**The British College**

**KATHMANDU**

**Coursework Submission Coversheet**(individual coursework only)

**Faculty of Arts, Environment and Technology LBU Student Id:**

Please ensure all information is complete and correct and attach this form securely to the front of your work before posting it in a coursework collection box.

Award name: Bsc(Hons) in Computing

Module code: 24545

Module name: Advanced Software Engineering D

Module run: 2020

Coursework title: A Software Engineering Graphical Programming Language

Due Date:

Module leader: (In LBU): Dr. Duncan Mullier

Module tutor: (In TBC): Resham Pun

**TURNITIN** Checked: YES NO ***(please circle)***

Submission date& time: Date: 24th December, 2020 Time: 23:00

**Total Word Count: 5864 Total Number of Pages (including this front sheet): 58**

**In submitting this form with your assignment, you make the following declaration:**  
I declare, that the coursework submitted is my own work and has not (either in whole or part) been submitted towards the award of any other qualification either at LBU or elsewhere. I have fully attributed/referenced all sources of information used during the completion of my assignment, and I am aware that failure to do so constitutes an assessment offence.

Signed: Sudip Chaulagain Date: 2020-12-24

**You are strongly advised to retain a second copy of your work in case of any query about the assignment.**

77262167

**For checking by the student:**

Contents

[Introduction 3](#_Toc59739152)

[Version Control 3](#_Toc59739153)

[Home Page for GPL Application 5](#_Toc59739154)

[Execution of command to generate a shape 6](#_Toc59739155)

[Commands used in application 10](#_Toc59739156)

[Testing 10](#_Toc59739157)

[Types of testing 10](#_Toc59739158)

[Codes used in GPL application 31](#_Toc59739159)

### Introduction

GPL Application is a command based application that reads commands on command line one at a time. The application generates different shapes such as circle, rectangle and triangle based on the commands provided by the users. The users can provide command through the text boxes used in the application where appropriate shape is then generated in the panel section. The shapes are generated only if the user provides correct command otherwise the application would throw user friendly error. The users are able to save and import the commands even through the application.

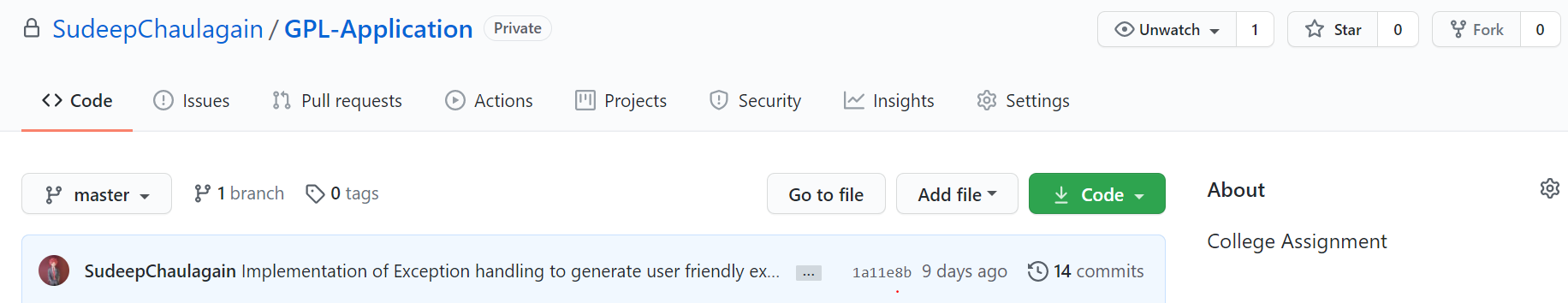
The application and its working mechanism have been developed overtime. The application working progress were tracked using version control. Git version control was used in order to track changes.

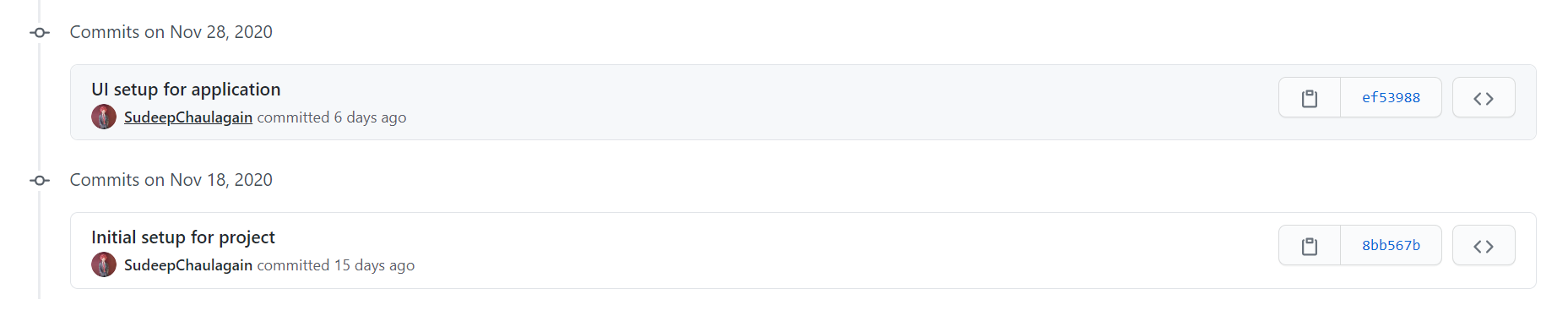
### Version Control

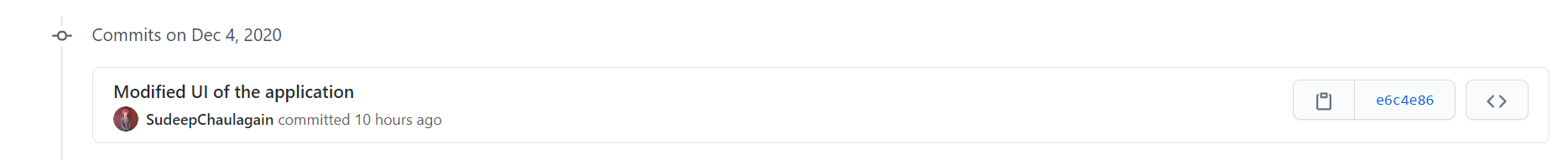
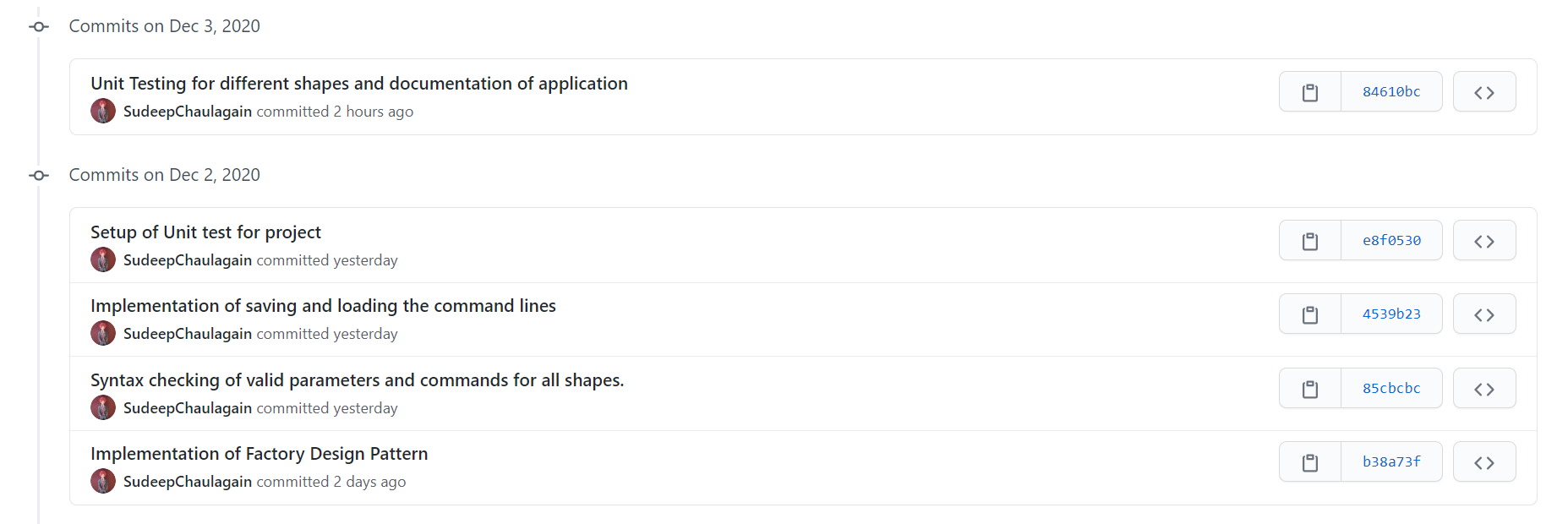
Version control is a system that records changes to a file or set of files over time so that you can recall specific versions later. It is even known as revision control where revisions can be compared, restored, and with some types of files, merged. The version control systems, commonly run as stand-alone applications, but may also be embedded in various types of software, including integrated development environments (IDEs).

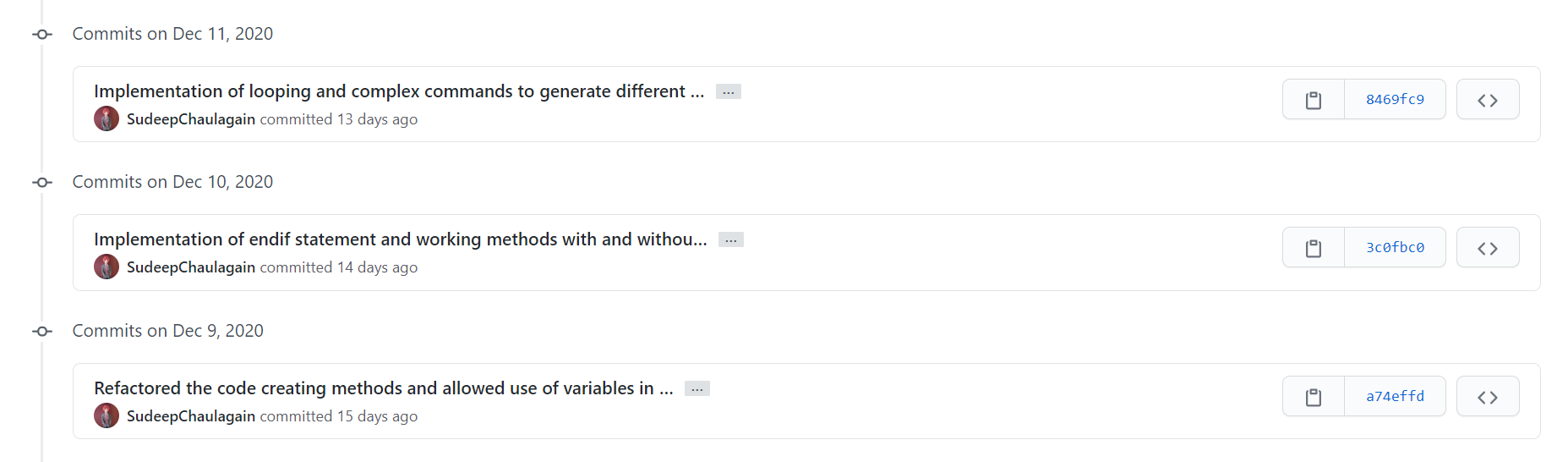
Version control systems help to track developments and changes in our files, record the changes you made to your file in a way that we will be able to understand later, experiment with different versions of a file while maintaining the original version, merge two versions of a file and manage conflicts between versions and revert changes, moving backward through your history to previous versions of our file.

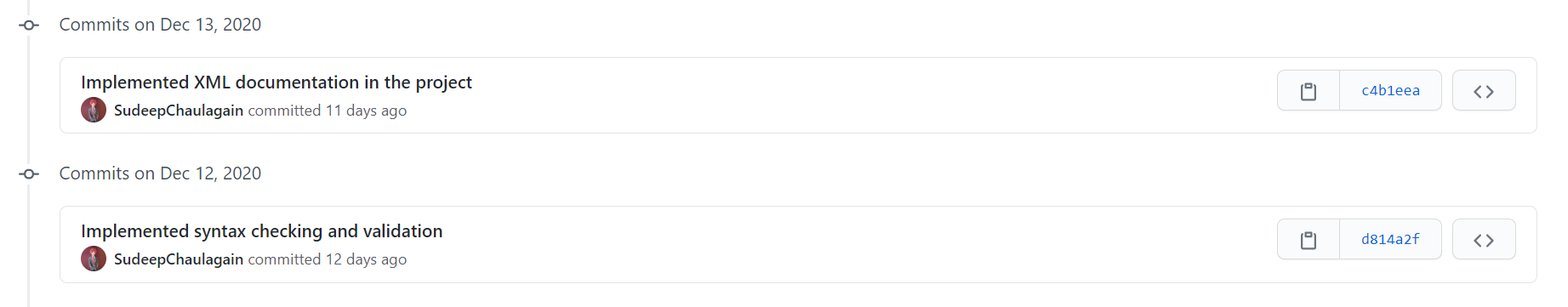
Evidences of commits that have been made in the GitHub have been presented in order to show that Git version control system was used in order to track the changes during the development of GPL Application.











