Name	Assignment 2 Rubric
Description	Assignments will be marked in each category by being given the level of achievement that corresponds to the description that most accurately describes the submitted assignment. If the marker believes that the submitted assignment fits somewhere between these levels of achievement, a score may be applied that is interpolated between them, at the marker's discretion. If there are problems with the assignment in a given category that are not quantified by any of the existing levels of achievement, the marker will give a different score and include a comment. The "Additional Functionality" criteria has special scoring rules for groups of 1. If you submit alone and do not implement additional functionality and have a score of 30 for the rest of the assignment, you will receive a score of 5 in the Additional Functionality section. If you have a score of 15, you will receive a score of 2.5, and if your score is 0, your score in Additional Functionality will be 0 as well.

Rubric Detail

	Levels of Achievement		
Criteria	**Poor (zero marks)**	**Fair (half marks)**	**Good (full marks)**
Source control	**0 Points** Project was submitted without the correct .git directory OR Git history does not demonstrate that Git was used properly (e.g. very few commits)	**0.625 Points** Project was submitted with Git repository and commit history shows that Git was used throughout the project, but not using best practices for source control (e.g. poor commit messages, not enough commits, commits aren't for distinctive feature additions / bug fixes etc.)	**1.25 Points** Project was submitted with Git repository and commit history shows that Git was used to a reasonable standard throughout the project.
Collaboration	**0 Points** Group is 2-4 people and project was submitted without the correct .git directory OR The submitted report does not describe what each member of the group did	**0.75 Points** Group is 2-4 people and the majority of people in the group appear in the commit history, but contributions are not terribly fairly distributed - some members have contributed far more than other members and/or the contributions as described in the	**1.5 Points** Group is 1 person OR Group is 2-4 people and everyone contributed towards the project with a fair distribution of work that approximately matches what the report describes. It is okay if one group member

	Levels of Achievement		
Criteria	**Poor (zero marks)**	**Fair (half marks)**	**Good (full marks)**
	in sufficient detail (for a 2-4 person group) OR Half (rounding up) of the group members did not contribute commits to the project under their own usernames.	report are not supported by the commit history as found in the Git repository.	does not appear in the commit history because that person wrote the report, as long as the report makes up a fair split with the other group members.
Report - Agile Methods	**0 Points** The report does not describe what Agile methods are used, or the methods described are not Agile.	**0.5 Points** The report describes the Agile methods used in the development of this project, but the Git repository was either not submitted or contains evidence in the commit history that contradicts the report in this area.	**1 Points** The report details the Agile methods used in the development of the project, the Git repository was submitted and the commit history does not contradict the report.
Report - Architecture	**0 Points** The software architecture is not documented in the report, or is highly deficient.	**0.625 Points** The report documents all or most of the classes, but detail is missing in terms of interactions between classes, or the language used in the report is not appropriate for this type of documentation.	**1.25 Points** The report documents all of the main classes (things like individual exception classes and the like are not important, just the classes responsible for significant functionality) and how they interact. The language used in the report is precise, detail-oriented and uses appropriate technical report nomenclature.
Report - OOP	**0 Points** The report only covers a	**0.5 Points** The report covers at least 3/4 of	**1 Points** The report covers all of

	Levels of Achievement		
Criteria	**Poor (zero marks)**	**Fair (half marks)**	**Good (full marks)**
	minority of OOP principles, or the principles discussed are not actually present in the software submitted.	abstraction, encapsulation, inheritance and polymorphism, and these principles are appropriately used in the software submitted.	abstraction, encapsulation, inheritance and polymorphism, and these principles are appropriately used in the software submitted.
Report - Manual	**0 Points** The report does not document how to use the software, or important details are missing.	**0.25 Points** The report documents how to use the software, but certain details are missing and are not immediately obvious/intuitive to users of the software.	**0.5 Points** The report documents how to use the software in a user-friendly way, making use of comprehension aids such as screenshots as appropriate, and the program's functionality is comprehensively documented.
Javadoc	**0 Points** Javadoc-style comments are not present in the source code, or there are entire classes or a substantial number of public methods missing from the documentation.	**0.5 Points** The code contains Javadoc-style comments, but the HTML documentation was not generated and submitted OR there are some classes/methods that should have been documented and weren't OR the Javadoc is not sufficiently comprehensive as documentation to allow the documented classes to be used as an API.	**1 Points** The code contains Javadoc-style comments, the HTML documentation was generated and submitted with the assignment and every public class and method is comprehensively documented and describes the class API such that the documentation could be used by other developers within to use these classes in their own program.
GUI function	**0 Points**	**1.125 Points**	**2.25 Points**

