Java上机题

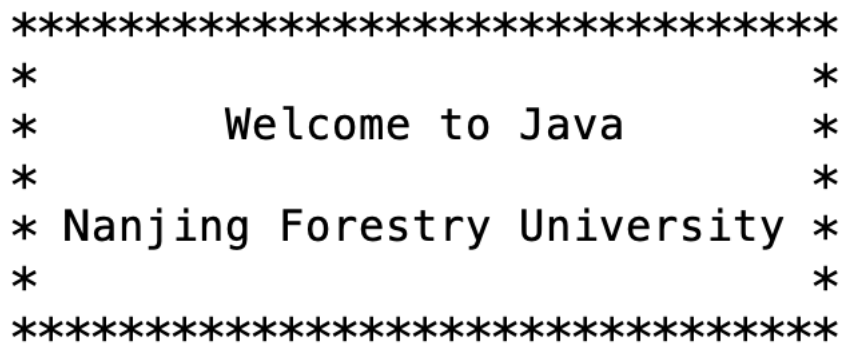
张威

南京林业大学信息学院

1.1 Greeter 1
1.2 Greeter2 1
1.3 Factorial 1
1.4 BreakJail 2
1.5 Hanoi 3
1.6 Converter 3
1.7 Stuff 4
1.8 Stuff2 4
1.9 BreakJail2 5
1.10 Guess 5
1.11 Poker 6
1.12 WireFrame3D 7
1.13 Spinner 10
1.14 Fibonacci 10

1.1 Greeter

编写一个名为Greeter的类，文件名为Greeter.java，在控制台上输出如下图所示的内容。具体要求如下：



1. 私有静态方法：void greet(int cols)，输出cols个星号；void greet(int cols, String words)，在开头输出一个星号，中间输出给定的文本words，最后再输出一个星号，星号和文本之间填充空格，使文本对于给定的列数居中；
2. 在void greet()方法中，用StringBuilder/StringBuffer通过循环生成目标输出字符串，然后通过System.out.println()一次性输出（提示：StringBuilder/StringBuffer.append()方法将给定的字符、数值、字符串等类型的参数追加到调用缓冲区中）；
3. 入口点main()：创建两个字符串变量，分别包含如图所示的内容，最长文本的左边为一个星号和一个空格，右边为一个空格和一个星号。文本行之间以及文本行与星号行之间必须用“空行”分隔；调用void greet()生成如图所示的输出(提示：String.length()方法返回以字符为单位的字符串长度）。

1.2 Greeter2

编写一个名为Greeter2的类，文件名为Greeter2.java。在控制台上输出的内容同1.1题。具体要求如下：

1. 私有静态方法：void greet(int cols)/void greet(int cols, String words)与1.1题相同；
2. 在void greet()方法中，不得使用StringBuilder/StringBuffer以及String的连接，直接在循环中完成输出；
3. 入口点main()：同1.1题。

1.3 Factorial

编写一个名为Factorial的类，它计算给定整数的阶乘；编写一个名为FactDriver的类，它包含入口点方法main()。这两个类都位于名为FactDriver.java的文件中。具体要求如下：

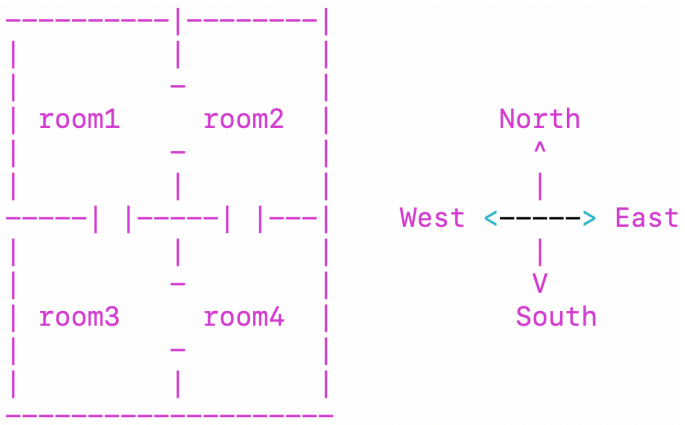
1. 以单例模式（singleton）实现Factorial类，即在当前运行的JVM中，仅仅存在该类的一个实例；要求采用惰性方式创建Factorial的实例；

2. 在Factorial类的实例方法long calculate(int n)中，计算给定整数n的阶乘并返回，要求缓存计算的结果（对外不可直接访问）；通过Factorial的实例方法getFact()，随时可以再次提取前一次计算的结果，如果之前未曾调用过calculate()方法，返回0；不许采用递归方方式；