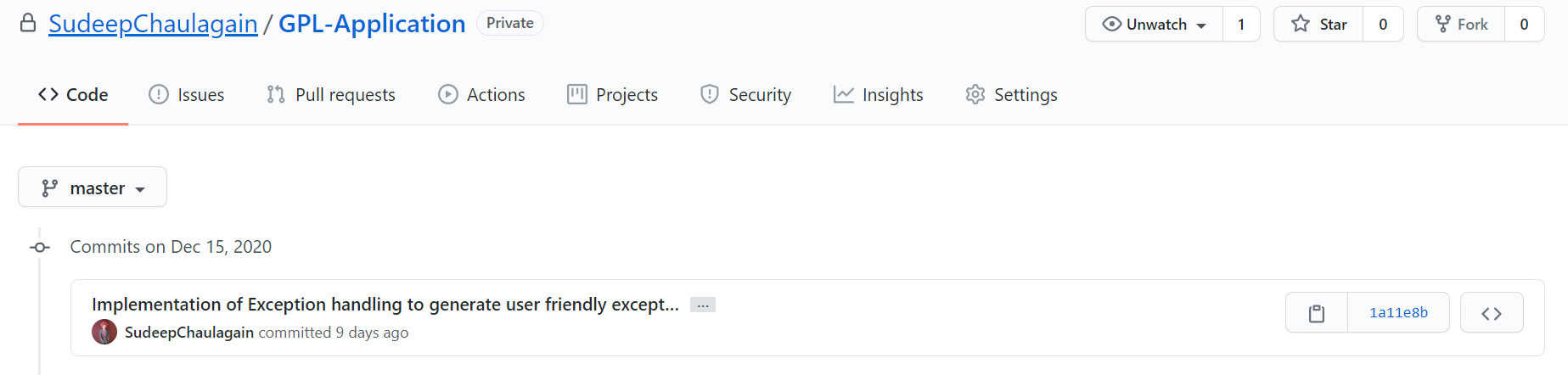


Figure 1 Commits made in GitHub For GPL Application

### Home Page for GPL Application

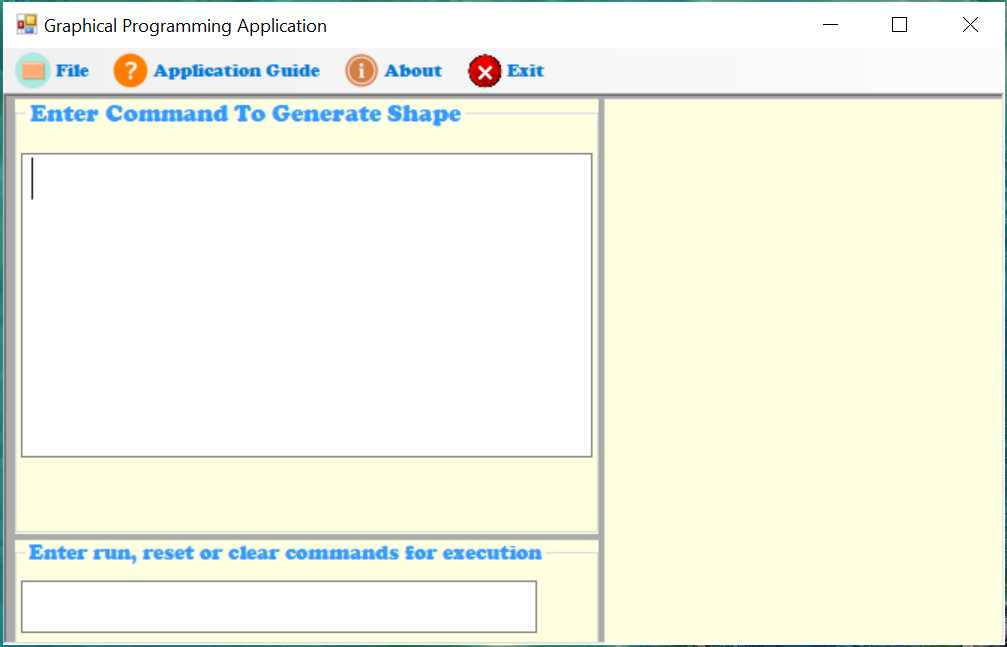


Figure 2 Home Page of GPL Application

The above figure is the home page of GPL Application that has designed using various toolbox present for windows form application. The menus for the application were designed using menu strip. On the left side is one multi line command box and on the right side there is drawing panel where the shapes are generated as per the commands given in the command box. The right below section contains single line command box where commands such as run, clear and reset can only be entered.

