Informatics 1: Object Oriented Programming

Assignment 2 - Report

<s2101367 Sean Choi>
<11-03-2021>

Basic

This section is about analysing given code and writing a specification

<u>List of desired program features:</u>

Holistically, the program has been made of three class. First class, which name is dataProvider, is a rainfall data generator so that this program uses the made precipitation data to describe a rainfall trend in a graph form. Second part has a graph generator with a made data in dataProvider class. The last third class, which is Test class, is used for implementing a program.

dataProvider class

- This class has a getRain instance of a class. getRain method produces random rainfall data with a given preset maximum and minimum rainfall data.
- getRain method
  - It creates synthetic rainfall data with a given pair of integers; the month and a day within that month.
  - A synthetic rainfall data is an approximate value made of instance variables; the maximum and minimum rainfall data, daysPerMonth, and rand.
    - The maximum data(maxRainInMM) and minimum rainfall data(minRainInMM)
      - These data have preset month values indexed by month. Data are saved in a form of an integer array
    - daysPerMonth instance variable
      - This integer array of the variable contains days per every each month.
    - rand instance variable
      - the rand instance variable has Random as a type so that it refers to a random value generator.

PrecipitationGraph class

- This class has several instance methods to print many kinds of a graph with a generated variable.
- Graphs have a horizontal graph and vertical graph
- For each graph, graphs show a monthly graph or yearly graph.

- When using a generated data getting from dataProvider, those data is adjusted by a given scale value. The default scale is 1. Thus, data adjusted by a scale value is called preparedData method in this program.
- Each graph are generated by using data of preparedData.

<u>Program specification:</u>

The program is about showing a precipitation graph with synthetic precipitation data for every day of any year. The precipitation data is needed to be displayed as a graph. The precipitation graph should display yearly precipitation and certain month precipitation. Each graph should have a common format regarding x and y axes. These graphs should be clearly drawn by representing a clear separation of the x and y-axis with the usage of some String, for instance, " | " or " - ". These graphs also should have each name above each graphs depending on the types of a graph; yearly graph or a certain month precipitation graph. To be specific, graphs, at first, will be divided into a vertical graph and horizontal graph. In the case of a vertical yearly graph, it has the y-axis which is consisted of each month like from January to December, and the x-axis of the graph does not have to have a specific name, but instead, each mm of rain is shown by one star. In the case of a certain month vertical precipitation graph, the y-axis would be made of each day in each month. Contrary to the vertical graph, the x-axis of a yearly horizontal graph has each month and its y-axis shows an amount of mm rainfall by representing one star. In the case of a certain month horizontal graph, the x-axis will be filled with each day in each month. All graphs have the same and default scale, which has 1 scale meaning that one star represents each mm of precipitation. For each graph, a scale used to generate the data is needed to show such as printing "Scale: 1.0". Regarding certain month precipitation, the programmer subsequently must be able to select a certain month as a parameter which they want to put. In other words, users want to check each days' precipitation data of a certain month at a glance so that they can figure out the maximum or minimum amount of precipitation of a certain month given by users' input. In addition, the program wants to check whether the yearly precipitation data is correctly rounded to the nearest integer.

Regarding precipitation data input and output, synthetic precipitation data for every day of any year is needed to be generated. Every synthetic rainfall data has an integer type. Maximum and minimum precipitation mm data in a given month are provided. For example, Minimum rain data contains 5, 6, 7, 4, 2, 0, 0, 0, 2, 3, 5, 7 which refers to an array of the minimum rainfall starting from January to December. Maximum rainfall data includes 35, 38, 40, 30, 15, 10, 12, 15, 25, 32, 35, 38. In addition, days per month should be exact. For instance, January has 31, and February has 28. Unlike maximum and minimum precipitation data, daily maximum rain data should be an integer and be automatically generated with given maximum and minimum rain data and a figure of distance through the month. A figure of distance through month refers to a value which is a certain day divided by days per a certain month. When the program randomly chooses daily rainfall data, a programmer should carefully think of how random data will be generated. The random rain data is made up of the multiplication of a random float value between 0,0 to 1,0 and the maximum of today's rain variable(I recommend you to use maxRainToday as a variable name). The maxRainToday variable refers to the summation of the maximum rainfall of this month and a maximum rainfall difference figure. The maximum rainfall difference figure is the multiplication of the two figures. The first one refers to a figure of distance maximum rainfall difference between this month and next month. The second one is the figure of distance through months, which has been defined above. Through randomising a daily rainfall value, the program can get daily random rainfall data. But, this rain data should be less than a maximum rain data of a certain month and greater than the minimum precipitation data of that. For example, let say that the maximum rainfall of February is 38mm and the minimum has 6mm. The maximum rainfall of March is 40mm. If a user wants to know the precipitation of 15th February, we can attain a random precipitation value of it by using given information. maxRainToday is 38 + ((40-38) * a figure of distance through month). Since a figure of distance through February is 15 / 28, int maxRainToday is 39. If a random float value is 0.5 by generating a random float value between 0.0 to 1.0 the daily random rainfall data is 39 * 0.5, which is 20. The result value should be an integer. Since the rain data is not less than the minimum rain data of February nor greater than the maximum data of that, the random precipitation data of the 15th of February refers to 20.

