Inf1B - OOP

Assignment 2 Report

s2134605

DoS: 11th March 2021

BEGINNER

Basic

List of desired program features:

- print yearly precipitation in a bar graph format horizontally
- print yearly precipitation in a bar graph format vertically
- print the precipitation for a month given its index in the year, and be able to do this vertically or horizontally
- scale and print a graph given a scaling coefficient
- generate random values of precipitation for given day in month

Program specification:

From a provided data set of min and max precipitation per month in a year, be able to generate and display yearly precipitation for the year in a bar graph format horizontally and vertically. Also from that same data, the program should be able to generate random rainfall values for days in a month given the month and the day, and using this be able to generate a bar graph of the rainfall per day given the month horizontally and vertically. Furthermore it is important that the user be able to scale the graphs by a given scaling coefficient

Additional requirements

- Display the max precipitation per month in bar graph format
- Display the min precipitation per month in bar graph format
- Display median precipitation per month in bar graph format
- Give a legend of scale (ex.: * = 2 cm rain) for each graph

INTERMEDIATE

Code Review

Functional Correctness

- In the method verticalGraph, first the max of the array of numbers passed as argument is found, and this is set as the increment variable i, and this is decremented until i == 0. The problem is that instead of doing this, an enhanced for loop should be used, which goes thru each element in the array and append the corresponding number of * or spaces. It does not make sense to loop the array based on the largest amount of rainfall rather the loop should loop until all the items in the array have had their corresponding number of stars appended to the StringBuilder. Another way of implementing the for loop is to use a for loop like the one in the method horizontalGraph.

  ```
  for(int i : array) {...}

       instead of

  for(int i = ArrayMax; i > 0; i--) {...}
  ```

Code Quality

- Modularity and reusability: The preparedData methods, etc. should be moved to the dataProvider.java class since they deal with the preparation of data.
- Robustness: Program is not able to handle odd user inputs like 13 for month index etc. Too easy to crash program.
- Robustness: Good use of access modifiers and final keywords to keep important data secure.
- Language Proficiency: Should use StringBuilder instead of String on PrecipitationGraph line 160, 189, 190, since Strings are immutable objects.

- Readability: More self-describing names needed for variables and methods, for example starString() should be changed to mmToStars().
- Readability: Comments need to genuinely add information for the programmer. Currently some comments are useless. Ex.: “Thirty days hath September, April, June, and November, All the rest have thirty-one, But February's twenty-eight, The leap year, which comes once in four, Gives February one day more.”.
- Readability: Not enough comments in the getRain method. The function and process of generating rain values is not clear.
- Readability: Int[] minRainInMM, maxRainInMM, daysPerMonth should be in SHOUT_CASE since they are non changing static constants.

Code documentation

1. Where more Comments would be helpful
    a. More comments would be helpful in the getRain method, specifically when the method starts creating random rain values using existing values from minRainInMM and maxRainInMM.
    b. More comments would be helpful in the preparedData(int[] month) method. A terse explanation of what the for loop does would be helpful.
2. Needless or uninformative comments
    a. Useless comment: “*/*** "Thirty days hath September,*

** April, June, and November,*

** All the rest have thirty-one,*

** But February's twenty-eight,*

** The leap year, which comes once in four,*

** Gives February one day more.*

**/*” (DataProvider 21-28)

    b. This comment is not needed since it is obvious that a new instance of the DataProvider class is being created. “An instance of DataProvider”

3. Comments are actively misleading, distracting or inappropriate
    a. Parts of comment that are underlined are verbose and unneeded, parts in bold are vague about the programming details they specify: “*This is like a temporary variable which does* ***massively complicated stuff in a loop****. Someone was saying its real efficient to sort of set it to like the first thing in the array but then I was like no why make it complicated so I'm just using a zero Anyway it's kinda working so why bother change it right?”* (Precipitation Graph 94-97)
    b. The verticalGraph() has more comments than are necessary. For example the following comment: “/ For every entry in the array:” merely describes the function of the for loop, evident without the comment. Or the following comment: “/ Print the row of stars” `for System.out.println(stars);` is simply not necessary. Comments should only be provided for the more complicated parts of the code. Too many comments only serves to clutter the code and make it harder to read.

ADVANCED

Legacy Code (Restructured and Renamed, with comments):

```
public class X {
	private int x,y;
	public void doubleXAndIncrement(boolean b){
		x<<=1;
		if(b==true){
			x++;y++;
		}
	public Boolean xAFactorOfTwo(){
		if(isYZero()==True) {
			return null;
		}
		return x%2!=0; // checking that x doesn't divide evenly by 2,
				// false if yes, true if no
				// (3%2 != 0) = 1!= 0 = true ; (4%2 != 0) = 0!=0 =
				false
	}
	public Boolean decrementyAndShiftX(){
		Boolean isXaNotFactorOfTwo = xAFactorOfTwo();
		if(isXaNotFactorOfTwo!=null){
			x>>>=1;
			y--;
		}
```

```
            return isXaNotFactorOfTwo;
      }
      public void setXandYtoZero(){
            x=y=0;
      }
      public int getY(){
            return y;
      }
      public boolean isYZero(){
            return y==0;
      }
}
```

Log of changes and description of method function

Previous Name	New Name	Function and Description (What it does is underlined)
m1()	*shiftAndIncrement()*	*Perform a left bitwise shift of 1 on x. Then if the parameter b is true, increment x & y by 1*
m2()	*xAFactorOfTwo()*	*If the result of the isYZero method is True (ie. Y is zero), return null, else return the value of x%2!=0 (Return true if x is not a multiple of 2, if x is a multiple of 2, return false)*

m3()	*decrementyAndShiftX()*	*Call the xAFactorOfTwo method, and store the result in isXaNotFactorOfTwo. If the value is not null, perform an unsigned right shift of 1 on x, and decrement y by 1. Return isXaNotFactorOfTwo*
m4()	*setXandYtoZero()*	*Sets both x and y to 0 when called.*
m5()	*getY()*	*Getter method for y, returns y*
m6()	*isYZero()*	*Returns whether y is zero. Checks whether y is equal to zero and returns the boolean result.*
z	*isXaNotFactorOfTwo()*	

1. Is there any justification for writing code like this (why/why not)?
    a. There is no justification for writing code in this way. It makes it incredibly difficult for fellow programmers to understand the code. The arbitrary method and variable names get confusing the more there are. It is hard to understand what method m1 does and how it is different from m3. Therefore one has to read every method. An intuitive name like GetY() or IncrementX() allows one to understand the general structure of the code without reading it all. Not to mention that the normal formatting of the code with tabs helps see the start and end of blocks of code. In a line one cannot do that.