**《面向对象程序设计(Java)》**

**实验报告**

姓 名 石家晖

学 号 2019081238

学 院 软件工程学院

专 业 软件工程

年 级 2019级

指导教师

**成都信息工程大学软件工程学院**

**2020年9月**

# 实验1：Java程序的编写、编译与运行

1. **实验目的**

1. 掌握下载Java SDK 软件包的方法；掌握Myeclipse或Eclipse或JetBrains IntelliJ IDEA等开发平台的下载安装方法

2. 掌握设置Java 程序运行环境的方法；

3. 掌握编写、编译、运行简单的Java Application程序的方法步骤，初步了解java程序的特点和基本结构。

1. **实验要求**

1. 实验在2学时内完成。

2. 1人1组独立完成。

1. **实验设备与环境 (填写你自己的实验环境)**

JDK13, NEOVIM，Linux

1. **设计思路与具体实现**

1.

1. 设计思路：

使用源代码编辑器VIM。创建类Hello.java，在编辑器种编写出程序源代码。并在shell里编译，运行。

(2)实验步骤

①、创建类Hello.java，

②、在主函数中调用System.out.println()方法

③、编译并运行

(3) 关键代码

**System.out.println("Hello World");**

(4) 程序测试过程

直接运行

(5) 运行结果（运行效果）



2.

(1) 设计思路

编写一个应用程序求三个整数的平均数，原始数字要求从命令行输入。应用程序中main方法的参数String类型的数组args能接受用户从命令行键入的参数。

(2)实验步骤

①、创建类AverageTest.java，

②、在主函数中调用Integer.valueOf()方法，将从命令行中读取的字符串转化成int类型的数据。

③、编译并运行。

(3) 关键代码

**int a = Integer.valueOf(args[0]);**

**int b = Integer.valueOf(args[1]);**

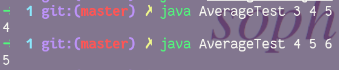
**int c = Integer.valueOf(args[2]);**

**System.out.println((a+b+c)/3);**

(4) 程序测试过程

先输入3 4 5三个整数，测出结果4，正确。再输入4 5 6三个整数，测出结果5，正确。

(5) 运行结果（运行效果）



1. **实验总结**

掌握了java基本的打印函数以及从终端编译运行并输入数据的基本方法。

1. **附录（程序代码）**

1.

public class Hello{

public static void main(String[] args){

System.out.println("Hello World");

}

}

2.

public class AverageTest{

public static void main(String[] args){

int a = Integer.valueOf(args[0]);

int b = Integer.valueOf(args[1]);

int c = Integer.valueOf(args[2]);

System.out.println((a+b+c)/3);

}

}

# 实验2：Java程序设计基础

1. **实验目的**

1. 掌握 Java 基本数据类型、运算符与表达式。

2. 理解 Java 程序语法结构，掌握顺序结构、选择结构和循环结构语法的程序设计方法。

3. 理解并掌握数组的使用

1. **实验要求**

1. 实验在2学时内完成。

2. 1人1组独立完成。

1. **实验设备与环境**

JDK13, NEOVIM，Linux

1. **设计思路与具体实现**

1.

(1) 设计思路

在循环中，只要除数不等于0，用较大数除以较小的数，将小的一个数作为下一轮循环的大数，取得的余数作为下一轮循环的较小的数，如此循环直到较小的数的值为0，返回较大的数，此数即为最大公约数，最小公倍数为两数之积除以最大公约数 。

例如输入：3 5

输出为：

最大公约数: 1

最小公倍数: 15

(2)实验步骤

①、创建类gcd

②、采用辗转相除法的思想，设计类gcd的构造函数，能够求出两个数的最大公约数。利用得到的两个数的最大公约数，可以求出两个数的最小公倍数。

③、在主函数中设置输入输出格式

(3) 关键代码

**public class gcd{**

**public static void main(String[] args){**

**Scanner s = new Scanner(System.in);**

**int a,b,ans;**

**System.out.println("请输入两个数：");**

**a = Integer.valueOf(s.next());**

**b = Integer.valueOf(s.next());**

**if (a > b)**

**ans = gcd(a,b);**

**else**

**ans = gcd(b,a);**

**System.out.println("最大公约数：" + ans);**

**System.out.println("最小公倍数：" + a\*b/ans);**

**}**

**public static int gcd(int a,int b){**

**return (a%b != 0)?gcd(b,a%b):b;**

**}**

**}**

(4) 程序测试过程

使用多组测试数据，观察输出的正确与否。

(5) 运行结果（运行效果）



2.

(1) 设计思路

用选择排序法，按升序顺序输出一维数组中的元素 。

(2)实验步骤

①、创建类ArraySort和方法sortChoose()

②、以选择排序法为核心算法，设计方法sortChoose()

③、在类ArraySort的主函数中设置测试数组，并使用增强型for循环作为打印输出的方法。

(3) 关键代码

**public class ArraySort{**

**public static void main(String[] args){**

**int[] a = {1,3,4,2,5};**

**sortChoose(a);**

**System.out.println("out:");**

**//增强型for循环**

**for( int i : a ){**

**System.out.print(i);**

**}**

**}**

**public static void sortChoose(int num[]){**

**System.out.println();**

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

**int index = 0;**

**for (int j = 0;j < num.length-i ;j++ ) {**

**if (num[j] > num[index]) {**

**index = j;**

**}**

**int temp = num[num.length-i-1];**

**num[num.length-i-1] = num[index];**

**num[index] = temp;**

**}**

**}**

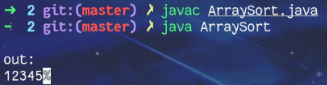
**}**

**}**

(4) 程序测试过程

改变主函数中的测试数组的各项数值，观察并检测输出结果的正确与否。

1. 运行结果（运行效果）



3.