Additional requirements

1. A rainfall comparison graph.

The original program shows one graph for a selected month. However, we sometimes need to compare the rainfall of the month with that of another month. So, we can add a feature to show a rainfall comparison graph with two monthly input data in one graph. This comparison graph will help readers to analyse the difference between the rainfall of two months. In addition, the graph has to be drawn not with a star, but with a line in order to show a comparison between two selected months. A line graph can clearly show the differences between a selected monthly rain data.

2. Precipitation graph transmission system.

These kinds of precipitation graph are able to be required for those who investigate a weather or have a job affected by the weather. Thus, we need to add a feature to transfer precipitation graphs to those who need them through email or text message. Since this graph is text-based data, the way to transfer the data has to follow the way to transmit text-based data.

Intermediate

This section is to review and improve a given code.

Program functionality		
Requirement	Issue	Proposed solution
The local variable, int maxRainNextMonth, should be initialised with maximum rain data of the next a given month.	Instead of using maxRainInMM[(month + 1) % 12] variable, the maxRainNextMonth local variable is initialised with minRainInMM[(month + 1) % 12] variable.	The error occurs on line 65 in dataProvider class. The local variable of getRain method should be able to be initialised with maxRainInMM[(month + 1) % 12] variable.
When getting a difference maximum data of rainfall, this difference should be absolute number.	If different maximum data of rainfall is not an absolute number, it can be a negative number. The negative number can cause a problem.	The error occurs on line 66 in dataProvider class. This local variable of int maxRainfallDifference should be able to be Math.abs(maxRainNextMonth – maxRainThisMonth).
The program should use the right parameter when using the function.	The parameters of getRain function, which is on line 65 in PrecipitationGraph class, is used wrongly.	In PrecipitationGraph class, the error occurs on line 65. the parameters of getRain function should be placed in the same position when it is defined. Thus, as it is defined, it should have the month and day parameter in order, not the day and month.
The program should use the right parameter when using the function.	In Test class on line 14, the programmer tried to show the precipitation for the month of January, but it shows February.	In order to see precipitation for the month of January, this graph.monthVertical should have 0 parameters, not 1.

Code quality			
Code affected	Issue	Proposed solution	Explanation/justification
Method "getRain" of dataProvider class	Structure: a good design program is needed to be split into manageable chunk.	Method "getRain" can be divided into three parts. The first chuck, which is from line 56 to 61, is to check the valid month and day variable. The second chuck, which is from line 63 to 69, is to create synthetic valid rain data with two integer inputs representing the month and a day within that month. The third chuck, which is from line 71 to 76, is to check and return a valid synthetic rain value.	When we can divide the code into manageable chunks, the reusability or modularity of the code can be increased.
PrecipitationGraph class	Structure : a repeated code is used unnecessarily.	Instead of declaring the same DAYS_PER_MONTH instance variable, we can access static dataProvider.daysPerMonth through class dataProvider reference.	Since public static daysPerMonth array constants are defined on line 29 in dataProvider class, the program does not have to re-define the same constants in other class such as PrecipitationGraph. If the program wants to use these constants, they can access them via class dataProvider reference. Thus, we can delete DAYS_PER_MONTH instance variables.
-Precipitation Graph class -dataProvider class	Structure (Extensibility) : whenever adding a new feature, the whole program can be redesigned.	We can use class hierarchy to improve the extensibility of this program. dataProvider class can be superclass and PrecipitationGraph class can be subclass. Thus, PrecipitationGraph subclass extends dataProvider superclass. In this code, the public class PrecipitationGraph extends dataProvider.	When PrecipitationGraph extends dataProvider class, subclass PrecipitationGraph class has all features of superclass and an extra feature can be added in sub-class. Therefore, class hierarchy makes a program add a new feature without redesigning the program.
An instance of DecimalFormat in PrecipitationGraph method	Robustness: using correct access modifier.	Using a private access modifier is needed to be used.	This instance of DecimalFormat is used only in this class, thus private is a more proper choice. Also, field may not be public.

