Development Report and Documentation

CAB302 - Assignment 2

Alex Butler, Bradley Fuller & Joshua O'Riordan

June 2, 2019

Contents

1 Statement of Completeness

The solution includes the following features:

Drawing - Line	Complete - Tested Working
Drawing - Plot	Complete - Tested Working
Drawing - Rectangle	Complete - Tested Working
Drawing - Ellipse	Complete - Tested Working
Drawing - Polygon	Complete - Tested Working
Drawing - Pen Colour	Complete - Tested Working
Drawing - Fill Colour	Complete - Tested Working
File Handling - Read	Complete - Tested Working
File Handling - Write	Still to be implemented
GUI - Layout	Complete - Tested Working
GUI - Buttons	Complete - Tested Working
GUI - Menu Bar	Complete - Tested Working
GUI - Colour Picker	Complete - Tested Working
Additional - Undo History	Complete - Tested Working
Additional - Bitmap Export	Complete - Tested Working

By the due date, we were able to complete all of the essential requirements and the required amount of two additional requirements - being Undo History, and Bitmap Export.

2 Statement of Contribution

Alex Butler

Alex implemented all of the Shape classes, as well as the Shape interface and enumerator. He also wrote the unit tests for testing each Shape class, and the first iteration of the GUI (mainly for testing purposes).

Bradley Fuller

Bradley implemented the file handling and validation classes, as well as the unit testing for said classes. He also wrote and configured the Apache Ant build script, and the TravisCI configuration for continuous integration and delivery to GitHub Releases. He also wrote most of the report, excluding Section 6.

Joshua O'Riordan

Joshua implemented the final iteration of the GUI, including the general layout, buttons, menu bar and colour picker. He also wrote the additional requirements functionality (being Undo History and Bitmap Export), as well as their respective unit tests.

3 Software Development Practices

The team utilised Agile's philosophy of iterative and incremental changes, through the use of continuous integration and deployment. This was achieved through the use of TravisCI and GitHub Releases. Whenever a change was made in any branch of the repository, TravisCI would create a temporary Docker container with Ubuntu 16.04 and Oracle JDK 11. All dependencies are then downloaded - in our case, JUnit 5.5.0 and JavaFX, as well as Apache Ant 1.10.6. Ant is used for creating all build folders, compiling the solution, and then running all unit tests (which are stored in the src/tests/ directory). We would then receive a report (via email) of the testing and if any errors occurred. However, if the change was made in the "master" branch of the repository, TravisCI would run Ant's "dist" script, which packages the compiled Java classes in to a JAR file. This JAR file would then be given a version number (which during development was TravisCI's build number) and uploaded to GitHub Releases. Pull requests would also be tested against the "master" branch of the repository to ensure there would be no breaking changes.

Due to other units and other assessment deadlines, we were not able to follow Agile's sprint methodology and strict timing requirements exactly. However, each feature of the solution was completed in sprints, each being in their own branch in the repository. Once these were complete, the branch would be merged in to master, and work on the next feature could begin. The only exception to this was the File Handling features, as they required the GUI to be fully implemented before they could be merged - however they were tested thoroughly beforehand without the use of a GUI, so the integration was very simple. We also had a kanban board of features that were to be implemented in the project, as well as an automated kanban board for bug triage and fixing.

Test-driven development was also used where possible, especially in the file read and write classes of the solution. The file reading class had over 20 tests of it's own, to make sure that exceptions were being thrown when they should have. Different test cases were used such as invalid commands, invalid co-ordinates, invalid colours, non-VEC file tests, incorrect file types etc.

4 Software Architecture

As with every Java project, there is a Main class - in this instance, the Main class calls the MainScreen class in order to initialise the GUI. When initialised, the GUI class instantiates the ComponentsClass constructor, which in turn instantiates every Shape class and prepares them for drawing. As well as this, the ShapesEnum enumerator is constructed which includes a list of all shapes. This enumerator is used to aid in drawing shapes for the GUI. The FileRead and FileWrite classes are also instantiated as part of the GUI class - this is in order to read to and write from the Components constructor.

5 Usage of Advanced Object-Oriented Principles

Abstraction

Some details about abstraction usage.