### Execution of command to generate a shape

1. **Generation of rectangle**

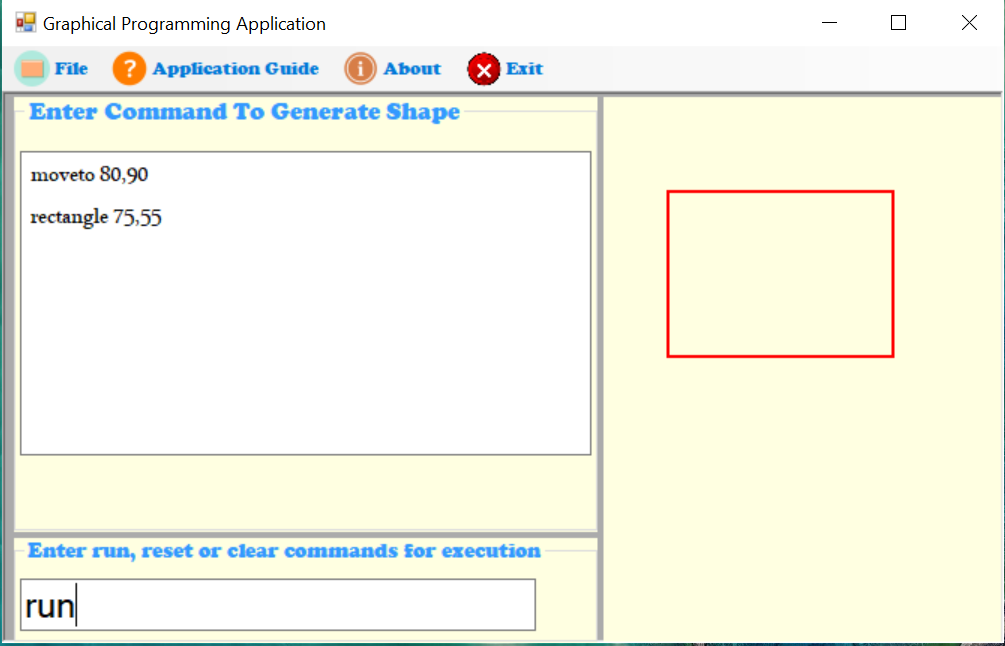


Figure 3 Execution of command to generate rectangle

1. **Generation of triangle**

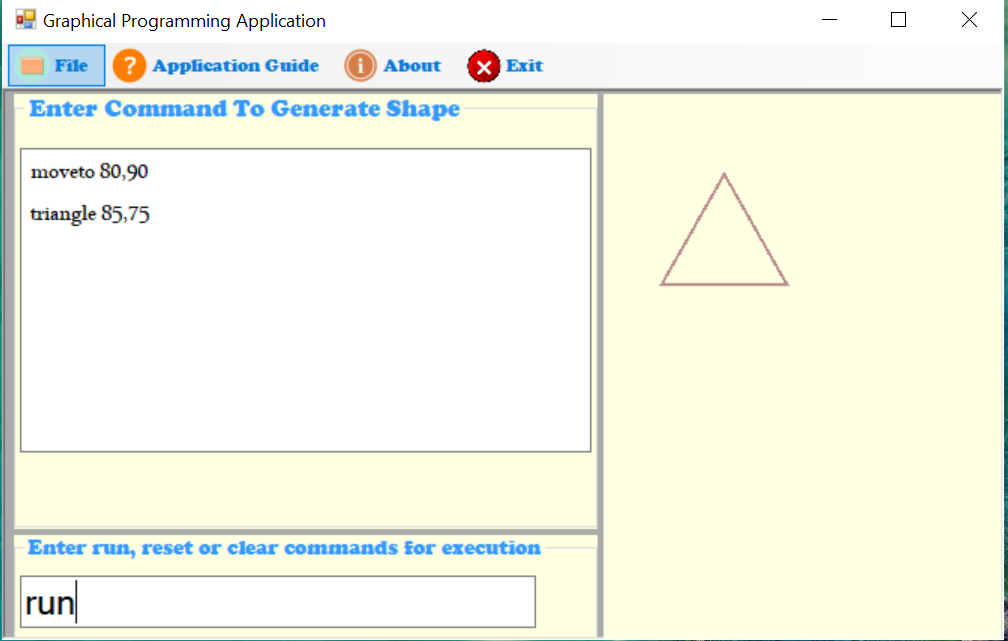


Figure 4 Execution of command to generate triangle

1. **Generation of circle**

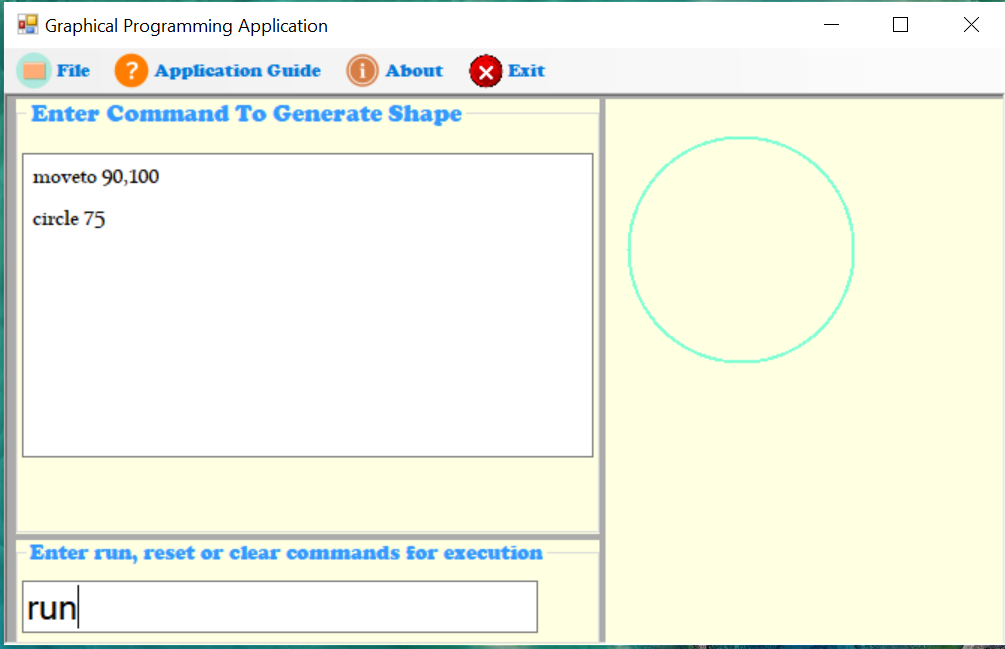


Figure 5 Execution of command to generate circle

1. **Generation of rectangle using variables**

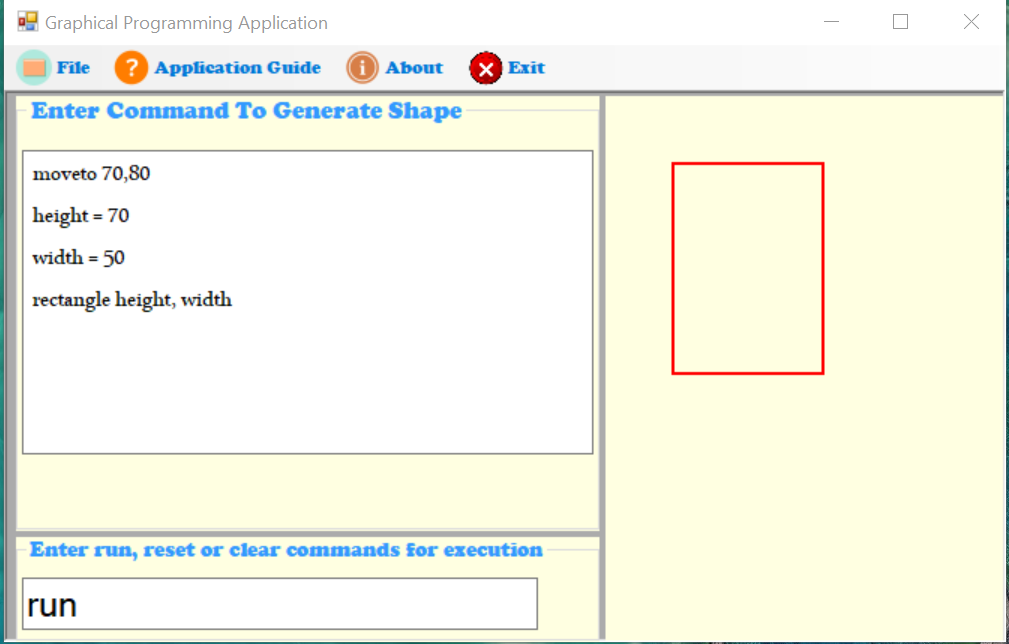


Figure 6 Execution of command to generate rectangle using variables

1. **Generation of circle using variables**

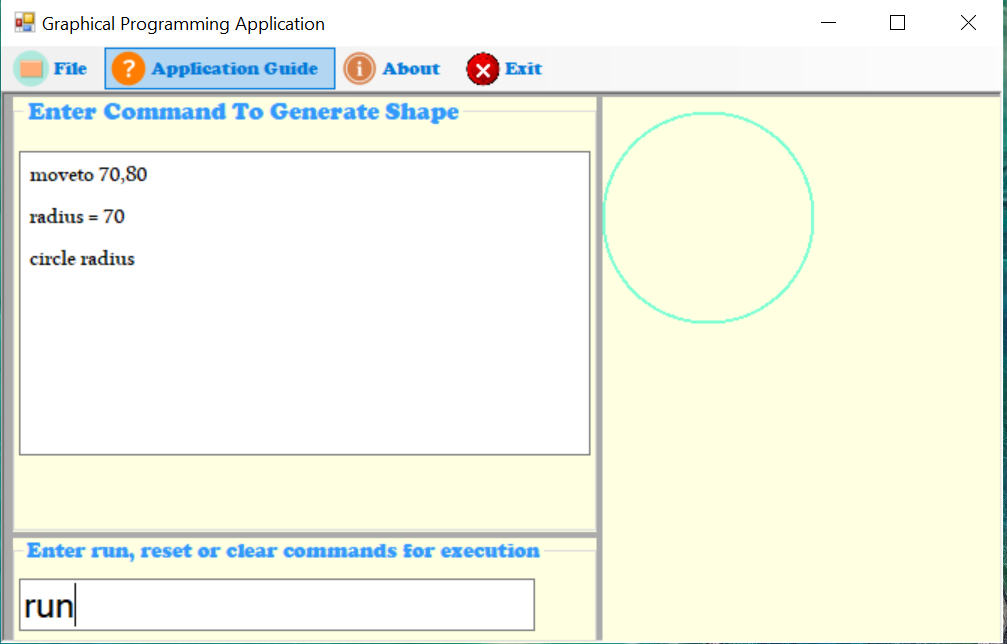


Figure 7 Execution of command to generate circle using variables

1. **Use of loop**

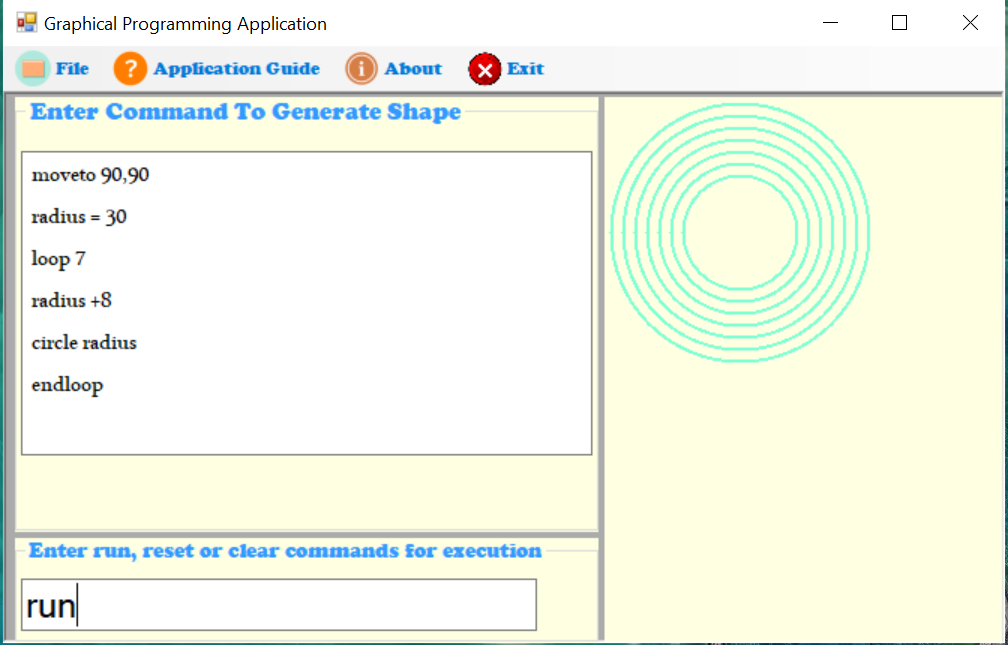


Figure 8 Execution of command to draw circle using loop

1. **Use of if and end if statement**

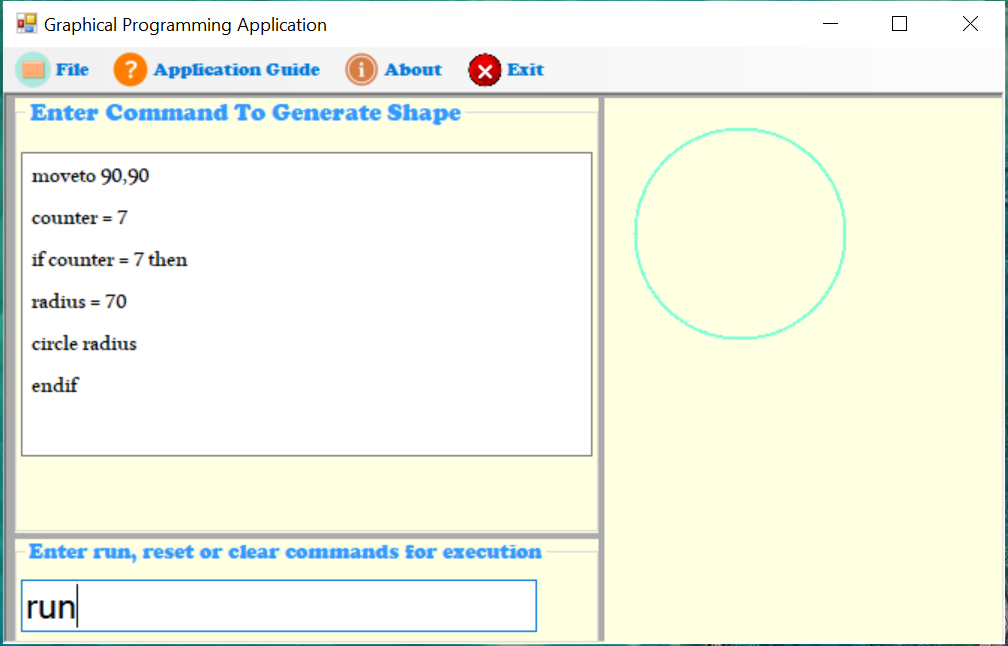


Figure 9 Execution of command to draw circle using end if statement

1. **Use of loop and end if statement**

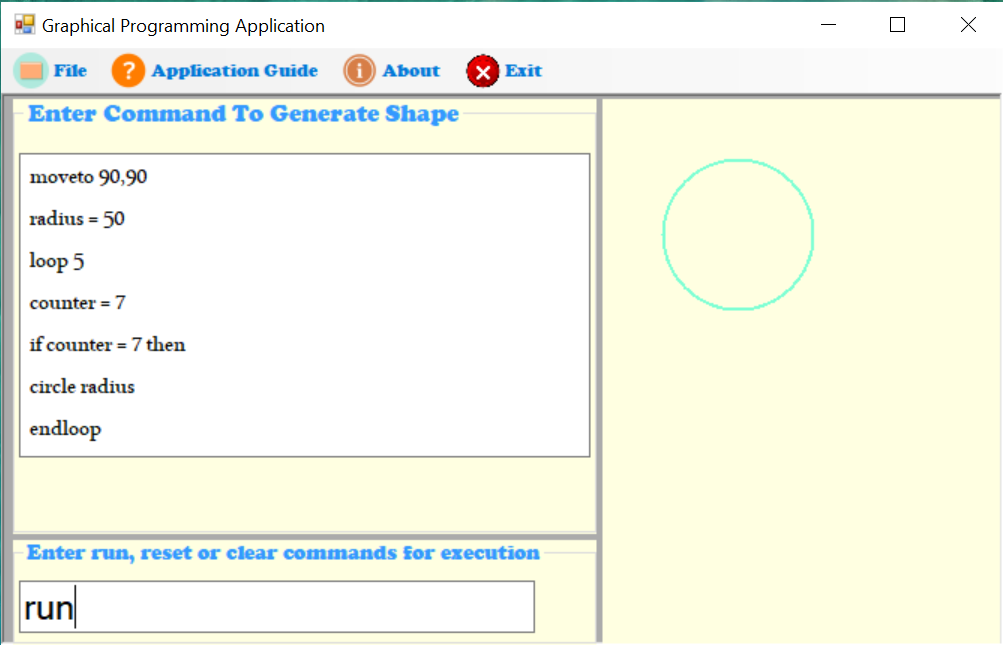


Figure 10 Execution of command to draw circle using end if statement and loop

### Commands used in application

1. Pen positioning

moveto x, y 🡪 moveto 70,80

1. Pen draw

drawto x, y 🡪drawto 70,80

1. Clearing the output area

clear

1. Reset the pen position to initial position

reset

1. Drawing basic shapes
2. Circle
3. circle <radius> 🡪 circle 70
4. Rectangle
5. rectangle <width>, <height> 🡪 rectangle 70,50
6. Triangle
7. triangle <side>,<side>,<side> 🡪triangle 70, 60, 50
8. If and end if statement
9. Looping

### Testing

Software testing is a process, to evaluate the functionality of a software application with an intent to find whether the developed software met the specified requirements or not and to identify the defects to ensure that the product is defect-free in order to produce a quality product.

### Types of testing

The different types of testing are explained below:

1. **Unit Testing:**

This software testing approach is followed by the programmer to test the unit of the program. It helps developers to know whether the individual unit of the code is working properly or not.

1. **Integration Testing:**

It focuses on the construction and design of the software. We need to see that the integrated units are working without errors or not.

1. **System Testing**

In this method, the software is compiled as a whole and then tested as a whole. This testing strategy checks the functionality, security, portability, amongst others.

1. **White box testing**

White Box Testing is based on the application’s internal code structure. In white-box testing, an internal perspective of the system, as well as programming skills, are used to design test cases. This testing is usually done at the unit level.

1. **Black box testing**

Black Box Testing is a software testing method in which testers evaluate the functionality of the software under test without looking at the internal code structure.

In the GPL application, three unit-testing were performed in order to determine that the individual unit of the code of the application is functioning correctly. Evidences below show that proper unit testing was carried out in order to ensure the proper working of the GPL application.

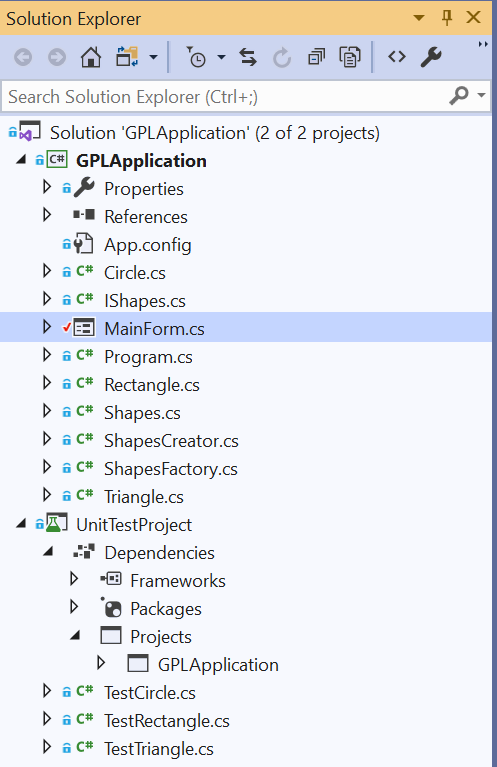


Figure 11 Unit testing available in GPL application



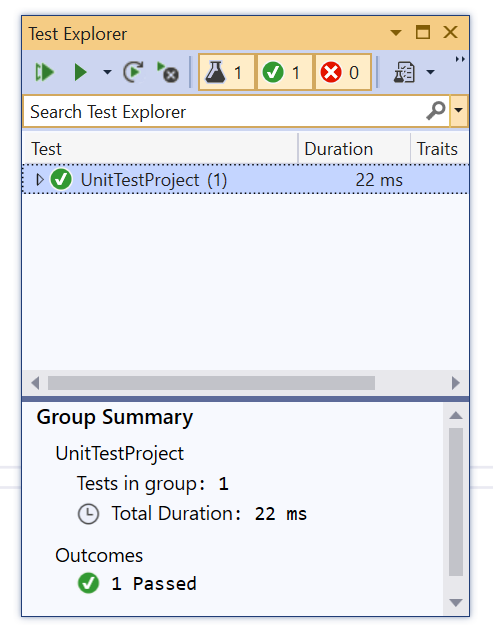


Figure 12 Unit Testing of Rectangle



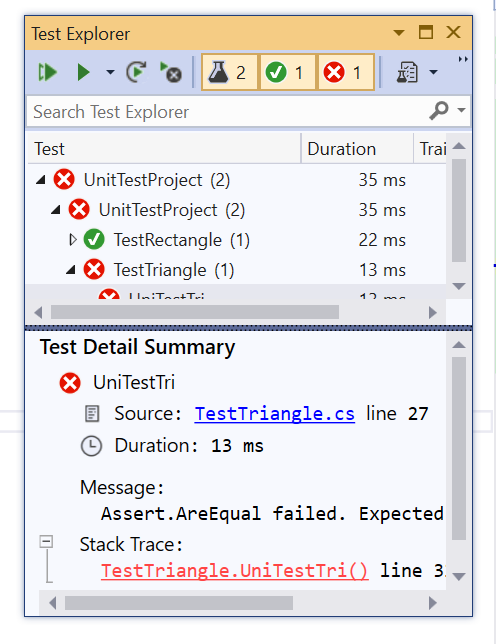


Figure 13 Failure of Unit Testing for Triangle

In the above figure, the failure of unit testing of triangle is seen as the values of height of triangle are not equal. After swapping the position of height and width, appropriate values where set to height and width and the test was success which is present in screenshot below.

