实验报告

实验名称	实验 4　异常与多线程			实验时间	第 11 周		

实验环境

Eclipse/Jcreator Pro/JDK。

实验目的和内容要求

一、实验目的

1. 掌握异常的概念、异常的处理方法；异常类的应用。

2. 掌握线程的概念、线程的生命周期；多线程的编程；使用多线程机制实现动画。

二、实验要求

1.提供所完成的各道题主要实验代码和运行结果的界面截图。

2.简述在实验过程中遇到的问题与解决方法。

3.简述实验过程中的发现与收获，未解决或需进一步解决的问题。

三、实验内容

1. 自定义非法年龄类 IllegalAgeException，定义一个 Person 类，包含年龄，姓名，性别等属性，编写属性设置和读取函数，在设置年龄函数中，判断参数是否符合要求（1~150），如果不符合则抛出异常，编写 main 函数进行测试。

2. 用继承 Thread 类的方法实现一个多线程程序，该程序先后启动三个线程，每个线程首先打印出一条线程创建信息，然后休眠一个随机时间，最后打印出线程结束信息退出。

3. 在一个线程中求 100 以内素数，求出后休眠一个随机时间。在另一线程中求 100 以内能被 3 整除的数，求出后休眠一个随机时间。输出数据时应有提示，指明哪个线程输出的数据。

实验过程、结果分析与总结

题目 1

源代码：

Person.java

```
public class Person {
    private int age;
    private String name;
    private String sex;

    public Person() {
    }

    public Person(int age, String name, String sex) {
```

```
        this.age = age;
        this.name = name;
        this.sex = sex;
    }

    /**
     * 获取
     * @return age
     */
    public int getAge() {
        return age;
    }

    /**
     * 设置
     * @param age
     */
    public void setAge(int age) {
        if (age < 1 || age > 150) {
            throw new IllegalAgeException("年龄应设置为1-150之间,而" + age + "
超出了范围。");
        } else {
            this.age = age;
        }
    }

    /**
     * 获取
     * @return name
     */
    public String getName() {
        return name;
    }

    /**
     * 设置
     * @param name
     */
    public void setName(String name) {
        this.name = name;
    }

    /**
     * 获取
     * @return sex
     */
    public String getSex() {
```

```
        return sex;
    }

    /**
     * 设置
     * @param sex
     */
    public void setSex(String sex) {
        this.sex = sex;
    }

    public String toString() {
        return "Person{age = " + age + ", name = " + name + ", sex = " + sex + "}";
    }
}
```

