实验 2. 异常处理-常用实用类

实验目的

（1）掌握 Java 的异常处理机制及相关实现方法；
（2）熟悉使用 JDK_API 进行软件开发；

实验步骤

步骤 1：创建工程，工程名以学号姓名方式命名，“2020010505-薛飞宇”。
步骤 2：为每道实习题目创建对应的包，如“work1”、“work2”……；
步骤 3：按照要求创建源代码，进行编写，注意编码格式，如缩进、命名规范等；
步骤 4：规范书写实习报告；
步骤 5：实现与测试。附上代码或以附件的形式提交，同时贴上必要的代码运行截图；
步骤 6：及时总结心得体会与备忘。

实验过程

实验题 1 对象排序
本项实验较为简单，先展示实验结果。

图 1.1 实验 1 结果

本实验的目的是按照奥运奖牌规则给相关的国家排序，主要通过 Arrays.sort()函数来实现，本实验只需要实现 compareTo 接口即可，在 C 语言中的 qsort()函数和 C++中的 sort()函数均有类似的 cmp()函数，因此本实验根据以往的经验实现即可。在 compareTo()函数的实现中，先比较两个对象的金牌数量，如果数量相等，再比较银牌数量，如果数量相等，再比较铜牌数量。最后改写 toString()方法，在 main 中排序输出即可。