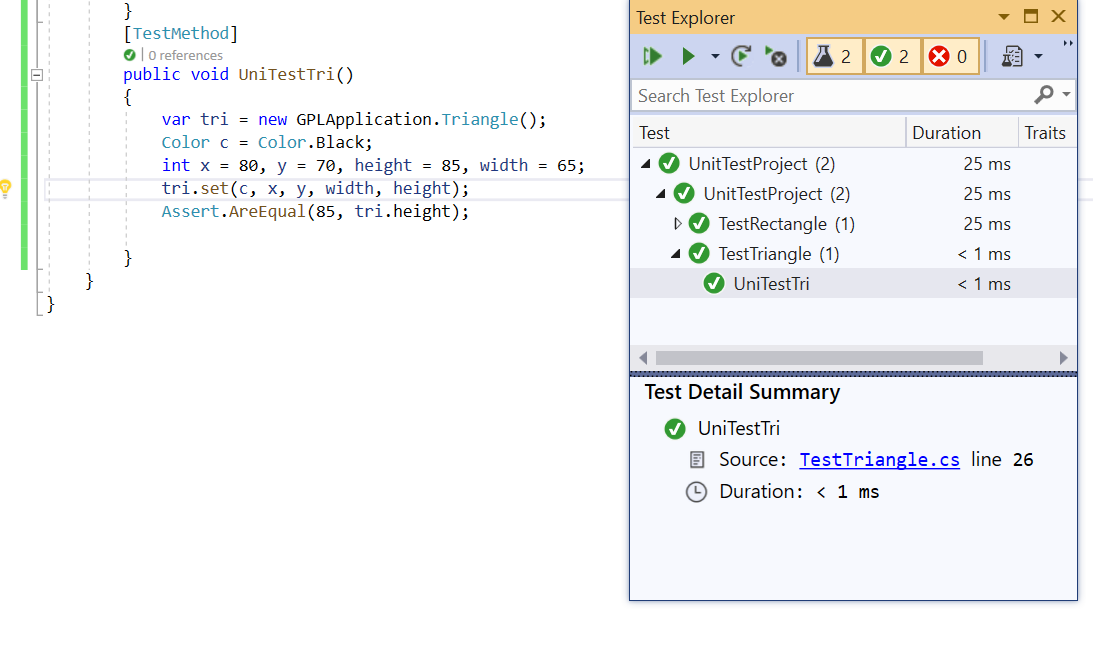


Figure 14 Successful unit testing of triangle



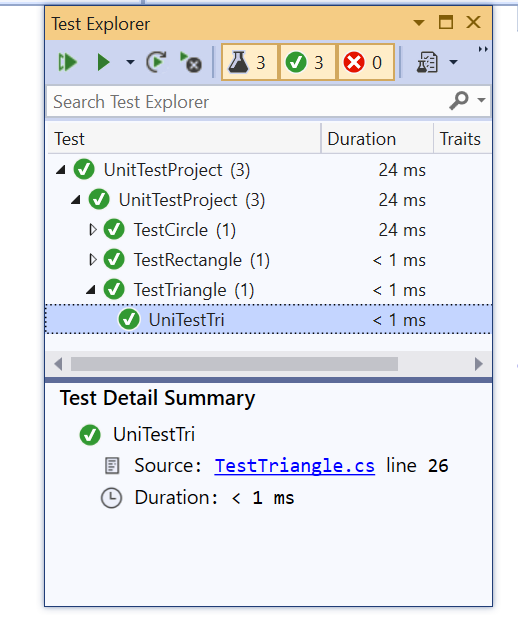
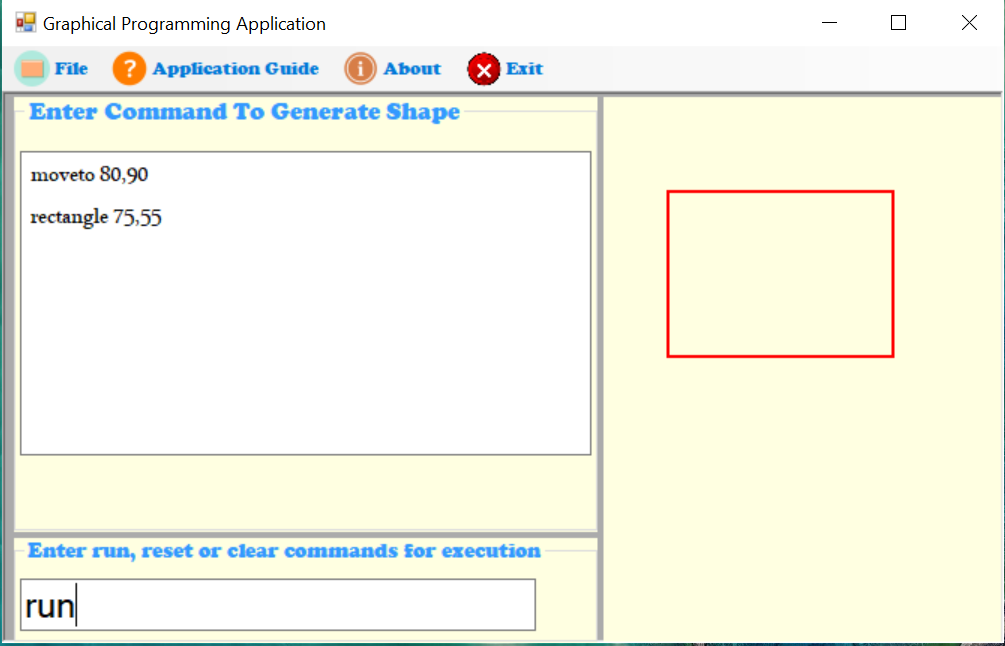
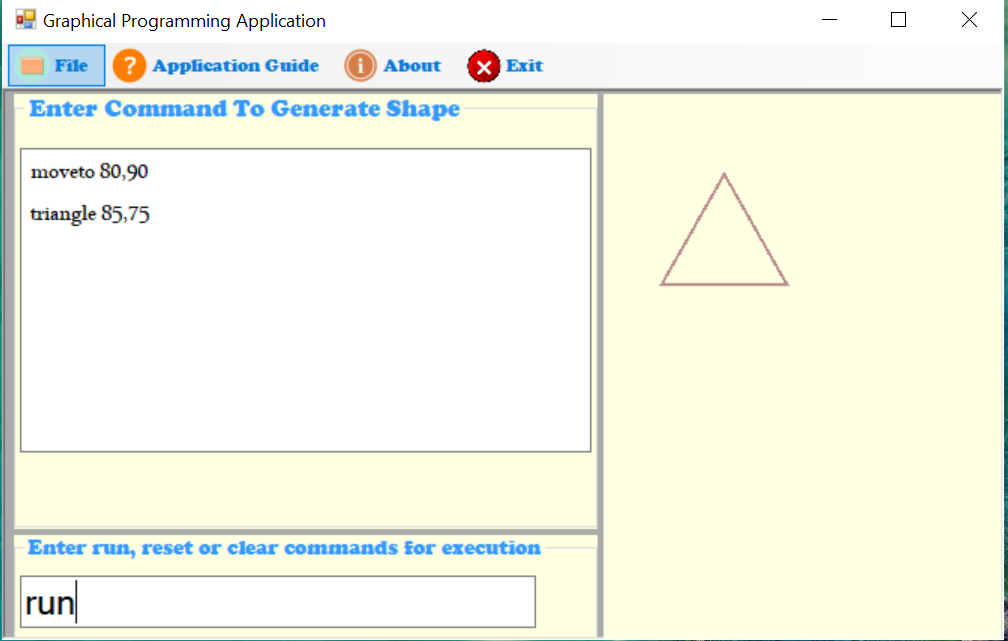


Figure 15 Successful unit testing of circle

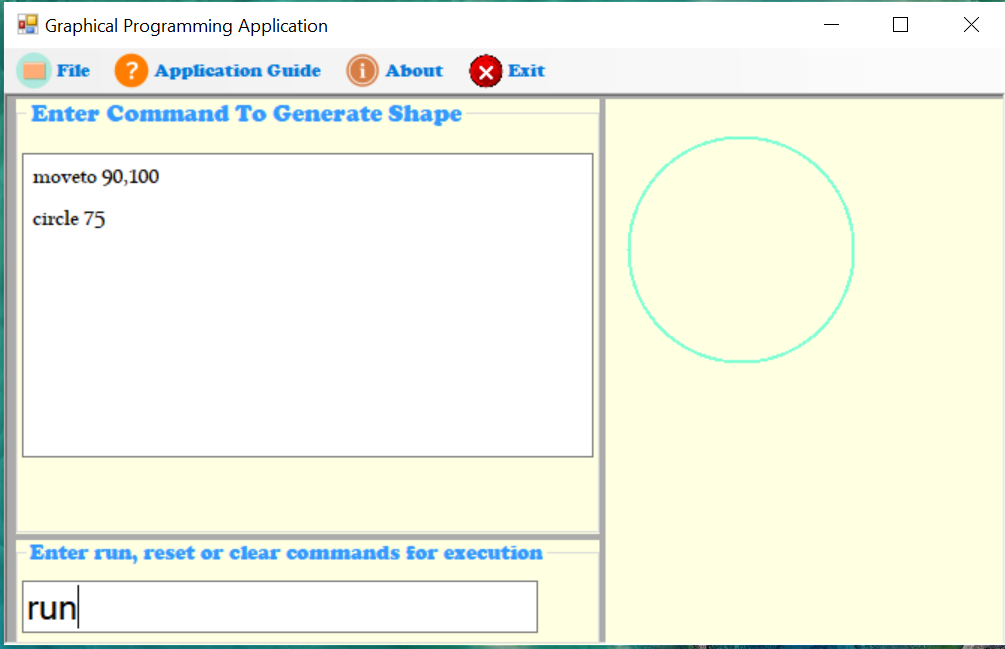
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Test Case ID** | | | | 01 | **Test Case Description** | | | Test the Functionality in Graphical Programming Application | | | | | | |
| **Created By** | | | | Sudeep | **Reviewed By** | | | The British College | | **Version** | | | 1.0.1 | |
|  | |  | |  |  | |  |  |  |  |  | |  |  |
| **QA Tester’s Log** | | | | Review comments from British college | | | | |  |  |  | |  |  |
|  | |  | |  |  | |  |  |  |  |  | |  |  |
| **Tester’s Name** | | | | Bijay | **Date Tested** | | | 3-december -2020 | | **Test Case (Pass/Fail/NOT)** | | | Pass | |
|  | |  | |  |  | |  |  |  |  |  | |  |  |
| **S#** | | **Prerequisites** | | | | |  | **S#** | | **Test Data** | | | | |
|  | |  | | | | |  | 1. | | moveto 80,90  rectangle 75,55 | | | | |
|  | |  | | | | |  |  | |  | | | | |
|  | |  | | | | |  |  | |  | | | | |
| **Test Scenario** | | Verifying on entering valid command, the user can change drawing position and draw rectangle. | | | | | | | |  |  | |  |  |
|  | |  | |  |  | |  |  |  |  |  | |  |  |
| **Step #** | **Step Details** | | **Expected Results** | | | **Actual Results** | | | | | | **Pass/Fail/Not executed/suspended** | | |
| **1** | Open Graphical programming application | | Application should open | | | As Expected | | | | | | Pass | | |
| **2** | Enter the command moveto x, y  rectangle <width>, <height> | | Text can be entered | | | As Expected | | | | | | Pass | | |
| **3** | Enter the command Run | | Drawing position has changed and rectangle will be drawn | | | As expected | | | | | | Pass | | |



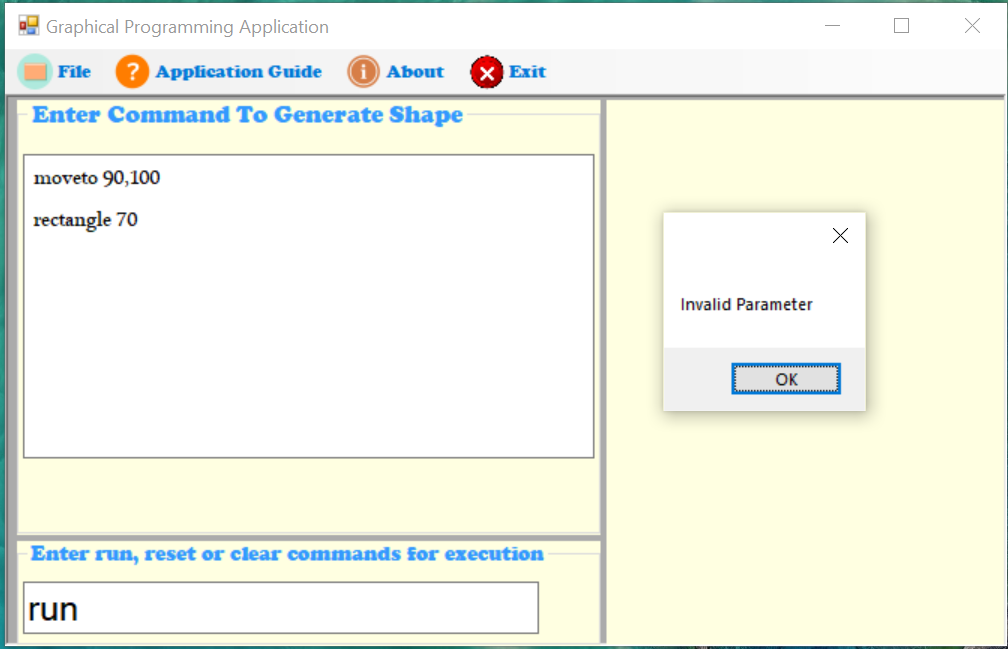
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Test Case ID** | | | | 02 | **Test Case Description** | | | Test the Functionality in Graphical Programming Application | | | | | | |
| **Created By** | | | | Sudeep | **Reviewed By** | | | The British College | | **Version** | | | 1.0.1 | |
|  | |  | |  |  | |  |  |  |  |  | |  |  |
| **QA Tester’s Log** | | | | Review comments from British college | | | | |  |  |  | |  |  |
|  | |  | |  |  | |  |  |  |  |  | |  |  |
| **Tester’s Name** | | | | Ravi | **Date Tested** | | | 3-december -2020 | | **Test Case (Pass/Fail/NOT)** | | | Pass | |
|  | |  | |  |  | |  |  |  |  |  | |  |  |
| **S#** | | **Prerequisites** | | | | |  | **S#** | | **Test Data** | | | | |
|  | |  | | | | |  | 1. | | moveto 80,90  triangle 85,75 | | | | |
|  | |  | | | | |  |  | |  | | | | |
|  | |  | | | | |  |  | |  | | | | |
| **Test Scenario** | | Verifying on entering valid command, the user can change drawing position and draw triangle | | | | | | | |  |  | |  |  |
|  | |  | |  |  | |  |  |  |  |  | |  |  |
| **Step #** | **Step Details** | | **Expected Results** | | | **Actual Results** | | | | | | **Pass/Fail/Not executed/suspended** | | |
| **1** | Open Graphical programming application | | Application should open | | | As Expected | | | | | | Pass | | |
| **2** | Enter the command moveto x, y  triangle<side>,<side> | | Text can be entered | | | As Expected | | | | | | Pass | | |
| **3** | Enter the command Run | | Drawing position will change and triangle will be drawn | | | As expected | | | | | | Pass | | |



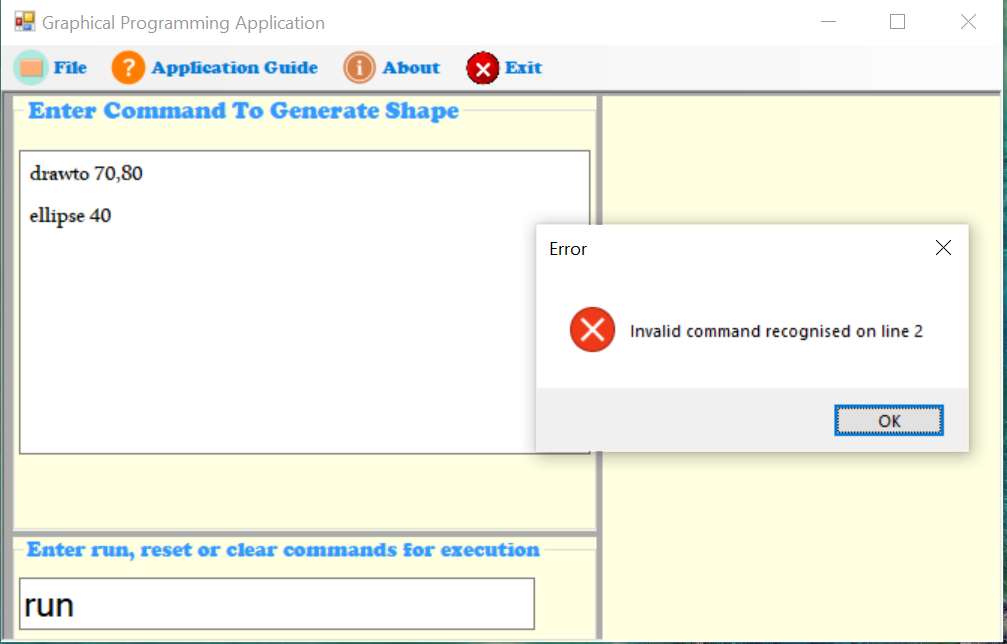
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Test Case ID** | | | | 03 | **Test Case Description** | | | Test the Functionality in Graphical Programming Application | | | | | | |
| **Created By** | | | | Sudeep | **Reviewed By** | | | The British College | | **Version** | | | 1.0.1 | |
|  | |  | |  |  | |  |  |  |  |  | |  |  |
| **QA Tester’s Log** | | | | Review comments from British college | | | | |  |  |  | |  |  |
|  | |  | |  |  | |  |  |  |  |  | |  |  |
| **Tester’s Name** | | | | Kiran | **Date Tested** | | | 3-december -2020 | | **Test Case (Pass/Fail/NOT)** | | | Pass | |
|  | |  | |  |  | |  |  |  |  |  | |  |  |
| **S#** | | **Prerequisites** | | | | |  | **S#** | | **Test Data** | | | | |
|  | |  | | | | |  | 1. | | moveto 90,100  circle 75 | | | | |
|  | |  | | | | |  |  | |  | | | | |
|  | |  | | | | |  |  | |  | | | | |
| **Test Scenario** | | Verifying on entering valid command, the user can change drawing position and draw circle | | | | | | | |  |  | |  |  |
|  | |  | |  |  | |  |  |  |  |  | |  |  |
| **Step #** | **Step Details** | | **Expected Results** | | | **Actual Results** | | | | | | **Pass/Fail/Not executed/suspended** | | |
| **1** | Open Graphical programming application | | Application should open | | | As Expected | | | | | | Pass | | |
| **2** | Enter the command  moveto x, y  circle<radius> | | Text can be entered | | | As Expected | | | | | | Pass | | |
| **3** | Enter the command Run | | Drawing position will change and circle will be drawn | | | As expected | | | | | | Pass | | |



