CSC 276 Rubric Contract Grading: Code Artifacts

The following rubric is used to assess the level of student learning in CSC 276 as it relates to creating code artifacts. See the class syllabus for details on how this rubric is used. For purposes of computing an assignment grade, any criteria deemed unacceptable shall have a numeric value of (50%). All criteria are weighted equally when averaging for an assignment grade.

Criteria	**Acceptable (C=75%)**	**Better (B=85%)**	**Best (A=100%)**
Separation of Concerns	• One method should be split into two to improve separation of concerns. • Solution makes use of one publicly scoped variable.	• One method should be split into two to improve separation of concerns. • No publicly scoped variables are used.	• Each method performs a distinct part of the processing and cannot be split into smaller methods. • Same.
Design for Reuse	• Two methods could be eliminated by generalizing other methods. • Missed two or more opportunities to create a more general method.	• One method could be eliminated by generalizing another method. • Missed an opportunity to create a more general method.	• No method could be eliminated by generalizing another method. • No (reasonable) opportunity to create a more general method.
Design Only What is Needed	• One "minor" assignment requirement has not been implemented. • Solution contains two "extra" processing steps that are relatively small and not required.	• All assignment requirements have been implemented. • Solution contains one "extra" processing step that is relatively small and not required.	• Same. • Solution contains zero "extra" processing steps that are relatively small and not required.
Class containing main function	• The main function does too much detailed processing.	• The main function only does high-level processing but has too many statements.	• The main function contains only a few statements.
Number of Logic Defects Found	• Two.	• One.	• Zero.
Number of Syntax Defects Found	• Not applicable.	• Zero.	• Not applicable.
Method Comments	• Each method has a comment that describes the purpose of the method.	• Same. • Describes each input parameter. • Describes each output parameter or return value.	• Same. • Describes the pre-conditions. • Describes the post-conditions.
Readability	• Three or four identifier names do not accurately describe the purpose of the variable/method. • A method contains too much code; it should be split into smaller methods.	• One or two identifier names do not accurately describe the purpose of the variable/method. • All methods contain an appropriate amount of code.	• All identifier names accurately describe the purpose of the variable/method. • Same.
Consistency with Design Artifacts	• Implementation is consistent with all but two design artifacts. Each inconsistency is considered "minor".	• Implementation is consistent with all but one design artifact. The inconsistency is considered "minor".	• Implementation is consistent with all design artifacts.
Language Features	• Inappropriate use of one library or feature.	• Uses appropriate libraries and features.	• Not applicable.