

- Solo programs must meet all the programming style requirements listed here. **No late work accepted or graded**
- A. At the top of your console output, display author, title and a brief introduction; but, no offensive remarks.
  - B. The main method must clearly show all major activities- the overall program structure, not header/body/trailer.
  - C. You must avoid excessive **chaining**.  
**Decompose problems into sub-methods** that can be **easily named with verbs or noun-verb phrases**.
  - D. You must have **at least 3 non-trivial methods other than main** in your program – see the next specification.
  - E. Code must be **easy to read**. You must follow the text book's indentation, { }. Limit lines to 60 characters.
  - F. **Factor or eliminate redundant code**: put any set of two or more lines duplicated in multiple places into a method. Review your code to find duplication that could be eliminated if you re-structured or reused methods. You must define other methods as needed for structure or to eliminate redundancy.
  - G. Follow the text book case standards on ClassNames, methodNames, variableNames, CONSTANT\_NAMES.
  - H. Code that does actual input, processing/calculation, or actual output must be in separate methods.
  - I. You must **not** do actual input or output directly in main. Main must call methods to do those tasks. You must use separate method(s) to do console output and separate method(s) to do any graphical output.
  - J. The code that asks the user for input must not be in the same method as code to read data from a file.
  - K. **You must include the required javadoc style Class comments with the description, author, etc.** See the Template.java file. You must remove the parenthetical explanations I provided such as “(required date)”.
  - L. You must have a **blank line & javadoc style method comments** before each method header.
  - M. **You must not use literals as “magic numbers”** (e.g. 3.141 or “enter”) in methods. Instead, use **CONSTANTS** that clarify your code. The use of the literals -1, 0, 1 and 2 are ok if they are simply loop increments.
  - N. You must use **Good names** and designs to reduce the need for explanatory comments. You must use easy to understand, meaningful identifiers for method and variable names.
  - O. You must structure your solution using static methods that accept parameters and return values where needed.
  - P. By the **first 5 minutes** of class, you must submit an **easy to read printed listing of your code with line numbers**. **Screen captures** showing your program test runs with 3 different, specified inputs must be attached.
  - Q. All file names must include the assignment ID and all your initials, for example, Sp1EBH\_Song.java and Sp1EBH\_SongTest.java. You must not use spaces in filenames.
  - R. You must **remove redundant comments** that say the same thing as the code, for example: `x+=1 //increment x`. When the code might be confusing, first try to change it so that it is more obvious. If you cannot simplify the code, your comments should explain why you did it this way, for example, `// I ran out of time and this works`.
  - S. Limit variable **scope**. Do NOT use global variables, unless you get my prior approval.
  - T. You must put DrawingPanel.java and any data files in the same directory as your program.
  - U. You must not use java features prohibited or not yet covered in this course when the program is assigned.
  - V. Solo programs are designed for you to demonstrate what you have mastered. You may not use code or request aid from anyone else or communicate with anyone else about solo programs except the instructor.
  - W. Ask all questions about solo projects in class so that everyone benefits from the answers. I am often out of town and rarely reply to homework related questions between our design discussions and the due date
  - X. You may use any mix of standard design tools: pseudo-code, hierarchy diagrams, flow charts, DFDs, etc. You **must use them correctly**. You may change your design after submitting the required drafts. (For a set of guidelines, see [ironbark.bendigo.latrobe.edu.au/subjects/PE/2005s1/other\\_resources/pseudocode\\_guide.html](http://ironbark.bendigo.latrobe.edu.au/subjects/PE/2005s1/other_resources/pseudocode_guide.html).)
  - Y. Team .java file names must include both members' full initials. For example Te2EBH\_HRJ\_Victory.java.

**Grading:** Your program scores will be based on a 5 point scale comprising 2 major components:

1: **Output** correctness and 2: **Internal** correctness.

Your **Output** must match expectations as stated in the assignment and in class. When I try it in the unzipped folder I receive from you, if your program does not compile or does not run to completion or does not produce any output, it will receive no output correctness points. Missing any numbered specifications above costs -1, missing more costs -2.

Your **Internal** correctness score will be based on how well you follow all the specifications listed in the assignment, including due date, design, style, comment and other requirements referenced. Initially, missing < 2 costs -1, more costs -2. Point deductions for each internal or style error category can **increase each week**.

The SCC email server rejects message attachments that contains executable code (.class) even zipped.