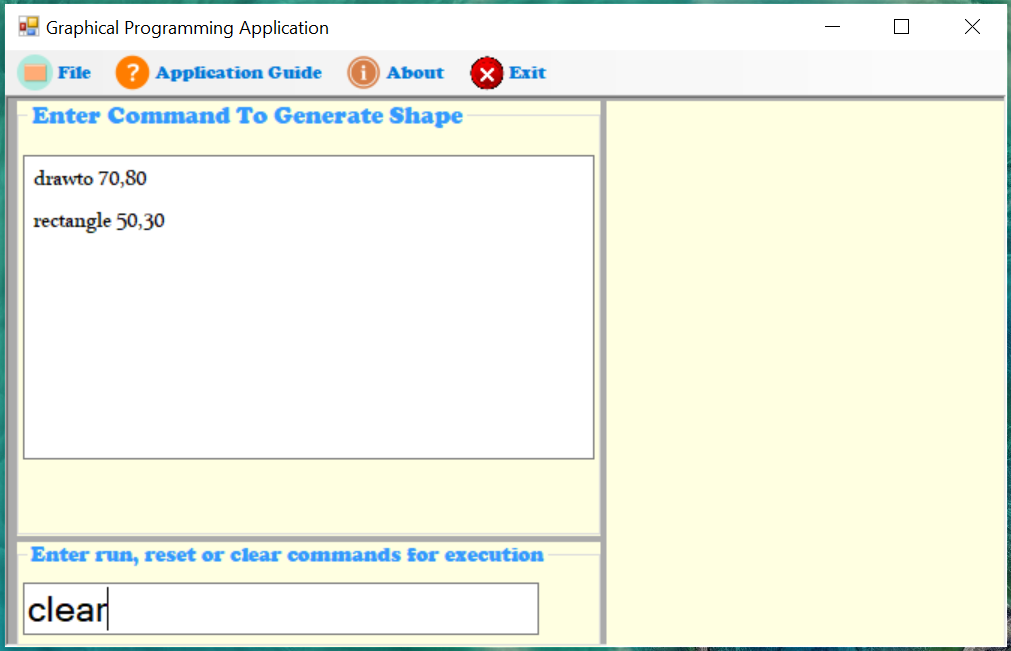
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Test Case ID** | | | | 04 | **Test Case Description** | | | Test the Functionality in Graphical Programming Application | | | | | | |
| **Created By** | | | | Sudeep | **Reviewed By** | | | The British College | | **Version** | | | 1.0.1 | |
|  | |  | |  |  | |  |  |  |  |  | |  |  |
| **QA Tester’s Log** | | | | Review comments from British college | | | | |  |  |  | |  |  |
|  | |  | |  |  | |  |  |  |  |  | |  |  |
| **Tester’s Name** | | | | Kiran | **Date Tested** | | | 3-december -2020 | | **Test Case (Pass/Fail/NOT)** | | | Pass | |
|  | |  | |  |  | |  |  |  |  |  | |  |  |
| **S#** | | **Prerequisites** | | | | |  | **S#** | | **Test Data** | | | | |
|  | |  | | | | |  | 1. | | moveto 90,100  rectangle 70 | | | | |
|  | |  | | | | |  |  | |  | | | | |
|  | |  | | | | |  |  | |  | | | | |
| **Test Scenario** | | Verifying on entering invalid parameters, the error message box will be seen. | | | | | | | |  |  | |  |  |
|  | |  | |  |  | |  |  |  |  |  | |  |  |
| **Step #** | **Step Details** | | **Expected Results** | | | **Actual Results** | | | | | | **Pass/Fail/Not executed/suspended** | | |
| **1** | Open Graphical programming application | | Application should open | | | As Expected | | | | | | Pass | | |
| **2** | Enter the command  moveto x, y  rectangle<width> | | Text can be entered | | | As Expected | | | | | | Pass | | |
| **3** | Enter the command Run | | Message box will be displayed showing invalid parameters. | | | As expected | | | | | | Pass | | |



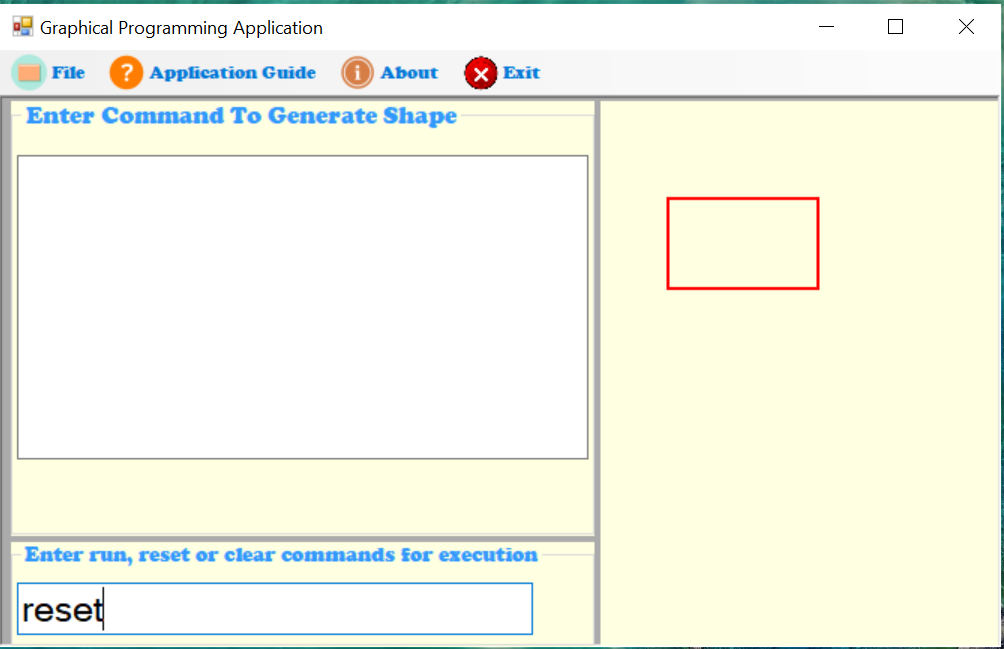
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Test Case ID** | | | | 05 | **Test Case Description** | | | Test the Functionality in Graphical Programming Application | | | | | | |
| **Created By** | | | | Sudeep | **Reviewed By** | | | The British College | | **Version** | | | 1.0.1 | |
|  | |  | |  |  | |  |  |  |  |  | |  |  |
| **QA Tester’s Log** | | | | Review comments from British college | | | | |  |  |  | |  |  |
|  | |  | |  |  | |  |  |  |  |  | |  |  |
| **Tester’s Name** | | | | Rajeev | **Date Tested** | | | 3-december -2020 | | **Test Case (Pass/Fail/NOT)** | | | Pass | |
|  | |  | |  |  | |  |  |  |  |  | |  |  |
| **S#** | | **Prerequisites** | | | | |  | **S#** | | **Test Data** | | | | |
|  | |  | | | | |  | 1. | | drawto 70,80  ellipse 40 | | | | |
|  | |  | | | | |  |  | |  | | | | |
|  | |  | | | | |  |  | |  | | | | |
| **Test Scenario** | | Verifying on entering invalid command, the error message box will be seen. | | | | | | | |  |  | |  |  |
|  | |  | |  |  | |  |  |  |  |  | |  |  |
| **Step #** | **Step Details** | | **Expected Results** | | | **Actual Results** | | | | | | **Pass/Fail/Not executed/suspended** | | |
| **1** | Open Graphical programming application | | Application should open | | | As Expected | | | | | | Pass | | |
| **2** | Enter the command  moveto x, y  ellipse<width> | | Text can be entered | | | As Expected | | | | | | Pass | | |
| **3** | Enter the command Run | | Message box will be displayed showing invalid command recognized on specific line. | | | As expected | | | | | | Pass | | |



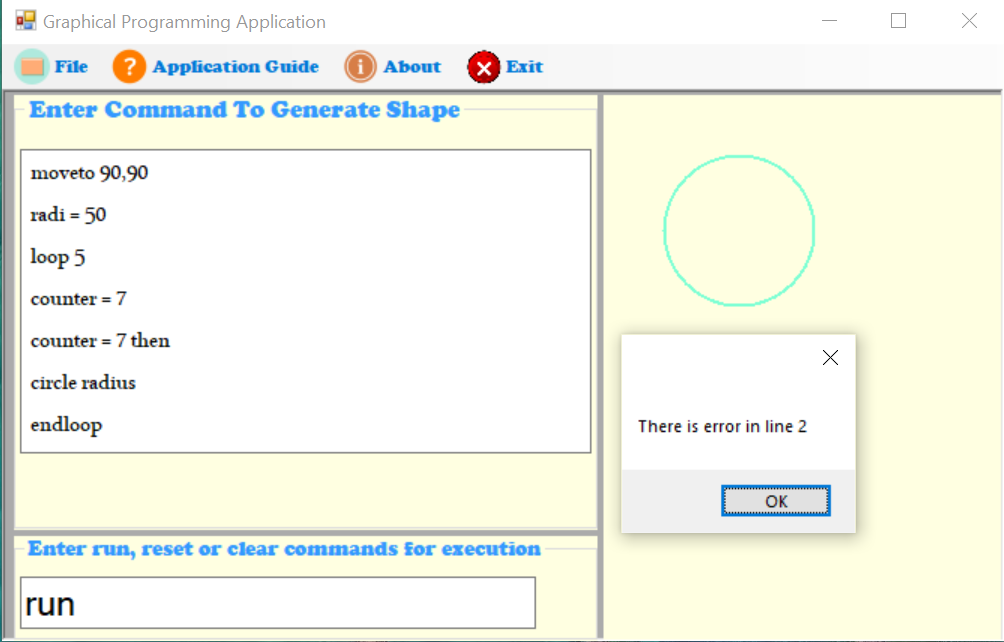
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Test Case ID** | | | | 06 | **Test Case Description** | | | Test the Functionality in Graphical Programming Application | | | | | | |
| **Created By** | | | | Sudeep | **Reviewed By** | | | The British College | | **Version** | | | 1.0.1 | |
|  | |  | |  |  | |  |  |  |  |  | |  |  |
| **QA Tester’s Log** | | | | Review comments from British college | | | | |  |  |  | |  |  |
|  | |  | |  |  | |  |  |  |  |  | |  |  |
| **Tester’s Name** | | | | Rajeev | **Date Tested** | | | 3-december -2020 | | **Test Case (Pass/Fail/NOT)** | | | Pass | |
|  | |  | |  |  | |  |  |  |  |  | |  |  |
| **S#** | | **Prerequisites** | | | | |  | **S#** | | **Test Data** | | | | |
|  | |  | | | | |  | 1. | | drawto 70,80  rectangle 50, 30  clear | | | | |
|  | |  | | | | |  |  | |  | | | | |
|  | |  | | | | |  |  | |  | | | | |
| **Test Scenario** | | Verifying on entering Clear command, the shape generated will be cleared. | | | | | | | |  |  | |  |  |
|  | |  | |  |  | |  |  |  |  |  | |  |  |
| **Step #** | **Step Details** | | **Expected Results** | | | **Actual Results** | | | | | | **Pass/Fail/Not executed/suspended** | | |
| **1** | Open Graphical programming application | | Application should open | | | As Expected | | | | | | Pass | | |
| **2** | Enter the command  moveto x, y  rectangle 50, 30 | | Text can be entered | | | As Expected | | | | | | Pass | | |
| **3** | Enter the command Run | | Drawing position has changed and rectangle will be drawn | | | As expected | | | | | | Pass | | |
| **4** | Enter the command Clear | | The rectangle drawn will be cleared | | | As expected | | | | | | Pass | | |