(1) 设计思路

有n个人围成一圈，顺序排号。从第一个人开始报数（从1到3报数），凡报到3的人退出圈子，问最后留下的是原来第几号的那位。

例如输入： 10

输出：原排在第4位的人留下了。

(2)实验步骤

①、创建类Circle

②、通过对于题目的逻辑分析，选择int数组进行数据的存储，使用while和for控制流程。

③、编译并使用多组测试数据进行实例化测试。

(3) 关键代码

**import java.util.Scanner;**

**public class Circle{**

**public static void main(String[] args){**

**Scanner s = new Scanner(System.in);**

**int i,j,k,m,n;**

**n = Integer.valueOf(s.next());**

**int[] a = new int[n];**

**for (i = 0; i < n; i++) {**

**a[i] = i+1;**

**}**

**i = 0;k = 0;m = 0;**

**while( m < n-1 ){**

**if ( a[i] != 0 ) {**

**k++;**

**}**

**if ( k == 3 ) {**

**a[i] = 0;**

**k = 0;**

**m ++;**

**}**

**i++;**

**if (i == n) {**

**i = 0;**

**}**

**}**

**i = 0;**

**while( a[i] == 0 ){**

**i++;**

**}**

**System.out.println(a[i]);**

**}**

**}**

(4) 程序测试过程

使用多组测试数据，观察输出的正确与否。

(5) 运行结果（运行效果）



1. **实验总结**

掌握 Java 基本数据类型、运算符与表达式。理解了 Java 程序语法结构，掌握了顺序结构、选择结构和循环结构语法的程序设计方法。理解并掌握了数组的使用

1. **附录（程序代码）**

1.import java.util.Scanner;

public class gcd{

public static void main(String[] args){

Scanner s = new Scanner(System.in);

int a,b,ans;

System.out.println("请输入两个数：");

a = Integer.valueOf(s.next());

b = Integer.valueOf(s.next());

if (a > b)

ans = gcd(a,b);

else

ans = gcd(b,a);

System.out.println("最大公约数：" + ans);

System.out.println("最小公倍数：" + a\*b/ans);

}

public static int gcd(int a,int b){

return (a%b != 0)?gcd(b,a%b):b;

}

}

2.

public class ArraySort{

public static void main(String[] args){

int[] a = {1,3,4,2,5};

sortChoose(a);

System.out.println("out:");

//增强型for循环

for( int i : a ){

System.out.print(i);

}

}

public static void sortChoose(int num[]){

System.out.println();

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

int index = 0;

for (int j = 0;j < num.length-i ;j++ ) {

if (num[j] > num[index]) {

index = j;

}

int temp = num[num.length-i-1];

num[num.length-i-1] = num[index];

num[index] = temp;

}

}

}

}

3.

import java.util.Scanner;

public class Circle{

public static void main(String[] args){

Scanner s = new Scanner(System.in);

int i,j,k,m,n;

n = Integer.valueOf(s.next());

int[] a = new int[n];

for (i = 0; i < n; i++) {

a[i] = i+1;

}

i = 0;k = 0;m = 0;

while( m < n-1 ){

if ( a[i] != 0 ) {

k++;

}

if ( k == 3 ) {

a[i] = 0;

k = 0;

m ++;

}

i++;

if (i == n) {

i = 0;

}

}

i = 0;

while( a[i] == 0 ){

i++;

}

System.out.println(a[i]);

}

}

# 实验3：

1. **实验目的**

1. 通过编程和上机实验理解 Java 语言是如何体现面向对象编程基本思想，了解类的封装方法，以及 如何创建类和对象，了解成员变量和成员方法的特性，掌握 OOP 方式进行程序设计的方法 。

1. **实验要求**

1. 实验在2学时内完成。

2. 1人1组独立完成。

1. **实验设备与环境**

JDK13, NEOVIM，Linux

1. **设计思路与具体实现**

1.

(1) 设计思路

编写一个圆环类 Ring 的 Java 程序。圆环类有 3 个数据成员 , 分别是内半径 innerRadius, 外半径 outerRadius 和颜色 color, 这些属性可以查看 get 也可以重新设置 set, 另外 , 圆环还可以返回其面积 area 。

(2) 实验步骤

①、创建类Ring

②、在类中创建 3 个数据成员 , 分别是内半径 innerRadius, 外半径 outerRadius 和颜色 color。并编写get方法和set方法以查看和修改这些属性。最后 , 编写方法返回其面积 area 。 并设计Ring的构造函数。

③、在main函数中创建Ring对象，并且对其get方法、set方法、和返回其area的方法进行测试。

(3) 关键代码

class Ring {

private double innerRadius;

private double outerRadius;

public String color;

public static double PI = 3.1415926;

public Ring (double iRadius, double oRadius, String c) {

this.innerRadius = iRadius;

this.outerRadius = oRadius;

this.color = c;

num++;

}

public double getInnerRadius() {

return innerRadius;

}

public void setInnerRadius (double innerRadius) {

this.innerRadius = innerRadius;

}

public double getOuterRadius() {

return outerRadius;

}

public void setOuterRadius (double outerRadius) {

this.outerRadius = outerRadius ;

