

黑龙江大学软件学院

“面向对象程序设计”

实验讲义

2021 年 3 月

# 目 录

第一部分 课程目的与任务.....	1
一、课程基础 .....	1
二、适应对象 .....	1
三、教学目的 .....	1
四、内容提要 .....	1
五、实验成绩构成 .....	2
第二部分 主要内容及基本要求.....	3
实验 1 Java 程序的编辑、编译、运行环境.....	3
实验 2 类和对象的使用.....	7
实验 3 封装性、继承性与包.....	11
实验 4 Object 类.....	16
实验 5 数组与异常处理.....	18
实验 6-1 链表类的实现 .....	21
实验 6-2 宠物商店 .....	25
实验 7 简易计算器 .....	29
实验 8 文本编辑器 .....	32
实验 9 世界时钟（选做） .....	35

## 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) 通过上机实践，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 6-1、6-2 (第 13-14 周，课外完成，从 2 个题目中至少完成 1 个)	链表类的实现 宠物商店	(4)	见正文	14-20
实验 7(第 15 周)	简易计算器	2	见正文	10
实验 8(第 16-17 周)	文本编辑器	4	见正文	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”。

### 实验 1 Java 程序的编辑、编译、运行环境

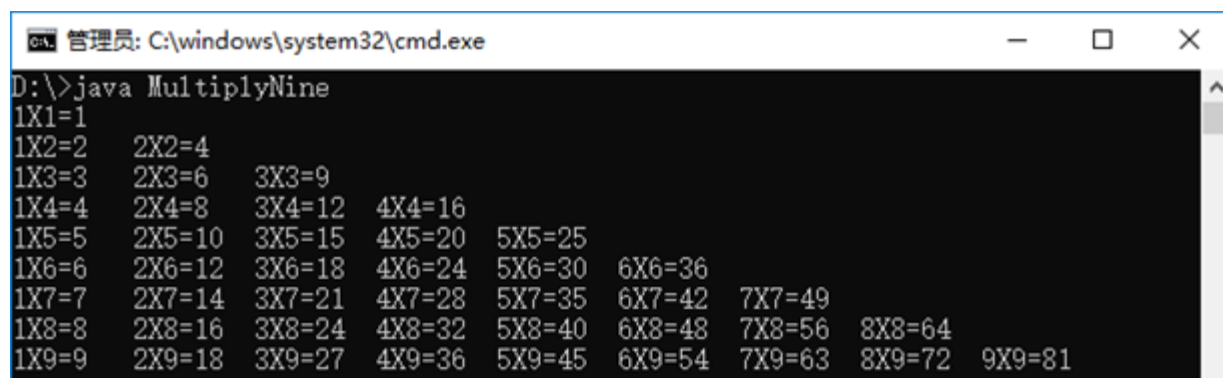
#### ●实验目的：

- (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 环境下编写计算整数 N（ $N \leq 20$ ）的阶乘程序 FactorialTest 类；
- (4) 在 Eclipse 环境下编写按图 1.1 格式输出乘法口诀表的程序 NineMultiplication。



```
D:\>java MultiplyNine
1X1=1
1X2=2    2X2=4
1X3=3    2X3=6    3X3=9
1X4=4    2X4=8    3X4=12    4X4=16
1X5=5    2X5=10   3X5=15    4X5=20    5X5=25
1X6=6    2X6=12   3X6=18    4X6=24    5X6=30    6X6=36
1X7=7    2X7=14   3X7=21    4X7=28    5X7=35    6X7=42    7X7=49
1X8=8    2X8=16   3X8=24    4X8=32    5X8=40    6X8=48    7X8=56    8X8=64
1X9=9    2X9=18   3X9=27    4X9=36    5X9=45    6X9=54    7X9=63    8X9=72    9X9=81
```