Encapsulation

As with most software projects that incorporate the use of object-oriented practice, encapsulation has been used to a large extent in the development of this solution.

Inheritance

Some details about inheritance usage.

Polymorphism

Some details about polymorphism usage.

6 Using the Software

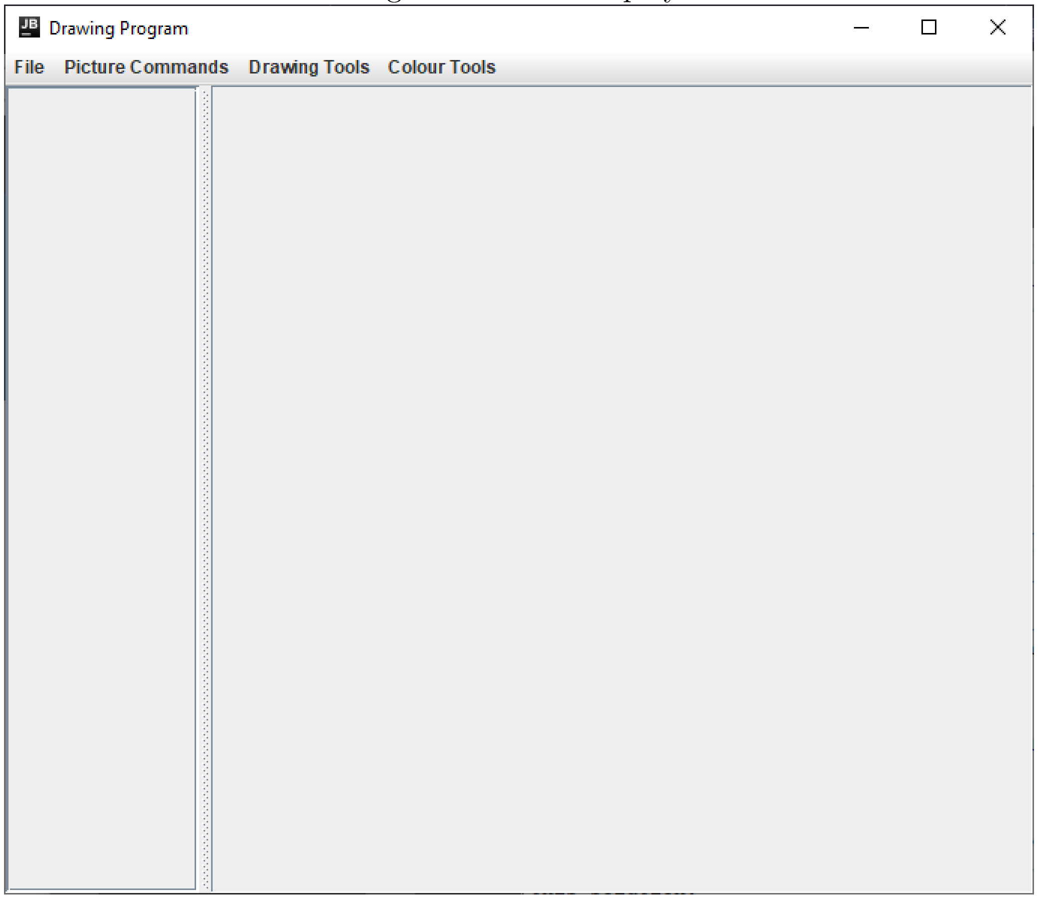
6.1 Preamble

This guide is designed to detail and instruct in the usage of the VEC drawing program. Each section describes related functions, such as file handling or keyboard shortcuts.

6.2 Running the Program

Upon opening the program a display such as that pictured is shown.

Figure 1: Blank display



The bar on the left hand side is a history bar, and the panel on the right is the location wherein you may draw shapes. Commands are situated in dropdown bars on the top of the program in the menu bar.

6.3 Keyboard Shortcuts

Some commands have keyboard shortcuts, such as Save File. These shortcuts are shown in parentheses after the name of the command. In the case of Save File, pressing ctrl+s (ctrl and s at the same time) will open the dialogue to save the currently displayed picture. Using a keyboard shortcut is the exact same as pressing on the corresponding button.