3. 在FactDriver的main()方法中，调用Factorial类的实例方法calculate()计算5的阶乘，并以形式“5! = XXX”的形式输出。

1.4 BreakJail

编写表示“越狱游戏”的BreakJail类，如以下ASCII码图所示，监狱共有4个房间（在实际的游戏中，房间的数量及布局可以通过脚本或“地图”而动态配置），这里为了简单起见，4个房间分别由实例方法room1()、room2()、room3()和room4()表示。



其中房间1为初始位置，可以向东(East)移动到房间2或向南(South)移动到房间3；在房间2中，可以向西(West)移动到房间1或向南(South)移动房间4；在房间3中，可以向北(North)移动到房间1或者向东(East)移动到房间4。这里将房间4看作是监狱的出口，表示越狱成功，并在控制台上打印输出文本“Congratulations, you have broken the jail!”（在实际的游戏中，除了房间的数量和布局之外，目标出口以及其它房间中的移动可以通过脚本或“地图”进行动态配置）。具体要求如下：

（1）第一个版本，用System.in.read()读取用户的键盘输入（作为单个字符），N或n表示向北移动，S或s表示向南移动，W或w表示向西移动，而E或e表示向东移动。

（2）第二个版本，用java.util.Scanner类读取用户的键盘输入(next()方法)，输入的含义与(1)相同。

（3）在入口点main方法中，创建BreakJail的一个实例，并调用room1()方法，表示当前处于房间1中，从而开始游戏。

（4）无论是版本1还是版本2，在除了房间4之外的其它3个房间中，如果移动方向不正确，打印输出提示信息：Can't move to: x，其中的x表示用户的实际输入。

（5）在所有的方法中，包括main方法，不得使用try~catch捕获异常。

1.5 Hanoi

以下是对梵塔问题的描述：在A座上有自上而下、从小到大堆叠的n个圆盘，以B座为中转，将这n个圆盘从A座移动到C座，每次只能移动一个，并且在移动过程中不能出现大盘压小盘的现象。完成以下任务：

1. 编写名为Hanoi的类，以单例模式实现，采用饥渴方式创建其全局唯一的实例；
2. 编写HanoiDriver类，它包含入口点main()方法；
3. Hanoi和HanoiDriver类均位于文件HanoiDriver.java文件中；
4. 在Hanoi类中，实现公有实例方法void move(int n, char source, char station, char target)，其中，n表示圆盘数量，source表示源座，即n个圆盘初始所在的座，station表示中转座，初始为空；target表示目标座，即n个圆盘应该被移动至的座；
5. 在Hanoi类中，实现私有辅助方法void show(int n, char source, char target)，其中，n表示第n号圆盘，source表示源座，target表示目标座；显示格式为：#n：source -> target；
6. 在HanoiDriver的main()方法中，调用Hanoi的实例方法move()，将5个圆盘，从’A’座以’B’座为中转，移动到’C’座上。

1.6 Converter

编写一个名为Converter的类。其中有两个构造方法，一个为默认，而另外一个接收一个int整数作为参数public Converter(int value)。

1. 在Converter类中，添加数据字段int value，缓存从构造方法接收到的整数参数；为该数据字段提供访问器方法（可读/可写）；
2. 在Converter类中，添加String toBase(int base)方法，它将当前缓存的值转换为参数base所指定的进制数的字符串并返回；
3. 在Converter类中，添加String toBin()方法，它将当前缓存的值转换为2进制数的字符串并返回；
4. 在Converter类中，添加String toOct()方法，它将当前缓存的值转换为8进制数的字符串并返回；

5. 在Converter类中，添加String toHex()方法，它将当前缓存的值转换为16进制数的字符串并返回；

6. 以上String toXXX()方法中，所返回的字符串形式以127的16进制形式为例：7_16，即以单个下划线分隔各位数字；

7. 编写名为ConverterDriver的类，在其入口点main()方法中，创建Converter类的一个实例，传入的整数值为127，输入以下形式：

base = 2, 127 => 1_1_1_1_1_1_1

base = 8, 127 => 1_7_7

base = 16, 127 => 7_15

base = 3, 127 => 1_1_2_0_1

1.7 Stuff

假设开发教务管理系统，需要对学生以及教职员工进行建模。用长整数long唯一地标识学生和教职员工。请根据以下要求编写表示学生和教职员工的类并在入口点中进行测试：

1. Stuff，抽象类。数据字段：id，long，标识ID；no，String，编号；name，String，名称；age，int，年龄；gender，char，性别（男：m；女：f）；抽象方法public void Print()；

2. Student，学生，Stuff子类。数据字段：major，String，专业；

3. Teacher，教师，Stuff子类。数据字段：title，String，职称；

4. Executive，行政人员，Stuff子类。数据字段：position，String，岗位；

5. 所有类的所有数据字段均为私有，提供相应的访问方法（可读、可写）；

6. 编写University类，在入口点中分别创建Student、Teacher和Executive类的一个实例，其变量类型为Stuff ，调用Print()成员函数输出相应的信息，具体格式为：

