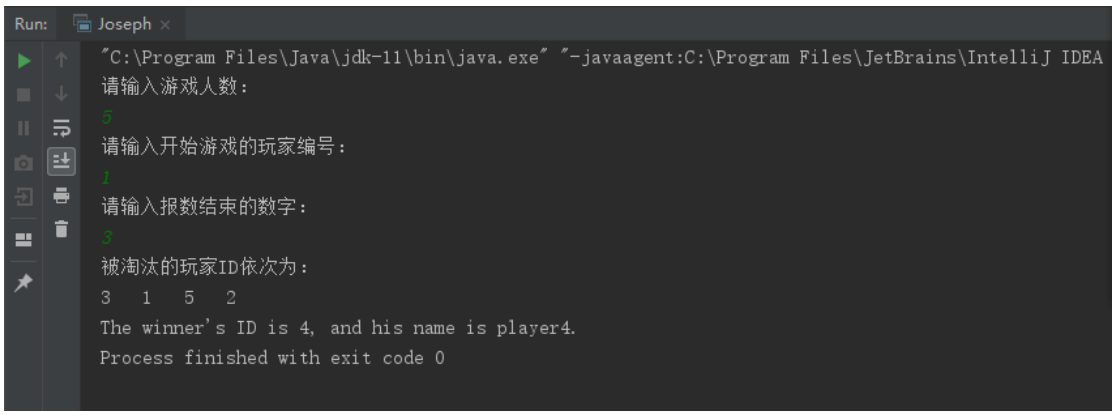
b. 从某个编号（如：K）开始报数；

c. 数到某个数（如：M）的时候，此人出列，下一个人重新报数；

d. 一直循环，直到所有人出列，约瑟夫环结束。

3．结果

程序运行结果如下



4．小结

约瑟夫环问题对我们来说并不陌生，第一次接触是大一 C++期末课设的时候，当时觉得还挺难的。然后就是大二的数据结构课上老师讲了使用**循环队列**的方法求解约瑟夫环。而本次 Java 实习使用数组求解约瑟夫环的目的是让我们掌握 Java 中数组的使用。

实习过程中也遇到了一些小问题，如空指针异常**(NullPointerException)**，最后通过查阅相关书籍和网上的资料都成功解决了。自己也对 Java 数组有了更深入的理解，也认识到了实践的重要性。因为很多问题你不动手去做，是很难发现的。

5．附录

源代码如下

```
import java.util.Scanner;

class player
{
    int ID;
    String name;
}

public class Joseph
{
    public static void main(String[] args)
    {
        Scanner in=new Scanner(System.in);
        System.out.println("请输入游戏人数：");
        int N=in.nextInt();
        System.out.println("请输入开始游戏的玩家编号：");
        int S=in.nextInt();
        System.out.println("请输入报数结束的数字：");
        int D=in.nextInt();
        int n=N;    //剩余人数
        player[] a=new player[N];
        for(int i=0;i<a.length;i++)
        {
            a[i]=new player();  //初始化，注意：没有此步程序运行会报空指
针异常错误
            a[i].ID=i+1;
            a[i].name="player"+Integer.toString(i+1);
        }
        int key=(S+D-2)%n;

        System.out.println("被淘汰的玩家 ID 依次为：");

        for(int i=1;i<=N-1;i++)
        {
            System.out.print(a[key].ID+"\t");
            if(key<n-1)
            {
                for(int j=key;j<=n-2;j++)
                {
                    a[j]=a[j+1];    //数组元素前移覆盖
                }
            }
            n--;
            key=(key+D-1)%n;
```

```
        }
        System.out.println();
        System.out.printf("The winner's ID is %d, and his name
is %s.",a[0].ID,a[0].name);
    }
}
```