以下是本实验的 UML 类图。

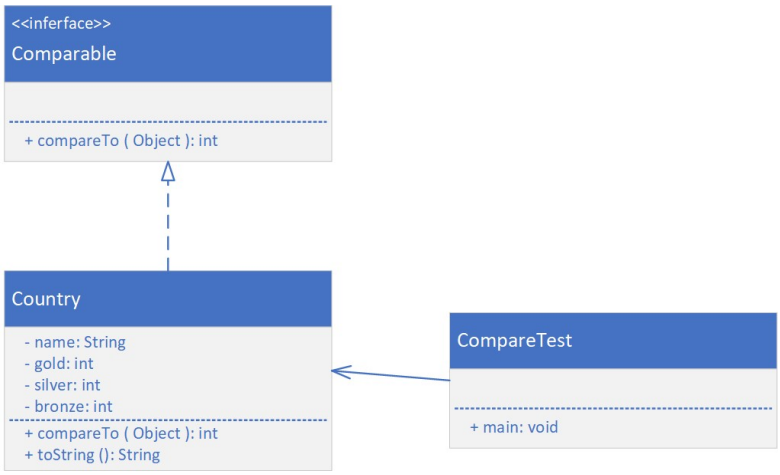

图 1.2 实验 1 的 UML 类图

实验题 2 常用实用类 Calendar 练习

本实验相对较难，主要考察对 JDK_API 的查阅和使用，先展示实验结果。

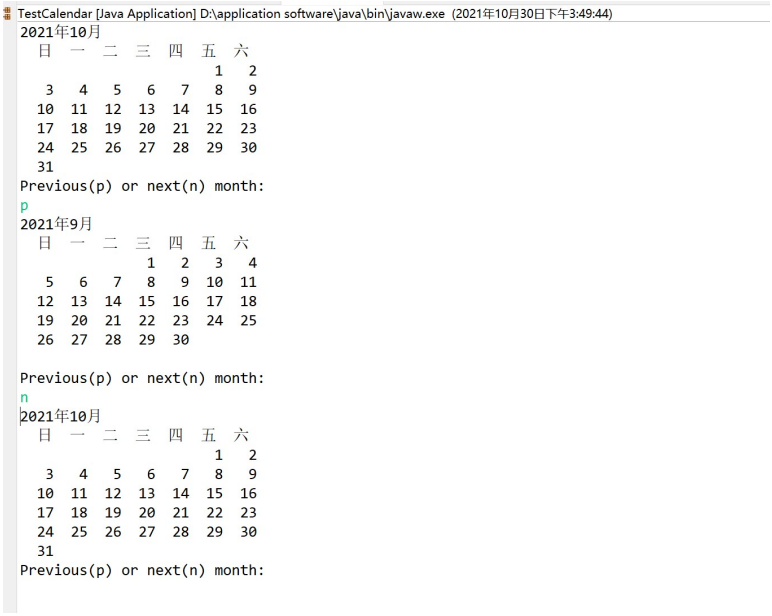

图 2.1 实验 2 结果

首先采用面向对象的思想，将日历封装为一个类，在这个类中导入 Calendar 包，再增加对 calendar 对象的一些方法和属性。

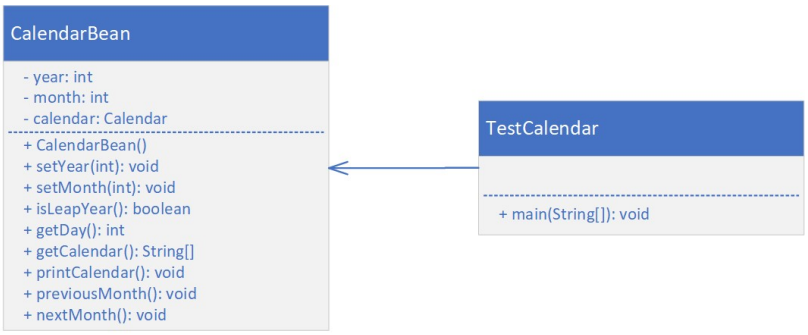

图 2.2 实验 2 的 UML 类图

CalendarBean 类中有 3 个实例成员：calendar，year，month；

CalendarBean 类中有如下方法：

CalendarBean()构造方法；

setYear(int), setMonth(int)修改器；

isLeapYear()用于判断当前年份是否为闰年；

getDay()根据当前月份和是否为闰年来确定当前月份的天数

getCalendar()用于获得一个日历的字符串数组的对象，方便输出，在这个方法中调用 getDay()方法，获得当前月份的天数，通过 calendar.get(Calendar.DAY_OF_WEEK)方法获取当前月的 1 号位于星期几，最后根据获取到的信息，给字符串数组对象赋值，并返回这个字符串对象；

previousMonth()，通过调用 calendar.roll(false)方法，对当前的月份和年份做出相应的改变；

nextMonth()，通过调用 calendar.roll(true)方法，对当前的月份和年份做出相应的改变；

printCalendar()，打印日历，先打印当前年份月份的信息，然后调用 getCalendar()获得字符串数组对象，最后格式化打印这个字符串数组

疑难问题：

1. 当月份为 1 时，对日历进行 p 操作，月份变为 12 月但是相应的年份不会减 1，同样地，月份为 12 时，对日历进行 n 操作，月份变为 1 月，但是相应的年份不会加 1。解决的方法也很简单，在 nextMonth()和 previousMonth()中加入特判，当月份为 1 月或 12 月时，对年份做出对应的改变即可。
2. 字符串对象的“判等”操作，由于 C++的习惯，我直接使用==对两个字符串对象进行判等。但是这两个对象保存的不是字符串的内容，而是字符串对象的引用，因此使用==的结果永远是 false。解决方案也很简单，调用字符串对象.equals(另一个字符串对象)即可。

解决这两个问题后，程序可以正常执行，如下图。

```
2021年11月
  日  一  二  三  四  五  六
      1   2   3   4   5   6
   7   8   9  10  11  12  13
  14  15  16  17  18  19  20
  21  22  23  24  25  26  27
  28  29  30

Previous(p) or next(n) month:
n
2021年12月
  日  一  二  三  四  五  六
              1   2   3   4
   5   6   7   8   9  10  11
  12  13  14  15  16  17  18
  19  20  21  22  23  24  25
  26  27  28  29  30  31

Previous(p) or next(n) month:
n
2022年1月
  日  一  二  三  四  五  六
                      1
   2   3   4   5   6   7   8
   9  10  11  12  13  14  15
  16  17  18  19  20  21  22
  23  24  25  26  27  28  29
  30  31
Previous(p) or next(n) month:
```

