FULL SAIL UNIVERSITY

COP2334

PG2 – LAB: GRAPHICS LIBRARY

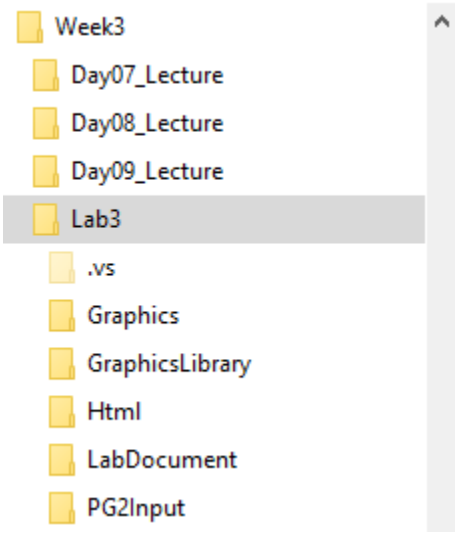
CONTENTS

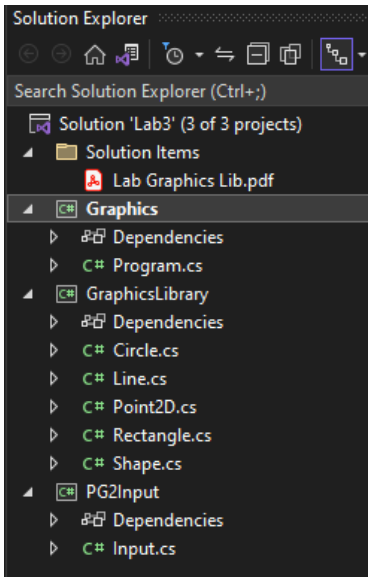
FULL SAIL UNIVERSITY

COP2334

SETUP

A **C# .NET Core console application** has been provided for you in your **GitHub repo**. **Use the provided solution.**





Lab Video

Here's a video showing what the lab could look like when completed:

https://web.microsoftstream.com/video/0781f701-2425-434f-bfe0-cb9ecb484fe7

PART A

Lecture Videos for Part A

CLASSES LECTURE:

https://fullsailedu-my.sharepoint.com/:v:/g/personal/ggirod_fullsail_com/Ed1AbYPCDttDmTHCVd2OYEIB163MblB-uek9w5YKnezucA?e=JNtR3Z

FULL SAIL UNIVERSITY

COP2334

Part A-1: Point2D struct

Tips: Structure types - C# reference | Microsoft Learn

Add a **Point2D** struct to the **GraphicsLibrary** project. An easy way to create a struct is to add a new class then change it from a class to a struct.

Add 2 fields: X and Y. The type of these fields is int. (They need to be visible outside of the struct)

Add a constructor that takes 2 int parameters. Use these to initialize X and Y.

Part A-2: Shape class

Add a **Shape** class to the **GraphicsLibrary** project.

Add **2 properties**: a **Point2D** property called **StartPt** and a **ConsoleColor** property called **Color**.

Add **2 constructors**:

- First constructor takes 2 parameters: a Point2D and a ConsoleColor. Set the properties with these parameters.
- Second constructor takes 3 parameters: 2 ints for the x and y and a ConsoleColor. Use the x and y parameters to initialize the StartPt property. Use the ConsoleColor parameter to set the Color property.

Add a **Draw** method.

- Set the background color.
- Move the cursor to the Point2D position.
- Print a space.
- Reset the color in the console.

Part A-3: Draw Shape menu

In Main, add code to case 1 of the menu switch. Clear the screen. Generate a random Point2D with an x,y anywhere in the console. Use that point to create a Shape instance with any color you want. Call Draw on the shape instance.

Example shape:

FULL SAIL UNIVERSITY

COP2334

PART B

Lecture Videos for Part B

INHERITANCE LECTURE:

https://fullsailedu-my.sharepoint.com/:v:/g/personal/ggirod_fullsail_com/EUy7jcYXBChAuTwodBxTUIoBUXQqP9MTtvDAzkzp03PHiA?e=Xv9LxH

Tips: Inheritance in C# | Microsoft Learn

POLYMORPHISM LECTURE:

https://fullsailedu-my.sharepoint.com/:v:/g/personal/ggirod_fullsail_com/EVX51bT4Ad9JqZIlHeu9FlMByySZpakxmOtbgCKDO5dRww?e=I28W7g

Tips: Polymorphism | Microsoft Learn

FULL SAIL UNIVERSITY

COP2334

Part B-1: Line class

Add a **Line** class to the **GraphicsLibrary** project. The Line class should **derive from Shape**.