图 1.1 乘法口诀表程序运行效果

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）：

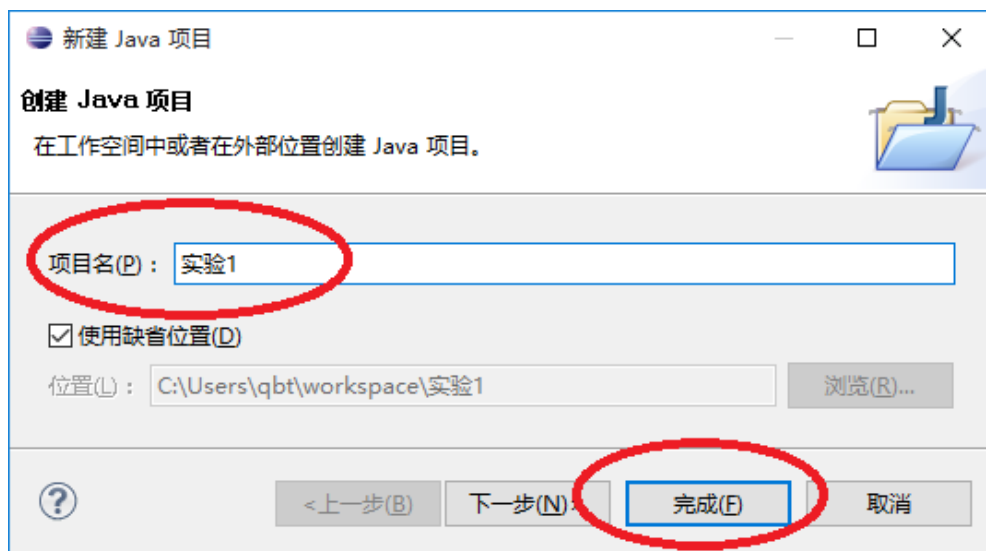


图 1.2 Eclipse 环境下创建 Java 项目

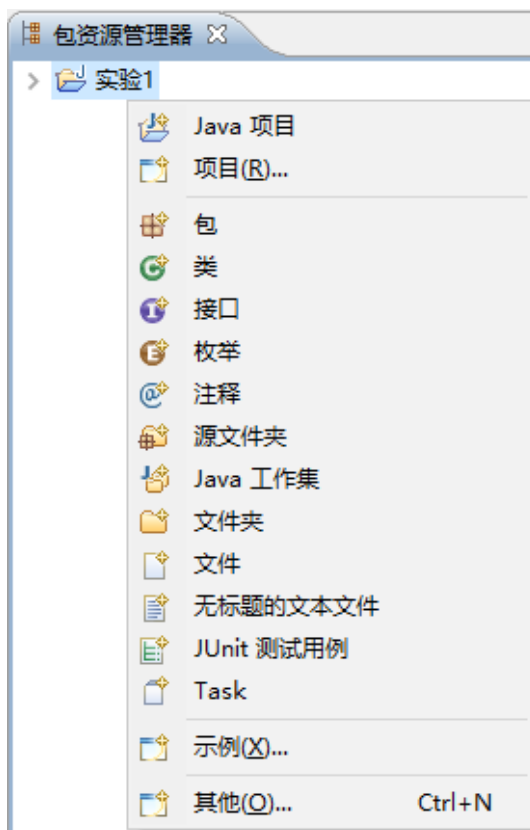


图 1.3“Alt-Shift-n”菜单

(3) 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){
    ...
}
```

●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 <code>java</code> 和 <code>javac</code> 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？

```
1X1= 1  1X2= 2  1X3= 3  1X4= 4  1X5= 5  1X6= 6  1X7= 7  1X8= 8  1X9= 9
      2X2= 4  2X3= 6  2X4= 8  2X5=10  2X6=12  2X7=14  2X8=16  2X9=18
            3X3= 9  3X4=12  3X5=15  3X6=18  3X7=21  3X8=24  3X9=27
                  4X4=16  4X5=20  4X6=24  4X7=28  4X8=32  4X9=36
                        5X5=25  5X6=30  5X7=35  5X8=40  5X9=45
                              6X6=36  6X7=42  6X8=48  6X9=54
                                    7X7=49  7X8=56  7X9=63
                                          8X8=64  8X9=72
                                                9X9=81
```

图 1.4 第二种输出格式

1X1= 1  
1X2= 2    2X2= 4  
1X3= 3    2X3= 6    3X3= 9  
1X4= 4    2X4= 8    3X4=12    4X4=16  
1X5= 5    2X5=10    3X5=15    4X5=20    5X5=25  
1X6= 6    2X6=12    3X6=18    4X6=24    5X6=30    6X6=36  
1X7= 7    2X7=14    3X7=21    4X7=28    5X7=35    6X7=42    7X7=49  
1X8= 8    2X8=16    3X8=24    4X8=32    5X8=40    6X8=48    7X8=56    8X8=64  
1X9= 9    2X9=18    3X9=27    4X9=36    5X9=45    6X9=54    7X9=63    8X9=72    9X9=81

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。 $\pi$  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：