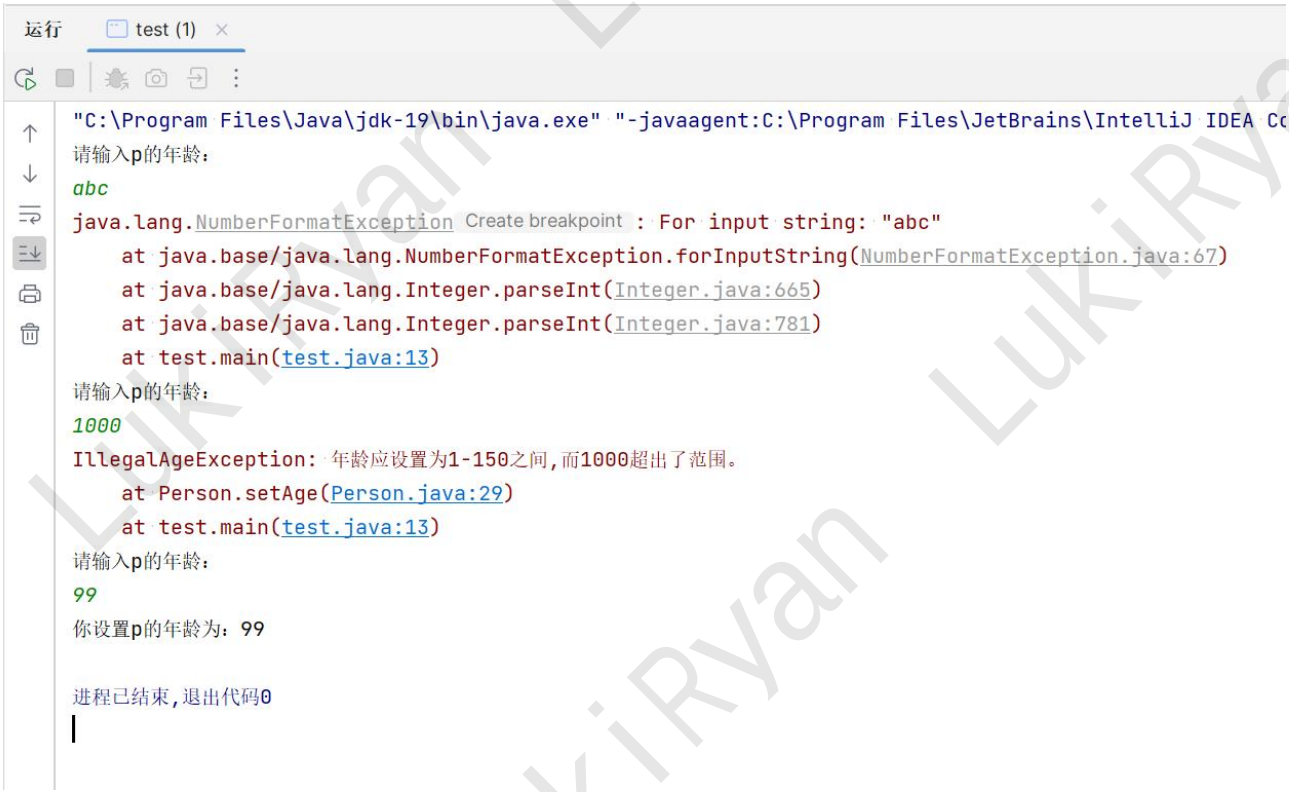
IllegalAgeException.java

```
public class IllegalAgeException extends RuntimeException {
    public IllegalAgeException() {
    }

    public IllegalAgeException(String message) {
        super(message);
    }
}
```

test.java

```
import java.util.Scanner;

public class test {
        public static void main(String[] args) throws IllegalAgeException {
            Scanner sc = new Scanner(System.in);

            Person p = new Person();

            while (true) {
                try {
                    System.out.println("请输入 p 的年龄：");
                    String age = sc.nextLine();
                    p.setAge(Integer.parseInt(age));
                    break;
                } catch (NumberFormatException e) {
                    e.printStackTrace();
                } catch (IllegalAgeException e) {
                    e.printStackTrace();
```

```
                }
            }
            System.out.println("你设置 p 的年龄为：" + p.getAge());
        }
}
```