}

public String getColor() {

return color;

}

public void setColor (String color) {

this.color = color;

}

public double getArea() {

return PI \* (getOuterRadius() \* getOuterRadius() - getInnerRadius() \* getInnerRadius() );

}

public double getInCircle() {

return 2 \* PI \* getInnerRadius();

}

public double getOutCircle() {

return 2 \* PI \* getOuterRadius();

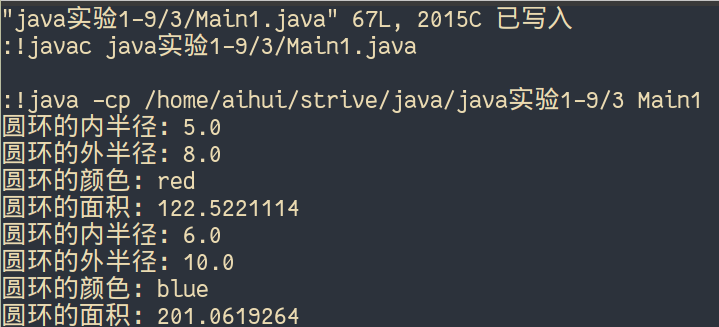
}

}

(4) 程序测试过程

在main函数中输入测试数据，调用Ring类的所有方法，观察输出的正确与否。

1. 运行结果（运行效果）



2.

(1) 设计思路

第1题中增加两个static 成员 ：圆周率和圆对象个数, 增加两个 static 方法，分别是设置圆周率和显示当前圆对象个数的, 仔细体会静态成员与实例成员的使用方法和区别。

(2) 实验步骤

①、在Ring类中增加两个static 成员 ：圆周率PI和圆对象个数num。

②、增加两个 static 方法，分别是设置圆周率和显示当前圆对象个数的。

③、在主函数中进行测试。

(3) 关键代码

public static double PI = 3.1415926;

public static int num = 0;

//static

public static void setPI (double pi) {

PI = pi;

}

public static int getNum() {

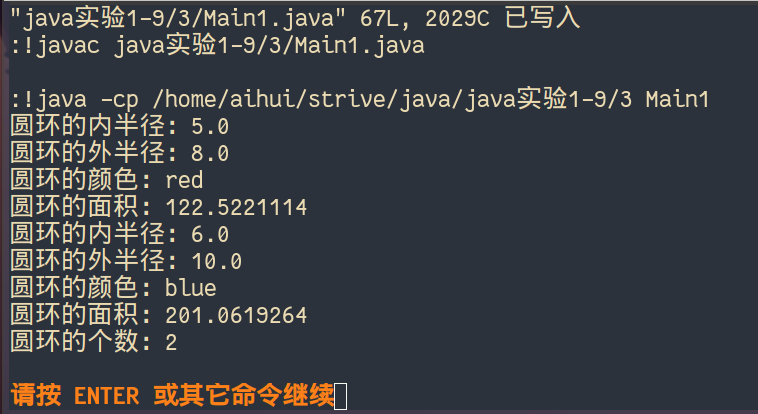
return num;

}

(4) 程序测试过程

在main函数中输入测试数据，调用Ring类的两个static方法，观察输出的正确与否。

(5) 运行结果（运行效果）



3.

(1) 设计思路

设计一个教材类，一个课程类，及一个为某门课程指定参考教材的类。一门课程可以有多本参考教材，再设计一个测试类。

(2) 实验步骤

①、创建4个类，分别是教材类、课程类、为某门课程指定参考教材的类、测试类。

②、确定好四个类的关系，设计实现这四个类。

③、创建测试类对象进行类的测试。

(3) 关键代码

class Classes {

private String name;

private NameBooks nameBooks = new NameBooks();

public Classes (String name) {

this.name = name;

}

public String getName() {

return name;

}

public void addBooks (String booksName) {

nameBooks.addBooks (booksName);

}

public void putAllBooks() {

for (int i = 0; i < nameBooks.getNum() ; i++ )

System.out.println ("第" + (i + 1) + "教材:" + nameBooks.getBooks (i) );

}

}

class Books {

private String name;

public void setName (String name) {

this.name = name;

}

public String getName() {

return name;

}

}

class NameBooks {

Books books[] = new Books[100];

int num = 0;

public void addBooks (String booksName) {

books[num] = new Books();

books[num].setName (booksName);

num++;

}

public String getBooks (int num) {

return books[num].getName();

}

public int getNum() {

return num;

}

}

class Test {

public void testClasses() {

Classes a = new Classes ("Chinese");

a.addBooks ("Chinese Book I");

a.addBooks ("Chinese Book II");

a.addBooks ("Chinese Book III");

a.putAllBooks();

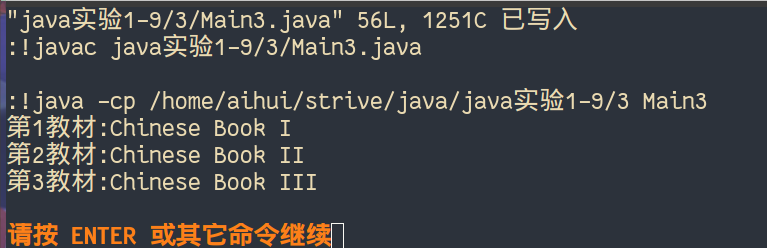
}

}

(4) 程序测试过程

由于设计了测试类，可以直接在main函数中创建测试类对象进行类的测试。

(5) 运行结果（运行效果）



1. **实验总结**

理解了 Java 语言是如何体现面向对象编程基本思想，了解了类的封装方法，以及如何创建类和对象。

了解了成员变量和成员方法的特性，基本掌握了OOP 方式进行程序设计的方法 。

1. **附录（程序代码）**

1&2.

