黑龙江大学软件学院

“面向对象程序设计”实验讲义

**2024年3月**

**目 录**

[第一部分 课程目的与任务 1](#_Toc162458334)

[一、课程基础 1](#_Toc162458335)

[二、适应对象 1](#_Toc162458336)

[三、教学目的 1](#_Toc162458337)

[四、内容提要 1](#_Toc162458338)

[五、实验成绩构成 2](#_Toc162458339)

[第二部分 主要内容及基本要求 3](#_Toc162458340)

[实验1 Java 程序的编辑、编译、运行环境 3](#_Toc162458341)

[实验2 类和对象的使用 7](#_Toc162458342)

[实验3 封装性、继承性与包 11](#_Toc162458343)

[实验4 Object类 15](#_Toc162458344)

[实验5 数组与异常处理 17](#_Toc162458345)

[实验6 链表类及应用 20](#_Toc162458346)

[实验7 文件管理器 26](#_Toc162458347)

## 

# 第一部分 课程目的与任务

一、课程基础

在学这门课之前，学生必须修读过C语言程序设计课程，并具备一定的C语言程序设计能力。

二、适应对象

软件工程专业

三、教学目的

结合对实例的开发，以学生为主体，充分发挥学生的主观能动性，引导学生自学教材部分内容，并锻炼学生自行开发各种小型程序模块；培养和锻炼学生利用Java语言进行网络编程的能力。

“面向对象程序设计”实验由若干个独立的实验项目组成，其主要目标是：

1. 加深学生对课堂讲授内容的理解，从计算机语言的基本概念、面向对象程序设计的基本方法、语法规则等方面加深理解，打好面向对象程序设计、软件开发的良好基础。
2. 在上机实验中，提高学生对Java 语言各部分内容的综合使用能力，逐步掌握Java语言程序设计的规律与技巧。
3. 在对Java 程序的调试过程中，提高学生分析程序中出现的错误和排除这些错误的能力。
4. 培养学生编写程序过程中形成良好的编码规范。
5. 通过上机实践，掌握IDEA、Eclipse等开发环境的使用，加深学生对计算机软件运行环境、Java 集成开发环境的了解。

四、内容提要

1. 本课程以实验为主，配合理论课的教学。任课教师需向学生讲清课程的性质，任务，要求，课程安排和进度，平时考核内容，期末考试办法，实验守则及实验室安全制度等。
2. 本课程主要设置下列类型的实验：验证性、设计性实验以及综合实验。整个实验过程包括课前准备，实验操作，实验报告等环节。学生在实验前必须进行准备。
3. 根据实验内容不同，基本操作性的实验1 人1 组，设计性实验可根据情况多人一组，每个实验要求在规定时间内由学生独立完成。
4. 实验过程中，老师应在实验室进行巡视，及时回答问题，纠正学生的错误操作，检查学生的实验报告。
5. 任课教师要认真备课，提前预做实验，上好每一堂课。实验前清点学生人数。
6. 实验的验收将分为两个部分。第一部分是上机操作，包括设计结果的源程序的验收与检查。第二部分是提交书面的实验报告。

五、实验成绩构成

表0.1 实验成绩构成表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **项目号** | **实验项目名称** | **学时** | **考核标准** | **分值** |
| 实验1(第8周) | Java 程序的编辑、编译、运行环境 | 2 | 见正文 | 10 |
| 实验2(第9周) | 类和对象的使用 | 2 | 见正文 | 10 |
| 实验3(第10周) | 封装性、继承性与包 | 2 | 见正文 | 10 |
| 实验4(第11周) | Object类 | 2 | 见正文 | 10 |
| 实验5(第12周) | 数组与异常处理 | 2 | 见正文 | 10 |
| 实验6(第15周) | 链表类及应用 | 2 | 见正文 | 10 |
| 实验7(第16-17周) | 文件管理器 | 4 | 见正文 | 20 |
| 实验报告，课堂表现 |  |  |  | 20 |
| 实验总成绩 | | | | 100 |

# 第二部分 主要内容及基本要求

请在开始做实验前认真阅读“基本要求”和“实验提示”，并在实验后领会“实验目的”，适当完成“思考题”。

（本讲义中的红色文字为注意事项或重要提示）

## 实验1 Java 程序的编辑、编译、运行环境

**●实验目的：**

1. 掌握JDK的安装与配置方法；
2. 能够编写简单的Java程序。

**●参考学时：2学时**

**●基本要求：**

1. 设置path环境变量（若尚未安装JDK，先安装该软件；若Path已设置，重新设置该值）；
2. 编写HelloWorld程序，并在命令模式下编译运行；
3. 在Eclipse或IDEA环境下编写计算整数N（N<=20）的阶乘程序FactorialTest类；
4. 在Eclipse或IDEA环境下编写按图1.1格式输出乘法口诀表的程序NineMultiplication。

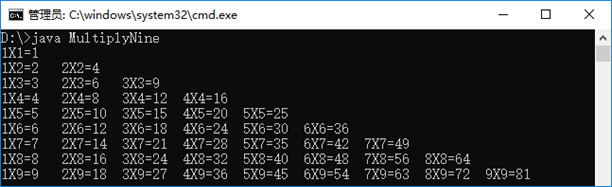


图1.1乘法口诀表程序运行效果

1. 标识符命名和代码格式符合规范。

注：其中，每行的多个乘法公式之间用‘\t’分割

**●实验提示：**

1. path是Windows系统的一个环境变量，内容为分号分隔的若干个文件夹名称（如“C:\Windows;C:\Windows\System32;C:\java\bin”）。在CMD窗口中输入一个命令时，如果没有指定命令文件所在的文件夹位置，Windows首先在命令行提示符所表示的文件夹（称作“当前文件夹”或“默认文件夹”）内查找命令文件。如果此文件夹内确实存在命令文件，则开始运行该命令，否则依次查看path环境变量中的每个文件夹，直到找到命令文件或查找失败为止。根据上述特点，我们可以把包含“javac.exe”命令文件和“java.exe”命令文件的文件夹（通常为“c:\program files\java\jdk1.6.xxx\bin”）添加到path环境变量中，从而简化CMD窗口中输入的命令。
2. Eclipse环境下应首先点击菜单“文件”-“新建”-“Java项目”创建Java项目（图1.2）。创建项目后点击菜单“文件”-“新建”-“类”（或按下“Alt-Shift-n”并选择“类”）在项目中新建Java类（图1.3）：