Method “getRain" of dataProvider class	Robustness: If the return value of getRain method is less than minimum rain data so that this data will be initialised with minimum data, the return value becomes wrong when it can be sometimes more than maximum rain data.	In dataProvider class, there is an error on line 72. This getRain method should return the value which is less or equal to maximum rain data and greater or equal to minimum rain data. Thus, this getRain method has to return the maximum rain value if it is greater than the maximum rain data. When the return value of getRain method is less than minimum rain data, it returns minimum rain data.	If user input is out of the scope where a rain input is not less or equal to maximum data and not greater or equal than minimum data, the input is a wrong value and then the program can be crashed. Therefore, the program should check and return a valid rain value.

In dataProvider class	Robustness : There is no a constructor.	In dataProvider class, there is no a constructor. a class should always contain a constructor even if the body is empty.
		
		Add this below.
		public dataProvider() {
		}
			Including a constructor whether the body is empty or not is a coding convention. Following the convention can increase quality of code, robustness, and readability.

Method “starString” of PrecipitationGraph class	Technical proficiency : String class is immutable so that the change of String can lessen efficiency.	In PrecipitationGraph on 82line, it would be more efficient code when StringBuilder class is used instead of String.
		
		* This code below is the modified code.
		private StringBuilder starString(int amountOfRain)
		{
		StringBuilder stringBuilder = new StringBuilder();
		for(int i = 0; i < amountOfRain; i++)
		{
		stringBuilder.append("* “);
		}
		return stringBuilder;
		}
			Since String class is immutable, using String class is not memory efficient when there are continuous the manipulations of String class. Thus, StringBuilder helps you manipulate Strings better. This is because StringBuilder is mutable.

dataProvider class	Readability: class names start with a lower letter.	The name of dataProvider class should be changed in to DataProvider.	This is a convention to have a capital letter when using class name.
dataProvider class	Readability: variable name is written with SHOUT_CASE.	In PrecipitationGraph class, the array of DAYS_PER_MONTH on the line 37 should be daysPerMonth.	Variable name needs to start with a lower letter and should be in camel case notation.
- dataProvide r class - Test class	Readability: the program should have consistent formatting throughout all classes.	Unlike dataProvider class, these Test and PrecipitationGraph class have different formatting. For instance, curly brackets in class PrecipitationGraph do not open at the end of a line. The proper position of braces for classes, methods, if, else, for, do and while statements are at the end of a line.	Using consistent formatting is one of the naming conventions. The naming conventions leads the program to be understandable to make the programer read easily the program. Also, following a code convention increases a quality of code.
PrecipitationG raph class	Readability: There is no a class comment at the top of PrecipitationGraph class.	At the top of class PrecipitationGraph, the class comment needs. For example, /** * This class can be used to draw a graph. **/	Writing a class comment at the top in every class is the one of coding convention. A reader can easily understand the role of the class with the comment of it. Thus, this coding convention increases readability.
Methods of PrecipitationG raph class	Readability: each method in PrecipitationGraph class does not have a method comment. -<line 70> private int monthlyAverage(int month) -<line 83> private String starString(int amountOfRain) -<line 93> private int findMax(int[] array)	One examples of these issue. /** * create a rain data of monthly average. * @param month for which you want rain data(in the range [1, 12], inclusive) * @return the average rainfall per day **/ private int monthlyAverage(int month)	Wring a method comment for each method can lead to increase readability. Due to a method comment, a reader can easily understand a method and a code.

PrecipitationGraph class	Readability: the names of each method in PrecipitationGraph class do not have verb.	• (line69) monthlyAverage - > getMonthlyAverage • (line 82) starString -> displayStarString • (line 146) HorizontalGraph -> displayHorizontalGraph (line 157,210,219,228,237 have the same problems)	As for naming convention, a method in java should be verb.
- dataProvider class - Precipitation Graph class - Test class	Readability: this code uses tab, not space.	Every classes in this program are needed to be written with space.	a program often uses in general space, not tab., especially in Open -source software.