题目二　　学习使用反射

1. 需求规格说明

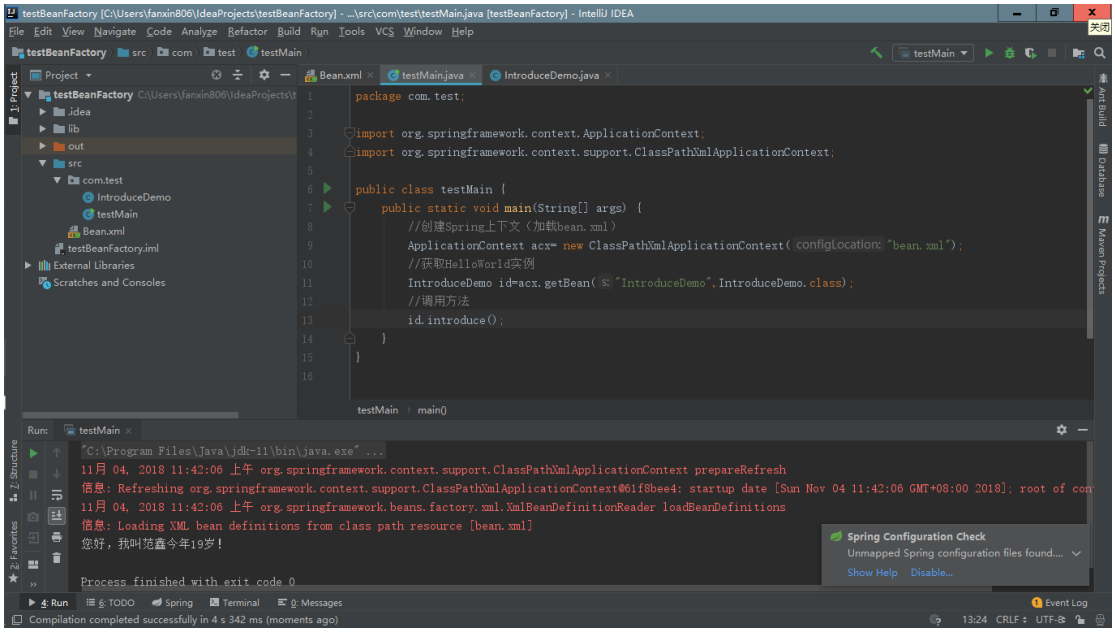
创建 beanFactory（使用 Spring 框架或自己写 BeanFactory 类）
用户用法：
BeanFactory.getBean(“bean1”)
BeanFactory.Invoke(“Class1.Test”)

2. 总体分析与设计

反射机制允许程序在运行时检查任意对象的内容，并调用它们的任意方法。JavaBeans 类提供了一个无参的构造函数、若干 getter/setter 方法对及一些其他方法。BeanFactory 是 Spring 的“心脏”，是 Spring 框架最核心的接口， 定义了 IoC 的基本功能，主要定义了 getBean 方法。可以实现控制反转/依赖注入，控制反转意思就是说，当我们调用一个方法或者类时，不再有我们主动去创建这个类的对象，控制权交给别人（spring）。依赖注入意思就是说，spring 主动创建被调用类的对象，然后把这个对象注入到我们自己的类中，使得我们可以使用它。

3. 结果

简单示例运行结果如下



4. 小结

Spring 是一个开源的轻量级 Java 开发框架，它使用基本的 JavaBean 来完成以前只可能由 EJB 完成的事情。通过本次实习我对 Spring 和 Java 的反射机制有了一定的认识，掌握了如何使用 Intellij IDEA 建立一个简单的 Spring 项目。后续有时间的话，会深入学习 Spring 框架。它在 Java 企业级的应用中非常广泛，这对以后工作也是很有价值的。

5. 附录

Beans.xml 配置文件

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans.xsd">
    <bean id="IntroduceDemo" class="com.test.IntroduceDemo">
    <property name="name" value="范鑫"/>
    <property name="age" value="19"/>
    </bean>
</beans>
```

Person 类

```
package com.test;

public class IntroduceDemo {
    //姓名
    private String name;
    //年龄
    private  int  age;

    public int getAge() {
        return age;
    }

    public void setAge(int age) {
        this.age = age;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

```

```
    /**
     * 自我介绍
     */
    public void introduce(){
        System.out.println("您好，我叫"+this.name+"今年"+this.age+"岁！
");
    }
}
```

test 类

```
package com.test;

import org.springframework.context.ApplicationContext;
import
org.springframework.context.support.ClassPathXmlApplicationContext;

public class testMain {
    public static void main(String[] args) {
        //创建 Spring 上下文（加载 bean.xml）
        ApplicationContext acx= new
ClassPathXmlApplicationContext("bean.xml");
        //获取 HelloWorld 实例
        IntroduceDemo
id=acx.getBean("IntroduceDemo",IntroduceDemo.class);
        //调用方法
        id.introduce();
    }
}
```