

CPT-206-Adv Event-Driven Program (C#)

Code Reviews (Peer Evaluation)

One of the goals of this course is to encourage good software development practices, especially when building a large or complex software system. For each project submission, you will have to choose four other random students in the course. You must review their code and offer constructive feedback. In industry, this process is known as a *code review* and is frequently used to improve software quality and to catch software defects early.

It is important to note that the goal of a code review is to improve the quality of the **code**, not to evaluate the quality of the **developer**. Accordingly, try to use "I-messages" (e.g., "I don't understand it" rather than "it's confusing"), and ask questions and make suggestions instead of making accusations. Keep all of your comments respectful and collegial.

The following table gives examples of both *non-constructive* and *constructive* feedback (aim for the latter!) for several questions. On the next page there are checklists with more specific questions to answer during reviews.

Question	Example of Non-constructive Feedback	Example of Constructive Feedback
<i>Is the code generally readable and well-documented?</i>	"Not enough documentation." "I liked the author's style."	"I didn't understand the goal of function X; I think the author should add some documentation regarding its inputs and outputs." "I liked the use of extra indentation to line up the array initializations in module X."
<i>Did you find any software defects?</i>	"Function X doesn't work." "The program crashed on one of my test cases."	"Function X does not produce the correct output for this input: 'continue' -- I believe the regex is missing this case." "The program crashed when I tried to run it on a program with a loop. I think this is because one of the labels is being emitted in the wrong place."
<i>What was one way that their solution differed from yours?</i>	"My code is faster." "My function X is shorter than this author's version. Mine is better."	"I chose to calculate the maximum value on every iteration of the outer loop while this author calculated it only once and cached the result. Their approach is more efficient, but my approach works even if the list is modified during iteration."

<i>Do you have any other constructive comments for the author?</i>	"This code sucks and needs to be rewritten." "This code is perfect."	"Function X has redundant ifconditions; the last two could be consolidated." "The use of recursion in function Y to avoid ugly class-level data structures is very elegant."
--	---	---

Checklists

As you work on the project, keep this review process in mind--your code will be reviewed by at least four people in addition to the instructor! Accordingly, here is a checklist you should use before you submit:

SUBMITTER:

- ☐ The code compiles without warnings and has been tested with unit and integration tests
 - ☐ The code is clean and follows a consistent coding style
 - ☐ All corner cases, workarounds, and non-obvious code have been documented ☐
- All dead code has been removed – dead code is code that doesn't do anything or code that has been commented out for testing purposes.

Here is a checklist of questions to answer as you complete your peer review:

REVIEWER:

- ☐ Does the code compile? Are there compiler warnings?
 - ☐ Is the code is generally readable and well-documented?
 - ☐ If not, what are some specific improvements that could be made?
 - ☐ Has repetitive code been factored out? Is there any dead code?
 - ☐ Did you find any software defects?
 - ☐ If so, what test case(s) exposed it? How would you attempt to fix the defect(s)?
 - ☐ Are all exceptions and corner cases handled appropriately, as far as you can tell?
 - ☐ Has the provided framework and existing code been used appropriately?
 - ☐ What was one way that their solution differed from yours?
 - ☐ Are there any explicit tradeoffs associated with the difference(s)?
 - ☐ Now that you have seen both approaches, which approach do you prefer? ☐
- Do you have any other constructive feedback for the author?

Code reviews should be **at least** two or three paragraphs of 3-4 sentences each, but in some cases could be much longer. Include as much detail as possible, and be as explicit as possible in your suggestions for improvement.

NOTE: To correctly provide a code review you will need a clone of their solution. Clone the solution, open it on your local computer, run the solution for accuracy, and review the code. After you have completed your code review file, feel free to delete the cloned solution to save space on your local machine.

Submit your code review on GitHub by uploading your file to the correct GitHub Repository and starting a merge pull request so the original author of the repository will be able to accept your file. **DO NOT COMMIT**

DIRECTLY TO ANYONE’S repository without making a pull request! That is the equivalent of kicking in someone’s door to look in their fridge without knocking. Don’t do it.

Your review will be compared with another person's review and my own assessment, and will be graded on the following scale:

Score	Description
4	Thorough and constructive criticism in all areas. <i>(often requires more than 2-3 paragraphs)</i>
3	Constructive feedback for some areas, but not thorough enough.
2	OK, but lacking (usually because of a focus on cosmetic style issues over deeper issues).
1	Mostly non-constructive or inconsistent written feedback; very superficial.
0	Nothing.