Note: If you want to refer to code by line number, you must use the line numbers in the provided code. These may change if you e.g. add comments to the code to help your own understanding!

Code documentation			
Code affected	Issue	Proposed solution	Explanation/justification
The line 17 code of main Method in Test class	The extra comments are able to be needed on line 17.	The extra comment: n scale means that each mm of rain is represented by n * one star. For example, the 0.5 scale shows 1 star represents 2 mm of rain.	When using setScale function, an extra explanation about scale in this program is required to understand this code.
Method of “getRain” in dataProvider class.	Method of “getRain” is too long that it is needed to be divided into manageable chunk, and then comments require for each chunks.	The line from 56 to 61 needs the comment which is “Having valid month and day variables.” The line from 63 to 69 needs the comment which is “Creating a synthetic rain value”. From line 71 to 76, the comment can be “Checking and return a valid synthetic rain value”.	The extra comments are able to be required on this method because a reader may have a difficulty of understanding or take time to understand this code without extra comments.
Method of “monthlyAverage” in PrecipitationGraph class	These comments are uninformative.	Those comments of method “monthlyAverage” in unnecessary comments so that these are needed to be deleted.	These comments are just describing basic functions of the codes.
Method of “findMax” in Precipitation class.	These comments are uninformative. It’s better not to leave notes in the code.	Those comments of method “findMax” in unnecessary comments so that these are needed to be deleted.	This comment is a note about what the programmer is thinking. This comment does not help for a code reader.
Method of “getScale” in Precipitation class.	This comment is uninformative.	Those comments of method “getScale” in unnecessary comments so that these are needed to be deleted.	This comment is just describing basic functions of the codes.
Method of “setScale” in Precipitation class.	This comment in “setScale” method is inappropriate. This comment is about the explanation of “setScale” method, not this.scale = scale;	The position of this comment is needed to be changed above “setScale” method to explain it.	A comment should be appropriate explanation for a subject.

daysPerMonth array comments in dataProvider class	This comment has typo to explain the code. As one example, “Thirty days hath September, April, June, and November … “	This comment having typo is needed to be changed into “Thirty days have September, April, June, and November … “	Having a typo in a comment can lead to misunderstand the code.

Advanced

- Appendix 1

This section analyses and evaluates a given legacy code. This code is a boolean stack data structure code so that a boolean type of data is put into a stack structure by using three main methods, including push, pop, and peek.

Proposed variable x, y, z name: x is intValue, y is size, and z is peekbit.

Method "m1":
Proposed method name: push

Description of what it does**:** This push method makes the program store a boolean type of data into a stack.

Description of how it works: This push method works like this. It makes an original bit to be shifted left and then put 1 or 0 into the end of far-right bit. In other words, a value of 1 or 0 is put into the first bit from the right. If a boolean data is true, the value is 1, or the value is 0 when a boolean is false.

Method "m2":
Proposed method name: peek

Description of what it does: This peek method checks the top value of the stack. What the top means is that it is the position of a bit that is located at the first bit from the right of the bit.

Description of how it works: If the first bit from the right is 0, it is an even number so that false data is stored. But, if the first bit from the right is 1, it is an odd number so that true data is stored.

Method "m3":
Proposed method name: pop

Description of what it does**:** This pop method makes a stored data to be extracted. The last added data is removed first.

Description of how it works: This method reads the first-bit value from the right, and then an original bit is shifted right by getting rid of the first bit from the right.

Method "m4":
Proposed method name: clear

Description of what it does: This method initialises the intValue and size of this class. What this means is that this method initialises a stack.

Description of how it works: This method initialises a stack by assigning 0 to the intValue and size of the class.

Method “m5”:
Proposed method name: size

Description of what it does: This method gets the size of this stack.

Method “m6”:
Proposed method name: empty

Description of what it does: This method checks if a stack is empty or not.

Question 1: What kind of data structure is this and what would be a better class name?

This code is about a stack data structure storing boolean type data. Therefore, this class name would be BooleanStack.

Question 2: What are the advantages and disadvantages of the chosen data representation?