==========

题目 2

源代码：

MyThread.java

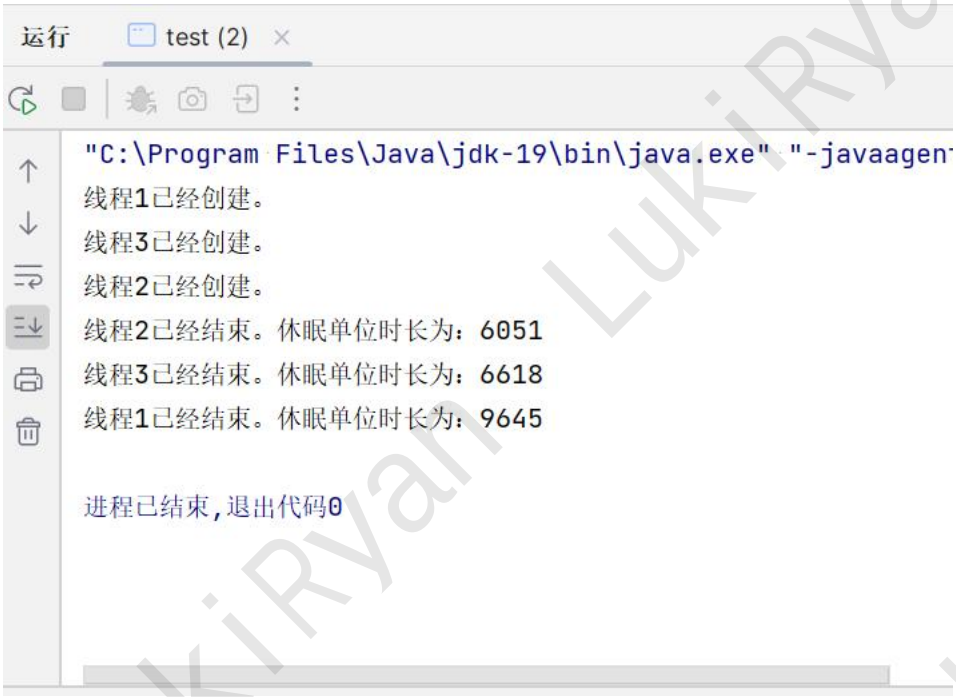
```
import java.util.Random;

public class MyThread extends Thread{
    @Override
    public void run() {
```

```
        System.out.println(getName() + "已经创建。");

        Random random = new Random();
        int t = random.nextInt(10000);
        try {
            Thread.sleep(t);
        } catch (InterruptedException e) {
            throw new RuntimeException(e);
        }

        System.out.println(getName() + "已经结束。休眠单位时长为：" + t);
    }
}
```

test.java

```
public class test {
    public static void main(String[] args) {
        MyThread t1 = new MyThread();
        MyThread t2 = new MyThread();
        MyThread t3 = new MyThread();
        t1.setName("线程 1");
        t2.setName("线程 2");
        t3.setName("线程 3");
        // start 方法表示开启线程
        t1.start();
        t2.start();
        t3.start();
    }
}
```

运行结果如下：



==========

题目 3

源代码：

PrimeThread.java

```
import java.util.Random;

public class PrimeThread extends Thread{
    @Override
    public void run() {

        //在一个线程中求 100 以内素数，
        for (int i = 1; i <= 100; i++) {
            if (i == 1 || i==2) {
                //输出数据时应有提示，指明哪个线程输出的数据。
                System.out.println(i + ":来自" + getName());
            }
            for (int j = 2; j < i; j++) {
                if (i % j == 0) {
```

```
                    break;
                }
                if (j == i-1) {
                    //输出数据时应有提示，指明哪个线程输出的数据。
                    System.out.println(i + ":来自" + getName());
                }
            }
        }
        //求出后休眠一个随机时间。
        Random random = new Random();
        int t = random.nextInt(10000);
         try {
            Thread.sleep(t);
        } catch (InterruptedException e) {
            throw new RuntimeException(e);
        }
        System.out.println(getName() + "休眠了" + t + "单位时间后结束");
    }
}
```