	Levels of Achievement		
Criteria	**Poor (zero marks)**	**Fair (half marks)**	**Good (full marks)**
	The submitted program does not use an AWT, Swing or JavaFX GUI, OR the program does use one of these toolkits but there are such serious problems with the GUI that it interferes with the application's functionality.	The submitted program uses an AWT, Swing or JavaFX GUI to provide a modern graphical interface. The interface does not stop the user from completing tasks with the application. (It's okay if there are parts of the spec that weren't completed, as long as it's not the GUI that's stopping the user from using them).	The submitted program uses an AWT, Swing or JavaFX GUI to provide a modern graphical interface. The interface does not stop the user from completing tasks with the application, and is able to gracefully handle exceptional situations, presenting appropriate error messages to the user instead of bugging out, printing stack dumps, silently swallowing events or behaving in unexpected ways.
GUI design	**0 Points** The submitted program does not use an AWT, Swing or JavaFX GUI, OR the GUI is so poorly designed it impacts usability.	**1.25 Points** The submitted program uses an AWT, Swing or JavaFX GUI that is sufficiently well designed to not interfere with use of the program, even if it could be better designed or more intuitive.	**2.5 Points** The submitted program features a polished, high quality AWT, Swing or JavaFX GUI that meets client objectives and allows the user to load and manipulate VEC images with a level of ease and grace comparable to modern image manipulation software packages. Just like MS Paint doesn't require comprehensive documentation just to use it, the submitted program is similarly intuitive and can be picked up and used by anyone, even if they don't know the whole story about VEC files etc.

	Levels of Achievement		
Criteria	**Poor (zero marks)**	**Fair (half marks)**	**Good (full marks)**
Functionality - Loading VEC images	**0 Points** The submitted program can't load proper VEC images, or the outcome of doing so does not resemble what the images are actually supposed to look like, as described in the assignment specification.	**1.375 Points** The submitted program can load and display properly formatted VEC files, which look mostly as they are supposed to, but aren't perfect OR The program is able to correctly load VEC images but does not present this option with an appropriate file open dialog allowing the user to navigate their file system and find the VEC file they wish to load, or the file open dialog does not filter so that only .VEC files are displayed by default.	**2.75 Points** The submitted program can load and display properly formatted VEC files correctly, and this is handled with an appropriate file open dialog allowing the user to navigate their file system and find the VEC file they wish to load (with the file open dialog filtering so that only .VEC files are displayed by default)
Functionality - PLOT	**0 Points** The submitted program does not provide a PLOT tool, or the PLOT tool does not allow the user to draw dots onto the image.	**0.125 Points** The submitted program provides a PLOT tool that allows dots to be drawn with clicks of the mouse (1 click = 1 dot), but there are areas where the implementation could have been handled better.	**0.25 Points** The submitted program provides a PLOT tool that allows dots to be drawn with clicks of the mouse (1 click = 1 dot), and the implementation is perfect, with no issues.
Functionality - LINE	**0 Points** The submitted program does not provide a line drawing tool or the line drawing tool does not allow lines to be drawn onto the image with a mouse.	**0.25 Points** The submitted program provides a line drawing tool that allows lines to be drawn onto the image with a mouse, but there are areas where the implementation could have been handled better.	**0.5 Points** The submitted program provides a line drawing tool that allows lines to be drawn onto the image with a mouse, and the implementation is perfect, with no issues.