public class Main1 {

public static void main (String[] args) {

Ring ring1 = new Ring (5, 8, "red");

System.out.println ("圆环的内半径：" + ring1.getInnerRadius() );

System.out.println ("圆环的外半径：" + ring1.getOuterRadius() );

System.out.println ("圆环的颜色：" + ring1.getColor() );

System.out.println ("圆环的面积：" + ring1.getArea() );

Ring ring2 = new Ring (5, 8, "red");

ring2.setInnerRadius (6);

ring2.setOuterRadius (10);

ring2.setColor ("blue");

System.out.println ("圆环的内半径：" + ring2.getInnerRadius() );

System.out.println ("圆环的外半径：" + ring2.getOuterRadius() );

System.out.println ("圆环的颜色：" + ring2.getColor() );

System.out.println ("圆环的面积：" + ring2.getArea() );

System.out.println ("圆环的个数：" + ring2.getNum() );

}

}

class Ring {

private double innerRadius;

private double outerRadius;

public String color;

public static double PI = 3.1415926;

public static int num = 0;

public Ring (double iRadius, double oRadius, String c) {

this.innerRadius = iRadius;

this.outerRadius = oRadius;

this.color = c;

num++;

}

public double getInnerRadius() {

return innerRadius;

}

public void setInnerRadius (double innerRadius) {

this.innerRadius = innerRadius;

}

public double getOuterRadius() {

return outerRadius;

}

public void setOuterRadius (double outerRadius) {

this.outerRadius = outerRadius ;

}

public String getColor() {

return color;

}

public void setColor (String color) {

this.color = color;

}

public double getArea() {

return PI \* (getOuterRadius() \* getOuterRadius() - getInnerRadius() \* getInnerRadius() );

}

public double getInCircle() {

return 2 \* PI \* getInnerRadius();

}

public double getOutCircle() {

return 2 \* PI \* getOuterRadius();

}

//static

public static void setPI (double pi) {

PI = pi;

}

public static int getNum() {

return num;

}

}

3.

public class Main3 {

public static void main (String[] args) {

Test test = new Test();

test.testClasses();

}

}

class Classes {

private String name;

private NameBooks nameBooks = new NameBooks();

public Classes (String name) {

this.name = name;

}

public String getName() {

return name;

}

public void addBooks (String booksName) {

nameBooks.addBooks (booksName);

}

public void putAllBooks() {

for (int i = 0; i < nameBooks.getNum() ; i++ )

System.out.println ("第" + (i + 1) + "教材:" + nameBooks.getBooks (i) );

}

}

class Books {

private String name;

public void setName (String name) {

this.name = name;

}

public String getName() {

return name;

}

}

class NameBooks {

Books books[] = new Books[100];

int num = 0;

public void addBooks (String booksName) {

books[num] = new Books();

books[num].setName (booksName);

num++;

}

public String getBooks (int num) {

return books[num].getName();

}

public int getNum() {

return num;

}

}

class Test {

public void testClasses() {

Classes a = new Classes ("Chinese");

a.addBooks ("Chinese Book I");

a.addBooks ("Chinese Book II");

a.addBooks ("Chinese Book III");

a.putAllBooks();

}

}

# 实验4：

1. **实验目的**

1. 掌握Java的继承与多态，能编写体现类的继承性（成员变量、成员方法、成员变量隐藏）、类的多态性（成员方法重载、构造方法重载）的程序。

2. 掌握抽象类与接口

3. 掌握Object类的equals方法和toString方法的使用

1. **实验要求**

1. 实验在2学时内完成。

2. 1人1组独立完成。

1. **实验设备与环境**

JDK13, NEOVIM，Linux

1. **设计思路与具体实现**

1.

(1) 设计思路

设计一个简单的学校人事管理系统,管理教师和学生信息。实现教师与学生基本信息的添加、删除、修改、查询。其中编号、姓名、性别、出生日期为共有的，教师包括部门、职称、工资；学生包括高考分数、专业field、班级等。

(2) 实验步骤

①、创建父类Person和它的子类Student和Teacher

②、实现教师与学生基本信息的添加、删除、修改、查询。其中编号、姓名、性别、出生日期为共有的，教师包括部门、职称、工资；学生包括高考分数、专业field、班级等。

③、 在主函数中输入一些测试用数据进行测试

(3) 关键代码

class Person {

private int num;

private String name;

private String sex;

private int birthYear, birthMonth, birthDay;

public void setNum (int num) {

this.num = num;

}

public void setName (String name) {

this.name = name;

}

public void setSex (String sex) {

this.sex = sex;

}

public void setbirth (int y, int m, int d) {

this.birthYear = y;

this.birthMonth = m;

this.birthDay = d;

}

public int getNum() {

return num;

}

public String getSex() {

return sex;

}

public String getName() {

return name;

}

public int getBirthYear() {

return birthYear;

}

public int getBirthMonth() {

return birthMonth;

}

public int getbirthDay() {

return birthDay;

}

public void check() {

System.out.println ("编号：" + num);

System.out.println ("姓名：" + name);

System.out.println ("性别：" + sex);

System.out.println ("出生日期：" + birthYear + "年" + birthMonth + "月" + birthDay + "日");