This int intValue in BooleanStack class has a very good memory efficiency compared to when using Boolean arr[]. This is because the method of “peek” in BooleanStack checks the top of value by just peeking the first bit from the right. This method uses just one bit by checking true or false. Instead of BooleanStack, the usage of Boolean arr[] is memory inefficient. This is because each element of an array is object, and the object is bigger than 4 bytes. Using an object to store true or false data is memory inefficient. Therefore, in terms of memory efficiency, this BooleanStack is efficient.

On the other hand, int intValue is only able to store up to 32 bit, which is 4 byte. In other words, a data value is crashed when pushing data more than 32.

Question 3: Is there any justification for writing code like this (why/why not)?

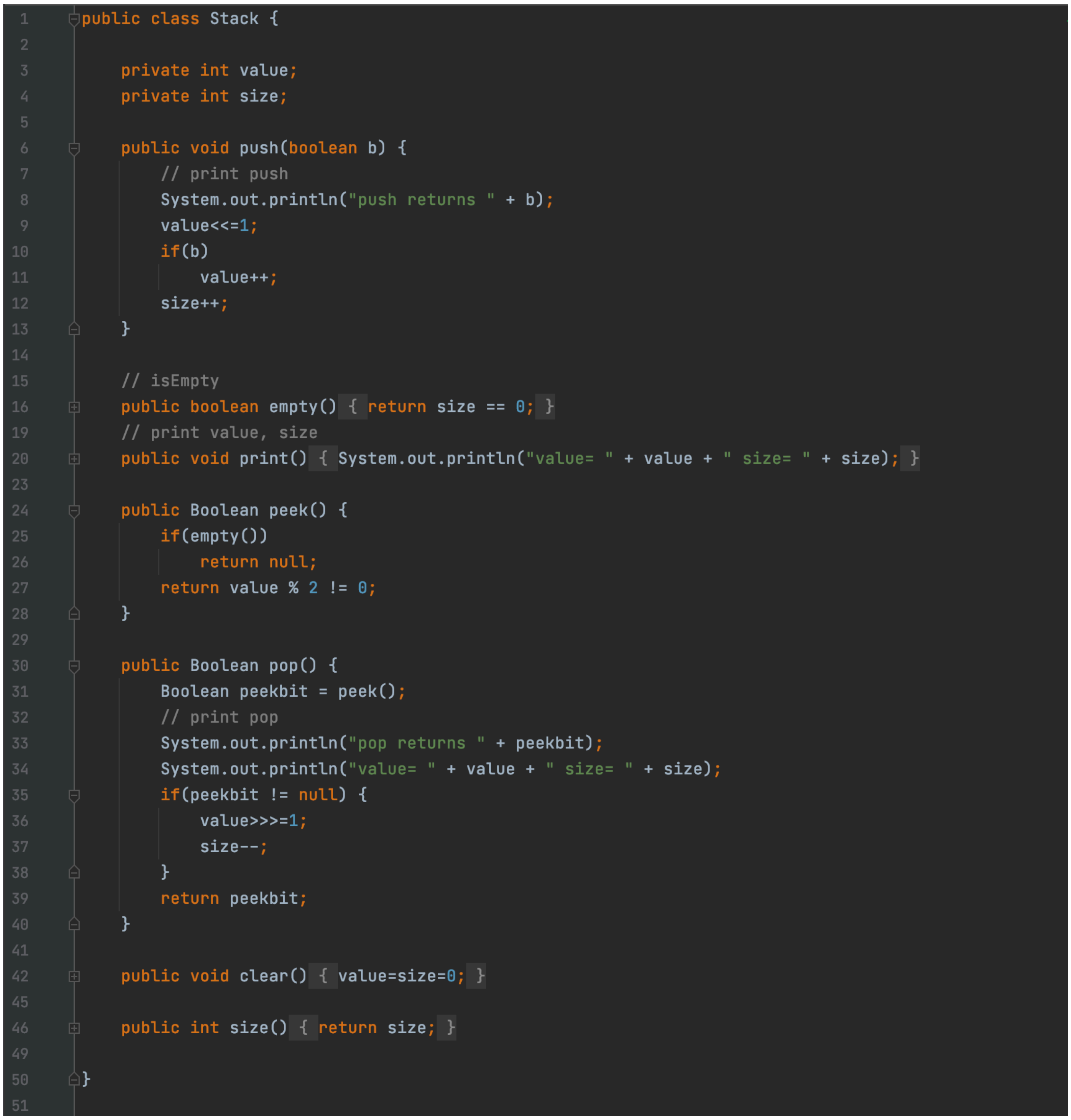
In terms of coding style, the given code before modifying can not read well and obscure. In order to improve code qualities, using a descriptive but clear names for variable, methods and classes is required.