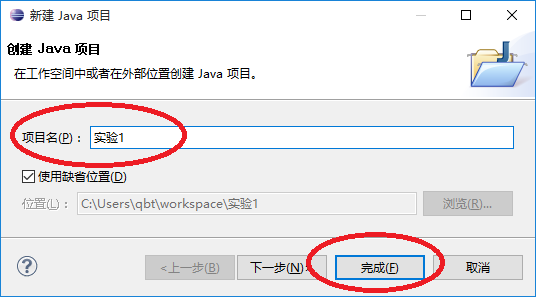


图1.2 Eclipse环境下创建Java项目

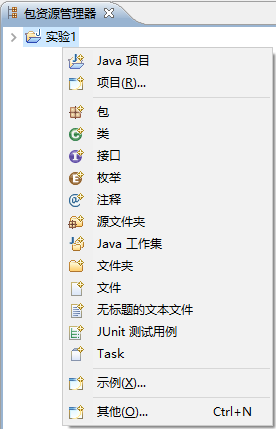


图1.3“Alt-Shift-n”菜单

1. 如果用函数的递归调用来实现阶乘计算部分，定义函数时应声明为static，如：

static long getFact(int i){

…

}

1. IDEA环境下，选择“File”-“New”-“Project…”菜单打开“New Project”窗口，并输入项目名称。整个课程的所有实验都在一个项目下完成，因此项目名称应为实验相关的全局名称，而不是某一个实验的名称：

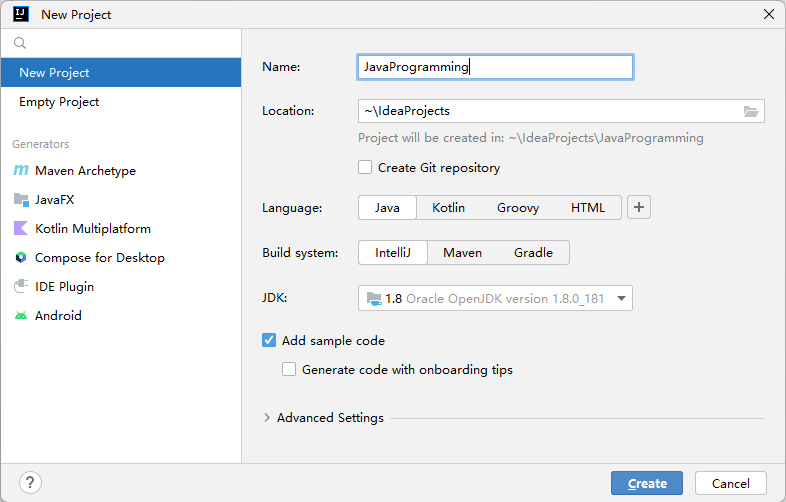


图1.4 IDEA环境下创建Java项目

创建项目后，在“src”文件夹上点击鼠标右键，选择“New”-“Java Class”，或选择“File”-“New”-“Java Class”菜单，打开“New Java Class”窗口，并输入Java类名称并回车（注意，暂时不要切换此时列表中的“Class”）：

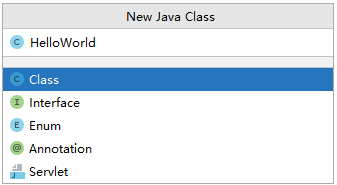


图1.5 IDEA环境下创建Java类

**●评分标准：满分10分**

1. 按要求正确完成所有任务（4项任务各2分，合计8分）
2. 编码、调试操作熟练；输出正确，界面符合要求（1分）
3. 实验报告撰写规范，内容完整；（1分）

**●测试案例：**

表1.1 测试案例（仅用于编码测试）

|  |  |  |  |
| --- | --- | --- | --- |
| 编号 | 测试目的 | 输入或测试数据 | 期望结果 |
| 1 | path环境变量设置 | 在cmd窗口中分别输入java和javac命令 | 不会出现“…不是内部或外部命令，也不是可运行的程序…” |
| 2 | 阶乘算法 | 无 | 阶乘结果正确 |
| 3 | 乘法口诀 | 无 | 乘法口诀表显示格式正确 |

**●思考题**

1. 如果阶乘结果溢出，该怎么办？
2. 若要输出如下乘法口诀表，该如何修改代码？

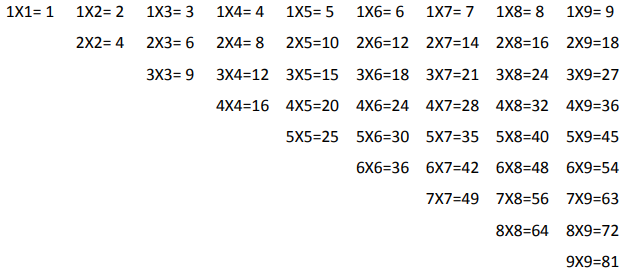


图1.4 第二种输出格式

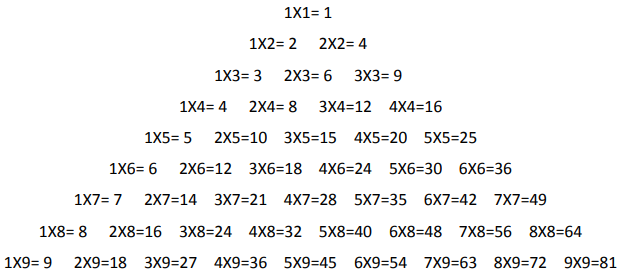


图1.5第三种输出格式

## 实验2 类和对象的使用

**●实验目的：**

1. 掌握类的定义过程，理解利用类进行封装的意义：定义属性和成员方法，定义构造方法，构造方法重载，this关键字（this前缀，this方法）；理解构造方法的作用
2. 掌握编写setter和getter方法，理解setter和getter方法的作用
3. 掌握对象的创建过程：对象与类的关系，构造方法与对象创建的关系
4. 掌握类的组合关系：通过组合定义更为灵活和复杂的类，同时达到代码重用目的

**●参考学时：2学时**

**●基本要求：**

1. 实现一个Point类，该类包含表示坐标的两个int型变量x、y，构造方法Point()和Point(int x, int y),返回x值和y值的int getX()和int getY()方法，设置x值和y值的void setX(int x)和void setY(int y)方法，计算两点间距离的double distance(Point p)方法。其中计算平方根的方法是Math.sqrt()，如：double d=Math.sqrt(2);
2. 实现一个Circle类，该类包含表示圆心的Point型变量center，表示半径的int型变量radius，以及构造方法Circle()、Circle(int x,int y,int r)、Circle(Point c,int r)，返回周长和面积的double perimeter()、double area()方法，返回两个圆是否为同一个圆（返回0）、同心圆（返回1）、相交的圆（返回2）、分离的圆（返回3）、包含的圆（返回4）等关系的int relation(Circle c)等方法。π值可以用Math.PI常量。
3. 实现测试上述两个类的Test类。该类在main方法中分别创建若干个Point对象和Circle对象，并调用相关方法，输出方法的返回值，验证其正确性。
4. 根据relation()方法的返回值，main方法输出“同一圆”、“同心圆”等内容。

**●实验提示：**

1. 同一类对象的属性值不同，对象之间才会有区别，所以创建对象后需要对其属性进行适当赋值，此过程俗称对象的“初始化”。由于对象创建时自动调用构造方法，所以把初始化工作交给构造方法完成更加合理，这就是为什么定义构造方法的原因。
2. 为了初始化方便，面向对象技术要求类的编写人员应提供各种构造方法，即运用构造方法重载。其结果是使用该类的其他开发人员可以根据自己的需要决定调用哪一个构造方法。
3. 不能在构造方法中编写任何输入输出有关的代码，否则该方法不仅具有初始化功能，还具有其它功能，从而违背面向对象程序设计的一个重要准则：“单一职责”原则。
4. Circle类将圆心定义为Point对象，这种现象称之为类和类之间的“组合关系”（Composition）。利用构造方法初始化Circle对象时，必须实例化center成员对象，以保证此类属性非空，否则将会引起空对象错误。如以下两个构造方法都没有对center实例化：（本讲义中的深色背景文字为示例代码）

Circle(){

radius=0;

}

Circle(int x,int y,int r){

center.x=x;

center.y=y;

radius=r;

}

正确的代码应该是：

Circle(){

center=new Point(0,0);

radius=0;

}

Circle(int x,int y,int r){

center=new Point(x,y);

radius=r;

}

1. Circle(Point c,int r)构造方法中，可以把c对象直接赋给center，因为c是调用该构造方法之前实例化的Point对象的引用。
2. Test类可以根据relation()方法的返回值显示两个圆的位置关系信息。在relation()方法中不应输出任何信息。

**●评分标准：满分10分**

1. 按要求正确完成所有任务；标识符命名规范；（Point和Circle各3分，Test 2分，合计8分）
2. 编码、调试操作熟练；输出正确，界面符合要求；（1分）
3. 实验报告撰写规范，内容完整。（1分）