```
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;
}
```

- (5) 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。
- (6) 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()`
- (3) 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

●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：

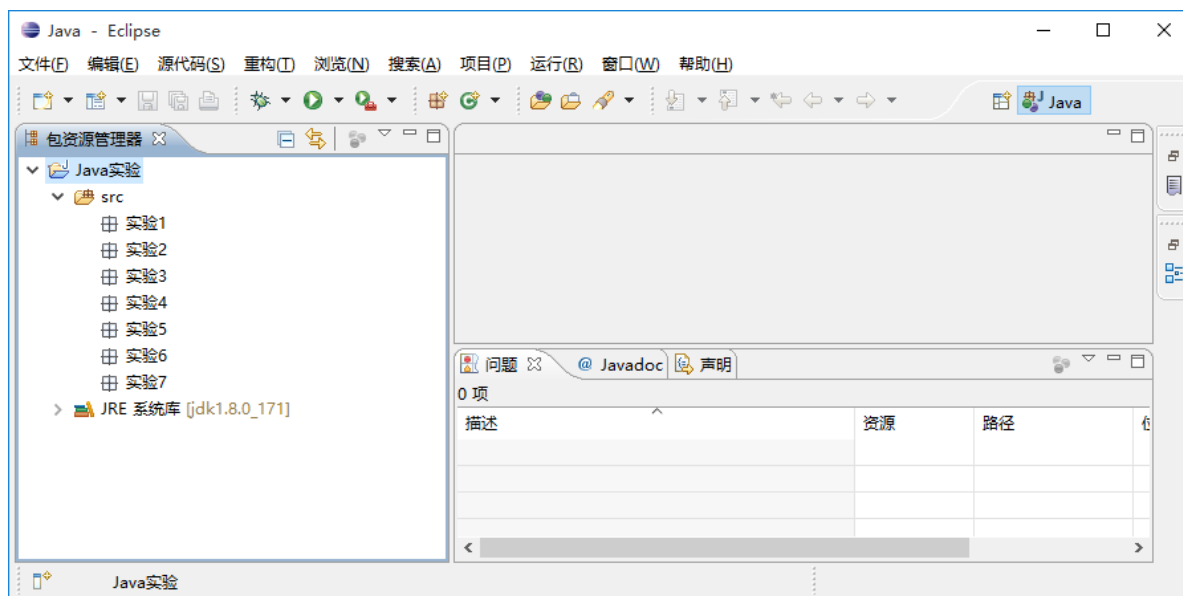


图 3.1 实验项目文件夹的浏览效果

(2) 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。

- (3) 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。
- (4) 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。
- (5) 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）。
- (6) 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	显示“同心圆”



---

```
centerColor 为(1,1,1), borderColor 为  
(1,1,1);调用 ColoredCircle()创建圆  
c2, 并调用 c2.relation(c1)
```

---

### ●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？

```
public static final Color RED=new Color(255,0,0);  
public static final Color GREEN=new Color(0,255,0);  
public static final Color BLUE=new Color(0,0,255);  
public static final Color WHITE=new Color(255,255,255);  
public static final Color BLACK=new Color(0,0,0);  
public 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”，并重写其构造方法、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。

### ●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。
- (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`
  - 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。

### ●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。实例化矩阵对象之后，可生成一组随机数，或从键盘读入一组数，并循环调用 `setData(...)`方法为矩阵某行某列赋值。建议采用随机数方式，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 IllegalArgumentException 异常（行号 <0，行号>=行数）
16	setter 方法	m.setData(0,-1) m.setData(0,m.getCols())	抛出 IllegalArgumentException 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) 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 6-1 链表类的实现

### ● 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。

### ● 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 LinkedList 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”。