}

public Person (int num,

String name,

String sex,

int birthYear,

int birthMonth,

int birthDay) {

this.num = num;

this.name = name;

this.sex = sex;

this.birthYear = birthYear;

this.birthMonth = birthMonth;

this.birthDay = birthDay;

}

}

class Teacher extends Person {

private String department, title;

private int salary;

public String getDepartment() {

return department;

}

public String getTitle() {

return title;

}

public int getSalary() {

return salary;

}

public void setDepartment (String department) {

this.department = department;

}

public void setTitle (String title) {

this.title = title;

}

public void setSalary (int salary) {

this.salary = salary;

}

public void check() {

System.out.println ("编号：" + getNum() );

System.out.println ("姓名：" + getName() );

System.out.println ("性别：" + getSex() );

System.out.println ("出生日期：" + getBirthYear() + "年" + getBirthMonth() + "月" + getbirthDay() + "日");

System.out.println ("部门：" + department);

System.out.println ("职称：" + title);

System.out.println ("薪资：" + salary);

}

public Teacher (int num,

String name,

String sex,

int birthYear,

int birthMonth,

int birthDay,

String department,

String title,

int salary) {

super (num, name, sex, birthYear, birthMonth, birthDay);

this.department = department;

this.title = title;

this.salary = salary;

}

}

class Student extends Person {

private String field, sClass;

private int score;

public void setScore (int score) {

this.score = score;

}

public void setField (String field) {

this.field = field;

}

public void setSClass (String sClass) {

this.sClass = sClass;

}

public int getScore() {

return score;

}

public String getField() {

return field;

}

public String getSClass() {

return sClass;

}

public void check() {

System.out.println ("编号：" + getNum() );

System.out.println ("姓名：" + getName() );

System.out.println ("性别：" + getSex() );

System.out.println ("出生日期：" + getBirthYear() + "年" + getBirthMonth() + "月" + getbirthDay() + "日");

System.out.println ("高考分数：" + score);

System.out.println ("专业：" + field);

System.out.println ("班级：" + sClass);

}

public Student (int num,

String name,

String sex,

int birthYear,

int birthMonth,

int birthDay,

int score,

String field,

String sClass) {

super (num, name, sex, birthYear, birthMonth, birthDay);

this.score = score;

this.field = field;

this.sClass = sClass;

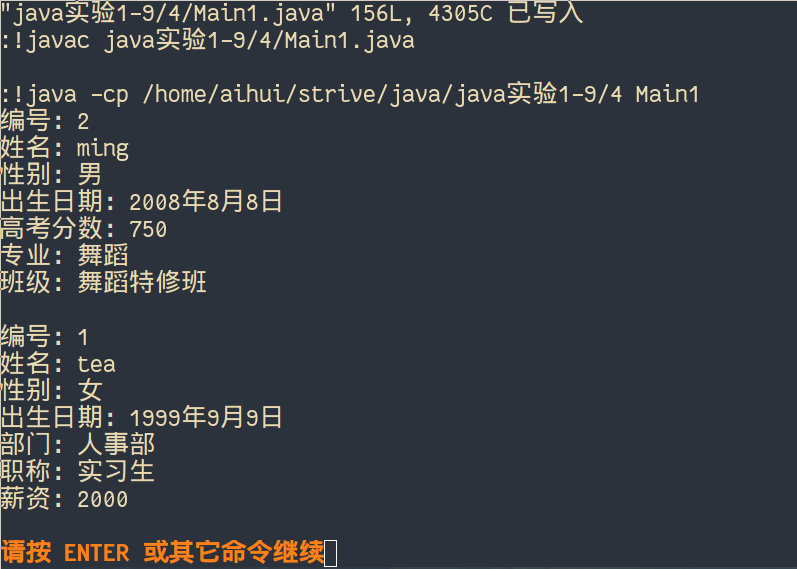
}

}

(4) 程序测试过程

在主函数中分别对类Teacher和Student进行输入输出测试。

(5) 运行结果（运行效果）



2.

(1) 设计思路

设计 3 个类 , 分别是学生类 Student, 本科生类 Undergaduate, 研究生类 Postgraduate, 其中 Student 类是一个抽象类 , 它包含一些基本的学生信息如姓名、所学课程、课程成绩等 , 而 Undergraduate 类和 Postgraduate 都是 Student 类的子类 , 它们之间的主要差别是计算课程成绩等级的方法有所不同 , 研究生的标准要比本科生的标准高一些 , 如下表所示。

|  |  |  |  |
| --- | --- | --- | --- |
| 本科生标准 | | 研究生标准 | |
| 80-100 | 优秀 | 90-100 | 优秀 |
| 70-80 | 良好 | 80-90 | 良好 |
| 60-70 | 一般 | 70-80 | 一般 |
| 50-60 | 及格 | 60-70 | 及格 |
| 50以下 | 不及格 | 60以下 | 不及格 |

假设某班级里既有本科生也有研究生 , 请编写程序统计出全班学生的成绩等级并显示出来。此题关键是设计一个学生数组 , 既能存放本科生对象 , 又能存放研究生对象。

(2) 实验步骤

①、创建父类Student和它的子类本科生类 Undergaduate和 研究生类 Postgraduate

②、Student 类是一个抽象类 , 填充它包含一些基本的学生信息如姓名、所学课程、课程成绩等 。而 Undergraduate 类和 Postgraduate 都是 Student 类的子类 , 分别实现本科生类 Undergaduate和 研究生类 Postgraduate所涉及的功能