图 2.3 实验 2 测试结果 1

```
p
2021年2月
  日  一  二  三  四  五  六
        1   2   3   4   5   6
   7   8   9  10  11  12  13
  14  15  16  17  18  19  20
  21  22  23  24  25  26  27
  28

Previous(p) or next(n) month:
p
2021年1月
  日  一  二  三  四  五  六
                      1   2
   3   4   5   6   7   8   9
  10  11  12  13  14  15  16
  17  18  19  20  21  22  23
  24  25  26  27  28  29  30
  31
Previous(p) or next(n) month:
p
2020年12月
  日  一  二  三  四  五  六
            1   2   3   4   5
   6   7   8   9  10  11  12
  13  14  15  16  17  18  19
  20  21  22  23  24  25  26
  27  28  29  30  31

Previous(p) or next(n) month:
```

图 2.3 实验 2 测试结果 1

实验题 3 自定义异常及异常处理

本实验比较简单，先看实验结果。

```
ExceptionTest [Java Application] D:\application software\java\bin\javaw.exe (2021年10月30日下午3:52:49)
账户当前金额：2000
输入操作及金额：draw 1000
取出1000，当前余额：1000
输入操作及金额：save -1000
存入数据非法：-1000
输入操作及金额：draw -1000
取出数据非法：-1000
输入操作及金额：save 1000
存入1000，当前余额：2000
输入操作及金额：
```

图 3.1 实验 3 结果

本实验主要考察对自定义异常和异常处理的应用。

首先，根据要求设计 BankAccount 类，有一个数据成员 amount，设计相应的访问器和修改器，其他的方法及功能如下：

void deposite(int dAmount)：存款操作

void withdraw(int dAmount)：取款操作

void showBalance()：显示当前余额

自定义以下异常类：

InsufficientFundsException：余额不足异常。取款超出余额时，在 deposite(int)中抛出该

异常对象。

NegativeAmountException：负数金额异常。不管是 deposite(int)还是 withdraw(int)，金额都不应该是负数，否则抛出该异常。

此外，由于负数异常对应的操作有“存入”和“取出”两种，因此在 NegativeAmountException (String msg, int mode)的构造方法中，异常的消息内容可以用参数 msg 指定，操作由 mode 指定，getMessage()可以获取该消息字符串。