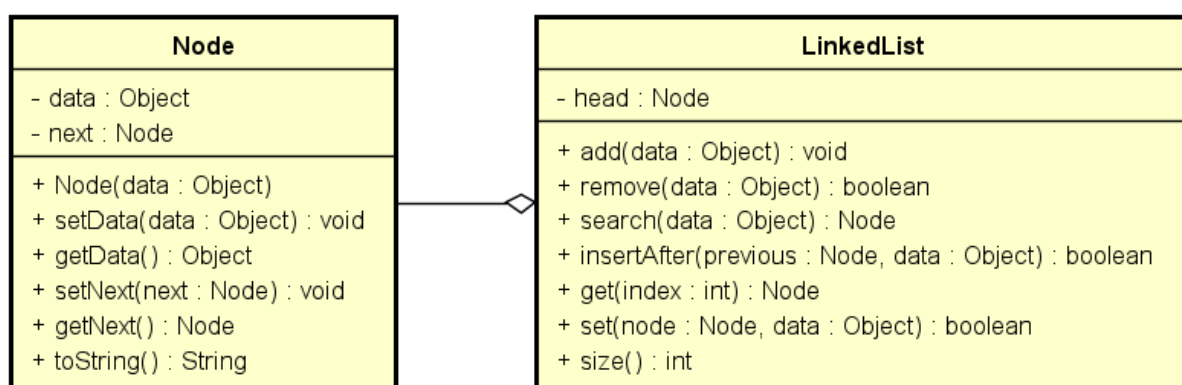


图 6-1 Node 类和 LinkedList 类之间的关系

### (3) Node 类：

#### ● 要求 Node 类拥有如下 2 个私有属性：

- a) **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。
- b) **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)。

#### (4) 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 LinkedList 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。
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 node 引用节点的内容改为 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
        }
    }
}
```



```
}

```

#### (5) 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。

#### ● 评分标准：满分 14 分

- (1) 按要求正确完成所有任务（LinkedList 类 8 分，Node 类 2 分，Test 类 2 分，合计 12 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 6.2 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

- 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？

## 实验 6-2 宠物商店

- 实验目的：

- (1) 掌握接口的概念和语法，并能够应用于实际。
- (2) 利用实验 61.LinkedList 实现复杂应用。

- 参考学时：2 学时

- 基本要求：

本题要求实现宠物商店的宠物上架、下架、查询等操作。要求用链表存储宠物商店中的宠物。因此，本选题需要在实验 6-1 中实现的链表类“实验 61.LinkedList”和“实验 61.Node”。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）。

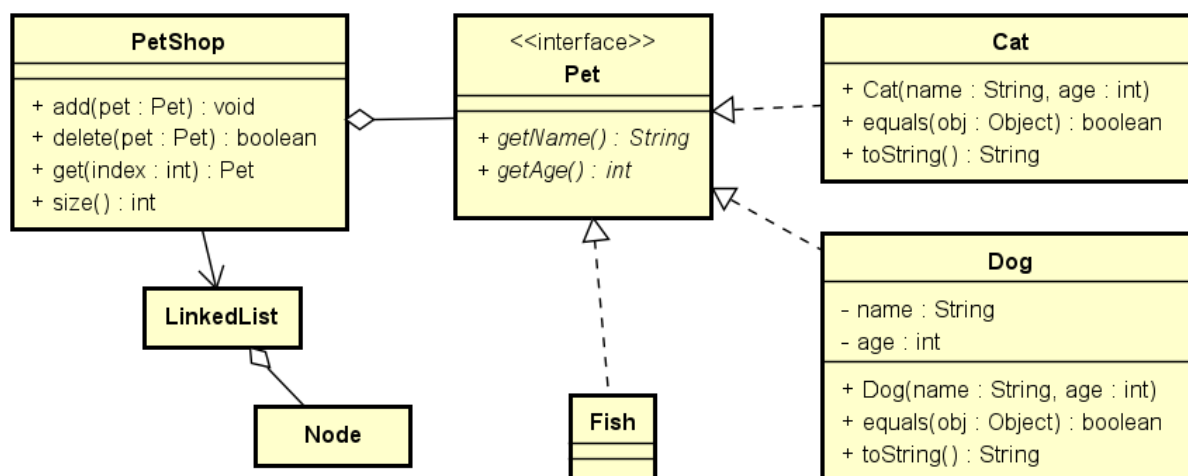


图 6.2 实验相关的类、接口及其关系

### (1) 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();

}
  
```

## (2) 定义 LinkedList 类

具体要求见实验 6-1。

## (3) 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（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（比如告知名字或年龄）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 6.3 宠物商店类的方法

编号	方法名称	描述
1	void add(Pet pet)	将宠物 pet 上架，即添加到链表中。
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 void add(Pet pet) {          //新宠物上架操作，以接口对象为参数!

        pets.add(pet);

    }

    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

        return (Pet)( pets.get(index).getData() );

    }

    ...