	Levels of Achievement		
Criteria	**Poor (zero marks)**	**Fair (half marks)**	**Good (full marks)**
Functionality - RECTANGLE	**0 Points** The submitted program does not provide a rectangle drawing tool or the rectangle drawing tool does not allow rectangles to be drawn onto the image with a mouse.	**0.125 Points** The submitted program provides a rectangle drawing tool that allows rectangles to be drawn onto the image with a mouse, but there are areas where the implementation could have been handled better.	**0.25 Points** The submitted program provides a rectangle drawing tool that allows rectangles to be drawn onto the image with a mouse, and the implementation is perfect, with no issues.
Functionality - ELLIPSE	**0 Points** The submitted program does not provide an ellipse drawing tool or the ellipse drawing tool does not allow ellipses to be drawn onto the image with a mouse.	**0.125 Points** The submitted program provides an ellipse drawing tool that allows ellipses to be drawn onto the image with a mouse, but there are areas where the implementation could have been handled better.	**0.25 Points** The submitted program provides an ellipse drawing tool that allows ellipses to be drawn onto the image with a mouse, and the implementation is perfect, with no issues.
Functionality - POLYGON	**0 Points** The submitted program does not provide a polygon drawing tool or the polygon drawing tool does not allow the user to craft a polygon, placing polygon vertices with clicks of the mouse and with some way of ending the drawing and closing off the polygon.	**0.5 Points** The submitted program provides a polygon drawing tool that allows the user to craft a polygon, placing polygon vertices with clicks of the mouse and with some way of ending the drawing and closing off the polygon, but there are areas where the implementation could have been handled better.	**1 Points** The submitted program provides a polygon drawing tool that allows the user to craft a polygon, placing polygon vertices with clicks of the mouse and with some way of ending the drawing and closing off the polygon, and the implementation is perfect, with no issues.

	Levels of Achievement		
Criteria	**Poor (zero marks)**	**Fair (half marks)**	**Good (full marks)**
Functionality - Undo	**0 Points** The submitted program does not have an undo feature or the undo feature does not work.	**0.625 Points** The submitted program has an undo feature that undoes the latest graphical operation (last plot, last line, last rectangle etc.) so that the operation is both no longer visible on the screen and is also not present in the VEC file if the file was saved after performing the undo- however, there is some problem with the functionality (e.g. either the key combination or menu option/button is missing, or the undoing is visually imperfect in some way, or undoing multiple times does not work, or undoes don't work on freshly loaded VEC files, or there is some other bug/undesirable operation).	**1.25 Points** The submitted program has an undo feature that undoes the latest graphical operation (last plot, last line, last rectangle etc.) so that the operation is both no longer visible on the screen and is also not present in the VEC file if the file was saved after performing the undo. The undo feature can be accessed via a button or menu option, and additionally can be accessed via Ctrl+Z) and it can be used repeatedly, allowing the entire image to be gradually undone, one operation at a time until the image is totally blank, and this even works on freshly loaded VEC files.
Functionality - Saving VEC images	**0 Points** The submitted program is unable to save VEC images, OR the saved VEC images are highly invalid (e.g. so broken they can't even be loaded by the same software that created them to reproduce the original image).	**1.125 Points** The submitted program is able to save VEC images, but the user is unable to fully select the path those images are saved to with an appropriate file save dialog (that also automatically inserts the .VEC extension if the user tried to save the file without giving an extension) OR There are minor formatting errors in the VEC images- however, the saved VEC images can still be reloaded by the same software.	**2.25 Points** The submitted program is able to save perfectly formatted VEC images and the user is able to fully select the path those images are saved to with an appropriate file save dialog (that also automatically inserts the .VEC extension if the user tried to save the file without giving an extension)