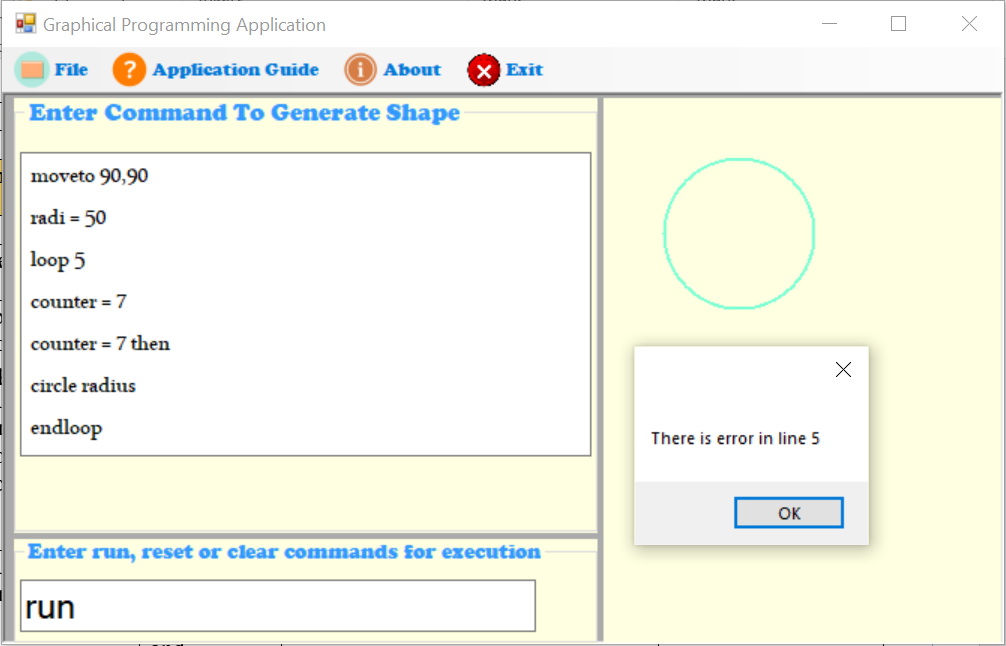


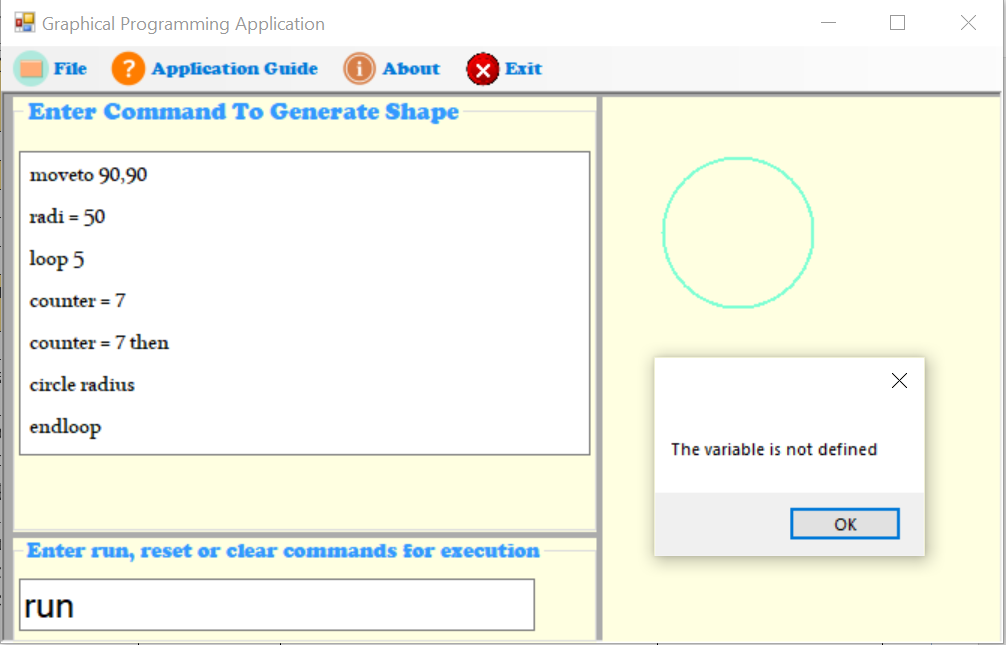
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Test Case ID** | | | | 07 | **Test Case Description** | | | Test the Functionality in Graphical Programming Application | | | | | | |
| **Created By** | | | | Sudeep | **Reviewed By** | | | The British College | | **Version** | | | 1.0.1 | |
|  | |  | |  |  | |  |  |  |  |  | |  |  |
| **QA Tester’s Log** | | | | Review comments from British college | | | | |  |  |  | |  |  |
|  | |  | |  |  | |  |  |  |  |  | |  |  |
| **Tester’s Name** | | | | Simran | **Date Tested** | | | 3-december -2020 | | **Test Case (Pass/Fail/NOT)** | | | Pass | |
|  | |  | |  |  | |  |  |  |  |  | |  |  |
| **S#** | | **Prerequisites** | | | | |  | **S#** | | **Test Data** | | | | |
|  | |  | | | | |  | 1. | | drawto 70,80  rectangle 50, 30  reset | | | | |
|  | |  | | | | |  |  | |  | | | | |
|  | |  | | | | |  |  | |  | | | | |
| **Test Scenario** | | Verifying on entering Reset command, the command written will be cleared. | | | | | | | |  |  | |  |  |
|  | |  | |  |  | |  |  |  |  |  | |  |  |
| **Step #** | **Step Details** | | **Expected Results** | | | **Actual Results** | | | | | | **Pass/Fail/Not executed/suspended** | | |
| **1** | Open Graphical programming application | | Application should open | | | As Expected | | | | | | Pass | | |
| **2** | Enter the command  moveto x, y  rectangle 50, 30 | | Text can be entered | | | As Expected | | | | | | Pass | | |
| **3** | Enter the command Run | | Drawing position has changed and rectangle will be drawn | | | As expected | | | | | | Pass | | |
| **4** | Enter the command Reset | | The command entered to generate shape will be cleared | | | As expected | | | | | | Pass | | |

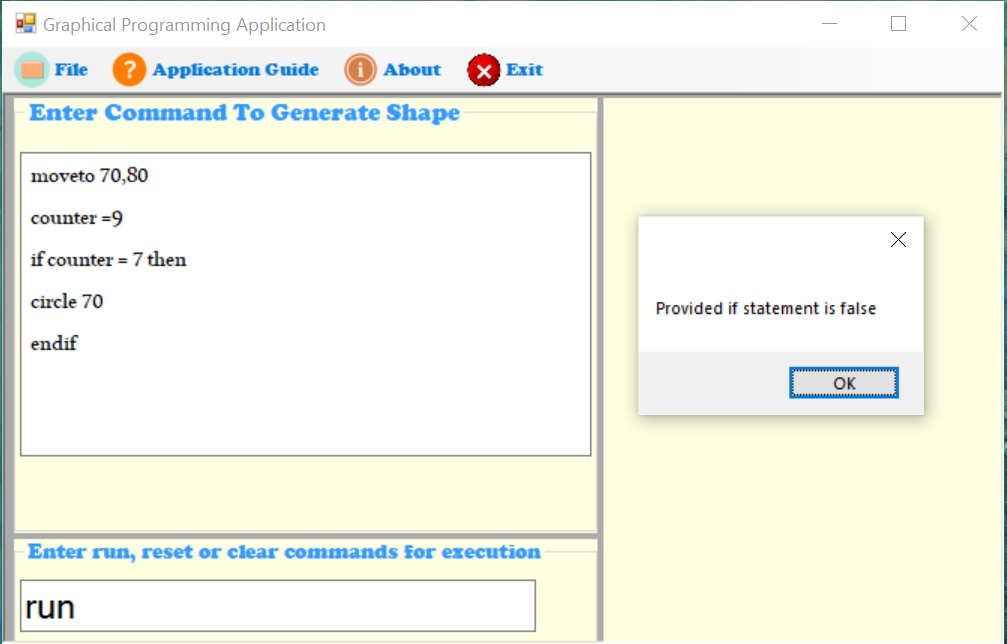


|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Test Case ID** | | | | 08 | **Test Case Description** | | | Test the Functionality in Graphical Programming Application | | | | | | |
| **Created By** | | | | Sudeep | **Reviewed By** | | | The British College | | **Version** | | | 1.0.1 | |
|  | |  | |  |  | |  |  |  |  |  | |  |  |
| **QA Tester’s Log** | | | | Review comments from British college | | | | |  |  |  | |  |  |
|  | |  | |  |  | |  |  |  |  |  | |  |  |
| **Tester’s Name** | | | | Aayush | **Date Tested** | | | 7-december -2020 | | **Test Case (Pass/Fail/NOT)** | | | Pass | |
|  | |  | |  |  | |  |  |  |  |  | |  |  |
| **S#** | | **Prerequisites** | | | | |  | **S#** | | **Test Data** | | | | |
|  | |  | | | | |  | 1. | | moveto 90,90  radi = 50  loop 5  counter = 7  counter = 7 then  circle radius  endloop  endif | | | | |
|  | |  | | | | |  |  | |  | | | | |
|  | |  | | | | |  |  | |  | | | | |
| **Test Scenario** | | Verifying on entering invalid command, the error message will be displayed. | | | | | | | |  |  | |  |  |
|  | |  | |  |  | |  |  |  |  |  | |  |  |
| **Step #** | **Step Details** | | **Expected Results** | | | **Actual Results** | | | | | | **Pass/Fail/Not executed/suspended** | | |
| **1** | Open Graphical programming application | | Application should open | | | As Expected | | | | | | Pass | | |
| **2** | Enter the command moveto 90,90  radi = 50  loop 5  counter = 7  counter = 7 then  circle radius  endloop  endif | | Text can be entered | | | As Expected | | | | | | Pass | | |
| **3** | Enter the command Run | | Error message will be shown checking on each line where the errors are present | | | As expected | | | | | | Pass | | |

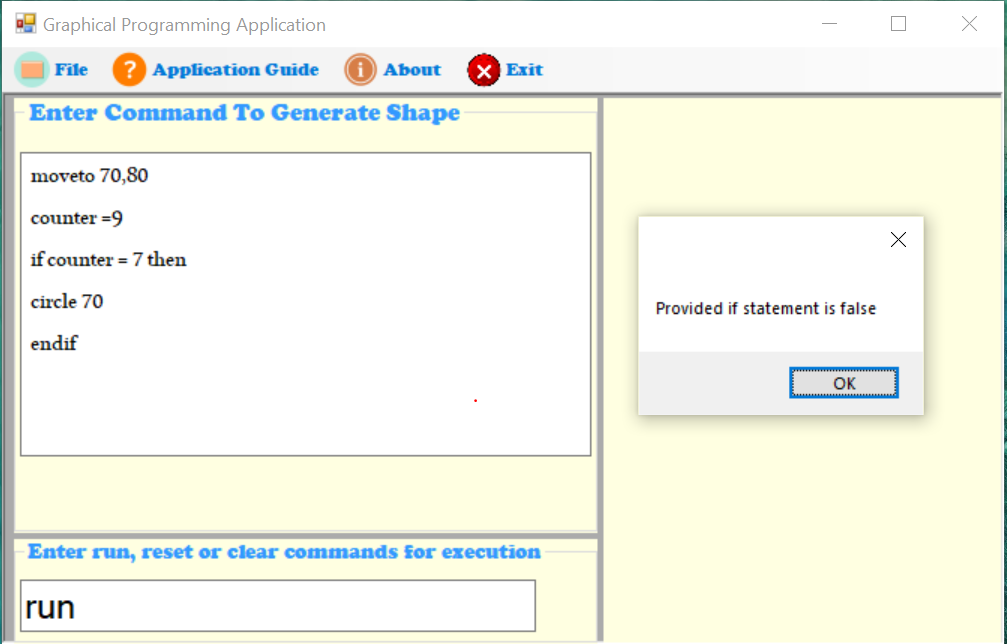




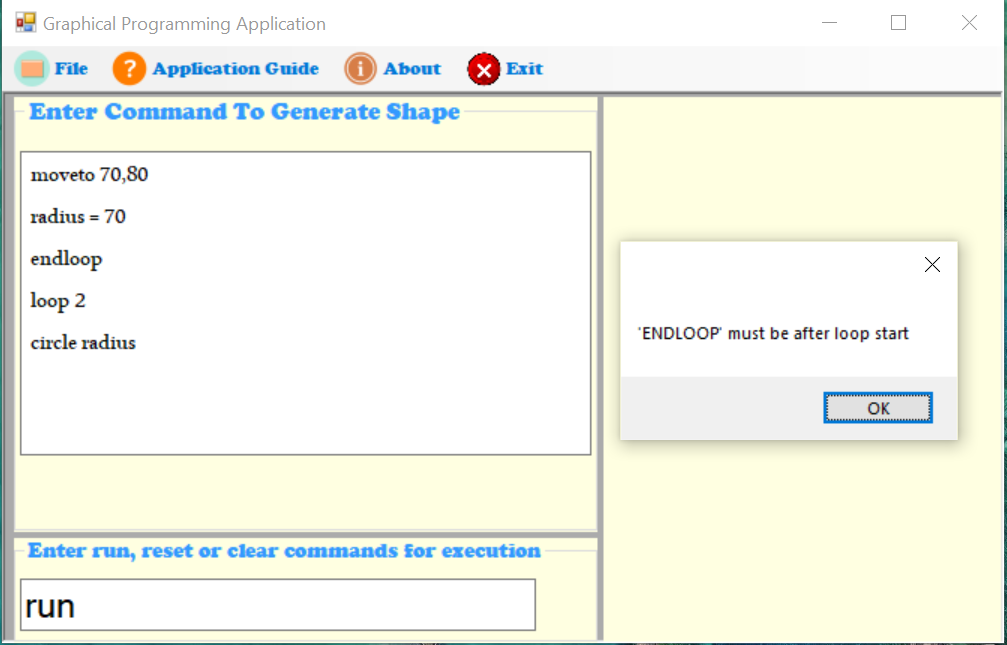




|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Test Case ID** | | | | 09 | **Test Case Description** | | | Test the Functionality in Graphical Programming Application | | | | | | |
| **Created By** | | | | Sudeep | **Reviewed By** | | | The British College | | **Version** | | | 1.0.1 | |
|  | |  | |  |  | |  |  |  |  |  | |  |  |
| **QA Tester’s Log** | | | | Review comments from British college | | | | |  |  |  | |  |  |
|  | |  | |  |  | |  |  |  |  |  | |  |  |
| **Tester’s Name** | | | | Subham | **Date Tested** | | | 7-december -2020 | | **Test Case (Pass/Fail/NOT)** | | | Pass | |
|  | |  | |  |  | |  |  |  |  |  | |  |  |
| **S#** | | **Prerequisites** | | | | |  | **S#** | | **Test Data** | | | | |
|  | |  | | | | |  | 1. | | moveto 70,80  counter =9  if counter = 7 then  circle 70  endif | | | | |
|  | |  | | | | |  |  | |  | | | | |
|  | |  | | | | |  |  | |  | | | | |
| **Test Scenario** | | Display of error message when if statement is false | | | | | | | |  |  | |  |  |
|  | |  | |  |  | |  |  |  |  |  | |  |  |
| **Step #** | **Step Details** | | **Expected Results** | | | **Actual Results** | | | | | | **Pass/Fail/Not executed/suspended** | | |
| **1** | Open Graphical programming application | | Application should open | | | As Expected | | | | | | Pass | | |
| **2** | Enter the command moveto 70,80  counter =9  if counter = 7 then  circle 70  endif | | Text can be entered | | | As Expected | | | | | | Pass | | |
| **3** | Enter the command Run | | Error message will be shown that provided statement is false | | | As expected | | | | | | Pass | | |



|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Test Case ID** | | | | 10 | **Test Case Description** | | | Test the Functionality in Graphical Programming Application | | | | | | |
| **Created By** | | | | Sudeep | **Reviewed By** | | | The British College | | **Version** | | | 1.0.1 | |
|  | |  | |  |  | |  |  |  |  |  | |  |  |
| **QA Tester’s Log** | | | | Review comments from British college | | | | |  |  |  | |  |  |
|  | |  | |  |  | |  |  |  |  |  | |  |  |
| **Tester’s Name** | | | | Asmita | **Date Tested** | | | 7-december -2020 | | **Test Case (Pass/Fail/NOT)** | | | Pass | |
|  | |  | |  |  | |  |  |  |  |  | |  |  |
| **S#** | | **Prerequisites** | | | | |  | **S#** | | **Test Data** | | | | |
|  | |  | | | | |  | 1. | | moveto 70,80  radius = 70  endloop  loop 2  circle radius | | | | |
|  | |  | | | | |  |  | |  | | | | |
|  | |  | | | | |  |  | |  | | | | |
| **Test Scenario** | | Syntax checking and display of error when validation is not fulfilled | | | | | | | |  |  | |  |  |
|  | |  | |  |  | |  |  |  |  |  | |  |  |
| **Step #** | **Step Details** | | **Expected Results** | | | **Actual Results** | | | | | | **Pass/Fail/Not executed/suspended** | | |
| **1** | Open Graphical programming application | | Application should open | | | As Expected | | | | | | Pass | | |
| **2** | Enter the command  moveto 70,80  radius = 70  endloop  loop 2  circle radius | | Text can be entered | | | As Expected | | | | | | Pass | | |
| **3** | Enter the command Run | | Error message will be shown that endloop must be after loop start | | | As expected | | | | | | Pass | | |



### Codes used in GPL application

1. **For interface (IShapes.cs)**

using System;

using System.Collections.Generic;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace GPLApplication

{

/// <summary>

/// to provide shape of objects

/// </summary>

public interface IShapes

{

/// <summary>

/// used to set color and values of the shapes

/// </summary>

/// <param name="c">define the color</param>

/// <param name="list">list of parameters that will be passed inside the function</param>

void set(Color c, params int[] list);