}
```

## (4) 定义宠物猫子类实验 62.Cat

要求 Cat 类实现接口 Pet。要求 Cat 类有两个私有属性 name 和 age。Cat 类中除了

getName()和 getAge() 方法外，还要求为其实现 3 个 public 方法，关于这 3 个方法的要求如表 6.4 所示。Cat 类的类图如图 6.3 所示。为了简洁起见，在图 6.3 所示的 Cat 类图中省略了 getName()方法和 getAge() 方法。

表 6.4 Cat 类的方法表

编号	方法名称	描述
1	Cat(String name, int age)	创建 Cat 类的实例，其名字为 name，年龄为 age。
2	equals(Object obj)	覆写 Object 中的 equals()方法。判断猫类的两个对象值是否相等。
3	toString()	覆写 Object 中的方法。返回猫完整信息。

#### (5) 定义宠物狗子类实验 62.Dog

要求 Dog 类实现接口 Pet。要求 Dog 类有两个私有属性 name 和 age。Dog 类中除了 getName()和 getAge() 方法外，还要求为其要求实现 3 个 public 方法，对这 3 个方法的要求，与表 6.4 中对 Cat 类中方法的要求类似，因此不再赘述。Dog 类的类图如图 6.4 所示。

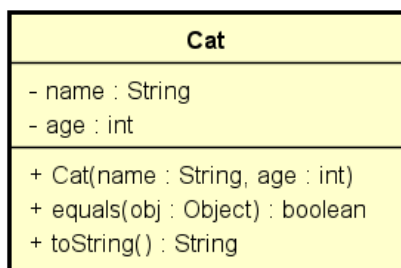


图 6.3 Cat 类图

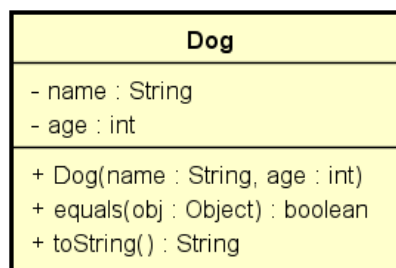


图 6.4 Dog 类图

#### (6) 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”（注：波尔多是一种狗名字）；
- 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。

● 评分标准：满分 6 分

- (1) 按要求正确完成所有任务（PetShop2 分，Dog 和 Cat 类 1 分，Pet 接口和 Test 类 1 分，合计 4 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 6.5 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",2));	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) Cat、Dog 类中重写 equals()有什么特殊意义？不重写会有什么后果？
- (2) 还可以为 Pet 接口或 PetShop 类添加哪些方法？

## 实验 7 简易计算器

### ●实验目的：

- (1) 掌握图形界面程序的构造过程以及布局管理器的应用；
- (2) 掌握常用组件对象的使用；
- (3) 掌握 Action 事件和 Window 事件的处理过程。

### ●参考学时：2 学时

### ●基本要求：

- (1) 编写具有图形界面的简易计算器类实验 7.SimpleCaculator, 该类包含 AWT 界面构造代码。界面如图 7.1 所示：

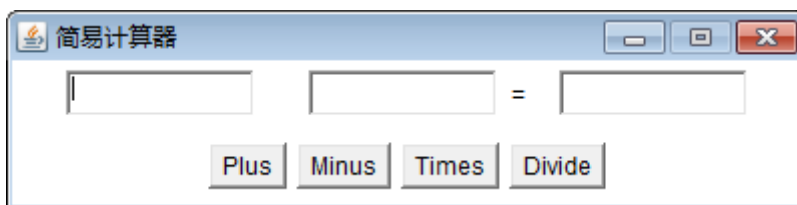


图 7.1 简易计算器界面示意图

- (2) 编写四个按钮的单击事件以及窗口关闭事件的处理类实验 7.MyListener:

- 当按下“+”按钮时，两个数值文本框之间应显示“+”号，同时相加结果显示在第三个文本框内（如图 7.2 所示）。类似处理“-”、“\*”和“/”按钮。

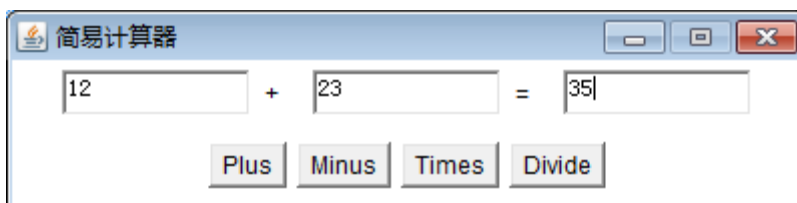


图 7.2 单击 Plus 按钮后的运行结果示意图

- 应处理除 0 异常和数值格式非法异常，如：

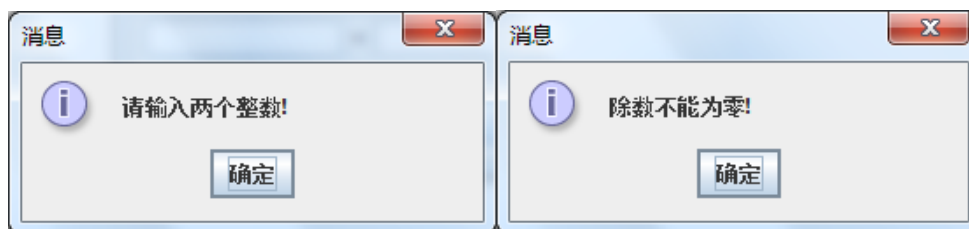


图 7.3 异常处理示意图

- 点击关闭按钮时，结束程序的运行。

- (3) 编写计算器类的测试类实验 7.Test。

### ●实验提示：

- (1) 文本框（TextField 类）的宽度可以用构造函数设置，如：