	Levels of Achievement		
Criteria	**Poor (zero marks)**	**Fair (half marks)**	**Good (full marks)**
		(Note that this means that marks are not awarded for saving VEC images if the program cannot properly load VEC images).	
Unit Testing	**0 Points** Unit tests are not present, or unit test coverage is very sparse, to the point where there are entire classes that could be unit tested but aren't OR The program is architected in such a way that there is nothing or almost nothing that can be unit tested OR There is so little code in the submitted product that there isn't anything to unit test. OR Unit tests are present, but they do not use JUnit 5 Jupiter.	**1.625 Points** Unit testing with JUnit 5 is present, but they do not cover the entire codebase of reasonably unit-testable code. OR Unit testing with JUnit 5 is present, but tests do not properly test for exceptions (using appropriate assertThrows() calls). OR Unit testing with JUnit 5 is present, but the code has been architected in such a way that no unit-testable code throws exceptions, and therefore exceptions have not been unit tested	**3.25 Points** Unit testing with JUnit 5 is present and comprehensively covers the parts of the codebase that can be tested; and the program has been architected with appropriate use of exceptions and appropriate classes to make unit testing these possible.
Code Quality	**0 Points** The code quality is very low, with problems like poor indentation, inconsistent formatting, bad variable names, poorly designed methods that are too large or too small and missing, useless	**1.125 Points** The code quality is adequate - there are areas where the formatting could be improved or be more consistent, and there are areas where some refactoring needs to be performed, but the code is at least understandable and well commented and able to be followed without a great deal of	**2.25 Points** The code quality is excellent. Formatting is consistent, variable names, method names and class names are all good and follow Java conventions and there are no refactorings that need to be performed. The code is already so that it

Criteria	Levels of Achievement		
	Poor (zero marks)	Fair (half marks)	Good (full marks)
	or misleading comments or large amounts of code duplication. The code is in major need of refactoring. OR The submitted assignment has so little code that the code quality cannot be evaluated.	difficulty.	doesn't need comments, but it has comments and those make it even better.
Design	**0 Points** The software design is highly deficient, bizarre or unprofessional. Too few classes or used, or too many classes (and there is serious amount of duplication between them). Good OOP design principles were not employed in the development of this software. OR The submitted assignment is so incomplete the design cannot be evaluated.	**1.25 Points** The software design is reasonable, with appropriate classes, but the interactions between classes are too tightly coupled, or the classes are designed with methods and fields with inappropriate levels of visibility, and in general there are problems with the code that would make maintaining or extending the software more difficult than necessary.	**2.5 Points** The software design is excellent, with classes and interactions between those classes demonstrating good OOP design principles, such that the design does not get in the way of maintaining or extending the code.
Deployment	**0 Points** The marker was not able to load the project into IntelliJ and get it running, or was only able to after a considerable amount of	**0.625 Points** The marker was able to load the project into IntelliJ, but there were minor changes required to reconfigure the project before the program or its unit tests ran correctly.	**1.25 Points** The marker was able to load the project into IntelliJ and run it and the unit tests without any modifications.

	Levels of Achievement		
Criteria	**Poor (zero marks)**	**Fair (half marks)**	**Good (full marks)**
	work, including potentially modifying the code itself to get it running. OR There is so little submitted, or what was submitted does not compile at all, that the marker is unable to get the project running.		
Additional Functionality (Note: If group has 1 person, the score for this section is the GREATER of: whatever mark the student would get for this section OR the student's score for the rest of the assignment minus this section, divided by 30 and multiplied by 5)	**0 Points** No additional functionality was implemented, OR the additional functionality was implemented incorrectly or poorly	**2.5 Points** If the group consists of 3 people, only 1 item of additional functionality was implemented correctly OR If the group consists of 4 people, only 2 items of additional functionality were implemented correctly OR All items of additional functionality were implemented correctly, but there are deficiencies in the way they were implemented OR The group chose to come up with their own items of additional functionality in lieu of the provided ones, and these worked correctly but did not amount to the same level of complexity or utility as the provided ones. (Note that only the necessary items of additional functionality for the group size need to be of sufficient quality- if the group consists of 3 members and the submitted assignment has 3 items of additional functionality, only 2 need to	**5 Points** The group submitted sufficient items of additional functionality for the size of the group (at least 3 items for a group of 4, at least 2 items for a group of 3 and at least 1 item otherwise) and these items of additional functionality worked, are of high quality and are of sufficient complexity and utility to equal the example items of additional functionality described in the assignment specification.

	Levels of Achievement		
Criteria	**Poor (zero marks)**	**Fair (half marks)**	**Good (full marks)**
		be of high quality for full marks to be awarded)	

View Associated Items

Print Close Window