Also, using consistent formatting in this class leads to improve this quality. These efforts can bring about improving readability and modularity of the code.

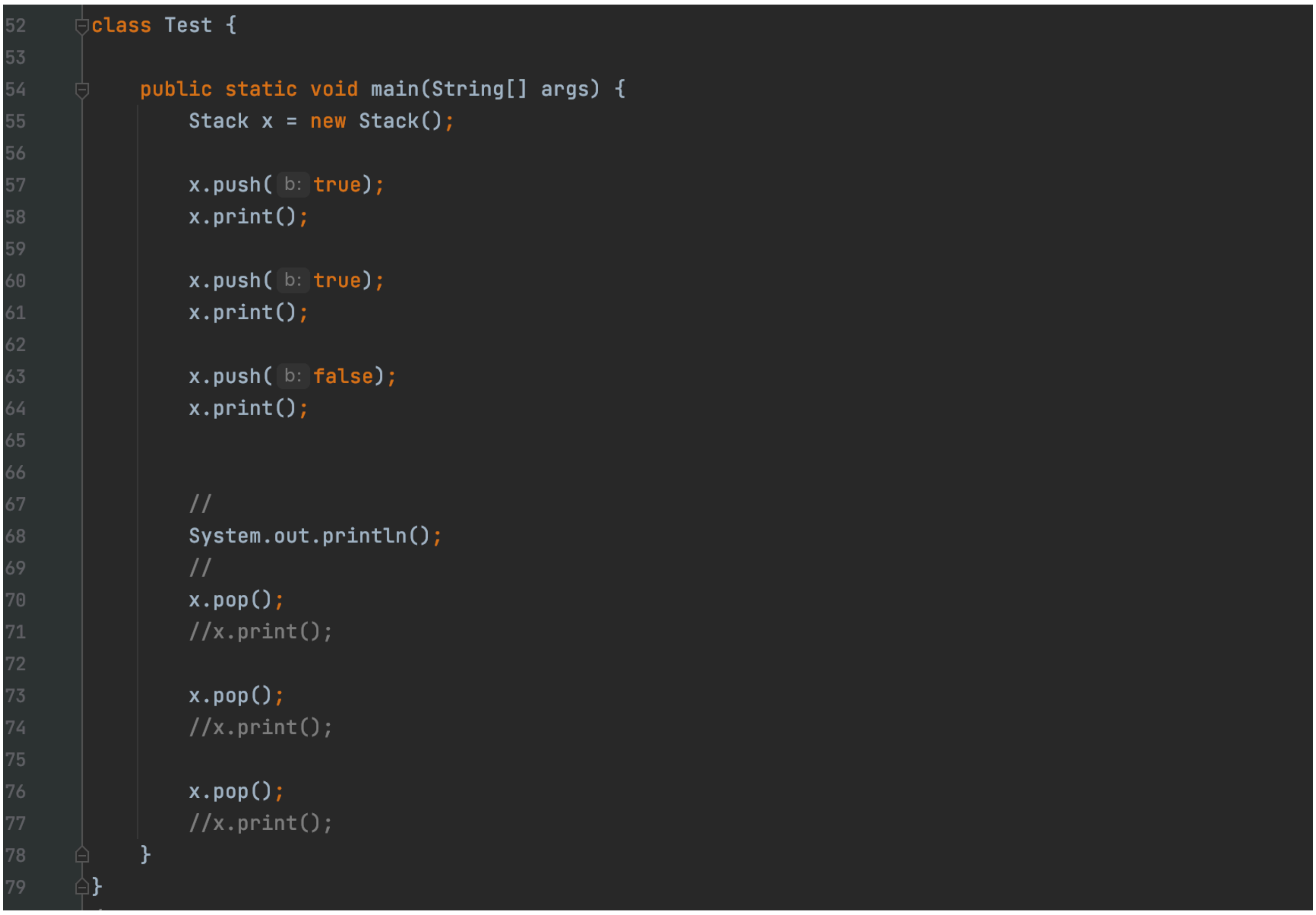
Furthermore, an appropriate usage of a certain method improve the memory efficiency of the given class. This code is about a stack data structure storing boolean type data. Since a boolean is 1-bit data, int intValue is used for storing the data to improve the memory efficiency. Although we can use others to store data like an array, your predecessor uses integer data type for the memory efficiency. If the size of pushed data is less than 33 bit, using integer data type might be efficient. if not, other data type is required to store them.

