Program 4 – Shape Histogram Report

The purpose of this assignment is to practice using the various concepts learned in Chapter 18 of our textbook, with a particular focus on using the Streams API presented in Section 18.6. Specifically, the goal of this assignment is to write a program that analyzes our Shape data file to produce a width versus height histogram displaying minimum width, maximum width, average width, minimum height, maximum height, average height, and total number of shape records within a sequential set of width ranges.

1 User Interface Specifications

The program should be written to meet the following User Interface specifications. This section corresponds to Item 3 of the "Program Rubric" at the end of this specification. Please refer to the sample output in Section 4. The following command-line interface should be implemented.

There will be no user input with this program. The user interface will consist of a well-formatted report that displays a table of data like that shown in Section 4. The table must include 10 rows of data in the following 8 columns.

1. Range - an integer value indicating the width range number, or simply the "row number" of the table starting at 1.
2. Minimum Width - The minimum width for the current width range. Will start at 0 and increment by 10 for each row.
3. Maximum Width - The maximum width for the current width range. Will start at 10 and increment by 10 for each row.
4. Average Width - The average width of all shapes found in the current width range.
5. Minimum Height - The minimum height of all shapes found in the current width range.
6. Maximum Height - The maximum height of all shapes found in the current width range.
7. Average Height - The average height of all shapes found in the current width range.
8. Count - The total number of shapes found in the current width range.

Each column header and row of data must be well formatted and easy to read. You may duplicate the example shown in Section 4 or produce a well formatted style of your own.

2 Java Implementation

The program should be implemented in Java using the Eclipse IDE as specified below. This section corresponds to Item 4 of the "Program Rubric". This assignment will require a single Main program file as described below.

2.1 Shape.java

Update your Shape class to override both the `+hashCode():int` and `+equals(other : Object) : boolean` methods of `Object`.

2.1.1 `+hashCode():int`

For this method, simply create a String that contains the ShapeType value, width, and height of the Shape. You could use String.format, StringBuilder, or simply concatenate the values. After the String is built, simply return its hash code using its .hashCode() method.

HINT: this is not complicated so don't overthink it.

2.1.2 `+equals(other : Object) : boolean`

For this method do the following:

1. If `(this == other)` then return true (addresses are the same so it's the same object).
2. If `(other instanceof Shape)` is false, then return false (other is not a Shape).
3. Type case other to a Shape using a local variable then compare the type, width, and height of this and other. If all three values are the same then return true, otherwise return false.

You may write the if-statements any way you choose so long as the logic is as described above.

DO NOT use "Source -> generate hashCode() and equals()" in Eclipse to write these methods.

DO NOT use Objects.hash, or Objects.equals (we have not covered them in our class).

2.2 Main.java

Write your implementation using the Stream API described in Section 18.6 of the textbook for all data filtering and processing. There are many possible ways to write the code to complete this project and you are free to develop your own solution. For the best grade, your code should adhere to the following specifications:

1. Load the "shapes_20230701.csv" file only once at the start of the program and use a Set implementation of your choice (See Section 18.3) in which to store the data using a reference variable of type Set<Boundable>.

   HINT: rewrite the loadData method from Program 3 to use a Set<Boundable> instead of ArrayList<Boundable> return type. Since both Set and List are Collections, the code for your loadData method will be very similar and require very little change. Call loadData at the top of your program. <u>**"ArrayList" should appear nowhere in this program.**</u>

2. Each time you need a Stream, use the Set interface's stream() method to create an instance of Stream<Boundable> from the Set<Boundable> instance of Item 1 above.

3. For each width range (HINT: Use a *for* loop), use the Stream<Boundable>, filter method, to create a filtered stream that includes only the Boundable objects within the current

width range. (See Intermediate Operations on Streams, pg. 1177). NOTE: inside your loop, you can create local variables for the minimum and maximum width of the current width range. You can use these local variables inside your lambda expression for the filter method so long as you declare them with the "final" keyword.

4. Inside the loop of Item 3, To find the average width, first use the Stream reduce method (Section 18.6, pg. 1182) to compute the sum of all Boundable widths, then divide by the number of shapes found within the current width range to compute the average. **Do not use a loop of any kind to compute the sum**. NOTE: if no shapes are found in the current width range, then the average width value should be 0.

5. Inside the loop of Item 3, to find the minimum height within the current width range, use either the Collections.min method (Section 18.5, pg. 1170) or the Stream min method (Section 18.6, pg. 1174). **Do not use a loop of any kind to compute the minimum height**. NOTE: if no shapes are found in the current width range, then the minimum value should be 0.

6. Inside the loop of Item 3, To find the maximum height within the current width range, use either the Collections.max method (Section 18.5, pg. 1170) or the Stream max method (Section 18.6, pg. 1174). **Do not use a loop of any kind to compute the maximum height**. NOTE: if no shapes are found in the current width range, then the maximum value should be 0.