6.4 File

The options under file handle the saving, opening, clearing, and exporting of drawn shapes. The commands available in the dropdown are: New File, Open, Save, and Export BMP.

6.4.1 New File

New File clears the program of all history and drawn shapes. This wipes all memory of previous shapes and is irreversible. New File will also disable Undo History if it is already active. The following screenshots demonstrate what pressing New File does.

Figure 2: New Fill - before command

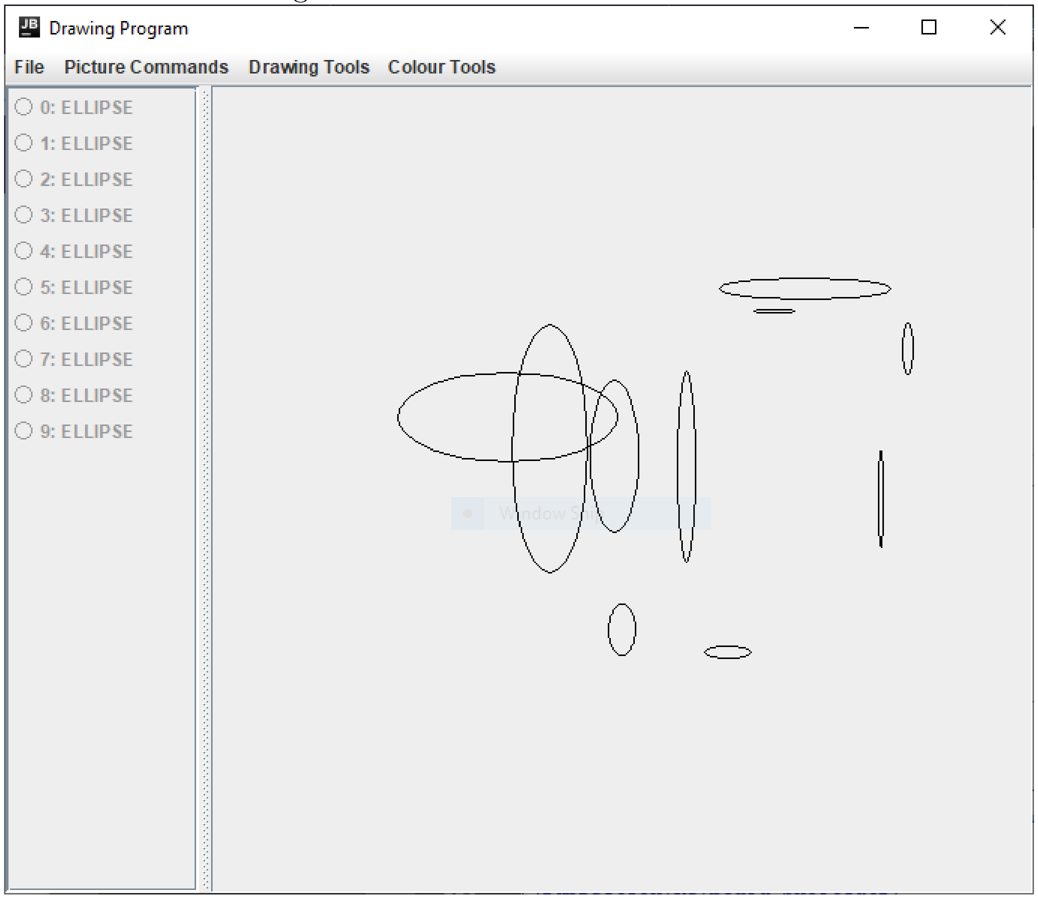
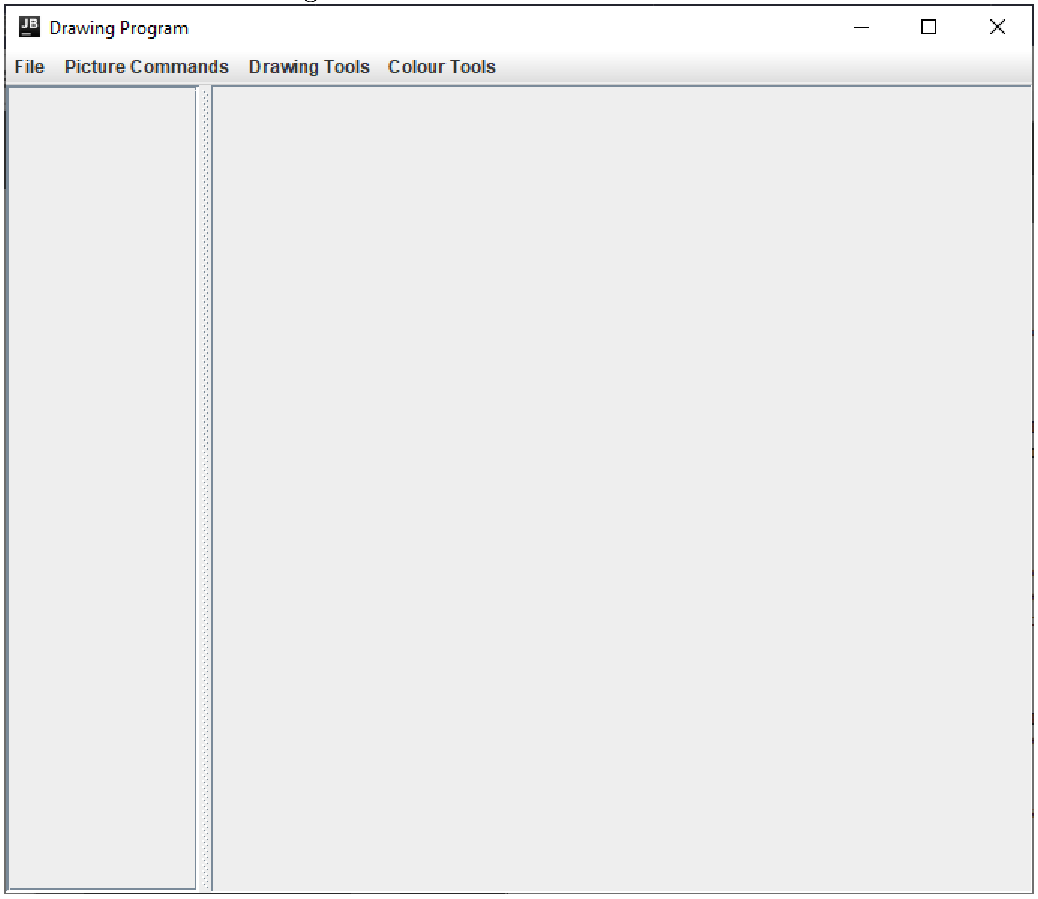