**●测试案例：**

表2.1 测试案例（仅用于编码测试）

|  |  |  |  |
| --- | --- | --- | --- |
| 编号 | 测试目的 | 输入或测试数据 | 期望结果 |
| 1 | Point类成员方法 | 创建p1=new Point()和p2=new Point(0,3) | P1.getX()值为0，p1.getY()值为0，p2.getX()值为0，p2.getY()值为3，p1.distance(p2) 值为3 |
| 2 | Circle()构造方法 | 创建Circle() | 圆心应为(0,0)，半径、面积和周长皆为0 |
| 3 | Circle(int,int,int)构造方法、其它成员方法 | 创建Circle(1,1,10) | 圆心应为(1,1)，半径为10，面积为314.159，周长为62.831 |
| 4 | Circle(int,int,int)构造方法中radius为负数 | 创建Circle(1,1,-1) | 圆心应为(1,1)，半径、面积和周长皆为0 |
| 5 | Circle(Point,int)构造方法 | 利用(1,1)点p创建Circle(p,10) | 圆心应为(1,1)，半径为10，面积为314.159，周长为62.831 |
| 6 | Circle(Point,int)构造方法中radius为负数 | 利用(1,1)点p创建Circle(p,-1) | 圆心应为(1,1)，半径、面积和周长皆为0 |
| 7 | Circle(Point,int)构造方法中Point为空对象 | 创建Circle(null,10) | 圆心应为(0,0)，半径为10，面积为314.159，周长为62.831 |
| 8 | Point类的set方法 | p.setX(10),p.setY(10) | 该点的坐标被变更为(10,10) |
| 9 | Circle类的set方法 | c.setRadius(20) | 该圆的半径被变更为20 |
| 10 | Circle类的set方法 | c.setRadius(-20) | 该圆的半径保持不变 |
| 11 | Circle类的set方法 | c.setCenter(new Point(20,20)) | 该圆的圆心被变更为(20,20)点 |
| 12 | Circle类的set方法 | c.setCenter(null) | 该圆的圆心保持不变 |
| 13 | relation方法 | 创建Circle()和Circle(0,0,0)，并调用relation() | 输出“同一圆” |
| 14 | relation方法 | 创建Circle()和Circle(0,0,1) 并调用relation() | 输出“同心圆” |
| 15 | relation方法 | 创建Circle(0,0,10)和Circle(1,1,5) 并调用relation() | 输出“包含的圆” |
| 16 | relation方法 | 创建Circle(0,0,10)和Circle(0,5,10) 并调用relation() | “相交的圆” |
| 17 | relation方法 | 创建Circle(0,0,10)和Circle(0,20,10) 并调用relation() | 输出“分离的圆” |

**●思考题**

1. 为什么distance()和relation()方法只需要一个参数？提供两个或更多的参数有什么缺点？
2. relation()方法只返回整数，而不是在方法内部直接输出判断结果。这么做有什么优点？
3. Circle(Point c,int r)构造方法中，如果c为空对象该如何处理？
4. 利用Point类，还可以实现哪些类似于Circle的类？
5. 还可以定义哪些圆的关系？

## 实验3 封装性、继承性与包

**●实验目的：**

1. 掌握包的定义和引用语法；
2. 理解封装性的目的：为什么定义private变量、public方法？
3. 掌握父子类之间的继承原则：哪些内容可以被继承，哪些内容不可以被继承
4. 掌握super关键字的用法：super前缀，super方法
5. 掌握方法覆盖的语法、注意事项
6. 进一步熟练掌握类的组合关系：Color和ColoredCircle之间也是组合关系

**●参考学时：2学时**

**●基本要求：**

1. 创建项目“Java实验”，将实验一、实验二分别组织到“实验1”、“实验2”包中，并合理设置实验2中的属性、方法的访问权限修饰符；
2. 在“实验3”包中编写颜色类“实验3.Color”，其实现要求包括：

* 包含三个颜色分量red、green和blue（取值范围必须为0-255）；
* 构造方法Color()和Color(int r,int g,int b)：注意取值范围规定
* 设置颜色值方法
* void setRed(int v)：注意取值范围规定
* void setGreen(int v) ：注意取值范围规定
* void setBlue(int v) ：注意取值范围规定
* 获取颜色值方法
* int getRed()；int getGreen()；int getBlue()

1. 编写“实验2.Circle”类的子类“实验3.ColoredCircle”类，该类的圆心、半径等属性均使用Circle中的定义。其余实现要求包括：

* 表示圆周颜色的Color类对象borderColor和圆心颜色对象centerColor
* 至少定义如下几个ColoredCircle构造方法：
* ColoredCirle()：将半径赋为0，将圆心赋为(0,0)点，将两个颜色赋为Color(0,0,0)对象
* ColoredCirle(Point center,int radius)：半径为radius，圆心为center，颜色为(0,0,0)
* ColoredCirle(Color centerColor,Color borderColor)：半径为0，圆心为(0,0)点，颜色分别为centerColor和borderColor
* ColoredCirle(Point center, int radius, Color centerColor, Color borderColor)：圆心为center，半径为radius，圆心颜色为centerColor，边框颜色为borderColor
* 设置颜色方法：
* void setCenterColor(Color c)
* void setBorderColor(Color c)
* 获取颜色方法：
* Color getCenterColor()
* Color getBorderColor()
* 重新定义的relation(Circle c)方法（方法覆盖）
* 在“实验3”包中编写测试ColoredCircle类的测试类Test，并在该类的main方法中调用ColoredCircle类的所有方法（包括从Circle继承来的方法），输出返回值(如有必要)，以验证其正确性

1. 标识符命名和代码格式符合规范。

**实验提示：**

1. 创建“Java实验”项目的目的是将之前的实验1、实验2的内容，以及今后的所有实验文件统一组织在一个项目内，不同的实验具有不同的包名，实验与实验之间可跨包复用。完整的项目结构应符合图3.1所示内容：

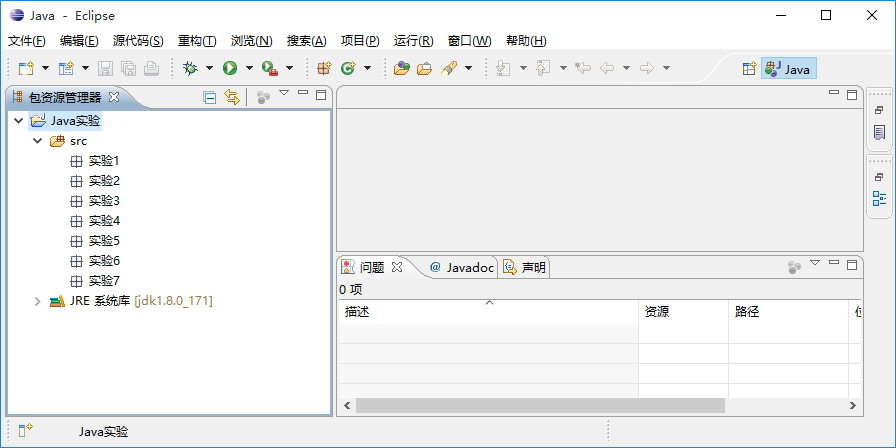


图3.1实验项目文件夹的浏览效果

1. Color类所表示的对象是“红绿蓝颜色模型”的简单抽象，即用三个整数（0-255范围）代表某颜色的三种原色取值。比如(red=0,green=0,blue=0)代表黑色，(red=255,green=0,blue=0)代表红色，(red=255,green=255,blue=255)代表白色。
2. ColoredCircle是具有额外属性和方法的Circle子类，但是其本质仍然是“圆”，因为ColoredCircle同样具有圆心、半径属性（隐藏于父类Circle中）和计算面积、计算周长的方法。
3. ColoredCircle类中唯一需要重写的父类方法是relation(Circle c)（注意relation(Circle c)方法的形参类型），因为在当前的抽象层次上，带颜色圆和一般圆的唯一不同点是圆和圆的关系不同，面积方法、周长方法、setter、getter方法无需改变。
4. relation(Circle c)方法首先应判断c是Circle对象还是ColoredCircle对象。如果c不是ColoredCircle对象，则调用父类的relation方法，并把同一圆关系调整为同心圆（即带颜色的圆和不带颜色的圆不可能是同一圆，最多是同心圆）。如果c是ColoredCircle对象，则可以在同一个圆的基础上进一步判断颜色是否一样（即首先调用父类的relation方法，并根据返回值判断是否为同一圆。如果是，则进一步判断颜色是否相同。只有圆心重合、半径相同，且颜色相同，则两个带颜色圆的关系才是同一圆；颜色不同的圆心重合、半径相同圆归类为同心圆。父类的relation方法返回值为非同一圆，则无需判断颜色）。
5. 任何一个类的缺省、无参构造方法将自动调用父类的无参构造方法。另一方面,任何子类都不能继承父类的构造方法。因此，在ColoredCircle类中编写上述若干构造方法，并在这些方法的开始处通过super()或super(…)调用父类的构造方法，以达到初始化父类中定义的成员变量目的。