(XXX: ID号，编号，姓名，年龄，性别，类型，...)。

1.8 Stuff2

接1.6题，从Stuff类中删除抽象方法public void Print()，重写public String toString()方法，以允许在System.out.print/ln()中直接打印，打印输出格式同1.6题。

1. 在Stuff类中添加方法protected void preFormat(StringBuilder sb)，它将员工的标签格式化到给定的参数中；

2. 在Stuff类中添加抽象方法abstract protected void postFormat(StringBuilder sb)；

3. 在Stuff类中的public String toString()方法中，创建一个StringBuilder对象，首先调用方法protected void preFormat(StringBuilder sb)，然后执行自己的格式化，接着调用protected void postFormat(StringBuilder sb)方法，最后返回格式化后的字符串；

4. 对于Teacher、Student和Executive类，执行相应的调整；

5. 编写University类，在入口点中分别创建Student、Teacher和Executive类的一个实例，其变量类型为Stuff ，调用System.out.println()输出相应的信息。

1.9 BreakJail2

类似于1.4，但文件名为BreakJail2.java，在相同的文件中，创建公有类BreakJail2和另外一个表示房间的类Room。

在Room类中，包含一个类型为boolean的私有字段target，如果其值为true，表示该房间为监狱出口，否则，表示普通房间；还包含类型为Room的4个名字分别为north、south、west和east的私有字段，表示当前当前房间的相邻房间（有的字段可以为空）。该类型中还包含一个move()公有方法，其作用与1.4中的roomX()等方法相同，如果当前房间为目标，输出文本与1.4相同，游戏也随之结束，否则，根据用户的输入（N|n、S|s、W|w、E|e）确定应该到用对应于north、south、west和east的move方法（当然，相关字段不能为空）。

在BreakJail2类中，添加private static Room build()私有方法，它模拟了游戏场景构建脚本而创建游戏地图（在GUI应用中，可以执行图形化绘制）。在该方法中，创建3×4个房间的监狱，并将第一个房间作为开始点返回，而将第2行第3列作为目标房间（出口），还应该建立所有房间的正确关联关系(例如，第一行的房间无北面的相邻房间，最后一行的房间没有南面的房间，第一列的房间没有西面的相邻房间，最后一列的房间无东面的相邻房间）。在该类中添加main入口点方法并调用build方法，从所返回的Room实例调用其move()方法而开始游戏。

为了简单起见，不根据获取用户的输入方式而分为两个版本。这里要求使用System.in.read()从键盘读取用户的输入（在GUI当然可以拦截光标移动键）。

1.10 Guess

简单猜数值游戏。编写一个名为Guess的类。在其构造方法用系统时间对随机数生成器种子化以防止每次生成的随机数序列相同（类java.util.Random）；成员方法Result judge(int value)对用户从键盘上输入的整数值进行判断并返回结果。编写Driver类，在入口点main()中实例化Guess的一个对象，提示用户输入一个整数。具体要求如下：

1. Guess的构造方法接收两个参数：bound，表示生成随机数的上界，默认值为10，即所生成的整数在范围[0, 10)中（Random的int nextInt(int bound)方法）；limit，表示允许用户猜测的最大次数，默认值为10；在构造方法中用系统时间（System. currentTimeMillis()方法）作为随机数生成器的种子；

2. 在成员方法void start()中，生成随机数并将其保存在成员变量value中，并将表示当前已猜测次数的成员变量count复位；

3. 在成员方法Result judge(int value)中，递增用户已猜测的次数count，对用户所输入的整数进行判断，并将判断结果以枚举Result（位于Guess类中）er 返回（EQUAL，value == this.value；LESS，this.value > value；GREAT，this.value < value；EXCEED，count >= limit）。

4. 在Driver类中，编写一个名为void guess(Guess guess)的方法，提示用户输入一个整数，并调用Guess的judge()成员方法对用户所输入的整数进行判断，根据其返回值在终端（控制台）上输出：

① 正确（EQUAL）：“congratulation!”