③、在主函数中输入一些测试用数据进行测试。设计一个学生数组 , 实现其既能存放本科生对象 , 又能存放研究生对象。

(3) 关键代码

abstract class Student {

String name;

String sClass;

int score;

public void setName (String name) {

this.name = name;

}

public void setsClass (String sClass) {

this.sClass = sClass;

}

public void setScore (int score) {

this.score = score;

}

public String getName() {

return name;

}

public String getsClass() {

return sClass;

}

public int getScore() {

return score;

}

public void print() {}

public Student (String name, String sClass, int score) {

this.name = name;

this.sClass = sClass;

this.score = score;

}

}

class Undergaduate extends Student {

public void print() {

System.out.print ("本科生");

System.out.print (" " + name);

System.out.print (" " + sClass);

System.out.print (" ");

if (score >= 80)

System.out.println ("优秀");

else if (score >= 70)

System.out.println ("良好");

else if (score >= 60)

System.out.println ("一般");

else if (score >= 50)

System.out.println ("及格");

else

System.out.println ("不及格");

}

public Undergaduate (String name, String sClass, int score) {

super (name, sClass, score);

}

}

class Postgraduate extends Student {

public void print() {

System.out.print ("研究生");

System.out.print (" " + name);

System.out.print (" " + sClass);

System.out.print (" ");

if (score >= 90)

System.out.println ("优秀");

else if (score >= 80)

System.out.println ("良好");

else if (score >= 70)

System.out.println ("一般");

else if (score >= 60)

System.out.println ("及格");

else

System.out.println ("不及格");

}

public Postgraduate (String name, String sClass, int score) {

super (name, sClass, score);

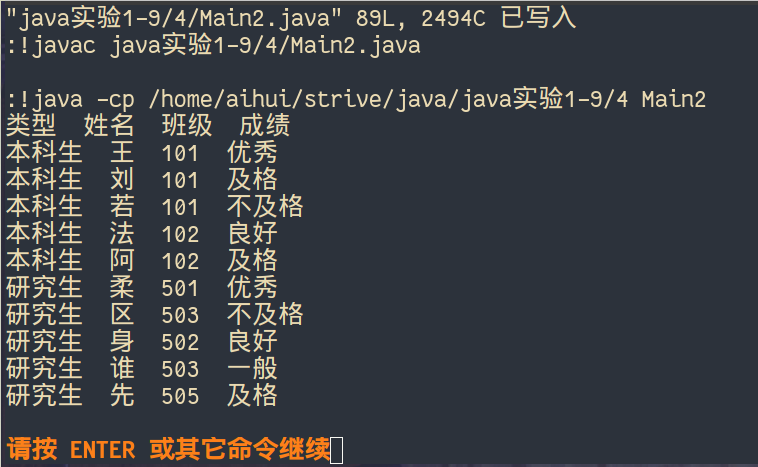
}

}

(4) 程序测试过程

在主函数中创建数组对类进行输入输出测试。

(5) 运行结果（运行效果）



1. **实验总结**

掌握了Java的继承与多态，能编写体现类的继承性（成员变量、成员方法、成员变量隐藏）、类的多态性（成员方法重载、构造方法重载）的程序。

另外，掌握了抽象类与接口并且掌握了Object类的equals方法和toString方法的使用

1. **附录（程序代码）**

1.public class Main1 {

public static void main (String[] args) {

Student s = new Student(2,"ming","男",2008,8,8,750,"舞蹈","舞蹈特修班");

Teacher t = new Teacher(1,"tea","女",1999,9,9,"人事部","实习生",2000);

s.check();

System.out.println();

t.check();

}

}

class Person {

private int num;

private String name;

private String sex;

private int birthYear, birthMonth, birthDay;

public void setNum (int num) {

this.num = num;

}

public void setName (String name) {

this.name = name;

}

public void setSex (String sex) {

this.sex = sex;

}

public void setbirth (int y, int m, int d) {

this.birthYear = y;

this.birthMonth = m;

this.birthDay = d;

}

public int getNum() {

return num;

}

public String getSex() {

return sex;

}

public String getName() {

return name;

}

public int getBirthYear() {

return birthYear;

}

public int getBirthMonth() {

return birthMonth;

}

public int getbirthDay() {

return birthDay;

}

public void check() {

System.out.println ("编号：" + num);

System.out.println ("姓名：" + name);

System.out.println ("性别：" + sex);

System.out.println ("出生日期：" + birthYear + "年" + birthMonth + "月" + birthDay + "日");

}

public Person (int num,

String name,

String sex,

int birthYear,

int birthMonth,

int birthDay) {

this.num = num;

this.name = name;

this.sex = sex;

this.birthYear = birthYear;

this.birthMonth = birthMonth;

this.birthDay = birthDay;

}

}

class Teacher extends Person {

private String department, title;

private int salary;

public String getDepartment() {

return department;

}

public String getTitle() {

return title;

}

public int getSalary() {

return salary;

}

public void setDepartment (String department) {

this.department = department;

}

public void setTitle (String title) {

this.title = title;

}

public void setSalary (int salary) {

this.salary = salary;

}

public void check() {

System.out.println ("编号：" + getNum() );

System.out.println ("姓名：" + getName() );

System.out.println ("性别：" + getSex() );

System.out.println ("出生日期：" + getBirthYear() + "年" + getBirthMonth() + "月" + getbirthDay() + "日");

System.out.println ("部门：" + department);

System.out.println ("职称：" + title);