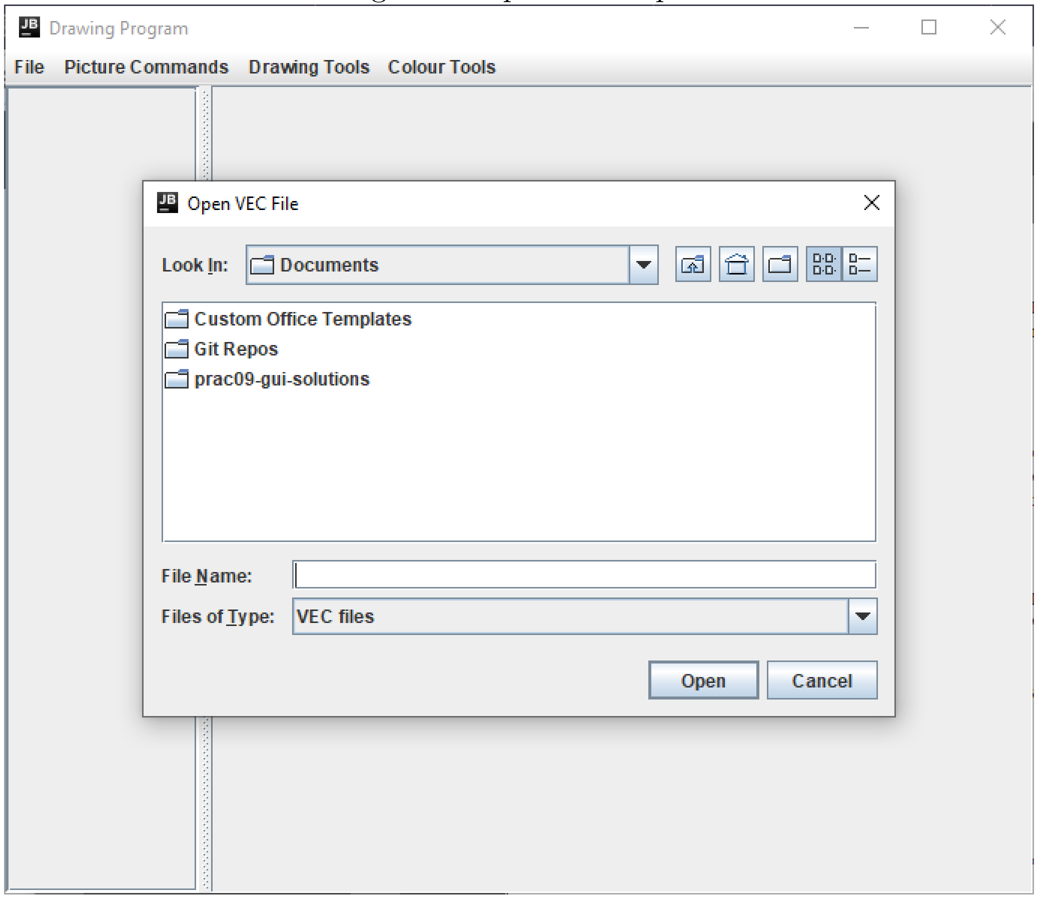
Figure 3: New Fill - after command



6.4.2 Open

Open allows you to load a VEC file. Loading a VEC file clears the current display of any working progress, and replaces it with that of selected VEC file. Searched file types are filtered by their extension, with the only files shown being those that have the .VEC extension. As Open sets the program to an entirely new image, Undo History upon the loading of the file.

