OpenAI

Code Refactoring and Optimization

Ex.1q5.) From my observation, I see that x2 more than x1 since x loop ascending in Ex.1q3.
So, I set the condition if x1 > x2 then swap variable from x1 to x2 and y1 to y2

To implemented to draw lines beautifully is to increase size of the plot to 5

Ex.2q5.) I found out that we need to make swap() function but the problem is Java variable cannot pass by reference, e.g., C language allow pass by reference variable by make it pointer.
So, I implemented to be array at first and implemented using swap with array

Next, we need to refactor DDALine to be readable by making swap() function,
findDifferential function (dx, dy), calculateSlope function (m) and drawDDA
to make it readable.

From refactor, swapValueArr(), findDifferential(), calculateSlope() are reusable, so I will use these functions/methods in
another drawing algorithm line.

Ex.3q5.) The bresenhamLine method should be drawn as the pseudo code above because of the following:
- Pseudo code of first is the original
- Pseudo code of second is edited from first.
- The difference between both are swap dy and dx
- The keypoint that I see is sx and sy. Normally we increase by one of each loop, but
in second pseudocode check that if each axis is less/more than other one, e.g., x1 < x2 ? or y1 < y2?
From those condition if x1|y1 < x2|y2 is true then each loop increase by 1, otherwise decrease by 1 from each axis
- From the keypoint, the program is more flexible and support with all coordinates.

fix my grammar pls

Here's your revised text with grammar and clarity improved while keeping your original structure and intent:

Ex.1q5.) From my observation, I see that `x2` is greater than `x1`, since the `x` loop increases in **Ex.1q3**.
So, I set a condition: if `x1 > x2`, then swap the variables `x1` with `x2` and `y1` with `y2`.

To make the lines draw more smoothly, I increased the plot size to 5.

Ex.2q5.) I found that we need to create a `swap()` function, but the problem is that Java does **not** support passing variables by reference (unlike C, which can do this using pointers).
So, I implemented it using an array instead and used a swap function with the array.

Next, we need to **refactor** the `DDALine` method to make it more readable by separating it into smaller functions:

- `swapValueArr()`
- `findDifferential(dx, dy)`
- `calculateSlope(m)`
- `drawDDA()`

These changes improve readability.

Functions like `swapValueArr()`, `findDifferential()`, and `calculateSlope()` are reusable, so I will use them in other line drawing algorithms as well.

Ex.3q5.) The `bresenhamLine` method should follow the pseudocode above for the following reasons:

- The first pseudocode is the original.
- The second pseudocode is an edited version of the first.
- The main difference between the two is that `dx` and `dy` are swapped.
- The key point is the use of `sx` and `sy`. Normally, in each loop, we increase by 1.
  But in the second pseudocode, it checks whether one axis is less than or greater than the other (`x1 < x2`? or `y1 < y2`?).
  Based on that condition, if `x1` or `y1 < x2` or `y2`, the loop increments by 1; otherwise, it decrements by 1 on that axis.

This makes the program more flexible and compatible with all types of coordinates.

Let me know if you'd like this polished further for a report or presentation!