CSC120 2025S Lab No.12
Precipitation Query

This assignment aims to write a program consisting of multiple class files. The application allows users to inquire about historical precipitation data in a city from 1931 to 2012. The application comprises five class files and one text file as the data source.

- `Const.java`: a file that defines constants used in another source files
- `OneDayInt.java`: an interface to be implemented in `OneDay.java`
- `OneDay.java`: an object class containing the precipitation of one day
- `Historical.java`: an object class for storing the historical record
- `PrecMain.java`: the source file containing the main method
- `precipitation-data.txt`: the data file

You must write `OneDay.java` and `Historical.java`.

Let us first look at some examples of execution to understand how the program works.

```
Start & end years: 1931 1932
Start & end months: 1 12
Start & end days: 1 31
The number of days is 731
The average Precipitation is 3.933
The maximum precipitation is 37.310 on 1931-06-14
The minimum precipitation is 0.000 on 1931-03-17
Try again (y/n)? y
Start & end years: 1931 2012
Start & end months: 1 12
Start & end days: 1 31
The number of days is 29951
The average Precipitation is 3.794
The maximum precipitation is 42.220 on 2009-11-01
The minimum precipitation is 0.000 on 1931-03-17
Try again (y/n)? y
Start & end years: 1966 1966
Start & end months: 1 1
Start & end days: 6 6
The number of days is 1
```

```
21 The average Precipitation is 1.490
22 The maximum precipitation is 1.490 on 1966-01-06
23 The minimum precipitation is 1.490 on 1966-01-06
24 Try again (y/n)? n
```

The user input appears after the colon.

The program receives a range of years, months, and days from the user and explores the data specified by the combination of the ranges. Since the range combination may contain non-existing dates (e.g., April 31st), we must exclude such combinations from consideration. The program counts how many combinations are valid and computes the average of the precipitation amounts across all the valid data points, the maximum, and the minimum. Lines 4-7, 11-14, and 18-21 are the lines that produce the results of exploration.

1 Class OneDay

The class must implement the two methods appearing in `OneDayInt.java`, `double getAmount()` and `String toString()`. The method `toString()` returns an encoding of the date and the amount as encoded in the output lines in the average starting with `Maximum` and `Minimum` The actual number is the amount in the record, and the texts appearing after `"on"` in these lines are the date information in the record.

Naturally, the constructor for `OneDay` takes three `int` parameters representing the year, the month, and the day and a `double` parameter representing the amount and stores these in its instance variables. You are free to choose how to encode the date information. You can use three integers (year, month, and day) or the `yyyy-mm-dd` texts to encode the information because we use the date information only for the output.

2 The Data File

The data is provided in a file `precipitation-data.txt` attached to this assignment. You must download it into the directory in which you have the class file. Each line of the data file is in the following format:

YEAR MONTH VALUE1 ... VALUE31

VALUE1 through VALUE31 represent the values for the 31 days of the month of the year. For months with fewer than 31 days, `-99.0` is the value for the non-existing dates (since precipitations cannot be negative). The lines appear in the increasing order of year and month. Because of this format, each year has $12 * 31 = 372$ entries. We will read values from the file and store them in a three-dimensional array of `double`, ignoring the year and month values. The first dimension of the array is the year, where the index zero represents 1931, and the last index 83 represents 2013. Here are the first 12 lines with the first four and the last 4 for the day entries:

```
1932    1  10.69  26.46  13.09   ...   0.27   0.25   1.77
1932    2   0.14   0.90   0.30   ...   0.05 -99.99 -99.99
1932    3   0.14   0.03   0.08   ...   7.41  11.65   2.14
1932    4   6.35   9.11   1.66   ...   2.12   1.05 -99.99
```

```
 1932    5   0.17   0.08   0.05   ...   0.75   0.98   0.00
 1932    6   0.03   0.00   0.08   ...   7.22  15.37 -99.99
 1932    7   1.80   6.01  13.47   ...   9.29   9.30   6.52
 1932    8   0.08   1.55   1.58   ...   2.55   4.45  11.97
 1932    9  14.82   5.91   7.67   ...   0.03   5.39 -99.99
 1932   10   4.19   3.22   0.65   ...  15.85   7.44   2.80
 1932   11  11.56   8.29  22.57   ...  17.49   4.35 -99.99
 1932   12  15.50  15.10   3.42   ...   4.15   0.15   6.50
```

For convenience, these quantities appear in class `Const` as constants as follows:

```
public interface Const {
  public static final double NODATA = -99.99;
  public static final int FIRST_YEAR = 1931;
  public static final int LAST_YEAR = 2012;
  public static final int LENGTH = LAST_YEAR - FIRST_YEAR + 1;
  public static final String FILE_NAME = "precipitation-data.txt";
}
```

3 Class Historical

Class `Historical` has one method `explore`, which receives six integers representing the range of years, the range of months, and the range of days and reports the results of exploration. For this purpose, the constructor for `Historical` reads from the data file and stores it in a three-dimensional array of `OneDay` objects `data`. The first dimension of the array is the year, the second is the month, and the third is the day. For example, you can design the indices so that `data[22][10][4]` is the one-day precipitation for the year `22 + Const.FIRST_YEAR`, the month `1 + 10`, and the day `1 + 4`; i.e., November 5th, 1954.

Here is how the constructor may work.

```
  public Historical() throws IOException {
    // instantiate a File object f with the file path "precipitation-data.txt"
    // instantiate a Scanner object using the File f
    // instantiate the three-dimensional OneDay array
    // Use a double-loop to iterate over the first two indices
    // ... read the first data as the year
    // ... read the second data as the month
    // ... use a for-loop to iterate over 31 days
    // ...... read the double data as the amount of precipitation
    // ...... instantiate a OneDay object with the year, the month, the day,
    // ...... and the amount and store it in the position specified
    // ...... the three indices
    //Close the Scanner
  }
```

The method `explore` explores `data` using triple indexing over the range corresponding to the offset indexing for the year, the month, and the day. During the triple-indexing examination, count how many of the data points are valid (that is, the amount the method `getAmount()` is not equal to `Const.NODATA`), computes the average, the maximum, and the minimum.

- Let's say that the triple indexing positions are `i`, `j`, and `k`.
- At the start of exploration, initialize an `int` variable `count` with the value of 0, a `double` variable `sum` with the value of 0, and initialize two `OneDay` variables `theMax` and `theMin` with the value of `null`.
- During the triple iteration, if the data has the amount not equal to `Const.NODATA`, it is a valid. If the data is valid, then do the following.
  - If `theMax` is `null`, store the `OneDay` object at the position in `theMax` and in `theMin`. Otherwise, compare the amount with the amount that `theMax` has. If the new value is greater, replace `theMax` with the current data. Do a similar updating on `theMin`.
  - Increase `count` by 1.
  - Update the `sum` by adding the amount.
- Report the value of `count`, the value of `sum/count`, the value of `theMax`, and the value of `theMin`. For the last two, use the `toString` method of `OneDay`.