② 太小（LESS）：“Sorry，too small，remaining：XXX”

③ 太大（GREAT）：“Sorry，too large，remaining：XXX”

④ 超限（EXCEED）：“Sorry，exhausted，value：YYY”

其中XXX表示剩余的猜测次数（提示：在Guess中提供成员方法以访问其私有数据成员）而YYY表示未被猜中的随机值。在太小或太大情况下，允许用户继续猜测。

5. 在main()方法中创建Guess的一个对象，在一个循环中，开始游戏，并在每一轮游戏之后询问用户是否继续，如果用户回答否，游戏结束。

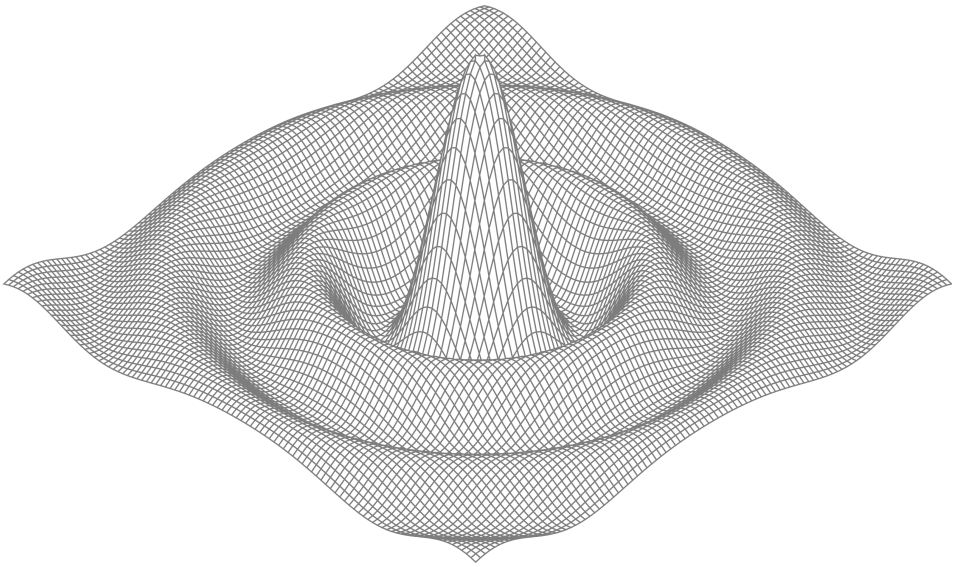
1.11 Poker

假设开发斗地主纸牌游戏。一副牌总共54张，顺序为A、2~10、J~K、小王、大王；除了大小王之外，其它依次为红桃、黑桃、方块和梅花；类Card，它包含数据字段：flush，表示花色；number，范围[1, 13]（大小王除外）。花色由枚举Flush定义：HEART，0，红桃；SPADE，1，黑桃；DIAMOND，2，方块；CLUB，3，梅花；JOKER，4，大小王（大王的number为1，小王的number为0）；类Deck，其私有数据字段List<Card> cards，包含所有的纸牌；成员方法boolean add(Card card)和boolean add(Flush fluash, int number)向其中添加一张牌，如果已经有了18张牌，返回false；否则，返回true；成员方法boolean contain(Flush flush, int number)确定是否包含参数给定的纸牌；成员方法bool isFull()确定是否已经包含了18张牌；成员方法void reset()执行复

位，即清空所有的纸牌；类Poker，私有数据字段：deckA、deckB、deckC，类型为Deck，表示发给三个玩家的牌，要求能够获取它们；公有成员方法void deal()实现为三个玩家随机发牌并确保在发牌之前，将deckA、deckB、deckC复位；可根据情况添加辅助方法；PokerGame，在其入口点main()中创建Poker的一个实例，调用其deal()函数随机发牌，获取三个玩家的牌，并输出到控制台（终端）窗口中。

1.12 WireFrame3D

下图是函数$\frac{\sin(r)}{r}$的3-D线框图，其中，$r = \sqrt{x^2 + y^2}$。为了绘制3-D曲面，一般用面片逼近该曲面。这里不是在GUI中直接在窗口中绘制该3-D线框图，而是创建可伸缩向量图形(SVG)文件。在以下所给定的类WireFrame3D中，常量WIDTH和HEIGHT分别为所给定的SVG画布的画布的宽度和高度；CELLS为SVG中的单元格数，即面片的数量大致为CELLS × CELLS；RANGE为x和y坐标轴的范围。