Add 1 property: a **Point2D** property called **EndPt**.

Add a **constructor** that takes a start Point2D, end Point2D, and console color. Make sure to call the base constructor. Don't duplicate what the base constructors do.

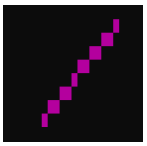
Implement the pseudocode for the **PlotLine** and **Plot** methods. (see the pseudo-code for plotLine below). NOTE: Plot is a method that moves the cursor to the x,y position and prints a space.

Override the Draw method of the Shape class (that means you need to mark the base as virtual and Line's Draw method as override). Do not call the base. Instead, set the background color, call PlotLine, then reset the color.

```
plotLine(x0, y0, x1, y1)
    dx = abs(x1 - x0)
    sx = x0 < x1 ? 1 : -1
    dy = -abs(y1 - y0)
    sy = y0 < y1 ? 1 : -1
    error = dx + dy

    while true
        plot(x0, y0)
        if x0 == x1 && y0 == y1 break
        e2 = 2 * error
        if e2 >= dy
            if x0 == x1 break
            error = error + dy
            x0 = x0 + sx
        end if
        if e2 <= dx
            if y0 == y1 break
            error = error + dx
            y0 = y0 + sy
        end if
    end while
```

In Main, add code to case 2 of the menu switch. Clear the screen. Generate 2 random Point2D points with an x,y anywhere in the console. Use those points to create a Line instance with any color you want. Call Draw on the Line instance.
Example line:



FULL SAIL UNIVERSITY

Part B-2: Rectangle class

Add a **Rectangle** class to the **GraphicsLibrary** project. The Rectangle class should **derive from Shape**.

Add 2 int properties: Width and Height.

Add 1 List<Line> field: _lines.

Add 1 constructor with the following parameters: width, height, startPt, color. Pass the startPt and color to the base constructor. Use width and height to set the properties. The constructor should create 4 lines and add them to the _lines field.

The 4 lines:

- top left to top right
- top right to bottom right
- bottom left to bottom right
- top left to bottom left

Override the Draw method of the Shape class (that means you need to mark the base as virtual and Rectangle's Draw method as override). Do not call the base. Instead, call the Draw method of each line in the _lines list.

In Main, add code to case 3 of the menu switch.

- Clear the screen.
- Generate a random Point2D point with an x,y anywhere in the console. This point will be the top-left position of the rectangle.
- Calculate a random width and height – ensure that it will NOT extend the rectangle beyond the bounds of the console.
- Use the point, width, and height to create a Rectangle instance with any color you want.
- Call Draw on the Rectangle instance.

Example output:

FULL SAIL UNIVERSITY

COP2334

Part B-3: Triangle class

Add a **Triangle** class to the **GraphicsLibrary** project. The **Triangle** class should **derive from Shape**.

Add 2 Point2D properties: P2 and P3.

Add 1 List<Line> field: _lines.

Add 1 constructor with the following parameters: p1, p2, p3, color. Pass the p1 and color to the base constructor. Use p2 and p3 to set the properties. The constructor should create 3 lines connecting the points and add them to the _lines field.

The 3 lines:

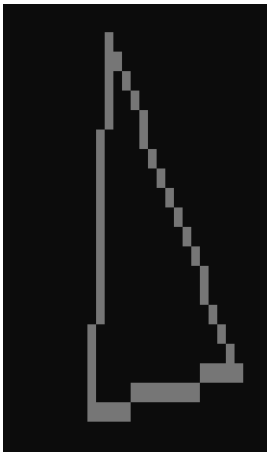
- P1 to p2
- P2 to p3
- P3 to p1

Override the Draw method of the Shape class (that means you need to mark the base as virtual and **Triangle**'s Draw method as override). Do not call the base. Instead, call the Draw method of each line in the _lines list.

In Main, add code to case 4 of the menu switch.