**●评分标准：满分10分**

1. 按要求正确完成所有任务（ColoredCircle3分，Color和包各2分，Test1分，合计8分）
2. 编码、调试操作熟练；输出正确，界面符合要求（1分）
3. 实验报告撰写规范，内容完整；（1分）

**●测试案例：**

表3.1 测试案例（仅用于编码测试）

|  |  |  |  |
| --- | --- | --- | --- |
| 编号 | 测试目的 | 输入或测试数据 | 期望结果 |
| 1 | 成员的private、public修饰 | 创建实验2.Point()对象p和实验2.Circle()对象c | p.x、p.y、c.center、c.radius提示语法错误 |
| 2 | Color类构造方法 | 创建Color() | getRed()、getGreen()和getBlue()皆返回0 |
| 3 | Color类构造方法 | 创建Color(255,255,255) | getRed()、getGreen()和getBlue()皆返回255 |
| 4 | Color类构造方法 | 创建Color(355,355,355) | getRed()、getGreen()和getBlue()皆返回0 |
| 5 | Color类getter、setter方法 | 调用setRed(355)、setGreen(355)、setBlue(355) | getRed()、getGreen()和getBlue()皆返回创建颜色对象时的原始red、green、blue值 |
| 6 | ColoredCircle类构造方法 | 调用ColoredCirle()创建圆；  调用ColoredCirle(null,-1)；  调用ColoredCirle(null,null) | getRadius()、area()、perimeter()、getCenter().getX()、getCenter(). getY()、getBorderColor().getRed()、getCenterColor().getRed()等值皆为0 |
| 7 | ColoredCircle类构造方法 | 调用ColoredCirle(new Point(0,0), 10)； | getRadius()为10，area()为314.159，perimeter()为62.831，getCenter().getX()、getCenter(). getY()、getBorderColor().getRed()、getCenterColor().getRed()等值皆为0 |
| 8 | ColoredCircle类构造方法 | 调用ColoredCirle(  new Color(0,0,0),  new Color(1,1,1) ) | getRadius()为0；area()为0；perimeter()为0；getCenter(). getX()为0；getCenter().getY()为0；centerColor的所有get方法返回值皆为0；borderColor的所有get方法返回值皆为1 |
| 9 | ColoredCircle类构造方法 | 调用ColoredCirle(Point center, int radius, Color centerColor, Color borderColor) 创建圆：center为(1,1),radius为10，centerColor为(0,0,0)，borderColor为(1,1,1) | getRadius()为10；area()为314.159；perimeter()为62.831；getCenter(). getX()为1；getCenter().getY()为1；centerColor的所有get方法返回值皆为0；borderColor的所有get方法返回值皆为1 |
| 10 | ColoredCircle类构造方法 | 调用ColoredCirle(Point center, int radius, Color centerColor, Color borderColor) 创建圆：center为null,radius为-10，centerColor为null，borderColor为null | getRadius()为0；area()为0；perimeter()为1；getCenter(). getX()为0；getCenter().getY()为0；centerColor的所有get方法返回值皆为0；borderColor的所有get方法返回值皆为0 |
| 11 | ColoredCircle类setter方法 | setCenterColor(null) | 创建ColoredCircle时的原始centerColor值不变 |
| 12 | ColoredCircle类setter方法 | setBorderColor(null) | 创建ColoredCircle时的原始borderColor值不变 |
| 13 | 子类中编写的方法覆盖 | 调用Circle()创建圆c1，调用ColoredCircle()创建圆c2，并调用c2.relation(c1) | 显示“同心圆” |
| 14 | 子类中编写的方法覆盖 | 调用ColoredCircle()创建圆c1，调用ColoredCircle()创建圆c2，并调用c2.relation(c1) | 显示“同一圆” |
| 15 | 子类中编写的方法覆盖 | 调用ColoredCircle(centerColor， borderColor)创建圆c1，其中centerColor为(1,1,1)，borderColor为(1,1,1);调用ColoredCircle()创建圆c2，并调用c2.relation(c1) | 显示“同心圆” |

**●思考题**

1. 您还可以设计哪些Circle类的子类？
2. Color类所抽象的颜色种类总数是多少？
3. 可以在Color类中定义如下代码所示类常量，这么做有什么意义和优点？

pubic static final Color RED=new Color(255,0,0);

pubic static final Color GREEN=new Color(0,255,0);

pubic static final Color BLUE=new Color(0,0,255);

pubic static final Color WHITE=new Color(255,255,255);

pubic static final Color BLACK=new Color(0,0,0);

pubic static final Color GRAY=new Color(128,128,128);

## 实验4 Object类

**●实验目的：**

1. 掌握编写不同包的子类时设置成员访问权限；
2. 掌握Object类与其他类的关系以及toString()、equals(Object o)方法的作用；
3. 掌握方法覆盖时必须保证方法原型的一致性；
4. 掌握类型上转型（上溯）和下转型（下溯）的应用；

**●参考学时：2学时**

**●基本要求：**

1. 在“实验4”包中编写“实验2.Point”和“实验2.Circle”类的子类“实验4.Point”、“实验4.Circle”，并编写新的构造方法，重写equals()方法和toString()方法；
2. 在“实验4”包中编写测试类实验4.Test，并测试上述方法的正确性。
3. 标识符命名和代码格式符合规范。

**●实验提示：**

1. equals方法的参数类型必须是Object，访问修饰符必须是public，返回类型必须是boolean，否则该方法不是对从父类继承的public boolean equals(Object obj)方法的覆盖，至多算是该方法的重载。
2. Circle类的equals方法不能写成如下代码：

public boolean equals(Object obj) {

if(obj instanceof Circle){

Circle c=(Circle)obj;

return c.getRadius()==getRadius() &&

c.getCenter().equals(getCenter());

}

return false;

}

以上代码试图通过c.getCenter().equals(getCenter())语句调用实验4.Point类的equals方法，从而判断圆心是否为同一点。这种写法的错误之处在于实验4.Circle的center变量是实验2.Point类的对象，而实验2.Point类的equals方法并没有判断XY坐标值是否相同。

**●评分标准：满分10分**

1. 按要求正确完成所有任务（Point和Circle各3分，Test 2分，合计8分）
2. 编码、调试操作熟练；输出正确，界面符合要求（1分）
3. 实验报告撰写规范，内容完整；（1分）

