Queensland University of Technology

**Name – Liam Allan**

**Student Number** – 10220607

**Name – Patrick Gallagher**

**Student Number** –

**Name – Ben Bloss**

**Student Number** – 9815848

**Due date** – 02/06/19

Cab302 Assessment Item 2

Documentation

# Statement of Completion

The application has been completed to a reasonable degree, with the implementation of a proper amount of additional functionality. The necessary VEC file importing and exporting tools have been implemented, with all the shapes necessary to create the files requested. The ability to create shapes with a fill colour has been implemented for all of the shapes created in the program. Rectangles, Ellipses, Lines, Plots and Polygons are all supported, and are all accessible through the Vector Design tool. An Undo option for the drawing tool has also been implemented to erase the latest drawing operation. Additionally, the additional functionality of a Zoom tool and multi-image support have been implemented.

# Statement of Agile Software Development Processes Implementation

* Pre-Project
* Feasibility Study
* Functional Model Iteration
* Design and Build Iteration
* Implementation
* Post-Project

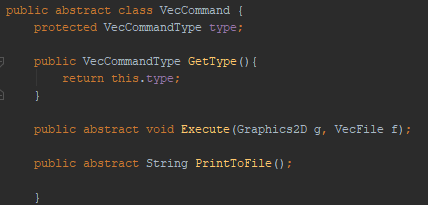
# Documentation of Software Architecture

# 

# Documentation of Advanced Object-Oriented Programming Principles

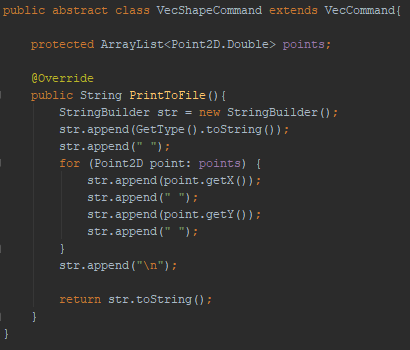
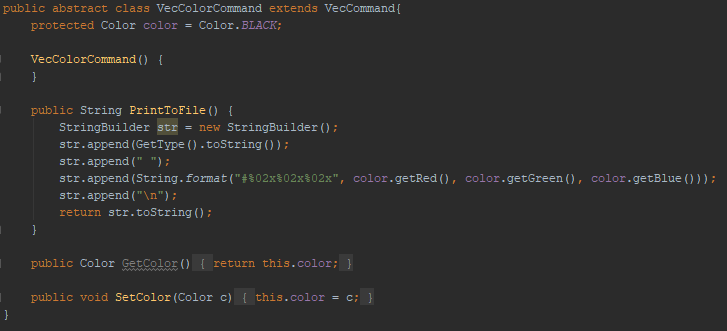
## For this assignment it was necessary to create several classes to represent commands used by a vector graphics printer. These commands were all required to behaviour in context specific ways while exposing a unified interface. For this purpose, we create the VecCommand class to capture at an abstract level the behaviour of all VecCommands, specifically all commands where required to be executable, each providing a method for changing the application state, and to be printable, returning a string representation of the command for writing to a “.vec” file to be read and executed by a printer. A VecCommandEnum was created to easily differentiate the types of commands and a VecCommandFactory was created to ensure the correct object was create based on a supplied type.

VecCommand:

The VecCommand class defines a private variable to store its VecCommandEnum type and a method for reading this type. It also outlines 2 abstract methods, “Execute” and “PrintToFile”. These methods are inherited by the child classes that extend VecCommand and in the case of the abstract methods the implementation is to be specified by the child classes.

VecColorCommand & VecShapeCommand:

2 classes extend the VecCommand class. These class represent the differentiation of the two fundamental types of VecCommands. VecColorCommands need to store color imformation and modified application state when excuted. VecShapeCommands store a list of vertices and when executed convert these vertices into a shape to be drawn onto a canvas. These classes are abstract passing the requirement to implement the abstract methods of its parent onto their child class where needed. The VecColorCommand and VecShapeCommand implement the “PrintToFile” method as it implementation was generalisable to all child classes.



VecColorCommand & VecShapeCommand subclasses:

Each of the VecColorCommands subclasses impements the “Execute” method required to render the shape it abstractly represents onto a canvas. The VecColorCommand subclass VecColorCommandFillOff overrides the inherited “PrintToFile” method and provides its on context specific implementation.

Advanced OOP priinciples:

Form this outline it is clear that we have sufficiently applied the principles of OOP of Abstraction, Encapsulation, Inheritance and Polymorphism.

# Documentation of Software Use

Guide to how to use software, with screenshots.