SVG文件本质上为XML格式，如下所示（其中的+号表示后面的部分应该紧随其后而并没有换行，style属性将线条颜色固定为灰色，填充色固定为白色，笔画宽度为0.7，而两个问号分别是画布的宽度和高度：

```
<svg xmlns='http://www.w3.org/2000/svg'  +
            style='stroke:grey; fill: white; stroke-width: 0.7' +
            width='?' height='?'>
<!-- the content to render goes here -->
</svg>
```

完成WireFrame3D中“TODO”部分的代码，要求：

1. 在入口点main方法中，检查将要保存的SVG文件名是否被传入，如果没有传递文件名，打印：Usage: WireFrame3D filename，然后退出；否则创建文件输出流，并以其作为参数调用方法build以创建SVG文件；添加大致度量该程序执行时间的代码（提示：System.nanoTime()返回以纳秒为单位的时间）；

2. 完成z3d()方法，根据给定的(x, y)坐标计算3-D中的Z坐标（提示，静态方法Math.hypot(x, y)计算$\sqrt{x^2 + y^2}$的值，即(x, y)到原点的距离）；

3. 在和WireFrame3D相同的文件中，编写Point类，以封装投影后的(x, y)坐标。

4. 在方法build中，调用project()方法以根据每个面片的单元格坐标(i, j)确定其3-D坐标并将其投影到SVG画布上，将其投影坐标作为Point对象返回；每个面片用SVG的多边形表示（四边形4对坐标）：

<polygon points='%f,%f %f,%f %f,%f %f,%f'/>

① 采用java.util.Formatter工具类，在构造函数中传入输入流，然后可以调用其format()方法对数据进行格式化（类似于C的printf，第一个参数为格式化字符串，随后是可变参数，转换指定符%d以及%f与C类似）。在构建完成后，调用其close()方法，它将会自动调用输出流的关闭方法（如果它实现了相关接口的话，例如文件输出流）。

② 采用StringBuilder，将SVG文件的内容作为字符串而构建，并将最终的内容写到输出流。注意需要明确地关闭输出流。

5. 编译并运行，然后在浏览器中打开所生成的SVG文件，验证输出结果。

```
public class WireFrame3D {
    // the width of the SVG canvas
    private static final int WIDTH = 600;
    // the height of the SVG canvas
    private static final int HEIGHT = 320;
    // the count of the 3-D surface patches
    private static final int CELLS = 100;
    private static final double RANGE = 30.0; // (-RANGE, +RANGE)
    // pixels per x or y unit
    private static final double SCALE = WIDTH / 2 / RANGE;
    // pixels per z unit
    private static final double SCALE_Z = HEIGHT * 0.4;
```

```
    // angle of x, y axes
    private static final double ANGLE = Math.PI / 6;
    private static final double SIN30 = Math.sin(ANGLE);
    private static final double COS30 = Math.cos(ANGLE);

    private static double z3d(double x, double y) {
        // TODO
    }

    private static Point project(int i, int j) {
        // determine the point (x, y) at corner of cell (i, j)
        double x = RANGE * (1.0 * i / CELLS - 0.5);
        double y = RANGE * (1.0 * j / CELLS - 0.5);
        // determine the z-coordinate
        double z = z3d(x, y);
        // project (x, y, z) isometrically onto 2-D SVG canvas
        double gx = 1.0 * WIDTH / 2 + (x - y) * COS30 * SCALE;
        double gy = 1.0 * HEIGHT / 2 + (x + y) * SIN30 * SCALE - z * SCALE_Z;
        return new Point(gx, gy);
    }
    public static void build(OutputStream os) throws Exception {
        // TODO
    }
    public static void main(String[] args) {
        // TODO
    }
}
```

1.13 Spinner

编写文件Spinner.java，在类Spinner的入口点方法中，显示一个进度指示器，即旋转的一个线段，对应四个字符：-、\、|和/，每个字符字符显示100毫秒，整个进度指示器持续大约40秒。提示：线程类的静态方法Thread.sleep()导致调用线程睡眠所指定的毫秒数（单参数版本）。

1.14 Fibonacci

编写Fibonacci类、Calculator类、Queue类和Item类，它们都位于各自的独立文件中。Item类表示当前所计算出来的Fibonacci项（long类型），以及一个标志finished指明Fibonacci项是否已经计算完毕；Calculator类负责计算Fibonacci项，将其包装到Item中，并发送到队列Queue中；而主类Fibonacci从队列中提取Item并显示。