**●测试案例：**

表4.1 测试案例（仅用于编码测试）

|  |  |  |  |
| --- | --- | --- | --- |
| 编号 | 测试目的 | 输入或测试数据 | 期望结果 |
| 1 | 成员的private、public修饰 | 构造方法中直接访问Point类的x、y或Circle类的center、radius | 提示语法错误 |
| 2 | 子类的构造方法 | 创建实验4.Point()对象p1和实验4.Point(0,10)对象p2 | p1.getX()结果为0，p1.getY()结果为0，p1.distance(p2)结果为10 |
| 3 | 子类的构造方法 | 创建实验4.Circle()对象c1和实验4.Circle(0,0,10)对象c2 | c1.getCenter().getX()结果为0，c1.getCenter().getY()结果为0，c1.area()为0，c1.perimeter()结果为0，调用c1.relation(c2)结果为“同心圆” |
| 4 | toString()方法 | 调用p1.toString() | 返回“(0,0)” |
| 5 | toString()方法 | 调用c1.toString() | 返回“(0,0),0” |
| 6 | equals()方法 | 创建实验4.Point()对象p1和实验4.Point(0,10)对象p2 | 调用p1.equals(p2)时返回false |
| 7 | equals()方法 | 创建实验4.Point()对象p1和实验4.Point(0, 0)对象p2 | 调用p1.equals(p2)时返回true |
| 8 | equals()方法 | 创建实验4.Point()对象p1和实验2.Point(0, 0)对象p2 | 调用p1.equals(p2)时返回true |
| 9 | equals()方法 | 创建实验4.Point()对象p1和实验2.Point(0, 0)对象p2 | 调用p2.equals(p1)时返回false（因为实验2.Point还没有重写equals方法） |
| 10 | equals()方法 | 创建实验4.Circle()对象c1和实验4.Circle(0,0,10)对象c2 | 调用c1.equals(c2)时返回false |
| 11 | equals()方法 | 创建实验4.Circle()对象c1和实验4.Circle(0,0,0)对象c2 | 调用c1.equals(c2)时返回true |
| 12 | equals()方法 | 创建实验4.Circle()对象c1、实验4.Point(0,0)对象p和实验4.Circle(p,0)对象c2 | 调用c1.equals(c2)时返回true  调用c1.equals(p)和p.equals(c1)皆返回false |
| 13 | equals()方法 | 创建实验4.Circle()对象c1、实验2.Point(0,0)对象p和实验4.Circle(p,0)对象c2 | 调用c1.equals(c2)时返回true |
| 14 | equals()方法 | 创建实验4.Circle()对象c1和实验2.Circle(0,0,0)对象c2 | 调用c1.equals(c2)时返回true，调用c2.equals(c1)时返回false |

**●思考题**

1. 为什么必须要重写子类的构造方法？
2. 为什么equals()方法和toString()方法的访问类型必须为public？
3. 在本实验中何处体现了面向对象技术的“上转型(上溯)”、“下转型(下溯)”概念？

## 实验5 数组与异常处理

**●实验目的：**

1. 掌握一维数组和二维数组的定义和初始化步骤；
2. 进一步掌握toString()方法的应用；
3. 掌握异常类的定义以及异常对象的实例化（new）、抛出（throw）、声明抛出（throws）、捕获（try…catch）等内容。

**●参考学时：2学时**

**●基本要求：**

1. 编写实现一维数组的反转类“实验5.ArrayReverser”。该类包含一个int[]数组data、构造方法ArrayReverser(int[] data)、获取data数组的方法int[] getData()、设置data数组的方法void setData(int[] data)、将data数组内容转换为字符串的toString()方法以及用于反转data数组内容（即第一个元素和最后一个元素互换、第二个元素和倒数第二个元素互换…）的方法void reverse()。
2. 编写ArrayReverser类的测试类实验5.ArrayTest。该类首先创建一个一维数组，并将其内容用随机数填充；其后利用上述一维数组创建ArrayReverser对象，并通过toString()方法查看初始数据；最后调用reverse()方法后再次调用toString()查看反转后的数据。
3. 编写矩阵类实验5.Matrix。该类包括矩阵数据数组double data[][]，构造方法Matrix(int rows,int cols)、Matrix(double data[][])，获取某元素值的方法double getData(int row,int col)，设置某元素值的方法void setData(int row,int col,double value)，获取矩阵行数方法int getRows()，获取矩阵列数方法int getCols()，计算两个矩阵的乘积的方法Matrix multiply(Matrix m)以及equals()、toString()等内容。注意，该类中不能定义行数成员变量rows或列数成员变量cols。
4. 编写测试类实验5.MatrixTest，测试Matrix类的正确性。
5. 编写如下三个异常类，这些类只需要包含toString方法即可：
   * + 矩阵行数或列数非法异常类IllegalArgumentException（注意，java.lang包中已经有同名的异常类，本实验要求不能直接使用该类，而是重新定义一个）
     + 矩阵行号或列号非法异常类IllegalIndexException
     + 矩阵无法相乘异常类MatrixMultiplicationException
6. 完善Matrix类的相关方法，使其在不正确的调用情况下抛出适当的异常对象。
7. 完善MatrixTest类，以测试异常类的定义和Matrix对异常类的应用是否有效。
8. 标识符命名和代码格式符合规范。

**●实验提示：**

1. ArrayReverser(int[] data)构造方法和setData(int[] data)中尽量申请新的this.data数组，并把data形参内容复制到新数组中。
2. Matrix类是对现实生活中矩阵的抽象，而不是对二维数组的抽象。通过实例化该类对象，并赋与适当数据值，得到所需矩阵。
3. 注意矩阵相乘方法的原型是Matrix multiply(Matrix m)，因为两个矩阵相乘的结果仍然是矩阵，而不是二维数组。
4. data不为空的前提下，矩阵行数是data.length值，列数是data[0].length值。
5. getData(row,col)方法的功能是得到矩阵的row行col列值。setData(row,col,value)方法的功能恰好相反，将矩阵的row行col列设置为value值。如getData(1,2)返回矩阵的1行2列值（一个double值），setData(2,3,1.5)将矩阵的2行3列值赋为1.5。
6. 在Matrix类的所有构造方法中，如果行数或列数值小于1，或形参data为空，则抛出IllegalArgumentException异常；
7. 在getData(…)和setData(…)方法中，如果行号或列号大于等于矩阵行数或列数，或小于0，则抛出IllegalIndexException异常；
8. 在multiply(…)方法中，如果形参为空对象，或两个矩阵的行列数不满足矩阵相乘规则，则抛出MatrixMultiplicationException异常。
9. MatrixTest类中，通过预先设定的常数或键盘输入的值确定所要创建的两个矩阵的行列数。建议实例化矩阵对象之后，生成一组随机数，并循环调用setData(…)方法为矩阵某行某列赋值。键盘输入方式效率低，不利于程序的调试。
10. 初始化矩阵对象之后，计算矩阵的乘积，并把结果通过toString方法的返回值输出到屏幕上，如某一2\*3矩阵的输出结果为：

0,2,3

2,1,1

输出两个矩阵的乘积语句应该是：

System.out.println(m.multiply(n));

**●评分标准：满分10分**

1. 按要求正确完成所有任务（ArrayReverser类1分，Matrix类3分，每个异常类各1分，两个Test类 1分，合计8分）
2. 编码、调试操作熟练；输出正确，界面符合要求（1分）
3. 实验报告撰写规范，内容完整；（1分）

**●测试案例：**

表5.1 测试案例（仅用于编码测试）

|  |  |  |  |
| --- | --- | --- | --- |
| 编号 | 测试目的 | 输入或测试数据 | 期望结果 |
| 1 | ArrayReverser构造方法 | new ArrayReverser(null) | 创建data数组长度为0的对象 |
| 2 | ArrayReverser构造方法  getData方法 | new ArrayReverser(data) | 利用data数组内容创建ArrayReverser，  getData()返回的数组内容应与原始data内容一样 |
| 3 | ArrayReverser类setData方法 | ar.setData(null) | ar对象data内容不变 |
| 4 | ArrayReverser类setData方法 | ar.setData(data) | ar对象data内容被形参data内容替换（数组长度也可能会发生变化） |
| 5 | ArrayReverser类reverse()方法 | ar.reverse() | ar对象data内容被反转 |
| 6 | ArrayReverser类toString()方法 | ar.toString() | 返回“3 6 1 5 7 34 65”形式字符串 |
| 7 | Matrix类构造方法 | new Matrix(2,3) | 创建2\*3矩阵，每项数据值全为0（数组被new后的默认值） |
| 8 | Matrix类构造方法 | new Matrix(data) | 创建以data数组为内容的矩阵对象 |
| 9 | Matrix类构造方法 | new Matrix(0,3) | 抛出IllegalArgumentException异常（行数列数必须大于0） |
| 10 | Matrix类构造方法 | new Matrix(null) | 抛出IllegalArgumentException异常（参数不能为空） |
| 11 | 获得列数 | m.getCols() | 返回列数（>0） |
| 12 | 获得行数 | m.getRows() | 返回行数（>0） |
| 13 | getter方法 | m.getData(0,0) | 返回0行0列double值 |
| 14 | setter方法 | m.setData(0,0,10) | 将0行0列值改为10 |
| 15 | getter方法 | m.getData(-1,0)  m.getData(m.getRows(),0) | 抛出IllegalIndexException异常（行号<0，行号>=行数） |
| 16 | setter方法 | m.setData(0,-1)  m.setData(0,m.getCols()) | 抛出IllegalIndexException异常（列号<0，列号>=列数） |
| 17 | multiply方法 | m1.multiply(m2) | 返回一个新矩阵，值为m1\*m2 |
| 18 | multiply方法 | m1.multiply(null) | 抛出MatrixMultiplicationException异常 |
| 19 | multiply方法 | m1.multiply(m2)，同时m1的列数不等于m2的行数 | 抛出MatrixMultiplicationException异常 |
| 20 | toString方法 | m1.toString() | 返回“1 2\n3 4\n”形式的字符串 |