System.out.println ("薪资：" + salary);

}

public Teacher (int num,

String name,

String sex,

int birthYear,

int birthMonth,

int birthDay,

String department,

String title,

int salary) {

super (num, name, sex, birthYear, birthMonth, birthDay);

this.department = department;

this.title = title;

this.salary = salary;

}

}

class Student extends Person {

private String field, sClass;

private int score;

public void setScore (int score) {

this.score = score;

}

public void setField (String field) {

this.field = field;

}

public void setSClass (String sClass) {

this.sClass = sClass;

}

public int getScore() {

return score;

}

public String getField() {

return field;

}

public String getSClass() {

return sClass;

}

public void check() {

System.out.println ("编号：" + getNum() );

System.out.println ("姓名：" + getName() );

System.out.println ("性别：" + getSex() );

System.out.println ("出生日期：" + getBirthYear() + "年" + getBirthMonth() + "月" + getbirthDay() + "日");

System.out.println ("高考分数：" + score);

System.out.println ("专业：" + field);

System.out.println ("班级：" + sClass);

}

public Student (int num,

String name,

String sex,

int birthYear,

int birthMonth,

int birthDay,

int score,

String field,

String sClass) {

super (num, name, sex, birthYear, birthMonth, birthDay);

this.score = score;

this.field = field;

this.sClass = sClass;

}

}

2.

public class Main2 {

public static void main (String[] args) {

Student []stu = new Student[10];

stu[0] = new Undergaduate ("王", "101", 80);

stu[1] = new Undergaduate ("刘", "101", 50);

stu[2] = new Undergaduate ("若", "101", 30);

stu[3] = new Undergaduate ("法", "102", 75);

stu[4] = new Undergaduate ("阿", "102", 55);

stu[5] = new Postgraduate ("柔", "501", 95);

stu[6] = new Postgraduate ("区", "503", 49);

stu[7] = new Postgraduate ("身", "502", 85);

stu[8] = new Postgraduate ("谁", "503", 75);

stu[9] = new Postgraduate ("先", "505", 65);

System.out.println ("类型 姓名 班级 成绩");

for(Student i:stu)

i.print();

}

}

abstract class Student {

String name;

String sClass;

int score;

public void setName (String name) {

this.name = name;

}

public void setsClass (String sClass) {

this.sClass = sClass;

}

public void setScore (int score) {

this.score = score;

}

public String getName() {

return name;

}

public String getsClass() {

return sClass;

}

public int getScore() {

return score;

}

public void print() {}

public Student (String name, String sClass, int score) {

this.name = name;

this.sClass = sClass;

this.score = score;

}

}

class Undergaduate extends Student {

public void print() {

System.out.print ("本科生");

System.out.print (" " + name);

System.out.print (" " + sClass);

System.out.print (" ");

if (score >= 80)

System.out.println ("优秀");

else if (score >= 70)

System.out.println ("良好");

else if (score >= 60)

System.out.println ("一般");

else if (score >= 50)

System.out.println ("及格");

else

System.out.println ("不及格");

}

public Undergaduate (String name, String sClass, int score) {

super (name, sClass, score);

}

}

class Postgraduate extends Student {

public void print() {

System.out.print ("研究生");

System.out.print (" " + name);

System.out.print (" " + sClass);

System.out.print (" ");

if (score >= 90)

System.out.println ("优秀");

else if (score >= 80)

System.out.println ("良好");

else if (score >= 70)

System.out.println ("一般");

else if (score >= 60)

System.out.println ("及格");

else

System.out.println ("不及格");

}

public Postgraduate (String name, String sClass, int score) {

super (name, sClass, score);

}

}

# 实验5：

1. **实验目的**

1.掌握异常处理的基本概念，学会使用try...catch块捕获常见的运行期异常。

2. 学会用throws从函数中抛出异常的用法。

3. 学会创建自定义异常，以及手工抛出（产生）异常的throw语句。

1. **实验要求**

1. 实验在2学时内完成。

2. 1人1组独立完成。

1. **实验设备与环境**

JDK13, NEOVIM，Linux

1. **设计思路与具体实现**

1.

(1) 设计思路

编写整除运算程序，要求捕获除数为0异常、数字格式异常、通用型异常。注意要把通用型异常的捕获顺序放在最后。

(2) 实验步骤

①、在main函数中利用try...catch块捕获除数为0异常、数字格式异常、通用型异常。

②、运行进行不同异常的测试。

(3) 关键代码

try {

System.out.print ("请输入被除数：");

a = sc.nextInt();

System.out.print ("请输入除数：");

b = sc.nextInt();

System.out.print ( (a / b) );

} catch (ArithmeticException e) {

System.out.println ("除数为0异常!");

} catch (InputMismatchException e) {

System.out.println ("数字格式异常!");

} catch (Exception e) {

System.out.println ("通用型异常!");

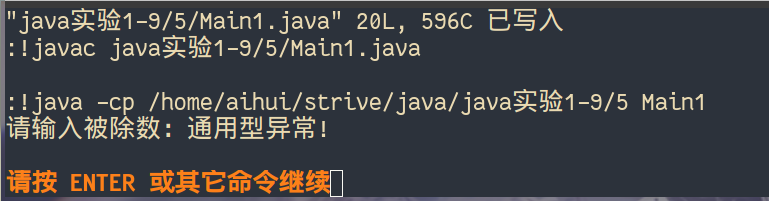
}

(4) 程序测试过程

输入导致异常的除数和被除数，观察输出。

(5) 运行结果（运行效果）





2.