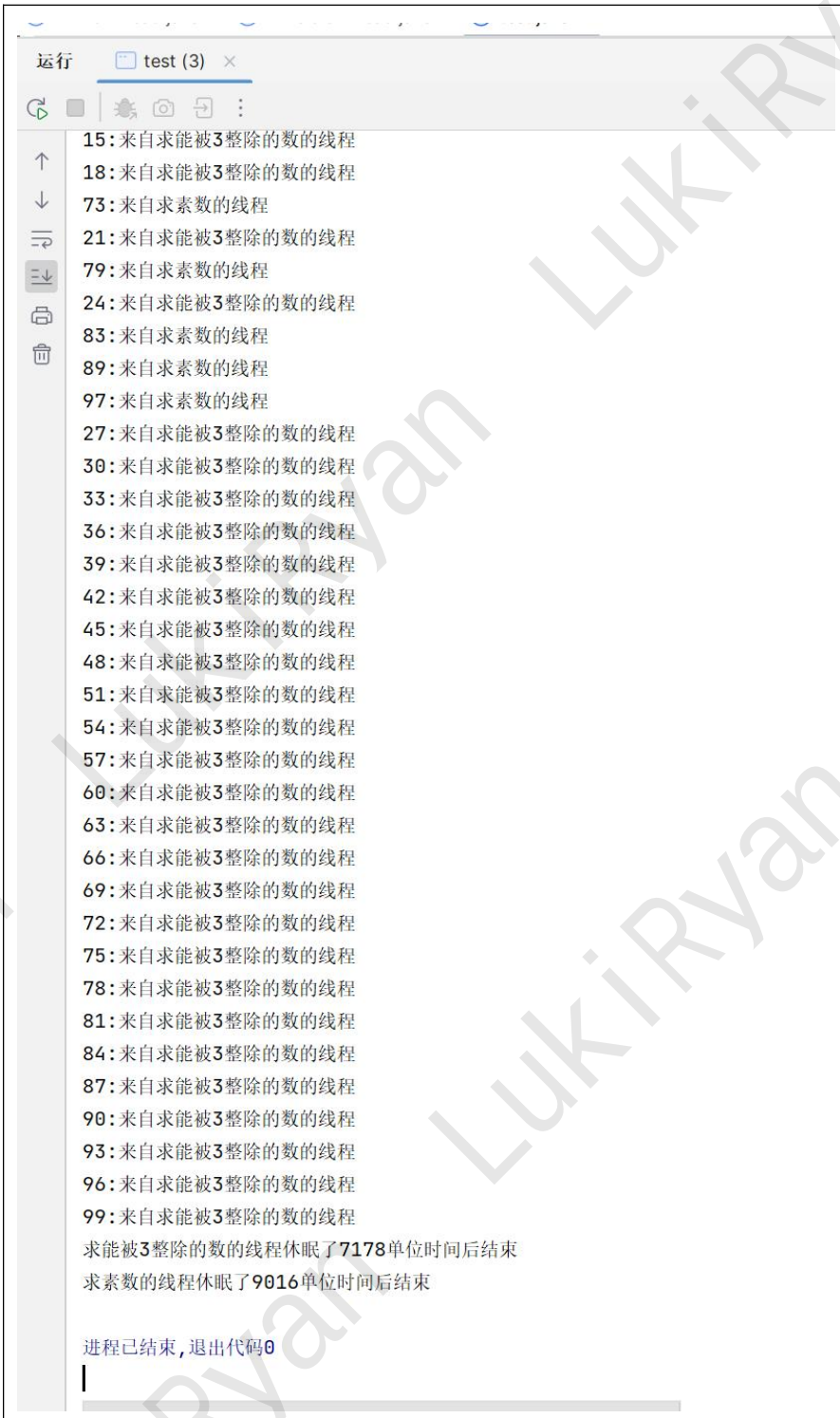
DivisibleThread.java

```
import java.util.Random;

public class DivisibleThread extends Thread{
    @Override
    public void run() {

        //在另一线程中求 100 以内能被 3 整除的数，
        for (int i = 1; i <= 100; i++) {
            if (i % 3 == 0) {
                //输出数据时应有提示，指明哪个线程输出的数据。
                System.out.println(i + ":来自" + getName());
            }
        }
        //求出后休眠一个随机时间。
        Random random = new Random();
        int t = random.nextInt(10000);
         try {
            Thread.sleep(t);
        } catch (InterruptedException e) {
            throw new RuntimeException(e);
        }
         System.out.println(getName() + "休眠了" + t + "单位时间后结束");
    }
}
```

test.java

```
public class test {
    public static void main(String[] args) {
        PrimeThread t1 = new PrimeThread();
        DivisibleThread t2 =new DivisibleThread();

        t1.setName("求素数的线程");
        t2.setName("求能被 3 整除的数的线程");

        t1.start();
        t2.start();
    }
}
```

