Lab 6 Comments

- wrap long lines – put in line breaks before 80 characters
- include your name and a description of the program in comments at the beginning of every program
- "better names"
  - conventions
    - `x`, `y` are common for pixel coordinates
    - `r`, `c` or `row`, `col` are common for rows and columns
    - `i`, `j` are common for array indexes or similar
    - for a counting loop, generic one letter names like `i`, `x`, or `c`, are OK only if there aren't indexes or pixel coordinates or columns anywhere nearby
      - better to use something like `count` if it is truly just a counter
  - it should be easy to identify what a variable is for
    - e.g. `x`, `y` are good names for coordinates but don't use just `x`, `y` if there are several things with positions in the program (even if you don't have variables for them)
      - identify what coordinates these are with names like `carX`, `carY`, etc.

Lab 6 Comments

- "separate variables for separate things"
  - unless there is an inherent relationship between two things, it is simpler (and more flexible) to have separate variables instead of a formula relating one to the other

Lab 6 Comments

- 1 – drawing
  - be sure to include all of the required elements
    - 5 different shape commands
    - 2 different stroke colors
    - 2 different line widths

Lab 6 Comments

- pattern
  - don't forget the window title
  - be careful with the sizes and positions
    - e.g. not leaving a margin on the top and left sides
  - "fixed window size" or "fixed grid size"
    - the problem said that the pattern should fill the window no matter the window size
    - use the `w` and `h` parameters to `draw` instead of hardcoding 800 and 500
    - use `x`, `y` loop variables with the conditions `x < w` and `y < h` instead of hardcoding 5 rows, 8 columns
  - only need a single set of `x`, `y` loop variables for the position of each four-shape pattern
    - the two rects and two ovals are in fixed positions relative to each other so `x`, `y` can refer to the upper left corner of the blue rect and the other values computed as `x+20` etc

Lab 6 Comments

- bar chart
  - be careful generating random numbers
    - want values 0-20, not 1-20 or 0-19
  - should be only 8 bars, not 10
  - initialize the array first, then draw the bars
    - this means separate loops, one after the other, instead of a single loop whose body initializes one array slot and immediately draws that bar
  - don't change the array values to scale the bar height to fill the window – compute the bar height from the array value
    - e.g. `g.fillRect(x,400-numbers[i]*20,50,numbers[i]*20);`
  - be careful with positioning and scaling
    - e.g.

      ```
      g.fillRect(x,numbers[i]*20,50,500-numbers[i]*20);
      ```

      makes the white space above the bar proportional to the values in the array (remember that (x,y) is the upper left corner), and also doesn't fill the full height of the window when `numbers[i]` is 20

Lab 6 Comments

- bar chart
  - draw one rect for each value in the array (8 values)
    - for the histogram you had to print the desired number of hashes one # at a time
    - for the bar chart you can instead draw a single rect that is the full size you want – you don't need one rect for each #
  - it is simpler to have separate loop variables for the current array index and the x position of the current bar rather than a formula for computing x given the index `i` (but the formula is OK)

    ```
    for ( int i = 0, x = 10 ; i < numbers.length ;
          i++, x = x+60 ) { … }
    ```