GCIEL Group: Ian, Kelton, Ethan, Josh, and Haobo
Coding Standards and Guidelines Document
CSC-324
Professor Jimenez
4/19/2024

Coding Standards and Guidelines

Functions, Methods, and Size:

- Functions and methods should be small and perform a singular function.
    - This allows repeated calls of functions, avoiding repeated code.
- If functions are too large, refactor them such that they're smaller.
- Smaller functions/methods require fewer tests as they are more straightforward.

Formatting, layout, and style:

- Use consistent indentation to define code block. An indentation is created by two spaces per indentation level instead of tab key.
- Include space around all operators (=, +, -, <-, etc.). Ex. x <- 2 * y
- Maintain the length of line below 100 characters and break lines to improve readability.

White Space:

- Empty lines to separate code blocks (i.e., group functions/methods together that serve the same intention)
- Make sure all statements within a block are equally aligned/indented and avoid double indentations.
- Avoid using irrelevant spaces around operators and functions/methods.

Block and Statement style guidelines:

- One statement per line, and format statements blocks identically (i.e., if opening bracket is on the same line as the function call [if {], then maintain that throughout
- Utilize more parentheses than necessary around functions.
- With conditionals, put each condition on its own line.

Declaration Style Guidelines

- Use only one declaration per line. However, you can group related variables together.
- Declare variables close to where they are used. (declaration before use rule).
- Order declarations sensibly: Group your declarations by types and usage.
- Use white space to separate your declarations.
- No nested headers and no source code in headers.

Commenting Style Guidelines:

- All comments should be identified with # followed by a space.
- Comments for a new function should preceded with two blank lines.

- For functions, use comments to provide the description of the function's purpose, parameters, and return value.
- Use TODO comments to indicate incomplete areas of your code that require further work or attention.

Identifier naming conventions:
- Useful and comprehensible variable and function names
- Camel case variables and functions when applicable ex. shipDist
- Use verb-noun pair for function names ex. getDistance()

Defensive programming: Questions:
- Need to protect from bad data.
- Need to Validate!
- On data file upload:
    - Check file operations: Did the file open? Did the read operation return anything? Did the write operation write anything? Did we reach EOF yet?
- Always initialize variables and don't depend on the system to do the initialization for you.

Error Handling:
- To avoid error on publication, keep consistent error handing practices with well-rounded tests.
- Recover:
    - Recovery means that your program needs to try to ignore the bad data, fix it, or substitute something else that's valid for the bad data.

References:

1. JEF Works. (n.d.). R Style Guide. Retrieved from https://jef.works/R-style-guide/ Accessed 04/19/2024.

2. Dooley, J. F. (2017). Software Development, Design and Coding With Patterns, Debugging, Unit Testing, and Refactoring (2nd ed. 2017.). Apress. https://doi.org/10.1007/978-1-4842-3153