**●思考题**

1. 还可以为Matrix类添加哪些方法？
2. 在Matrix类中不能再定义行数和列数成员变量rows或cols，为什么？
3. 为什么在Matrix类的构造方法中不能用随机数直接将二维数组赋值？

## 实验6 链表类及应用

**● 实验目的：**

1. 进一步掌握Object类的主要特点；
2. 进一步掌握上转型、下转型的含义；
3. 掌握equals方法的具体应用；
4. 掌握开发一个链表类的关键步骤。
5. 利用链表类实现复杂程序。
6. 掌握接口的概念和语法，并能够应用于实际。

**● 参考学时：2学时**

**● 基本要求：**

1. 定义实验61.Node类和实验61.LinkedList类，并通过测试类测试上述类的正确性。LinkedList类组合了Node类，Node类与LinkList类之间的关系如图6.1所示。图中空心菱形的含义是LinkedList类聚合(Aggregation)了Node类对象，聚合是组合关系的一种。
2. 注意，Java语言已经提供了一个java.util.LinkedList类，请不要在本实验中导入并使用该类，而是重新定义“实验61.LinkedList”。

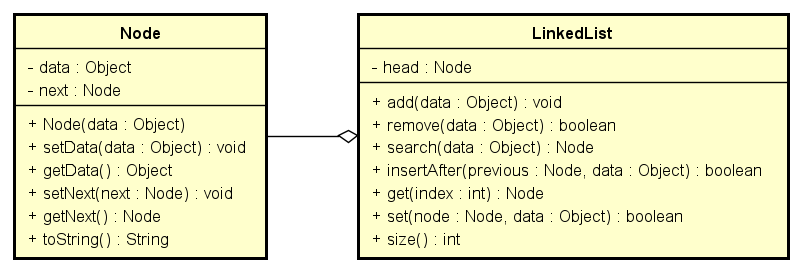


图6-1 Node类和LinkedList类之间的关系

1. Node类：

* 要求Node类拥有如下2个私有属性：
  1. data：用于存放节点的数据。为了能够实现常见数据类型的链表，data类型定义为Object，即可以在链表中保存任何类型的对象。如果要保存int、double等基本数据类型，需要用到Java的自动“装箱”和“拆箱”机制。
  2. next：用于保存本节点的下一个节点的引用。
* 要求Node类里有一个构造方法Node(Object data)，此外还拥有设置属性和获取属性的getter、setter方法以及简单的toString()方法（该方法将data转换为String并返回该值）。

1. LinkedList类

* 要求LinkedList类拥有1个private属性head，用于存储链表第一个节点的引用。
* 要求为LinkedList实现7个public方法，关于这7个方法的要求如表6.1所示。
* 实现search()方法和remove()方法时需要用到链表中节点的数据所属类提供的equals()方法，因为这两个方法都涉及到比较链表中某一Node的getData().equals(所要查找的data)，而不是getData()== data。

表6.1 LinkList类的方法

|  |  |  |
| --- | --- | --- |
| 编号 | 方法名称 | 描述 |
| 1 | void add(Object data) | 将值为data的节点插入到链表尾。 |
| 2 | boolean remove(Object data) | 在链表中删除值为data的节点。若删除成功，返回true；否则，返回false。注意，移除后的节点应与链表没有任何链接关系。 |
| 3 | Node search(Object data) | 在链表中查找值为data的节点。若找到，返回该节点的引用，若没找到，返回null。 |
| 4 | boolean insertAfter(Node previous, Object data) | 在引用previous指向的节点后插入一个值为data的节点。若插入成功，返回true；否则，返回false。 |
| 5 | Node get(int index) | 在链表中找第index个节点。若找到，返回第index节点的引用，若找不到，返回null。 |
| 6 | boolean set(Node node, Object data) | 将链表中的node节点的内容改为data。若修改成功，返回true；否则，返回false。 |
| 7 | int size() | 返回链表长度。 |

一个初步的具有add方法的LinkedList代码如下：

public class LinkedList {

private Node head;

public void add(Object data) {

Node node=new Node(data); //创建包含data值的Node节点

if(head==null) //链表为空，则当前节点就是第一个节点

head=node;

else { //链表不为空，需要找到最后一个节点

Node tmp=head; //利用tmp节点遍历

while(tmp.getNext()!=null) //tmp节点不是最后一个节点

tmp=tmp.getNext(); //tmp指向下一个节点

//循环结束时tmp为最后一个节点

tmp.setNext(node); //把当前节点链接在最后一个节点之后

}

}

}

1. 实验61.Test类（测试类，其中含主方法）

* 要求在测试类的主方法中，测试LinkedList类的方法，可按如下步骤执行：

① 调用无参构造方法LinkedList()，创建一个空链表。

② 连续三次调用add()方法，在链表中插入三个Integer对象，如add（1）； add（2）； add（3）（此处Java自动将1、2、3等int值装箱为Integer对象）。

③ 调用search()方法，查找值为2的节点，将该方法返回的节点的引用赋给Node型变量p；

④ 调用set()方法，将p指向的节点的内容修改为22。

⑤ 调用insertAfter()方法，在p指向的节点后面插入一个值为23的节点。

⑥ 调用remove()方法，删除值为22的节点。

⑦ 调用size()方法，获得链表长度赋给整型变量n。

⑧ 循环n次，每一轮通过调用get(i)方法得到第i个节点的引用，并将其内容输出。

1. 在“实验61.LinkedList”和“实验61.Node”基础上，实现宠物商店的宠物上架、下架、查询等操作，其中宠物商店中的宠物需要用链表存储。在本程序中，最重要的是定义宠物标准（定义一个宠物接口Pet），进入宠物商店销售的所有宠物（如Cat类、Dog类、Fish类）都需要实现这个标准（即实现接口Pet）。根据以上原则，给出了图6.2所示的类图（图中的虚线三角形箭头表示实现接口，实线普通箭头表示关联(Association)关系）。

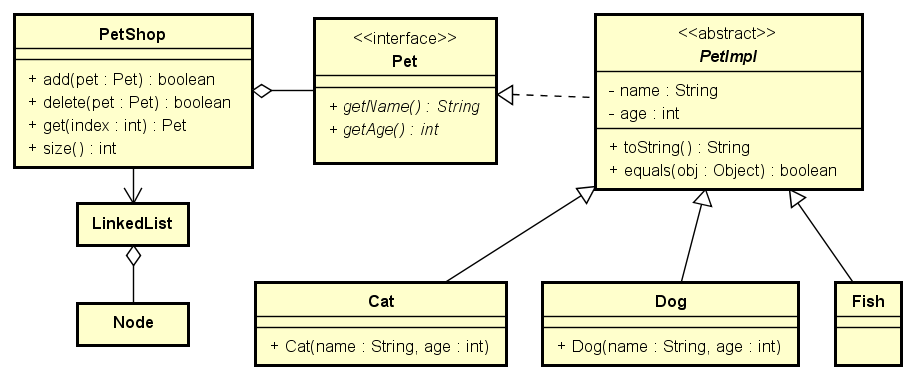
****

图6.2 实验相关的类、接口及其关系

* 定义宠物接口实验62.Pet

这个接口是宠物商店可处理的宠物标准。宠物商店并不关心具体的宠物是什么，只关心一点：只要是实现了标准接口的宠物，就可以进入宠物商店，并能够通过宠物商店进行各种操作。宠物接口Pet的定义如下：

package 实验62;

interface Pet {

public String getName();

public int getAge();