```
TextField tf1=new TextField(10);
```

- (2) 出错提示窗口可以利用 javax.swing.JOptionPane 类的相应方法，如：

```
javax.swing.JOptionPane.showMessageDialog(null, "请输入两个整数!");
```

- (3) 由于 SimpleCaculator 类和 MyListener 类分别是两个类，在 SimpleCaculator 类中应该把事件处理过程中需要访问的三个 TextField 对象以及第 1 个 Label 对象定义为成员变量（四个按钮也可以定义为成员变量），并实例化事件处理对象时将 SimpleCaculator 对象传给 MyListener。此后 MyListener 就可以直接访问需要的对象（除非这几个成员变量为 private）。这两个类和 Test 类的关系如以下代码：

```
public class Test{
    public static void main(){
        SimpleCaculator calc= new SimpleCaculator();
        calc.go();
    }
}

public class SimpleCaculator{
    TextField tf1,tf2,tf3;
    Label l1;
    Button[4] button;
    public void go(){
        ...
        MyListener ml=new MyListener(this);//this 即 Test 中的 calc 对象
        for(int i=0;i<4;i++)
            button[i].addActionListener(ml);
        ...
    }
}

public class MyListener...{
    SimpleCaculator calc;
    public MyListener(SimpleCaculator o){
        calc=o;
    }
    public void actionPerformed(ActionEvent e){
```



```

        if(e.getSource()==calc.button[0]){//Plus 按钮
            Stringv1=calc.tf1.getText();
            ...
        }
    }
}

```

(4) 将界面类和事件处理类分开写的优点是程序结构清晰，易于维护，便于多人合作开发复杂软件。

#### ●评分标准：满分 10 分

- (1) 按要求正确完成所有任务（SimpleCaculator 和 MyListener 类各 4 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 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	布局管理器	改变窗口大小	各组件的相对位置基本保持不变
2	Window 事件	点击关闭按钮	窗口能够正确关闭
3	四则运算	输入 10 和 0，点击加按钮	显示加号和 10
4	四则运算	输入 10 和 0，点击减按钮	显示减号和 10
5	四则运算	输入 10 和 0，点击乘按钮	显示乘号和 0
6	四则运算	输入 10 和 0，点击除按钮	显示“除数不能为零！”
7	异常处理	什么也不输入，依次点击加减乘除按钮	提示“请输入两个整数!”
8	异常处理	输入 a 和 b，依次点击加减乘除按钮	提示“请输入两个整数!”

#### ●思考题

- (1) 将事件处理代码直接写在 SimpleCaculator 有什么便利之处，有什么缺点？
- (2) 在示例代码中，如果将 MyListener ml=new MyListener(this)改为 MyListener ml=new MyListener(new SimpleCaculator())会带来什么后果？
- (3) 四个按钮的点击事件处理代码会有很多重复之处，如何减少这种冗余代码？

## 实验 8 文本编辑器

### ●实验目的：

- (1) 掌握图形界面环境下的 JTextArea 组件和菜单组件的应用；
- (2) 掌握 JFileChooser 类的使用方法；
- (3) 掌握字符流文件的读写。

### ●参考学时：4 学时

### ●基本要求：

- (1) 编写简易编辑器图形界面类实验 8.NotepadFrame，界面效果如图 8.1 所示：

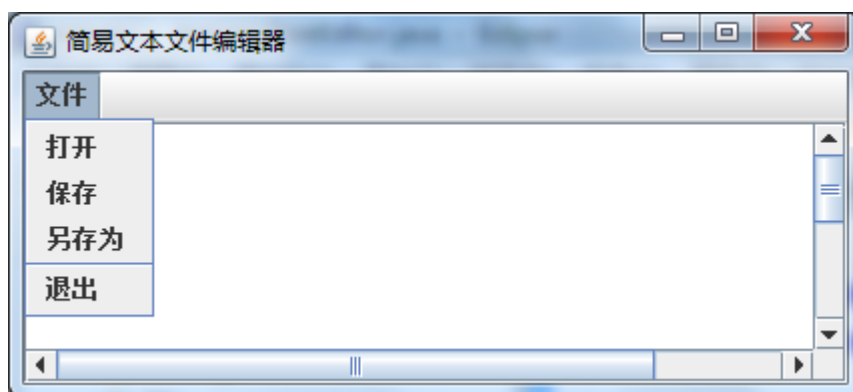


图 8.1 文本编辑器界面示意图

- (2) 编写菜单事件响应类实验 8.MyListener，并完成文本文件的编辑、保存功能。
- (3) 要求完成从磁盘读入文本文件，或从新开始编辑全新内容，并将内容显示到 JTextArea 中。所编辑的内容可以保存到新文件、原文件或另存到其它文件中。
- (4) 必须使用 JFileChooser 类来完成文件名称的输入或选取。

### ●实验提示：

- (1) 在 JFrame 中添加菜单可以用如下代码：