Figure 4: Open - example

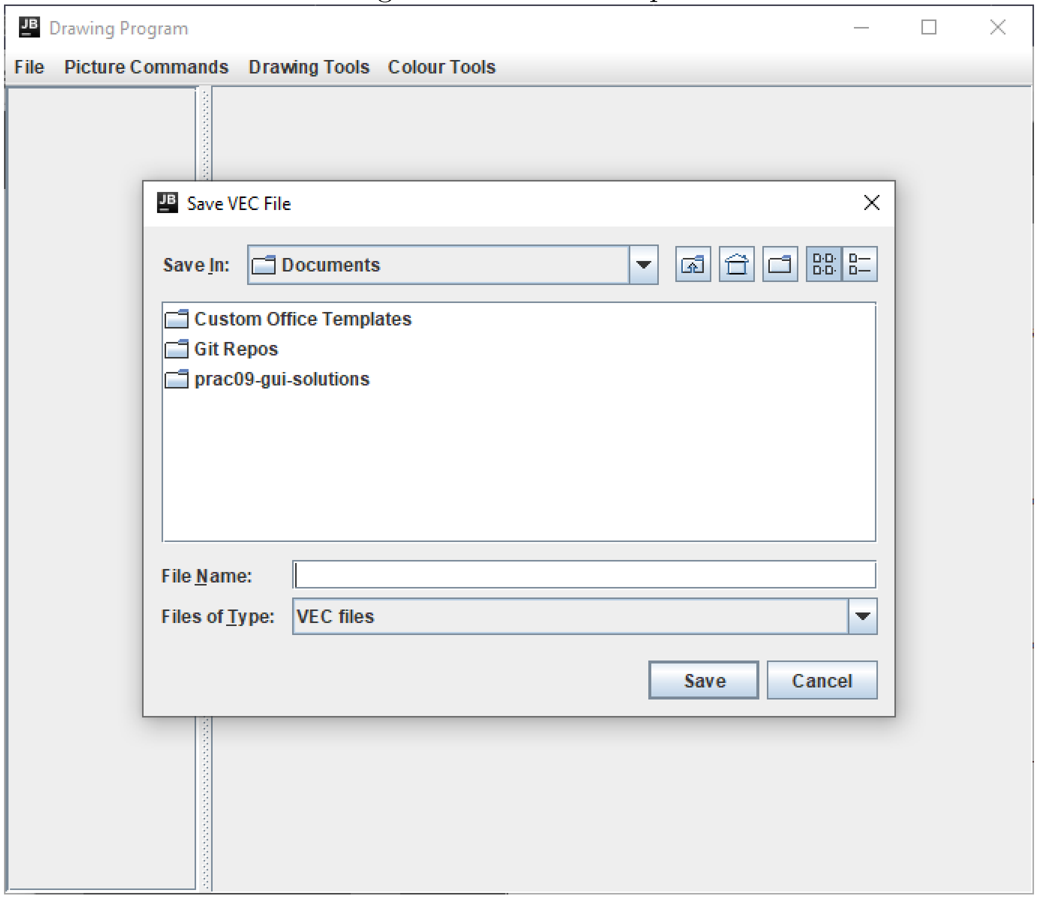


6.4.3 Save

Save is the opposite of Open, in that it allows for the current drawing to be saved to a VEC file. Save does not clear the program of previous shapes, allowing you to continuing working without having to open the newly saved file. Like Open, files are filtered so only VEC files are shown.