7. Inside the loop of Item 3, To find the average height, first use the Stream reduce method (Section 18.6, pg. 1182) to compute the sum of all Boundable heights, then divide by the number of shapes found within the current width range to compute the average. **Do not use a loop of any kind to compute the sum**. NOTE: if no shapes are found in the current width range, then the average height value should be 0.

8. Inside the loop of Item 3, at the end of the iteration of your loop, display all information to the console as shown in Section 4.

   HINT: Use one of the formatted String methods to get nice looking columns and tabular data output.

2.3 Comments

Your programs must always include a project documentation comment at the top of your main program file. For example.

```
/**
 * <Your Name Goes Here>
 * CPS142 – Summer 2023
 * <current date goes here>
 * Instructor: Adam Divelbiss
 * Assignment: Program04
 * Purpose: <write the purpose of your program here>
 */
```

All classes and methods, except for methods that override well documented superclass methods, must have a documentation comment.

Also be sure to use a single line comment to document/describe each major section in the code.

3 Deliverables

The deliverable files will be uploaded to the assignment in Blackboard. Deliverables consist of the following .java file(s). DO NOT SEND THE .class FILES. I CANNOT USE THEM.

- Main.java
- Shape.java – Updated with your changes for this assignment.
- Include the following files from your previous programs with any updates or corrections you may have written.
    - Boundable.java
    - ShapeType.java
    - BoundingBox.java
    - Rectangle.java
    - Circle.java
    - ShapeException.java
    - ShapeFactory.java
- Optionally, if you do not include the other files, I will reuse the most recent ones you previously submitted. Do not use this option if updates or corrections were needed.

The due date/time is:
Saturday, August 12, 2023, at 11:59PM

Materials that are Provided
Use the CSV file containing the Shape databases that was already provided with Program 01.

4 Sample Output

The following output demonstrates what the completed program will print to the console. Use it as a "sanity check" to ensure that your program is working properly.

```
Welcome to the Shape Database 4.0!

Width/Height Histogram Analysis:

         Mininum    Maximum    Average    Mininum    Maximum    Average
 Range     Width      Width      Width     Height     Height     Height      Count
----------------------------------------------------------------------------------
     1      0.00      10.00       4.99       0.03      97.03      26.79      1,018
     2     10.00      20.00      15.07       0.66      97.79      32.49        976
     3     20.00      30.00      24.91       0.62      97.98      37.30        917
     4     30.00      40.00      35.06       0.10      97.62      43.04      1,020
     5     40.00      50.00      45.14       0.31      97.85      46.83        960
     6     50.00      60.00      55.06       0.11      97.81      52.81      1,029
     7     60.00      70.00      65.06       0.05      97.72      56.80        985
     8     70.00      80.00      74.94       0.31      97.98      61.62        999
     9     80.00      90.00      85.19       0.01      97.88      67.46        995
    10     90.00     100.00      94.06       0.05      98.00      71.53        796
```

5 General Submission Policy

You may submit your code multiple times up until the due date/time. Any submission after the due date/time will be subject to the late assignment policy described in the course Syllabus. In all cases the last code you submit will be used for the grade.

6 General Requirements

All code you write for this course should meet the following general requirements:

1. All code should be your own work and not copied from another student, the internet, or any solution guide for the textbook.

2. Code from the following sources may be used or modified to complete the project:
    a. Any code provided by me in the class notes.
    b. Any code provided by the textbook itself (excluding any solution guides).
    c. Any code written by you for this or another course.

3. All Java classes should have a documentation comment including, author, date, course (i.e., CPS142-Summer 2023), and a purpose or brief description statement.

4. All methods must have properly formatted and descriptive documentation comments. The only exception is methods that override well documented interface or superclass methods. Private methods must also be commented.

5. Any method with more than just a few lines of code should have additional single-line comments describing important parts of the algorithm.

6. The use of break, continue, or exit statements inside loops is not acceptable in the implementation of any class. If found points will be subtracted from Item 5 of the Programming Rubric.

NOTE: The above requirements should be followed only where applicable.

7 Program Rubric

Your grade for this assignment will be computed using the following rubric.

Item	Description	Percent of Grade
1	<u>Compilation:</u> Compiles with no syntax errors.	10
2	<u>Execution:</u> Runs with no crashes or runtime errors.	10
3	<u>External Operation</u>: Written to meet all requirements for input and output as specified or implied by the problem.	20
4	<u>Internal Operation</u>: Written to meet all requirements for internal operation as specified as specified or implied by the problem statement.	40
5	<u>Coding Style:</u> Written and formatted in accordance with best practices presented in class (indentation, class names, variable names, method names, etc.).	5
6	<u>Project Comment:</u> Contains a **documentation comment** with the assignment name, author, date, class (i.e., CPS142-Summer 2023), and purpose/description at the top of each file.	5
7	<u>Documentation</u>: All methods, classes, and interfaces have **documentation comments**. Methods that override well-documented interface method and the main method are exempt.	5
8	<u>Comments</u>: Includes sufficient other regular comments to describe important parts of the code.	5
	Total	**100**