}

* 定义宠物商店类实验62.PetShop

宠物商店所售卖的宠物数量不定，因此要用链表（实验61.LinkedList）对象保存多个宠物。同时宠物商店能够售卖的宠物必须是具有宠物特征（比如具有名字和年龄）和行为（比如能够获知名字或年龄）的一切动物，因此在上架（添加）、卖出（删除）宠物时，接收的参数都应是Pet接口类型。

表6.2 宠物商店类的方法

|  |  |  |
| --- | --- | --- |
| 编号 | 方法名称 | 描述 |
| 1 | boolean add(Pet pet) | 将宠物pet上架，即添加到链表中。若上架成功返回true，否则返回false。注意，不能出现多个同名且同年龄的任何类型宠物。 |
| 2 | boolean delete(Pet pet) | 从宠物商店下架宠物pet，即从链表中查找并删除。若删除成功，返回true；否则，返回false。 |
| 3 | Pet get(int index) | 查找第index个宠物。若找到，返回该宠物的引用，若找不到，返回null。 |
| 4 | int size() | 返回宠物总数。 |

**提示**：在宠物商店类中，先定义一个LinkedList类变量pets，并实例化。

package 实验62;

import 实验61.\*;

public class PetShop { // 一个宠物商店要保存多个宠物信息

private LinkedList pets = new LinkedList(); //用链表保存宠物信息

public boolean add(Pet pet) { //新宠物上架操作，以接口对象为参数！

…//添加同名同年龄的任何宠物，都视为无效，返回false

}

public boolean delete(Pet pet) { //宠物下架操作，以接口对象为参数！

**…**

}

public Pet get(int index){

//pets.get()返回Node节点，Node节点的getData()返回Object，该Object就是Pet

**…**

}

**…**

**}**

* 定义宠物的一般实现类实验62.PetImpl

Pet接口的实现类，该类不代表具体哪种宠物，而是定义宠物类的一般框架：名字name、年龄age等属性，基本的构造方法PetImpl(String name, int age)，以及需要实现的getName、getAge、toString、equals等方法。由于该类不代表具体宠物的抽象，因此将其定义为abstract类。

* 定义宠物猫子类实验62.Cat

要求Cat类继承PetImpl。Cat类只需要有构造方法Cat(String name, int age)。

* 定义宠物狗子类实验62.Dog

要求Dog类继承PetImpl。Dog类只需要有构造方法Dog(String name, int age)。

* 编写测试类实验62.Test。要求在测试类的主方法中，完成如下动作：

① 定义一个宠物商店类变量shop，并实例化；

② 在宠物商店shop中添加1岁的“波斯猫”；

③ 在宠物商店shop中添加2岁的“橘猫”；

④ 在宠物商店shop中添加1岁的“折耳猫”；

⑤ 在宠物商店shop中添加1岁的“柯基犬”；

⑥ 在宠物商店shop中添加2岁的“波尔多狗”；

⑦ 显示宠物商店shop中所有宠物；

⑧ 删除宠物商店shop中2岁的“橘猫”；

⑨ 显示宠物商店shop中所有宠物；

通过运行结果可以发现，由于本程序是面向接口的编程，所以返回的结果中即包含了Cat类对象，也包含了Dog类对象。

1. 标识符命名和代码格式符合规范。

**● 评分标准：满分10分**

1. 按要求正确完成所有任务（8分）
2. 编码、调试操作熟练；输出正确，界面符合要求（1分）
3. 实验报告撰写规范，内容完整；（1分）

**●测试案例：**

表6.4 链表类测试案例（仅用于编码测试）

|  |  |  |  |
| --- | --- | --- | --- |
| 编号 | 测试目的 | 输入或测试数据 | 期望结果 |
| 1 | 构造方法 | list=new LinkedList() | list.size()返回0；  list.get(0)返回null；  list.search(1)返回null；  list.remove(1)返回false |
| 2 | add方法 | list.add(1); list.add(2); list.add(3); | list.size()返回3 |
| 3 | search方法 | Node p= list.search(22); | System.out.println(p)显示null |
| 4 | search方法 | Node p= list.search(2); | System.out.println(p)显示2 |
| 5 | set方法 | list.set(null,22); | set方法返回false |
| 6 | set方法 | list.set(p,22); | System.out.println(p)显示22 |
| 7 | insertAfter方法 | list.insertAfter(p,23); | list.size()返回4 |
| 8 | insertAfter方法 | list.insertAfter(null,23); | insertAfter方法返回null |
| 9 | remove方法 | list.remove(12) | 返回false |
| 10 | remove方法 | list.remove(22) | 返回true |
| 11 | get方法 | Node p=list.get(0); | System.out.println(p)显示1 |
| 12 | get方法 | Node p= list.get(-1); | System.out.println(p)显示null |
| 13 | get方法 | Node p= list.get(100); | System.out.println(p)显示null |

表6.5 宠物商店测试案例（仅用于编码测试）

|  |  |  |  |
| --- | --- | --- | --- |
| 编号 | 测试目的 | 输入或测试数据 | 期望结果 |
| 1 | 构造方法  size方法  get方法  delete方法 | shop=new PetShop() | shop.size()返回0；  shop.get(0)返回null；  shop.delete(null)返回false |
| 2 | add方法  get方法  toString方法 | shop.add(new Cat("波斯猫",1));  shop.add(new Cat("橘猫",2));  shop.add(new Cat("折耳猫",1));  shop.add(new Dog("柯基犬",1));  shop.add(new Dog("波尔多狗",2));  shop.add(new Dog("波尔多狗",2));  shop.add(new Cat("波尔多狗",2)); | 前五个add返回true，后两个返回false  shop.size()返回5  shop.get(0).toString()返回“波斯猫，1岁” |
| 3 | delete方法 | shop.delete(new Cat("橘猫",1)); | 返回false，因为商店中只有2岁橘猫，没有1岁橘猫  shop.size()返回5 |
| 4 | delete方法 | shop.delete(new Cat("波斯猫",1)); | 返回true  shop.size()返回4，因为已经删除了一个波斯猫 |

**● 思考题**

1. Node类中重写toString()有什么特殊意义？
2. 在本实验中Node中的data类型不是Object而是Integer可不可行？为什么？
3. 假如链表中保存任意一个自定义类型的对象，上述Node和LinkedList的功能是否依然正确？如果不正确，需要进行哪些补充？
4. Cat、Dog类的父类PetImpl重写equals()有什么特殊意义？不重写会有什么后果？
5. 还可以为Pet接口或PetShop类添加哪些方法？

## 实验7 文件管理器

**● 实验目的：**

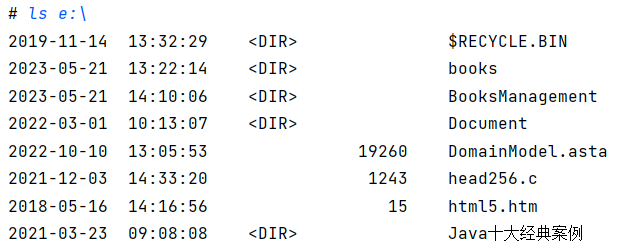
1. 掌握基本的目录和文件操作技术；
2. 掌握简单的命令行解释操作；

**●参考学时：4学时**

**●基本要求：**

1. 编写类似于Linux Shell或Windows CMD程序的字符界面文件管理程序SimpleShell类。
2. 实验过程中不允许使用java.nio.file.Files类，否则很多功能都可以利用该类实现，达不到熟悉和掌握文件操作基本过程的目的。
3. 该程序循环接收输入的每行命令，并根据所输入命令执行对应功能。命令提示符固定为“#空格”，即用户在该提示符后面输入命令。
4. 必须实现ls、cp、mv、rm四个命令功能和exit命令，也可额外实现md、rd、cat等命令，各命令的输入格式和功能说明如下（<文字>表示必须输入括号内文字，同时不能输入括号）：