Save is disabled if Undo History is active. This ensures that what is saved is always identical to that as is being displayed on the drawing panel.

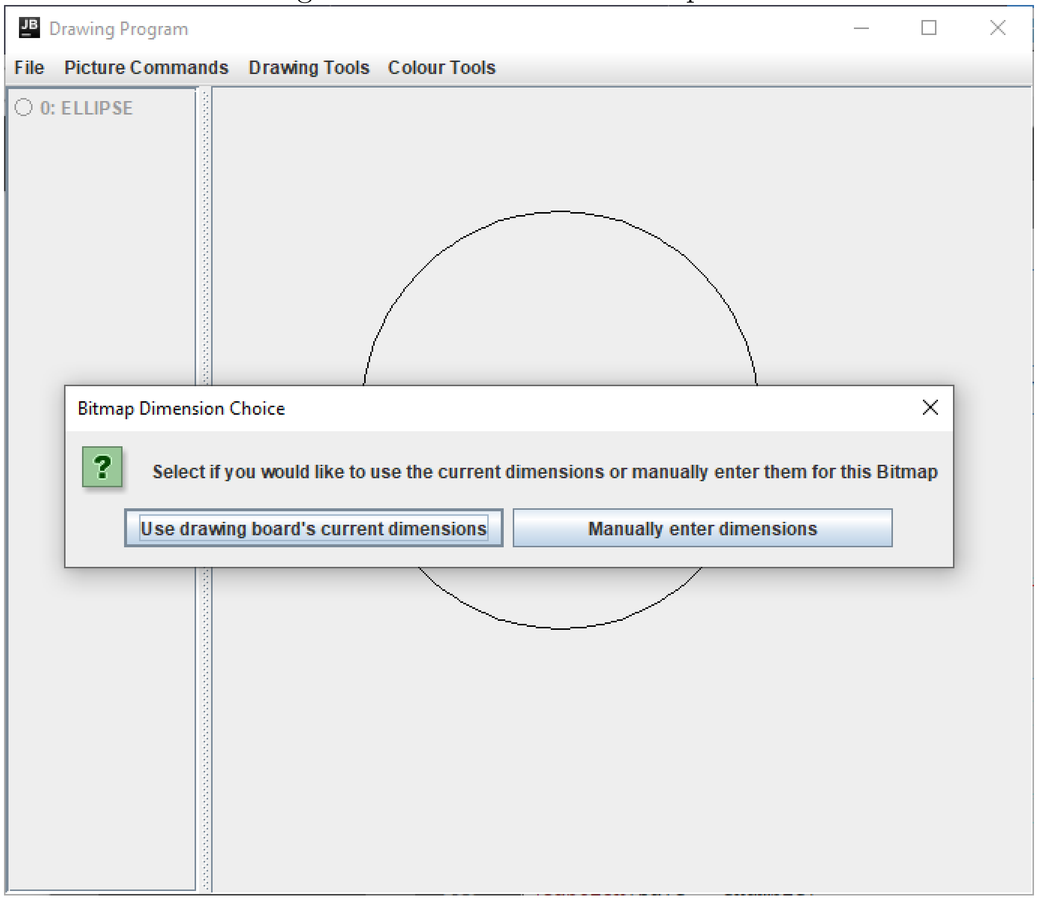
Figure 5: Save - example



6.4.4 Export BMP

Export BMP allows for the exporting of what is drawn on the display panel to a bitmap file. Upon pressing the command a window (as shown in Figure 6) is displayed.

