SE2852 Lab 2 Feedback

Dr. Yoder, Spring 2014, MSOE

The circled numbers correspond to circled numbers in the graded assignment.

UML Diagram

-0.5 Missing connections

-0.5 public/package instance variables that are not final. A "package" variable is an instance variable not declared public, private, or protected. It is available to all classes in the package, and thus is much more permissive than a private variable. On the other hand, protected variables are also available to every class in the package, as well as all classes that extend a class. Thus "package" is more protective than "protected!" To ensure your readers that you have not made a mistake, if you intend to use a package variable, put a /* package */ comment before the method, like this:

```
/* package */ int mySharedMethod(int x)
```

Don't use package instance variables unless they are immutable (final).

-0.5 more such variables

-0.5 Capitalize inner-class names

Code

① -1 Prefer local variables to instance variables. If an instance variable is only used in one or two methods, consider passing it between the methods using arguments instead of creating a new instance variable. This makes it easier for others to read your program because it is easier to trace where the variables are used.

② -1 Use interface references. Using references to interfaces rather than concrete classes can make it easier to switch from one implementation to anther – e.g. from ArrayList to Array.

③ -2 Be sure to close the file. Once you are done processing, close the file to unlock it (in Windows) and free up the resources. By putting the closing of the file in a Finally block, you can ensure the file is closed even if an error occurs while parsing it.

④ -1 Extract common functionality. Very similar code exists in two places. If you find yourself copy-pasting chunks of code, consider making a method that performs the shared functionality. This will often make the maintenance of your code easier in the future.

⑤ -1 Simplify and/or generify by setting plot boundaries. WinPlotter offers a setPlotBoundaries method that allows you to set the min/max. You can use plotter.setPlotBoundaries(0,0,1,1) and avoid the need to scale points yourself.

⑥ -1 Connect last point back to first. When drawing the curve, connect the last point to the first to complete the circuit.

⑦ +1 Use of “previous” and “next” references

⑧ +1 Dynamic updating of crit values to avoid need to recalculate all

⑨ -0 Can combine conditions. e.g. if(a) { g(); } else if(b) { g();} can be combined to if(a||b) {g();}

⑩ +1 Making x & y public and read-only for an immutable dot.

⑪ -1 Avoid static instance variables. Instead, pass the needed arguments from method to method. Digging deeper: Why do the instructions specify that the method should be static?

⑫ -0 Name methods using verbs (e.g. drawLines) rather than nouns (e.g. lineDrawer).

⑬ -4 Separate functionality. The lab specifies “A good design will result in logical separation of functionality (UI in one class), managing the "shape" in another class that keeps a list of points, where the points are likely a custom class that keeps track of the points coordinates and is able to calculate its critical value when it knows its neighbors.”

⑭-2 The lab specified that the code should contain a method with this signature: public static void getDesiredDots(List<Dot> original, List<Dot> result, int numDesired);

⑮ -1 The lab specified getDesiredDots is static. (See also ⑭,⑪)

⑯ -0 Break long actionPerformed methods out into their own methods… this causes this code to show up in the UML diagram and improves self-documentation. You can use the anonymous inner class to call the method.

⑰ -1 Code goes off page!