Appendix 1.

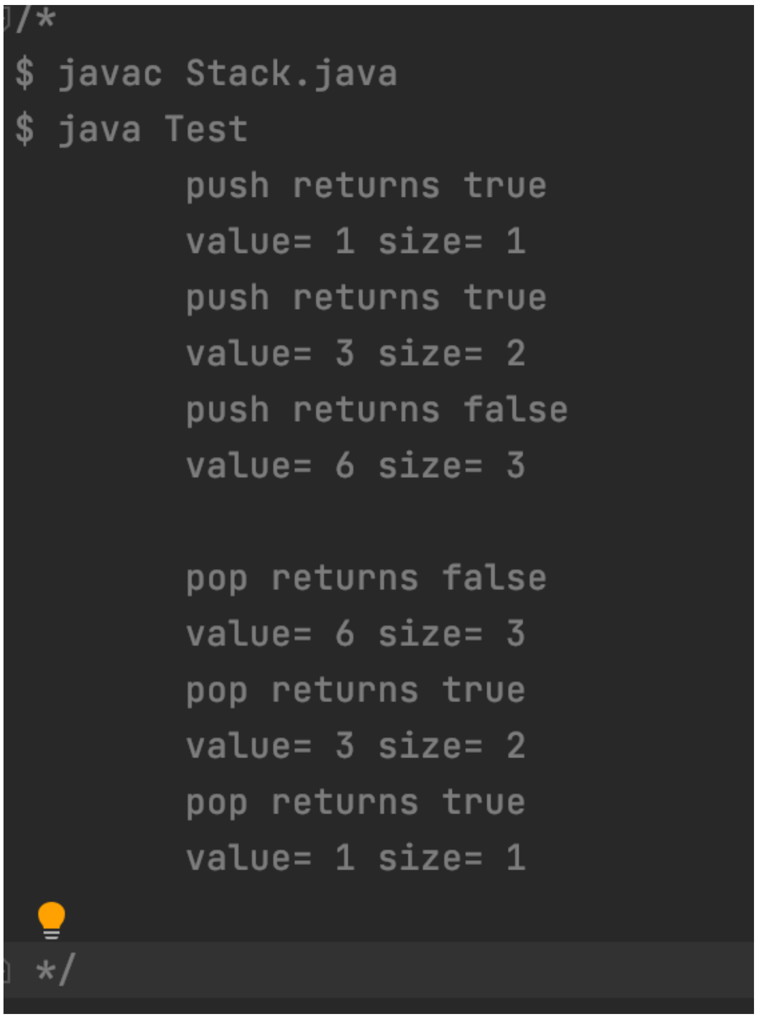
- The modified code of X.java



```java
public class Stack {

    private int value;
    private int size;

    public void push(boolean b) {
        // print push
        System.out.println("push returns " + b);
        value<<=1;
        if(b)
            value++;
        size++;
    }

    // isEmpty
    public boolean empty() { return size == 0; }
    // print value, size
    public void print() { System.out.println("value= " + value + " size= " + size); }

    public Boolean peek() {
        if(empty())
            return null;
        return value % 2 != 0;
    }

    public Boolean pop() {
        Boolean peekbit = peek();
        // print pop
        System.out.println("pop returns " + peekbit);
        System.out.println("value= " + value + " size= " + size);
        if(peekbit != null) {
            value>>>=1;
            size--;
        }
        return peekbit;
    }

    public void clear() { value=size=0; }

    public int size() { return size; }

}
```



```java
class Test {

    public static void main(String[] args) {
        Stack x = new Stack();

        x.push( b: true);
        x.print();

        x.push( b: true);
        x.print();

        x.push( b: false);
        x.print();


        //
        System.out.println();
        //
        x.pop();
        //x.print();

        x.pop();
        //x.print();

        x.pop();
        //x.print();
    }
}
```



```
/*
$ javac Stack.java
$ java Test
        push returns true
        value= 1 size= 1
        push returns true
        value= 3 size= 2
        push returns false
        value= 6 size= 3

        pop returns false
        value= 6 size= 3
        pop returns true
        value= 3 size= 2
        pop returns true
        value= 1 size= 1

 */
```