





## Code Quality Guidelines

In addition to producing the desired behavior, your code should be well-written and meet all expectations described in the [grading guidelines](#) and the [Code Quality Guide](#). For this assessment, pay particular attention to the following elements:

### Capturing Structure

As in assessment 1, you should use static methods to accurately capture the structure of the output in your program. In particular, your `main` method should be a concise summary of the program and reflect the structure of the figure being produced. Your program should not include any `System.out.println` or `System.out.print` statements or any `for` loops in `main`.

### Reducing Redundancy

You should continue to reduce redundancy as much as possible in your program by making good use of static methods. Specifically, you should not have any redundant code to produce the same full line of output. Instead, use methods to remove this redundancy. Similar to assessment 1, this only applies to full-line redundancy. You are not expected to deal with partial-line redundancy, such as the two groups of colons in this line of output:

```
__/::~::~:|:~::~:\__
```

### Code Aesthetics

Your code should be properly indented, make good use of blank lines and other whitespace, and include no lines longer than 100 characters. Your class, methods, variables, and constant should all have meaningful and descriptive names and follow the standard Java naming conventions. (e.g. `ClassName`, `methodOrVariableName`, `CONSTANT_NAME`) See the [Code Quality Guide](#) for more information.

### Commenting

Your code should include a header comment at the start of your program, following the same format described in assessment 1. Your code should also include a comment at the beginning of each method that describes that methods behavior. Comments should be written in your own words (i.e. not copied and pasted from this spec) and should not include implementation details (such as describing loops or expressions included in the code). See the [Code Quality Guide](#) for examples and more information.

## Getting Help

If you find you are struggling with this assessment, make use of all the course resources that are available to you, such as:

- Reviewing relevant examples from [lessons](#), [section](#), and [lab](#)
- Reading the textbook
- Visiting [support hours](#)
- Posting a question on the [message board](#)

## Collaboration Policy

Remember that, while you are encouraged to use all resources at your disposal, including your classmates, **all work you submit must be entirely your own**. In particular, you should **NEVER** look at a solution to this assessment from another source (a classmate, a former student, an online repository, etc.). Please review the [full policy](#) in the syllabus for more details and ask the course staff if you are unclear on whether or not a resource is OK to use.



Try to identify the large chunks of the output and write a method for each, as in the tapestry example from lecture.



This will result in *some* redundant code. This is OK, but **only** if that code is producing partial-line redundancy as described here. All other redundant code should be fixed.

## Reflection

In addition to your code, you must submit answers to short reflection questions. These questions will help you think about what you learned, what you struggled with, and how you can improve next time. The questions are given in the file `SpaceNeedleReflection.txt` in the Ed assessment; type your responses directly into that file.

## Running and Submitting

You can run your program by clicking the "Run" button in Ed. This will compile and execute your code and show you any errors, or the output of your program if it runs correctly. If you believe your output is correct, you can submit your work by clicking the "Mark" button in the Ed assessment. You will see the results of some automated tests along with tentative grades. **This grade is not final until you have received feedback from your TA.**

You may submit your work as often as you like until the deadline; we will always grade your most recent submission. Note the due date and time carefully—**work submitted after the due time will not be accepted.**