analysis and design process		1 2	3 4	5 6	7 8	9 10
	Specification	1) Student provides verbatim copy of requirements as spelled out in the assignment. 2) Student adds information about limits on the input.	3) Student writes clear specifications which include input and output requirements (format for input and output) 4) Specifications include general information about persistent data to be maintained /used.	5) Specifications include a section which provides details about the persistent data to be used. 6) Specifications include details which are the result of the student's problem analysis and potential interactions with the 'client'.	7) Specifications include several important use cases. 8) Specifications include evidence of validation, which shows that they are meeting the client's intent (validation: build the right thing)	User requirements – including functional and non-functional requirements - are clearly spelled out. Specifications are meaningful and complete.
		1 2	3 4	5 6	7 8	9 10
	Design	1) Student correctly identifies responsibilities (methods) of given class/component. 2) Student correctly identifies and informally describes most important class / component and its responsibilities (methods).	3) Student correctly identifies and describes set of classes/components - including their responsibilities (methods) – which is suitable. 4) Student correctly identifies major use cases and provides a detailed description of the each use case.	5) Classes, attributes, methods and interfaces are identified for each use case. 6) Extensive description of the flow of each use case	7) Data model related to each use case is identified. 8) Documentation of design includes explanation of other alternatives and/or trade-offs and why the current design was chosen.	Problem broken into reasonable elements of appropriate size, scope, and independence. Elements are reusable and have general application. Design choices were explained and trade-offs with other potential choices justified. Architecture, interface, and database (if appropriate) designs are described.
		1 2	3 4	5 6	7 8	9 10
	User Interface	1) User understand what to enter (but not why) and result is shown. 2) User understand what to enter and why (what program does) and result is shown.	3) Screen-based communications with the user are clear and properly formatted. 4) Output is presented in a format that is easy to understand.	5) Screen-based communications with the user are consistent to avoid user errors. 6) Program checks for simple entry errors and informs user about the error.	7) Program checks for entry errors and handles them in a reasonable way. 8) GUI is used for communication with the user.	Communications with the user are clear, intuitive, and at the appropriate level. The program facilitates user input with consistent prompting/information. Output is presented in a format that is easy to understand

		1 2	3 4	5 6	7 8	9 10
	Read-, and Maintainability	1) Program is properly formatted for readability (indentations, empty lines, length of lines.) 2) Identifiers are consistent with convention.	3) Program code includes more than one method. 4) Program uses local variables where appropriate.	5) Methods have only one responsibility to support re-use. 6) Program uses calls to same method to accomplish same steps.	7) Solutions to complex responsibilities are implemented using several methods. 8) Program is easy to read due to the incorporation of information hiding.	Code is exceptionally easy to read, understand, and maintain.
		1 2	3 4	5 6	7 8	9 10
	Internal documentation	1) Names of identifiers are meaningful. 2) Comments explain purpose of important variables.	3) Inline comments explain purposes of groups of statements 4) Comments for method headers clearly describe purposes of methods	5) Comments for method headers describe purpose of parameters and returned results 6) Comments describe purpose of each file/classes in each file.	7) Details of code / algorithm are explained where needed. 8) Important loops include (formal) pre-/post conditions.	Comments follow prescribed conventions. Method headers include pre/post conditions and clear statement of purpose. All variables and parameters have clearly defined purpose.
		1 2	3 4	5 6	7 8	9 10
Quality of implementation	**Evaluation and Test Cases**	1) Student shows that program compiles/can be executed 2) Program follows general specification of input/output.	3) Student shows that program generates correct output for several general cases. 4) Student explains, identifies, and tests (successfully) important problem cases (general and some boundary)	5) Student explains, identifies, and tests (successfully) most problem cases (general and nearly all boundary) 6) Program follows general specification for maintaining (persistent) data	7) Student provides context for test cases in overall evaluation of product. 8) Program follows nearly all specifications for input /output and (persistent) data	Test cases cover (all/most) important categories based on the problem specification. Evaluation includes information about strength/weaknesses of software system developed.

		1 2	3 4	5 6	7 8	9 10
	Efficiency & Analysis of Algorithm	1) Algorithm employed is straight forward or not selected based on efficiency. 2) Algorithm gets (most of) the job done.	3) Student states the runtime complexity of algorithm or methods as Big-O of variable n which is not input size. 4) Some of the methods/algorithms are efficient	5) Student provides and explains reasonable runtime analysis of algorithms in terms of input-size 6) Most of the methods are efficient	7) Student uses (very) efficient algorithm(s)/methods to solve the problem. 8) Student discusses possible algorithm choices, advantages and disadvantages, and tradeoffs and why s/he selected the used algorithm.	Analysis of run-time complexity & space requirement, as well as correctness of used algorithm at appropriate level. Efficient algorithms were used to solve the problem.