```
JMenuBar jMenuBar1 = new JMenuBar();
JMenu jMenu1 = new JMenu("文件");
JMenuItem jMenuItem1 = new JMenuItem("打开");
JMenuItem jMenuItem2 = new JMenuItem("保存");
JMenuItem jMenuItem3 = new JMenuItem("另存为");
JMenuItem jMenuItem4 = new JMenuItem("退出");
JSeparator jSeparator1 = new JSeparator();//分割线
jMenu1.add(jMenuItem1);
jMenu1.add(jMenuItem2);
```

```
jMenu1.add(jMenuItem3);
jMenu1.add(jSeparator1);
jMenu1.add(jMenuItem4);
jMenuBar1.add(jMenu1);
frame.setJMenuBar(jMenuBar1);
```

(2) 默认情况下 JTextArea 没有滚动条，若要添加滚动条，需要用以下代码：

```
jScrollPane1 = new JScrollPane();
jTextArea1 = new JTextArea();
jScrollPane1.setViewportView(jTextArea1);
frame.add(jScrollPane1);
```

(3) 应在程序中定义文件路径和文件名称变量，以便当打开某一文件，或将全新编辑的内容存储为某一文件时，保存文件路径和文件名称。

(4) 当前内容已被编辑时，退出程序或打开新文件都应提示是否需要保存，并根据作者的选择决定下一步处理。可通过 JTextArea 的键盘事件（KeyEvent）监听确定内容是否被修改，如：