最后在 main 方法中通过 try-catch 语句进行异常处理。

最后，经过测试，程序正常执行。

以下是本实验的 UML 类图。

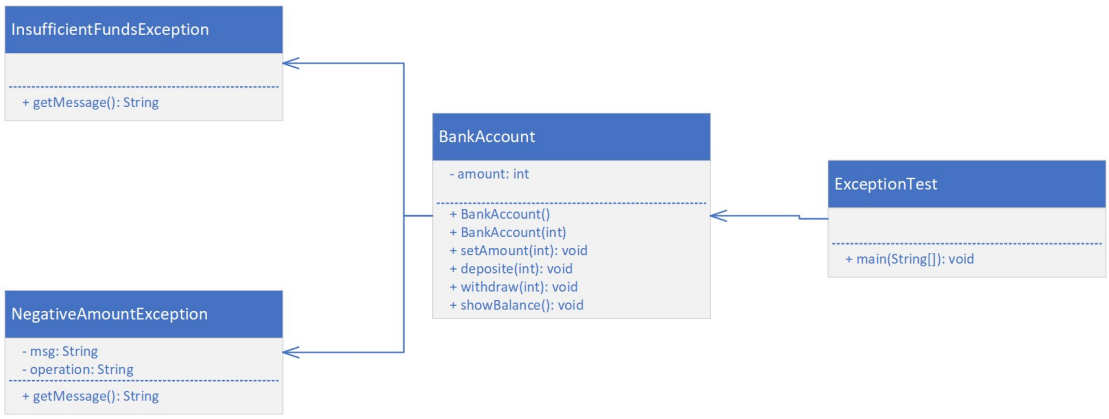

图 3.2 实验 3 的 UML 类图

实验题 4 正则表达式应用

正则表达式规则较为复杂，我还没有完全掌握，但本实验使用的正则表达式比较基础。实验要求匹配人名长度 2-3 个字符，手机号码长度 11 位数字，通过查询资料，得到中文的 Unicode 编码为\u4e00-\u9fa5，匹配其中任意一个字符即[\u4e00-\u9fa5]，重复 2-3 次，即为[\u4e00-\u9fa5]{2，3}，这是第一个捕获组，用括号（[\u4e00-\u9fa5]{2，3}），第二个捕获组是 11 个数字，也就是(\d{11})，在 java 字符串中，\需要重复 2 次，因此，最终的正则表达式是([\\u4e00-\\u9fa5]{2,3})(\\d{11})，通过 Pattern 类和 Matcher 类提供的相关接口，Pattern 可以新建一个正则表达式对象，而 Matcher 类可以新建一个需要匹配的原始字符串，再根据示例程序，完成本次实验，以下是实验结果。

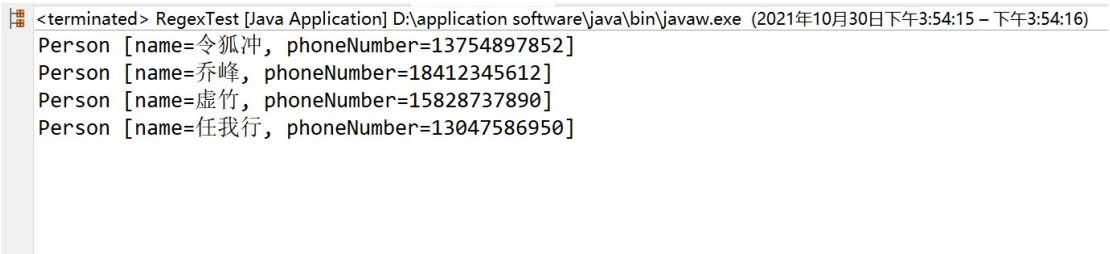

图 4.1 实验 4 结果

接下来，做选作数据，匹配座机号码，基本思路是使用“|”来匹配移动号码“或者”座机号码，因此写出第二个捕获组(\\d{11}|\\d{3,4}-\\d{7,8})，经过测试，程序可以正常执行。

```
<terminated> RegexTest [Java Application] D:\application software\java\bin\javaw.exe (2021年11月1日
Person [name=令狐冲, phoneNumber=13754897852]
Person [name=乔峰, phoneNumber=18412345612]
Person [name=虚竹, phoneNumber=15828737890]
Person [name=任我行, phoneNumber=13047586950]
Person [name=明教总部, phoneNumber=1939-3275648]
Person [name=思过崖, phoneNumber=993-27685954]
```

图 4.2 实验 4 的匹配结果

以下是本实验的 UML 类图。

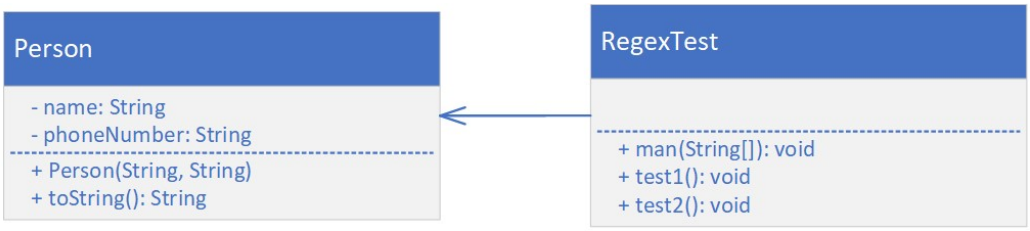

图 4.3 实验 4 的 UML 类图

实验总结

本次实验尚未解决的疑难：

本次实验相对简单，疑难问题已经全部解决。

心得体会：

在实验 1 中熟悉了 Comparable 接口和相关的 compareTo 函数，熟练掌握之后就可以方便地对对象数组进行排序。在实验 2 中学习了使用 JDK_API 进行开发，通过打印日历，增强了面向对象的思想。实验 3 自定义了异常类并且尝试处理，解决了程序异常或者出错的问题。在实验 4 中接触了正则表达式，但是完全掌握深奥的规则还需要多加练习。

课程建议：

课程讲的太快啦，如果可以，建议录制实习讲解视频。