/// <summary>

/// used to draw shape of the object

/// </summary>

/// <param name="g"></param>

void draw(Graphics g);

}

}

1. **For class (Shapes.cs)**

using System;

using System.Collections.Generic;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace GPLApplication

{

/// <summary>

/// Provide shape of the object

/// </summary>

abstract class Shapes : IShapes

{

protected Color colour;

protected int x, y;

public Shapes()

{

colour = Color.Red;

x = y = 100;

}

/// <summary>

/// Set color, x-axis and y-axis values

/// </summary>

/// <param name="colour"></param>

/// <param name="x"></param>

/// <param name="y"></param>

public Shapes(Color colour, int x, int y)

{

this.colour = colour;

this.x = x;

this.y = y;

}

/// <summary>

/// Used as drawing shape of any object

/// </summary>

/// <param name="g"></param>

public abstract void draw(Graphics g);

/// <summary>

/// Used to set values for different shapes

/// </summary>

/// <param name="colour"></param>

/// <param name="list"></param>

public virtual void set(Color colour, params int[] list)

{

this.colour = colour;

this.x = list[0];

this.y = list[1];

}

/// <summary>

/// Used to overwrite the values

/// </summary>

/// <returns></returns>

public override string ToString()

{

return base.ToString() + " " + this.x + "," + this.y + " : ";

}

}

}

1. **For class (ShapesCreator.cs)**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace GPLApplication

{

/// <summary>

/// Declare class name as ShapesCreator

/// </summary>

abstract class ShapesCreator

{

/// <summary>

/// Used to pass shape of any objects

/// </summary>

/// <param name="ShapeType">Shape parameter</param>

/// <returns></returns>

public abstract IShapes getShape(string ShapeType);

}

}

1. **For rectangle (Rectangle.cs)**

using System;

using System.Collections.Generic;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace GPLApplication

{

/// <summary>

/// Class is declared as rectangle and interface is implemented

/// </summary>

public class Rectangle : IShapes

{

/// <summary>

/// Ineteger values for x-axis, y-axis, width and height of the rectangle

/// </summary>

public int x, y, width, height;

/// <summary>

/// Provides width and height of the rectangle

/// </summary>

public Rectangle() : base()

{

width = 0;

height = 0;

}

/// <summary>

/// Pass integer values of x-axis, y-axis, width and height for the rectangle

/// </summary>

/// <param name="x">X-axis</param>

/// <param name="y">Y-axis</param>

/// <param name="width">Rectangle's width</param>

/// <param name="height">Rectangle's height</param>

public Rectangle(int x, int y, int width, int height)

{

this.width = width;

this.height = height;

}

/// <summary>

/// Draw rectangular shape

/// </summary>

/// <param name="g"></param>

public void draw(Graphics g)

{

try

{

Pen p = new Pen(Color.Red, 2);

//SolidBrush b = new SolidBrush(Color.Aquamarine);

//g.FillRectangle(b, x, y, width, height);

g.DrawRectangle(p, x - (width / 2), y - (height / 2), width \* 2, height \* 2);

}

catch (Exception ex)

{

// throw ex;

MessageBox.Show(ex.Message);

}

}

/// <summary>

/// Set values of x-axis, y-axis, height and width

/// </summary>

/// <param name="c">Color</param>

/// <param name="list">List of parameters</param>

public void set(Color c, params int[] list)

{

try

{

this.x = list[0];

this.y = list[1];

this.width = list[2];

this.height = list[3];

}

catch (Exception ex)

{

// throw ex;

MessageBox.Show(ex.Message);

}

}

}

}

1. **For triangle (Triangle.cs)**

using System;

using System.Collections.Generic;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace GPLApplication

{

/// <summary>

/// Declare class as triangle and implement the interface

/// </summary>

public class Triangle : IShapes

{

/// <summary>

/// Inetger values for the triangle sides

/// </summary>

public int xcordinate1, ycordinate1, xcordinate2, ycordinate2, xcordinate3, ycordinate3, xcordinate4, ycordinate4, xcordinate5, ycordinate5, xcordinate6, ycordinate6;

/// <summary>

/// Draw the triangular shape as per the sides provided

/// </summary>

/// <param name="g"></param>

public void draw(Graphics g)

{

try

{

Pen p = new Pen(Color.RosyBrown, 2);

g.DrawLine(p, xcordinate1, ycordinate1, xcordinate2, ycordinate2);

g.DrawLine(p, xcordinate3, ycordinate3, xcordinate4, ycordinate4);

g.DrawLine(p, xcordinate5, ycordinate5, xcordinate6, ycordinate6);

}

catch (Exception ex)

{

// Throw appropraite error message

//throw ex;

MessageBox.Show(ex.Message);

}

}

/// <summary>

/// Set values for different sides of the traingle

/// </summary>

/// <param name="c">Color</param>

/// <param name="list">List of parameters</param>

public void set(Color c, params int[] list)

{

this.xcordinate1 = list[0];

this.ycordinate1 = list[1];

this.xcordinate2 = list[2];

this.ycordinate2 = list[3];

this.xcordinate3 = list[4];

this.ycordinate3 = list[5];

this.xcordinate4 = list[6];

this.ycordinate4 = list[7];

this.xcordinate5 = list[8];

this.ycordinate5 = list[9];

this.xcordinate6 = list[10];

this.ycordinate6 = list[11];

}

}

}

1. **For circle (Circle.cs)**

using System;

using System.Collections.Generic;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace GPLApplication

{

/// <summary>

/// holds commands of the class: circle

/// </summary>

public class Circle : IShapes

{

/// <summary>

/// used to get integer values for circle

/// </summary>

public int x, y, radius;

public Circle() : base()

{

}

/// <summary>

/// used to pass values of the circle

/// </summary>

/// <param name="x">used to set x-cordinate value</param>

/// <param name="y">used to set y-cordinate value</param>

/// <param name="radius">used to set radius value for circle</param>

public Circle(int x, int y, int radius)

{

this.radius = radius;

}

/// <summary>

/// used to draw circle on the output panel

/// </summary>

/// <param name="g"></param>

public void draw(Graphics g)

{

try

{

Pen p = new Pen(Color.Aquamarine, 2);

g.DrawEllipse(p, x - radius, y - radius, radius \* 2, radius \* 2);

}

catch (Exception ex)

{

//throw ex;

MessageBox.Show(ex.Message);

}

}

/// <summary>

/// used to define color and set values for x, y and radius

/// </summary>

/// <param name="c">used to define color for circle</param>

/// <param name="list">list of parameters that will be passed inside the function</param>

public void set(Color c, params int[] list)

{

try

{

this.x = list[0];

this.y = list[1];

this.radius = list[2];

}

catch (Exception ex)

{

//throw ex;

MessageBox.Show(ex.Message);

}

}

}

}

1. **For form (MainForm.cs)**

using System;

using System.Collections;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.IO;

using System.Linq;

using System.Text;

using System.Text.RegularExpressions;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace GPLApplication

{

public partial class MainForm : Form

{

Graphics g;

CommandValidation cvalidate;

/// <summary>

/// Load all the program logic

/// </summary>

public MainForm()

{

InitializeComponent();

g = Output\_Box.CreateGraphics();

}

/// <summary>

/// variables to generate different shapes through the commands provided by the users

/// </summary>

ShapesCreator factory = new ShapesFactory();

Pen myPen = new Pen(Color.Red);

public Color newcolor;

int x = 0, y = 0;

int counterLoop;

public int counter = 0;

public int dgSize = 0;

public int radius = 0;

public int width = 0;

public int height = 0;

/// <summary>

/// used to save the ruuning commands in the specific drive as txt file

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void saveToolStripMenuItem\_Click(object sender, EventArgs e)

{

SaveFileDialog save = new SaveFileDialog();

save.Filter = "TXT files (.txt)|\*.txt|All files (.\*)|\*.\*";

if (save.ShowDialog() == DialogResult.OK)

{

StreamWriter write = new StreamWriter(File.Create(save.FileName));

write.WriteLine(txt\_Command\_Box.Text);

write.Close();

MessageBox.Show("File has been saved successfully!");

}

}

/// <summary>

/// used for loading the commands that were saved in the txt files

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void importToolStripMenuItem\_Click(object sender, EventArgs e)

{

Stream myStream = null;

OpenFileDialog openFileDialog1 = new OpenFileDialog();

openFileDialog1.Title = "Browse file from specified folder";

openFileDialog1.InitialDirectory = "c:\\";

openFileDialog1.Filter = "TXT files (.txt)|\*.txt|All files (.\*)|\*.\*";

openFileDialog1.Filter = "DOCX files (.docx)|\*.docx|All files (.\*)|\*.\*";

openFileDialog1.FilterIndex = 2;

openFileDialog1.RestoreDirectory = true;

//Browse .txt file from computer

if (openFileDialog1.ShowDialog() == DialogResult.OK)

{

try

{

if ((myStream = openFileDialog1.OpenFile()) != null)

{

using (myStream)

{

// Insert code to read the stream here.

}

}

}

catch (Exception ex)

{

MessageBox.Show("Error: Could not read file from disk. Original error: " + ex.Message);

}

//displays the text inside the file on TextBox named as txtInput

txt\_Command\_Box.Text = File.ReadAllText(openFileDialog1.FileName);

}

}

/// <summary>

/// used to close the application

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void closeToolStripMenuItem\_Click(object sender, EventArgs e)

{

Application.Exit();

}

/// <summary>

/// used to provide information about the application

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void aboutToolStripMenuItem\_Click(object sender, EventArgs e)

{

string des = "GPL Application. Version 1.1.0-Sudeep Chaulagain";

MessageBox.Show(des);

}

private void MainForm\_Load(object sender, EventArgs e)

{

}

/// <summary>

/// used to provide guidelines about the application and commands that can be used in the application

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void applicationGuideToolStripMenuItem\_Click(object sender, EventArgs e)

{

try

{

System.Diagnostics.Process.Start("file:///D:/GPLApplicationCommands.pdf");

}

catch (Exception ex)

{

MessageBox.Show("Error: Could not read file from disk." + ex.Message);

}

}

/// <summary>

/// check the validation and run the program logic as per the commands provided in the execution box

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void txt\_Execution\_Box\_TextChanged(object sender, EventArgs e)

{

if (txt\_Execution\_Box.Text.ToLower().Trim() == "run")

{

if (txt\_Command\_Box.Text != null && txt\_Command\_Box.Text != "")

{

cvalidate = new CommandValidation(txt\_Command\_Box);

if (!cvalidate.isSomethingInvalid)

{

commandLoad();

}

}

}

else

{

if (txt\_Execution\_Box.Text.ToLower().Trim() == "clear")

{

Output\_Box.Invalidate();

}

else if (txt\_Execution\_Box.Text.ToLower().Trim() == "reset")

{

txt\_Command\_Box.Clear();

}

}

}

/// <summary>

/// used to load the commands in the output panel

/// </summary>

private void commandLoad()

{

Graphics g = Output\_Box.CreateGraphics();

string command = txt\_Command\_Box.Text.ToLower();

string[] commandline = command.Split(new String[] { "\n" },

StringSplitOptions.RemoveEmptyEntries);

int numberOfLines = txt\_Command\_Box.Lines.Length;

for (int k = 0; k < commandline.Length; k++)

{

string[] cmd = commandline[k].Split(' ');

if (cmd[0].Equals("moveto") == true)

{

Output\_Box.Refresh();

string[] param = cmd[1].Split(',');

if (param.Length != 2)

{

MessageBox.Show("Incorrect Parameter");

}

else

{

Int32.TryParse(param[0], out x);

Int32.TryParse(param[1], out y);

moveTo(x, y);

}

}

for (counterLoop = 0; counterLoop < numberOfLines; counterLoop++)

{

String oneLineCommand = txt\_Command\_Box.Lines[counterLoop];

oneLineCommand = oneLineCommand.Trim();

if (!oneLineCommand.Equals(""))

{

commandRun(oneLineCommand);

}

}

}

}

/// <summary>

/// the codes are executed as per the users input after run command is provided in the execution box

/// </summary>

/// <param name="oneLineCommand"></param>

private void commandRun(String oneLineCommand)

{

Boolean hasPlus = oneLineCommand.Contains("+");

Boolean hasEquals = oneLineCommand.Contains("=");

if (hasEquals)

{

oneLineCommand = Regex.Replace(oneLineCommand, @"\s+", " ");

string[] cmd = oneLineCommand.Split(' ');

for (int i = 0; i < cmd.Length; i++)

{

cmd[i] = cmd[i].Trim();

}

String firstWord = cmd[0].ToLower();

if (firstWord.Equals("if"))

{

Boolean loop = false;

if (cmd[1].ToLower().Equals("radius"))

{

if (radius == int.Parse(cmd[3]))

{

loop = true;

}

}

else if (cmd[1].ToLower().Equals("width"))

{

if (width == int.Parse(cmd[3]))

{

loop = true;

}

}

else if (cmd[1].ToLower().Equals("height"))

{

if (height == int.Parse(cmd[3]))

{

loop = true;

}

}

else if (cmd[1].ToLower().Equals("counter"))

{

if (counter == int.Parse(cmd[3]))

{

loop = true;

}

}

int ifStartLine = (getIfStartLineNumber());

int ifEndLine = (getEndifEndLineNumber() - 1);

counterLoop = ifEndLine;

if (loop)

{

for (int j = ifStartLine; j <= ifEndLine; j++)

{

string oneLineCommand1 = txt\_Command\_Box.Lines[j];

oneLineCommand1 = oneLineCommand1.Trim();

if (!oneLineCommand1.Equals(""))

{

commandRun(oneLineCommand1);

}

}

}

else

{

MessageBox.Show("Provided if statement is false");

}

}

else

{

string[] cmd2 = oneLineCommand.Split('=');

for (int j = 0; j < cmd2.Length; j++)

{

cmd2[j] = cmd2[j].Trim();

}

if (cmd2[0].ToLower().Equals("radius"))

{

radius = int.Parse(cmd2[1]);

}

else if (cmd2[0].ToLower().Equals("width"))

{

width = int.Parse(cmd2[1]);

}

else if (cmd2[0].ToLower().Equals("height"))

{

height = int.Parse(cmd2[1]);

}

else if (cmd2[0].ToLower().Equals("counter"))

{

counter = int.Parse(cmd2[1]);

}

}

}

else if (hasPlus)

{

oneLineCommand = System.Text.RegularExpressions.Regex.Replace(oneLineCommand, @"\s+", " ");

string[] cmd = oneLineCommand.Split(' ');

if (cmd[0].ToLower().Equals("repeat"))

{

counter = int.Parse(cmd[1]);

if (cmd[2].ToLower().Equals("circle"))

{

int increaseValue = getSize(oneLineCommand);

radius = increaseValue;

for (int j = 0; j < counter; j++)

{

drawCircle(radius);

radius += increaseValue;