```
public void keyPressed(KeyEvent e){
    //edited 为当前内容是否已编辑（不代表内容肯定被修改）标志
    edited=true;
}
```

#### ●评分标准：满分 20 分

- (1) 按要求正确完成所有任务（NotepadFrame 2 分，关闭按钮和退出菜单 3 分，打开菜单 3 分，保存菜单 3 分，另存菜单 3 分，文件读写 4 分，合计 18 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 8.1 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	布局管理器	改变窗口大小	各组件的相对位置基本保持不变
2	Window 事件 退出菜单	没有修改过 JTextArea 内容（内容空白，同时没有键入任何内容）	关闭窗口时直接退出
3	Window 事件 退出菜单	键入某些内容（可将键入的内容全部删除）	关闭窗口时弹出对话框提示“是否保存当前内容？”和“是”、“否”、“取消”三个按钮。选择“是”则弹出 JFileChooser 对话框，获取文件路径后保存内容并退

			出；选择“否”则直接退出；选择“取消”则不退出，返回编辑状态
4	Window 事件退出菜单	打开一个文件后没有任何修改	关闭窗口时直接退出
5	Window 事件退出菜单	打开一个文件后进行了修改	关闭窗口时弹出对话框提示“是否保存当前内容?”和“是”、“否”、“取消”三个按钮。选择“是”则覆盖原有文件并退出；选择“否”则直接退出；选择“取消”则不退出，返回编辑状态
6	打开菜单	没有修改过 JTextArea 内容	弹出 JFileChooser 对话框，获取文件路径后读入内容并显示在 JTextArea 中
7	打开菜单	修改过 JTextArea 内容，同时该内容不属于任何文件	弹出对话框提示“是否保存当前内容?”和“是”、“否”、“取消”三个按钮。选择“是”则弹出 JFileChooser 对话框，获取文件路径后保存内容，再次显示 JFileChooser 对话框，获取新文件路径后读入内容并显示在 JTextArea 中；选择“否”则直接显示 JFileChooser 对话框，获取文件路径后读入内容并覆盖 JTextArea 原内容；选择“取消”则返回编辑状态
8	打开菜单	修改过 JTextArea 内容，同时该内容属于某一文件	弹出对话框提示“是否保存当前内容?”和“是”、“否”、“取消”三个按钮。选择“是”则保存原有文件内容，再次显示 JFileChooser 对话框，获取新文件路径后读入内容并覆盖 JTextArea 内容；选择“否”则直接显示 JFileChooser 对话框，获取文件路径后读入内容并覆盖 JTextArea 原内容；选择“取消”则返回编辑状态
9	保存菜单	JTextArea 内容为全新内容	弹出 JFileChooser 对话框，获取文件路径后保存内容，最后返回编辑状态
10	保存菜单	JTextArea 内容为某一文件内容	直接覆盖源文件内容，最后返回编辑状态
11	另存为菜单	JTextArea 内容为全新内容	弹出 JFileChooser 对话框，获取文件路径后保存内容，最后返回编辑状态
12	另存为菜单	JTextArea 内容为某一文件内容	和测试案例 11 相同

### ●思考题

- (1) 为了方便起见，通常将如前所述的文件路径、文件名称以及 edited 变量定义为事件监听类 MyListener 的成员变量。如果把上述变量定义为 NotepadFrame 的成员变量，有何特殊意义？
- (2) 还可以添加哪些功能菜单？

## 实验 9 世界时钟（选做）

### ●实验目的：

- (1) 掌握创建多线程应用；
- (2) 进一步熟练掌握图形界面和事件处理。

### ●参考学时：2 学时

### ●基本要求：

- (1) 编写一个世界时钟程序界面类实验 9.WorldClock，界面如图 9.1 所示：

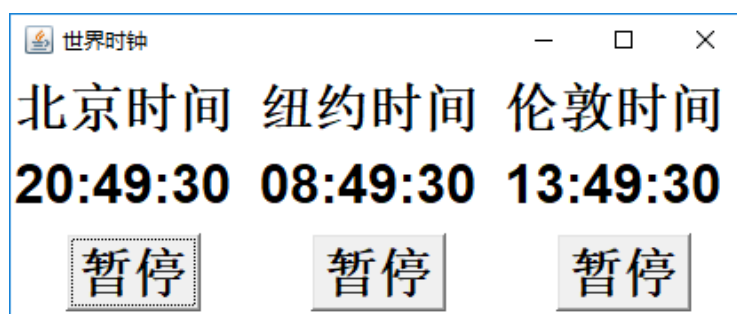


图 9.1 文世界时钟界面示意图

- (2) 编写时钟线程类实验 9.ClockThread，完成每隔 1 秒更新显示功能。
- (3) 编写三个按钮的事件处理类实验 9.MyListener，实现单击按钮时暂停或继续时钟显示。点击某一“暂停”按钮后，该按钮文字变成“继续”，再次点击该按钮（“继续”按钮）则又变成“暂停”按钮。

### ●实验提示：

- (1) 设置 Label 字体的代码如下：

```
Font font=new Font("宋体",Font.BOLD,30);
```

```
labelBeiJing=new Label("北京时间");
```

```
labelBeiJing.setFont(font);
```

- (2) 获取时间、换算时区以及转化为字符串的代码如下：

```
Calendar now = Calendar.getInstance();
```

```
//timeInterval 为与北京时间相差的时数，如纽约为-12，伦敦为-7，东京为+2
```

```
now.add(Calendar.HOUR, timeInterval);
```

```
String sdf=new SimpleDateFormat("HH:mm:ss").format(now.getTime());
```

- (3) 让某一线程 t 暂停执行的语句是：

```
t.suspend();//挂起，即暂停
```

- (4) 让某一线程 t 继续执行的语句是：

`t.resume();`//恢复，即继续执行

●评分标准：满分 10 分

- (1) 按要求正确完成所有任务（WorldClock 和 MyListener 各 3 分，ClockThread2 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）