(1) 设计思路

把第1题整除程序改为双精度型实数的除法运算程序，并更改有关提示信息，运行该程序若干次，每次输入不同的数据，观察分析程序的运行结果。

(2) 实验步骤

①、把第1题整除程序改为双精度型实数的除法运算程序，并更改有关提示信息。

②、运行程序若干次，每次输入不同的数据，观察分析程序的运行结果。

(3) 关键代码

double a, b;

try {

System.out.print ("请输入被除数：");

a = sc.nextDouble();

System.out.print ("请输入除数：");

b = sc.nextDouble();

System.out.print ( (a / b) );

} catch (ArithmeticException e) {

System.out.println ("除数为0异常!");

} catch (InputMismatchException e) {

System.out.println ("数字格式异常!");

} catch (Exception e) {

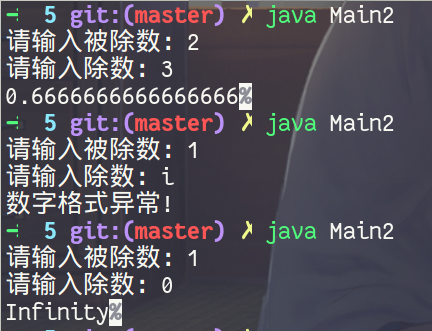
System.out.println ("通用型异常!");

}

(4) 程序测试过程

运行程序若干次，每次输入不同的数据，观察分析程序的运行结果。

1. 运行结果（运行效果）



3.

(1) 设计思路

在第2题基础上编写自定义异常类（必须继承系统的Exception类），在除数为0时抛出自定义异常，并捕获处理该异常。

(2) 实验步骤

①、创建继承系统Exception类的自定义异常类EX

②、设计自定义异常类EX和主函数，在除数为0时抛出自定义异常，并捕获处理该异常

③、编译并测试

(3) 关键代码

try {

System.out.print ("请输入被除数：");

a = sc.nextDouble();

System.out.print ("请输入除数：");

b = sc.nextDouble();

if (b == 0)

throw new EX ("异常：除数不能为0！");

else

System.out.print ( (a / b) );

} catch (EX e) {

System.out.println (e.getMessage() );

}

class EX extends Exception {

public EX (String message) {

super (message);

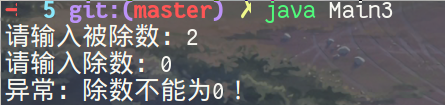
}

}

(4) 程序测试过程

类似题目2的测试方法，将除数设置为0，测试异常。

(5) 运行结果（运行效果）



1. **实验总结**

掌握了异常处理的基本概念，学会了使用try...catch块捕获常见的运行期异常。此外，学会了用throws从函数中抛出异常的用法。

学会了创建自定义异常，以及手工抛出（产生）异常的throw语句。

1. **附录（程序代码）**

1.import java.util.\*;

public class Main1 {

public static void main (String[] args) {

Scanner sc = new Scanner (System.in);

int a, b;

try {

System.out.print ("请输入被除数：");

a = sc.nextInt();

System.out.print ("请输入除数：");

b = sc.nextInt();

System.out.print ( (a / b) );

} catch (ArithmeticException e) {

System.out.println ("除数为0异常!");

} catch (InputMismatchException e) {

System.out.println ("数字格式异常!");

} catch (Exception e) {

System.out.println ("通用型异常!");

}

}

}

2.

import java.util.\*;

public class Main2 {

public static void main (String[] args) {

Scanner sc = new Scanner (System.in);

double a, b;

try {

System.out.print ("请输入被除数：");

a = sc.nextDouble();

System.out.print ("请输入除数：");

b = sc.nextDouble();

System.out.print ( (a / b) );

} catch (ArithmeticException e) {

System.out.println ("除数为0异常!");

} catch (InputMismatchException e) {

System.out.println ("数字格式异常!");

} catch (Exception e) {

System.out.println ("通用型异常!");

}

}

}

3.

import java.util.\*;

public class Main3 {

public static void main (String[] args) {

Scanner sc = new Scanner (System.in);

double a, b;

try {

System.out.print ("请输入被除数：");

a = sc.nextDouble();

System.out.print ("请输入除数：");

b = sc.nextDouble();

if (b == 0)

throw new EX ("异常：除数不能为0！");

else

System.out.print ( (a / b) );

} catch (EX e) {

System.out.println (e.getMessage() );

}

}

}

class EX extends Exception {

public EX (String message) {

super (message);

}

}

# 实验6：

1. **实验目的**
2. **实验要求**
3. **实验设备与环境**
4. **设计思路与具体实现**

1.

2.

1. **实验总结**
2. **附录（程序代码）**

# 实验7：

1. **实验目的**
2. **实验要求**
3. **实验设备与环境**
4. **设计思路与具体实现**

1.

2.

1. **实验总结**
2. **附录（程序代码）**

# 实验8：

1. **实验目的**
2. **实验要求**
3. **实验设备与环境**
4. **设计思路与具体实现**

1.

2.

1. **实验总结**
2. **附录（程序代码）**

# 实验9：

1. **实验目的**
2. **实验要求**
3. **实验设备与环境**
4. **设计思路与具体实现**

1.

(1) 设计思路

(2) 实验步骤

①、

②、

③、

(3) 关键代码

(4) 程序测试过程

(5) 运行结果（运行效果）

2.

1. **实验总结**
2. **附录（程序代码）**