}

}

else if (cmd[2].ToLower().Equals("rectangle"))

{

int increaseValue = getSize(oneLineCommand);

dgSize = increaseValue;

for (int j = 0; j < counter; j++)

{

drawRectangle(dgSize, dgSize);

dgSize += increaseValue;

}

}

else if (cmd[2].ToLower().Equals("triangle"))

{

int increaseValue = getSize(oneLineCommand);

dgSize = increaseValue;

for (int j = 0; j < counter; j++)

{

drawTriangle(dgSize, dgSize, dgSize);

dgSize += increaseValue;

}

}

}

else

{

string[] cmd2 = oneLineCommand.Split('+');

for (int j = 0; j < cmd2.Length; j++)

{

cmd2[j] = cmd2[j].Trim();

}

if (cmd2[0].ToLower().Equals("radius"))

{

radius += int.Parse(cmd2[1]);

}

else if (cmd2[0].ToLower().Equals("width"))

{

width += int.Parse(cmd2[1]);

}

else if (cmd2[0].ToLower().Equals("height"))

{

height += int.Parse(cmd2[1]);

}

}

}

else

{

generateDrawCommand(oneLineCommand);

}

}

/// <summary>

/// used for returning the size of the shapes as per the commands provided

/// </summary>

/// <param name="lineCommand"></param>

/// <returns></returns>

private int getSize(string lineCommand)

{

int value = 0;

if (lineCommand.ToLower().Contains("radius"))

{

int pos = (lineCommand.IndexOf("radius") + 6);

int size = lineCommand.Length;

String tempLine = lineCommand.Substring(pos, (size - pos));

tempLine = tempLine.Trim();

String newTempLine = tempLine.Substring(1, (tempLine.Length - 1));

newTempLine = newTempLine.Trim();

value = int.Parse(newTempLine);

}

else if (lineCommand.ToLower().Contains("size"))

{

int pos = (lineCommand.IndexOf("size") + 4);

int size = lineCommand.Length;

String tempLine = lineCommand.Substring(pos, (size - pos));

tempLine = tempLine.Trim();

String newTempLine = tempLine.Substring(1, (tempLine.Length - 1));

newTempLine = newTempLine.Trim();

value = int.Parse(newTempLine);

}

return value;

}

/// <summary>

/// Initiate the shapes as the per the commands given by the user in command box

/// </summary>

/// <param name="lineOfCommand"></param>

private void generateDrawCommand(string lineOfCommand)

{

String[] shapes = { "circle", "rectangle", "triangle" };

String[] variable = { "radius", "width", "height", "counter", "size" };

lineOfCommand = System.Text.RegularExpressions.Regex.Replace(lineOfCommand, @"\s+", " ");

string[] cmd = lineOfCommand.Split(' ');

for (int i = 0; i < cmd.Length; i++)

{

cmd[i] = cmd[i].Trim();

}

String firstWord = cmd[0].ToLower();

Boolean firstcmdhape = shapes.Contains(firstWord);

if (firstcmdhape)

{

if (firstWord.Equals("circle"))

{

Boolean secondWordIsVariable = variable.Contains(cmd[1].ToLower());

if (secondWordIsVariable)

{

if (cmd[1].ToLower().Equals("radius"))

{

drawCircle(radius);

}

}

else

{

drawCircle(Int32.Parse(cmd[1]));

}

}

else if (firstWord.Equals("rectangle"))

{

String args = lineOfCommand.Substring(9, (lineOfCommand.Length - 9));

String[] parms = args.Split(',');

for (int i = 0; i < parms.Length; i++)

{

parms[i] = parms[i].Trim();

}

Boolean secondWordIsVariable = variable.Contains(parms[0].ToLower());

Boolean thirdWordIsVariable = variable.Contains(parms[1].ToLower());

if (secondWordIsVariable)

{

if (thirdWordIsVariable)

{

drawRectangle(width, height);

}

else

{

drawRectangle(width, Int32.Parse(parms[1]));

}

}

else

{

if (thirdWordIsVariable)

{

drawRectangle(Int32.Parse(parms[0]), height);

}

else

{

drawRectangle(Int32.Parse(parms[0]), Int32.Parse(parms[1]));

}

}

}

else if (firstWord.Equals("triangle"))

{

String args = lineOfCommand.Substring(8, (lineOfCommand.Length - 8));

String[] parms = args.Split(',');

for (int i = 0; i < parms.Length; i++)

{

parms[i] = parms[i].Trim();

}

drawTriangle(Int32.Parse(parms[0]), Int32.Parse(parms[1]), Int32.Parse(parms[2]));

}

}

else

{

if (firstWord.Equals("loop"))

{

counter = int.Parse(cmd[1]);

int loopStartLine = (getLoopStartLineNumber());

int loopEndLine = (getLoopEndLineNumber() - 1);

counterLoop = loopEndLine;

for (int i = 0; i < counter; i++)

{

for (int j = loopStartLine; j <= loopEndLine; j++)

{

String oneLineCommand = txt\_Command\_Box.Lines[j];

oneLineCommand = oneLineCommand.Trim();

if (!oneLineCommand.Equals(""))

{

commandRun(oneLineCommand);

}

}

}

}

else if (firstWord.Equals("if"))

{

Boolean loop = false;

if (cmd[1].ToLower().Equals("radius"))

{

if (radius == int.Parse(cmd[1]))

{

loop = true;

}

}

else if (cmd[1].ToLower().Equals("width"))

{

if (width == int.Parse(cmd[1]))

{

loop = true;

}

}

else if (cmd[1].ToLower().Equals("height"))

{

if (height == int.Parse(cmd[1]))

{

loop = true;

}

}

else if (cmd[1].ToLower().Equals("counter"))

{

if (counter == int.Parse(cmd[1]))

{

loop = true;

}

}

int ifStartLine = (getIfStartLineNumber());

int ifEndLine = (getEndifEndLineNumber() - 1);

counterLoop = ifEndLine;

if (loop)

{

for (int j = ifStartLine; j <= ifEndLine; j++)

{

String oneLineCommand = txt\_Command\_Box.Lines[j];

oneLineCommand = oneLineCommand.Trim();

if (!oneLineCommand.Equals(""))

{

commandRun(oneLineCommand);

}

}

}

}

}

}

/// <summary>

/// Initiate whether if statement is present in the commands given in the command box

/// </summary>

/// <returns></returns>

private int getIfStartLineNumber()

{

int numberOfLines = txt\_Command\_Box.Lines.Length;

int lineNum = 0;

for (int i = 0; i < numberOfLines; i++)

{

String oneLineCommand = txt\_Command\_Box.Lines[i];

oneLineCommand = Regex.Replace(oneLineCommand, @"\s+", " ");

string[] cmd = oneLineCommand.Split(' ');

//removing white spaces in between cmd

for (int j = 0; j < cmd.Length; j++)

{

cmd[j] = cmd[j].Trim();

}

String firstWord = cmd[0].ToLower();

oneLineCommand = oneLineCommand.Trim();

if (firstWord.Equals("if"))

{

lineNum = i + 1;

}

}

return lineNum;

}

/// <summary>

/// Determine whether the if statment ended with the endif statement

/// </summary>

/// <returns></returns>

private int getEndifEndLineNumber()

{

int numberOfLines = txt\_Command\_Box.Lines.Length;

int lineNum = 0;

for (int i = 0; i < numberOfLines; i++)

{

String oneLineCommand = txt\_Command\_Box.Lines[i];

oneLineCommand = oneLineCommand.Trim();

if (oneLineCommand.ToLower().Equals("endif"))

{

lineNum = i + 1;

}

}

return lineNum;

}

/// <summary>

/// Initiate the loop as per the command given in the command box

/// </summary>

/// <returns></returns>

private int getLoopStartLineNumber()

{

int numberOfLines = txt\_Command\_Box.Lines.Length;

int lineNum = 0;

for (int i = 0; i < numberOfLines; i++)

{

String oneLineCommand = txt\_Command\_Box.Lines[i];

oneLineCommand = Regex.Replace(oneLineCommand, @"\s+", " ");

string[] cmd = oneLineCommand.Split(' ');

//removing white spaces in between cmd

for (int j = 0; j < cmd.Length; j++)

{

cmd[j] = cmd[j].Trim();

}

String firstWord = cmd[0].ToLower();

oneLineCommand = oneLineCommand.Trim();

if (firstWord.Equals("loop"))

{

lineNum = i + 1;

}

}

return lineNum;

}

/// <summary>

/// Determine whether the loop has ended with end looop

/// </summary>

/// <returns></returns>

private int getLoopEndLineNumber()

{

try

{

int numberOfLines = txt\_Command\_Box.Lines.Length;

int lineNum = 0;

for (int i = 0; i < numberOfLines; i++)

{

String oneLineCommand = txt\_Command\_Box.Lines[i];

oneLineCommand = oneLineCommand.Trim();

if (oneLineCommand.ToLower().Equals("endloop"))

{

lineNum = i + 1;

}

}

return lineNum;

}

catch (Exception e)

{

return 0;

}

}

/// <summary>

/// Draw the rectangle as per the command provided

/// </summary>

/// <param name="width"></param>

/// <param name="height"></param>

private void drawRectangle(int width, int height)

{

Pen p = new Pen(Color.Red, 2);

g.DrawRectangle(p, x - (width / 2), y - (height / 2), width \* 2, height \* 2);

}

/// <summary>

/// Draw circle as per the command provided

/// </summary>

/// <param name="radius"></param>

private void drawCircle(int radius)

{

Pen p = new Pen(Color.Aquamarine, 2);

g.DrawEllipse(p, x - radius, y - radius, radius \* 2, radius \* 2);

}

/// <summary>

/// Draw triangle as per the command provided

/// </summary>

/// <param name="rBase"></param>

/// <param name="adj"></param>

/// <param name="hyp"></param>

private void drawTriangle(int rBase, int adj, int hyp)

{

Pen po = new Pen(Color.RosyBrown, 2);

Point[] pnt = new Point[3];

pnt[0].X = x;

pnt[0].Y = y;

pnt[1].X = x - rBase;

pnt[1].Y = y;

pnt[2].X = x;

pnt[2].Y = y - adj;

g.DrawPolygon(po, pnt);

}

/// <summary>

/// Display the values of X-axis and Y-axis

/// </summary>

/// <param name="toX"></param>

/// <param name="toY"></param>

public void moveTo(int toX, int toY)

{

x = toX;

y = toY;

}

/// <summary>

/// Draw the pen position as per the X-cordinate and Y-cordinate

/// </summary>

/// <param name="toX"></param>

/// <param name="toY"></param>

public void drawTo(int toX, int toY)

{

x = toX;

y = toY;

}

}

}

# References

Baeldung, 2018. *baeldung.* [Online]   
Available at: https://www.baeldung.com/creational-design-patterns  
[Accessed 19 May 2019].

Carr, R., 2014. *blackwasp.* [Online]   
Available at: http://www.blackwasp.co.uk/Interpreter.aspx  
[Accessed 01 June 2019].

Chaniel, R., 2018. *medium.* [Online]   
Available at: https://medium.com/@ronnieschaniel/object-oriented-design-patterns-explained-using-practical-examples-84807445b092  
[Accessed 08 June 2019].

Chapple, M., 2019. *lifewire.* [Online]   
Available at: https://www.lifewire.com/definition-of-a-schema-in-a-database-1019262  
[Accessed 02 August 2019].

Charles, 2014. *sourcemaking.* [Online]   
Available at: https://sourcemaking.com/design\_patterns  
[Accessed 18 May 2019].

Grace, 2018. *geeksforgeeks.* [Online]   
Available at: https://www.geeksforgeeks.org/unified-modeling-language-uml-introduction/  
[Accessed 24 May 2019].

Guru99, n.d. *Data warehouse architecture.* [Online]   
Available at: https://www.guru99.com/data-warehouse-architecture.html  
[Accessed 09 november 2020].

Hebb, N., 2018. *breezetree.* [Online]   
Available at: https://www.breezetree.com/articles/what-is-a-flow-chart  
[Accessed 29 May 2019].

Javatpoint, n.d. *data warehouse design approaches.* [Online]   
Available at: https://www.javatpoint.com/data-warehouse-design  
[Accessed 09 november 2020].

Joe, 2015. *javapapers.* [Online]   
Available at: https://javapapers.com/design-patterns/proxy-design-pattern/  
[Accessed 02 June 2019].

Juliee, 2015. *met data.* [Online]   
Available at: https://www.ques10.com/p/14875/what-is-the-role-of-meta-data-in-a-data-warehouse/  
[Accessed 09 november 2020].

Kramer, 2019. *c-sharpcorner.* [Online]   
Available at: https://www.c-sharpcorner.com/UploadFile/nipuntomar/behavioral-design-pattern-for-net-part-1/  
[Accessed 29 May 2019].

Lucinda, S., 2016. *study.* [Online]   
Available at: https://study.com/academy/lesson/what-is-a-data-flow-diagram.html  
[Accessed 29 May 2019].

Marie, 2019. *visualparadigm.* [Online]   
Available at: https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-use-case-diagram/  
[Accessed 24 May 2019].

Molly, 2018. *sourcemaking.* [Online]   
Available at: https://sourcemaking.com/design\_patterns/composite  
[Accessed 01 June 2019].

Nipun, T., 2018. *sharpcorner.* [Online]   
Available at: https://www.c-sharpcorner.com/UploadFile/nipuntomar/behavioral-design-pattern-for-net-part-1/  
[Accessed 01 June 2019].

Richard, C., 2017. *blackwasp.* [Online]   
Available at: http://www.blackwasp.co.uk/ChainOfResponsibility.aspx  
[Accessed 01 June 2019].

Ronnie, s., 2018. *medium.* [Online]   
Available at: https://medium.com/@ronnieschaniel/object-oriented-design-patterns-explained-using-practical-examples-84807445b092  
[Accessed 26 May 2019].

Rouse, M., 2015. *techopedia.* [Online]   
Available at: https://www.techopedia.com/definition/27489/activity-diagram  
[Accessed 28 May 2019].

Shawn, 2017. *visual paradigm.* [Online]   
Available at: https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-class-diagram/  
[Accessed 28 May 2019].

Steve, S., 2018. *microsoft.* [Online]   
Available at: https://docs.microsoft.com/en-us/aspnet/core/mvc/overview?view=aspnetcore-2.2  
[Accessed 02 June 2019].

Stitch, n.d. *ETL process.* [Online]   
Available at: https://www.stitchdata.com/etldatabase/etl-process/  
[Accessed Sunday November 2020].

Ucmerced, n.d. *Data dictionary.* [Online]   
Available at: http://library.ucmerced.edu/data-dictionaries  
[Accessed 18 november 2020].

Vijay, B., 2011. *data warehouse design approaches.* [Online]   
Available at: https://www.folkstalk.com/2011/04/data-warehouse-design-approaches.html  
[Accessed 09 november 2020].