Figure 6: BMP - Dimension Options

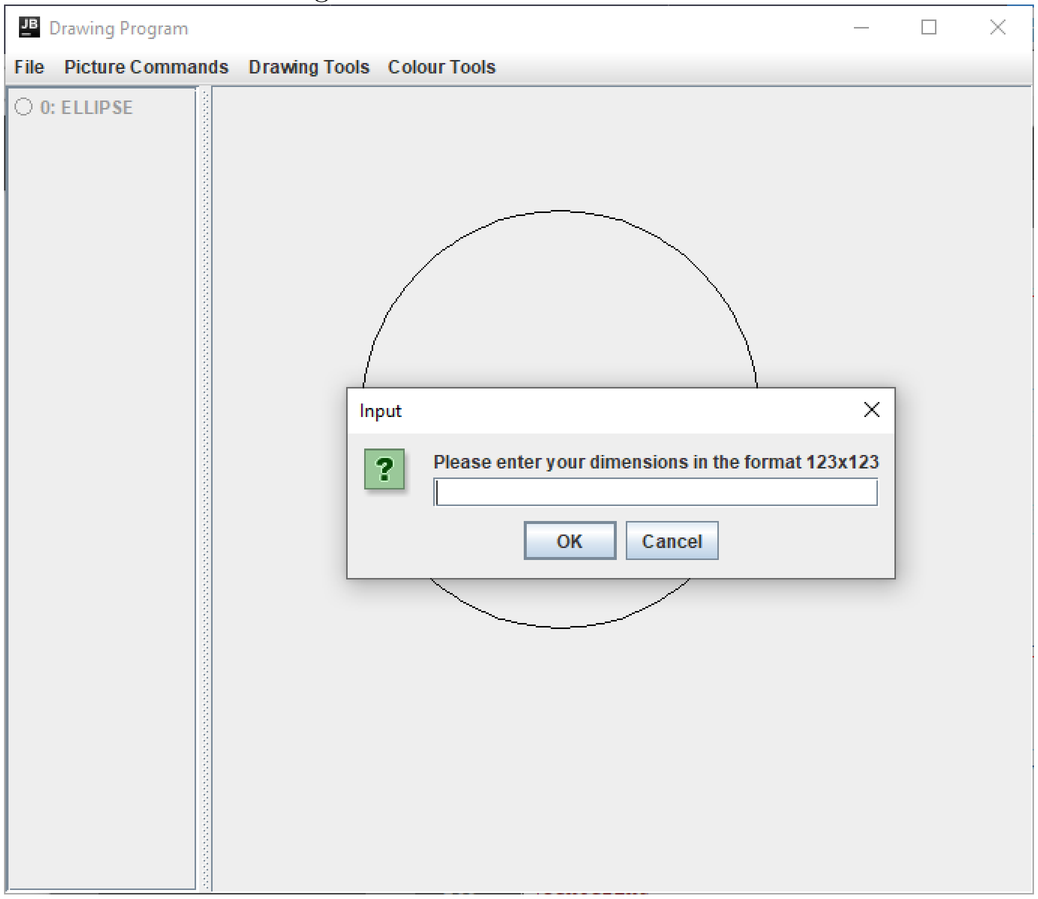


Select if you would like to use the current dimensions or manually enter them for this Bitmap

The option on the left (Use drawing board's current dimensions) which is highlighted by default will use the drawing panel's dimensions (in pixels) for the Bitmap. These are the dimensions of the drawing board at the time the Export BMP functionality was called.

The option on the right allows you to enter the width and height in pixels. If the right hand option (Manually enter dimensions) is pressed, a display such as below is shown.

Figure 7: BMP - User Dimensions



Please enter your dimensions in the format 123x123