运行结果如下：


运行
test (3)
15:来自求能被3整除的数的线程
18:来自求能被3整除的数的线程
73:来自求素数的线程
21:来自求能被3整除的数的线程
79:来自求素数的线程
24:来自求能被3整除的数的线程
83:来自求素数的线程
89:来自求素数的线程
97:来自求素数的线程
27:来自求能被3整除的数的线程
30:来自求能被3整除的数的线程
33:来自求能被3整除的数的线程
36:来自求能被3整除的数的线程
39:来自求能被3整除的数的线程
42:来自求能被3整除的数的线程
45:来自求能被3整除的数的线程
48:来自求能被3整除的数的线程
51:来自求能被3整除的数的线程
54:来自求能被3整除的数的线程
57:来自求能被3整除的数的线程
60:来自求能被3整除的数的线程
63:来自求能被3整除的数的线程
66:来自求能被3整除的数的线程
69:来自求能被3整除的数的线程
72:来自求能被3整除的数的线程
75:来自求能被3整除的数的线程
78:来自求能被3整除的数的线程
81:来自求能被3整除的数的线程
84:来自求能被3整除的数的线程
87:来自求能被3整除的数的线程
90:来自求能被3整除的数的线程
93:来自求能被3整除的数的线程
96:来自求能被3整除的数的线程
99:来自求能被3整除的数的线程
求能被3整除的数的线程休眠了7178单位时间后结束
求素数的线程休眠了9016单位时间后结束
进程已结束,退出代码0

通过这个实验，我学到了以下内容：

异常的概念和处理方法：我了解到异常是程序运行过程中可能发生的错误或异常情况，如输入错误、除零错误等。我学习了如何使用 try-catch 语句块捕获异常，并可以在 catch 块中进行相应的处理或提供错误提示信息。如果无法处理异常，可以将异常抛出给调用者处理。

线程的概念和生命周期：我学习了线程是程序执行的单个流程，可以并发执行多个线程。线程有不同的状态，包括新建、就绪、运行、阻塞和终止。我还了解了如何使用 Thread 类来创建线程，并可以通过调用 start()方法启动线程。

多线程编程和动画实现：我学习了如何使用多线程机制来实现并发执行的任务。通过继承 Thread 类，我可以重写 run()方法，在其中编写线程的具体逻辑。在实验中，我可以使用多线程编程来实现动画效果，其中每个线程负责打印信息、休眠一段时间等操作。

在实验过程中，我遇到了一些问题，并采取了相应的解决方法：

问题 1：在自定义非法年龄类 IllegalAgeException 中，如何判断年龄是否符合要求并抛出异常？
解决方法：我可以在设置年龄的函数中添加判断逻辑，检查年龄是否在 1 到 150 之间。如果不符合要求，我可以使用 throw 关键字抛出 IllegalAgeException 异常，并在调用该函数的地方使用 try-catch 块捕获并处理异常。

问题 2：在多线程程序中，如何控制线程的顺序和休眠时间？
解决方法：我可以使用 sleep()方法在每个线程中设置休眠时间，使其等待一段随机时间后再执行下一步操作。通过调整各个线程的顺序和休眠时间，可以实现指定的执行顺序和时间间隔。

发现和收获：

发现 1：异常处理是一种重要的编程技巧，可以帮助我们在程序出错时进行合理的处理和错误提示，提高程序的稳定性和可靠性。

收获 1：通过学习多线程编程，我了解到多线程可以提高程序的并发性和效率，适用于一些需要同时执行多个任务的场景，如动画效果的实现。

指导老师评阅意见

指导老师：　　年　　月　　日

填写内容时，可把表格扩大。