* exit：退出程序。
* ls<空格><路径>：在命令行下列出指定“路径”下的所有子目录和文件信息，显示格式如图所示。其中第一列和第二列为子目录或文件的最后一次修改年月日时分秒，第三列为“<DIR>”（表示该行对应项为子目录）或文件字节数（表示该行对应项为文件），第四列为子目录或文件名称。



* cp<空格><源文件路径><空格><目标文件路径>：将源文件内容复制到目标文件，文件名可以不同。
* mv<空格><源文件路径><空格><目标文件路径>：将源文件内容移动到目标文件，文件名可以不同。
* rm<空格><文件路径>：删除指定文件
* md<空格><子目录路径>：创建指定路径对应的子目录
* rd<空格><子目录路径>：删除指定路径对应的非空子目录
* cat<空格><文件路径>：显示指定路径对应的文本文件内容

1. 标识符命名和代码格式符合规范。

**实验提示：**

1. 注意，测试时不要移动或删除硬盘，尤其是C盘原有文件或文件夹，否则有误删系统文件的风险。
2. 实现命令行提示的基本语句如下：

//定义用于接收键盘输入信息的流kbin

BufferedReader kbin=new BufferedReader(new InputStreamReader(System.in));

while(true){

//显式命令行提示符

System.out.print("# ");

//接收命令行输入

String cmd=kbin.readLine();

//命令行内容为"exit"（不区分大小写）

if(cmd.toLowerCase().equals("exit")){

//退出程序

System.exit(0);

}

//命令行以"ls"开头（不区分大小写）

else if(cmd.toLowerCase().startsWith("ls")){

//调用完成ls命令功能的方法

ls(cmd);

}

//依次处理其它命令

else if(...){

}

else if(...){

}

//不能识别的命令，提示错误

else{

System.out.println("无法识别所输入命令:"+cmd);

}

}

1. 获取文件或目录的大小、日期等属性信息需要用到File对象；可以利用SimpleDateFormat类将文件或目录的日期信息转换为特定格式：

//创建某一已存在文件files[i]的File对象，dir为父目录

File f=new File(dir,files[i]);

//将文件日期转换为String对象的格式：4位年、2位月、2位日、24小时制小时、2位分、2位秒

String df="yyyy-MM-dd HH:mm:ss";

//将文件日期转换为Date对象

Date d=new Date(f.lastModified());

//将文件日期Date对象转换为给定格式String对象

String sdf=new SimpleDateFormat(df).format(d);

//利用printf输出日期时间

System.out.printf("%s\t",sdf);

//如果是文件，则输出文件大小，输出格式为为16位长度，靠右对齐

if (f.isFile())

System.out.printf("%16d\t",f.length());

//如果是目录，则输出"<DIR>"，格式为16位长度，靠左对齐（-符号）

else

System.out.printf("%-16s\t","<DIR>");

//输出文件或目录名称

System.out.println(f.getName());

1. cat命令只能显式文本文件（如.java、.c、.txt、.html等等）内容，如果直接显示非文本文件（如.class、.jpg、.mp3等等）内容，将会出现乱码。

**●评分标准：满分20分**

1. 按要求正确完成所有任务
2. 编码、调试操作熟练；输出正确，界面符合要求（1分）
3. 实验报告撰写规范，内容完整；（1分）

**●测试案例：**

表7.1 测试案例（仅用于编码测试）

|  |  |  |  |
| --- | --- | --- | --- |
| 编号 | 测试目的 | 输入或测试数据 | 期望结果 |
| 1 | exit命令 | Exit或exit | 退出程序 |
| 2 | 错误的命令 | Abc | 显示“无法识别所输入命令:Abc” |
| 3 | ls命令显示正确的目录 | ls c:\ | 显示C盘根目录内容，显示格式正确 |
| 4 | ls命令显示当前目录 | ls . | 显示当前目录（用一个英文句号表示）内容，显示格式正确 |
| 5 | ls命令显示上级目录 | ls .. | 显示上级目录（用两个英文句号表示）内容，显示格式正确 |
| 6 | 错误的ls命令格式 | ls | 显示“命令格式不正确” |
| 7 | ls命令的路径不存在 | ls c:\abcdef | 显示“找不到目录” |
|  | 正确的cp命令 | cp c:\a.java c:\b.java | 存在C:\a.java、不存在C:\b.java的前提下，将C:\a.java复制到c:\b.java，命令结束后不显示任何信息  注意：c:\也可以是其他任意路径，如d:\，c:\abc，.\。以下类同。 |
|  | 正确的cp命令 | cp c:\a.java c:\b.java | 同时存在C:\a.java、C:\b.java的前提下，首先提示“覆盖C:\b.java吗(Y/N)”。如果输入Y或y，将C:\a.java复制到c:\b.java，命令结束后不显示任何信息；如果输入N或n，无任何处理，无任何输出 |
|  | 正确的cp命令 | cp c:\a c:\b | 存在C:\a，但不是文件（即c:\a是目录），则显示“不是有效文件” |
|  | 错误的cp命令格式 | cp  或  cp c:\a.java | 显示“命令格式不正确” |
|  | 正确的mv命令 | mv c:\a.java c:\b.java | 存在C:\a.java、不存在C:\b.java的前提下，将C:\a.java更名为c:\b.java，命令结束后不显示任何信息 |
|  | 正确的mv命令 | mv c:\a.java c:\b.java | 同时存在C:\a.java、C:\b.java的前提下，首先提示“覆盖C:\b.java吗(Y/N)”。如果输入Y或y，将C:\a.java改名为c:\b.java，命令结束后不显示任何信息；如果输入N或n，无任何处理，无任何输出 |
|  | 正确的mv命令 | mv c:\a c:\b | 存在C:\a，但不是文件（即c:\a是目录），则显示“不是有效文件” |
|  | 错误的mv命令格式 | mv  或  mv c:\a.java | 显示“命令格式不正确” |
|  | 正确的rm命令 | rm c:\a.java | 存在C:\a.java的前提下，首先提示“删除C:\a.java吗(Y/N)”。如果输入Y或y，将C:\a.java删除，命令结束后不显示任何信息；如果输入N或n，无任何处理，无任何输出 |
|  | 正确的rm命令 | rm c:\a | 存在C:\a，但不是文件（即c:\a是目录），则显示“不是有效文件” |
|  | 错误的rm命令格式 | rm | 显示“命令格式不正确” |
|  | 正确的md命令 | md c:\a | 不存在c:\a的前提下，创建空目录c:\a，命令结束后不显示任何信息 |
|  | 正确的md命令 | md c:\a | 已存在c:\a的前提下（c:\a可以是文件，也可以是目录），显示“目录已存在” |
|  | 错误的md命令格式 | md | 显示“命令格式不正确” |
|  | 正确的rd命令 | rd c:\a | 已存在c:\a目录，并且该目录为空，则删除该目录，命令结束后不显示任何信息 |
|  | 正确的rd命令 | rd c:\a | 已存在c:\a目录，并且该目录不为空，则显示信息“目录非空” |
|  | 正确的rd命令 | rd c:\a | 已存在c:\a，并且c:\a不是目录，则显示信息“目录名称无效” |
|  | 错误的rd命令格式 | rd | 显示“命令格式不正确” |
|  | 正确的cat命令 | cat c:\a.java | 存在C:\a.java的前提下，显示文件内容 |
|  | 正确的cat命令 | cat c:\a.java | 不存在C:\a.java的前提下，显示“文件不存在” |
|  | 正确的cat命令 | cat c:\a | 存在C:\a，但不是文件（即c:\a是目录），则显示“不是有效文件” |
|  | 错误的cat命令格式 | cat | 显示“命令格式不正确” |

**●思考题**

1. 您还可以提供哪些其它命令？（可以参考Windows CMD或Linux Shell下的常用命令）
2. 程序中会有很多重复内容（如提示内容重复、处理代码重复），如何减少这种冗余代码？