**Checklist & Rubric (20 points; half point for each row)**

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Program Technical Requirement** | **Self check**  **X** | **Kaufman’s check** |
| **Design** | | | |
| 1 | All method names and signatures match those in the specifications; i.e. don’t change method names, parameter types, or return values from the requirements |  |  |
| 2 | Code has proper indentation, use of well-named variables, blank spaces as appropriate to break up sections of code, and comments to explain complex design. |  |  |
| **The LinearEquationRunner client class** | | | |
| 3 | In the main() method, the program welcomes the user. |  |  |
| 4 | Program uses the Scanner class to accept two different coordinates and store them as String variables. |  |  |
| 5 | Both coordinates are accepted and saved in the format “(x, y)” |  |  |
| 6 | Successfully use string methods to extract x and y from each point |  |  |
| 7 | Successfully use Integer.parseInt to convert string values to int values |  |  |
| 8 | Program checks to see if x1 == x2 before creating a LinearEquation object. |  |  |
| 9 | If user enters two points with the same x value (i.e. a vertical line), the program informs the user a message that the line is a vertical and outputs the equation of the line in x = \_\_\_ format, e.g. "x = 4" and the program ends. |  |  |
| 10 | Program generates a LinearEquation object from the x and y values parsed from the coordinates, i.e. x1, y1, x2, y2 |  |  |
| 11 | Program uses System.out.println(obj.lineInfo()), where obj is your LinearEquation object, to print the LinearEquation object’s info. |  |  |
| 12 | After line’s information is printed, the program asks user to enter an x value |  |  |
| 13 | The program accepts an x value from the user as a double |  |  |
| 14 | The program uses the LinearEquation object's coordinateForX method to obtain the String coordinate. |  |  |
| 15 | The program prints the resulting String coordinate for the user. |  |  |
| 17 | **All** printed decimal values are rounded to the nearest *hundredth*. |  |  |
| **The LinearEquation class** | | | |
| 17 | public LinearEquation(int x1, int y1, int x2, int y2) constructor creates a LinearEquation object. |  |  |
| 18 | public double distance() correctly calculates and returns distance between (x1, y1) and (x2, y2) |  |  |
| 19 | public double yIntercept() correctly calculates and returns the y-intercept of the line between (x1, y1) and (x2, y2) |  |  |
| 20 | public double slope() correctly calculates and returns the slope of the line between (x1, y1) and (x2, y2) |  |  |
| 21 | public String equation() correctly returns a String that represents the linear equation of the line through points (x1, y1) and (x2, y2) in slope-intercept (y = mx + b) form, where m is the "printable" slope and b is the y-intercept; e.g. "y = 3x + 1.5" |  |  |
| 22 | * printed slope is a whole number when the fraction reduces to a whole number (e.g. "4" instead of "8/2") |  |  |
| 23 | * printed slope is a fraction when it can’t be reduced to a whole number (e.g. "4/3") |  |  |
| 24 | * printed negative slopes have the negative sign in the correct place (e.g. "-4/3", ***not*** "4/-3") |  |  |
| 25 | * no "double negatives" appear in the printed slop (e.g. "4/3", ***not*** "-4/-3") |  |  |
| 26 | * correctly prints slopes of +1 and -1 (e.g. "y = x + 4.0" and "y = -x + 4.0", ***not*** y = "1x + 4.0" or "y = -1x + 4.0" |  |  |
| 27 | * printed y-intercepts are rounded to two decimal places. |  |  |
| 28 | * printed y-intercepts are not shown for a y-intercept of 0. |  |  |
| 29 | * printed negative y-intercepts appear as subtraction rather than plus a negative (e.g. "y = 2x - 4.0" and "y = 1/2x - 7.25", ***not*** y = "2x + -4.0" or "y = 1/2x + -7.25" |  |  |
| 30 | public String equation() handles horizontal lines (when *m* = 0, i.e. when y1 and y2 are equal) by returning an equation in the form *y = b*; for example, the line through points (2, 5) and (7, 5) should be returned as "y = 5" rather than "y = 0x + 5" |  |  |
| 31 | public String lineInfo() correctly generates and returns a printable String outputting all required information. |  |  |
| 32 | * Includes the original points: (x1, y1) and (x2, y2) |  |  |
| 33 | * The equation of the line in y = mx + b format (generated by calling the equation() method) |  |  |
| 34 | * The slope of the line, as a decimal (using slope() method) |  |  |
| 35 | * The y-intercept of the line (using yIntercept() method) |  |  |
| 36 | * The distance between the two points (using distance() method) |  |  |
| 37 | public String coordinateForX(double yValue) correctly computes the corresponding y value for the point on the line and returns a string representing the coordinate for the point (x, y) with values rounded to the nearest hundredth, e.g. (4.5, -1.27) |  |  |
| 38 | public double roundedToHundredth(double toRound)correctly returns the toRound value rounded to the nearest *hundredth.* |  |  |
| 39 | roundedToHundredth(double toRound) are used appropriately in your other methods so as to reduce redundancy |  |  |
| **Testing and submission** | | | |
| 40 | All 16 test cases are tested (including others you think up) to ensure accuracy in your method development. |  |  |
|  | **Rubric Row Total (out of 40)** |  |  |
|  | **Total Points (out of 20; 0.5 points per row)** |  |  |