- Clear the screen.
- Generate 3 random Point2D points with an x,y anywhere in the console.
- Use the points to create a Triangle instance with any color you want.
- Call Draw on the Triangle instance.

Example triangle:



FULL SAIL UNIVERSITY

Part B-4: Circle class

Add a **Circle** class to the **GraphicsLibrary** project. The Circle class should **derive from Shape**.

Add 1 int property: Radius.

Add 1 constructor with the following parameters: radius, startPt, color. Pass the startPt and color to the base constructor. Use radius parameter to set the Radius property.

Implement the pseudocode for the **DrawCircle, DrawCirclePoints,** and **Plot** methods. (see the pseudo-code below). NOTE: Plot is a method that moves the cursor to the x,y position and prints a space. **Override the Draw method** of the Shape class (that means you need to mark the base as virtual and Circle's Draw method as override). Do not call the base. Instead, set the background color, call DrawCircle, then reset the color.

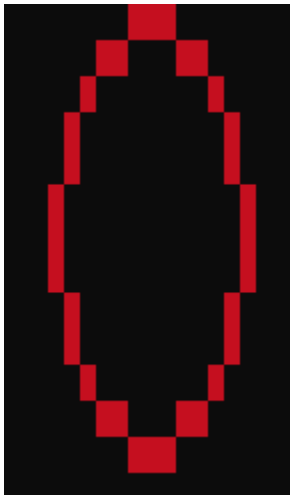
```
DrawCirclePoints(xc, yc, x, y)
    Plot(xc+x, yc+y)
    Plot(xc-x, yc+y)
    Plot(xc+x, yc-y)
    Plot(xc-x, yc-y)
    Plot(xc+y, yc+x)
    Plot(xc-y, yc+x)
    Plot(xc+y, yc-x)
    Plot(xc-y, yc-x)
end DrawCirclePoints

DrawCircle(xc, yc, r)
    x = 0, y = r
    d = 3 - 2 * r
    DrawCirclePoints(xc, yc, x, y)
    while y >= x
        x = x + 1
        if d > 0
            y = y - 1
            d = d + 4 * (x - y) + 10
        else
            d = d + 4 * x + 6
        DrawCirclePoints(xc, yc, x, y)
    end while
end DrawCircle
```

In Main, add code to case 5 of the menu switch.

- Clear the screen.
- Generate a random Point2D point with an x,y anywhere in the console. This point will be the center position of the circle.
- Calculate a random radius – ensure that it will NOT extend the circle beyond the bounds of the console.
- Use the point and radius to create a Circle instance with any color you want.
- Call Draw on the Circle instance.

Example circle:



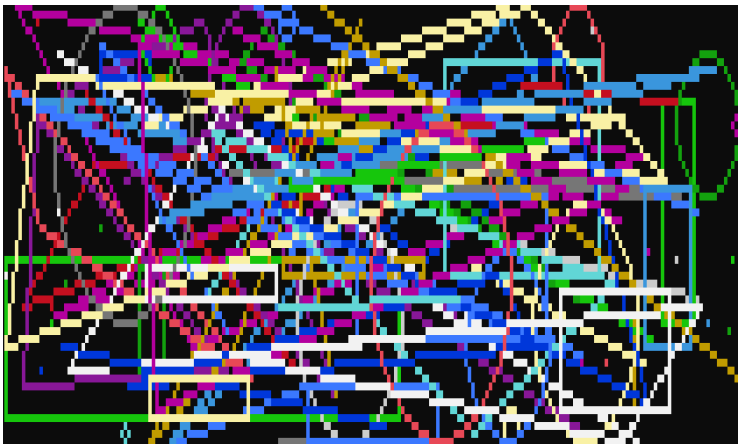
PART C

Part C-1: Random Shapes

In Main, add code to case 6 of the menu switch.

- Clear the screen.
- Create a List variable that holds Shapes.
- Create 100 random shapes and add them to the list.
    - Randomly pick which type of shape to create (Shape, Line, Rectangle, Triangle, Circle).
    - Create the instance according to its shape. (see the case statements before to create the different shapes)
- After this loop, loop over the shapes list and call Draw on each shape.

Example output:



FULL SAIL UNIVERSITY

COP2334

RUBRIC

Part A

FEATURE	POINTS
Part A-1: Point2D struct	10
Part A-2: Shape class	15
Part A-3: Draw Shape menu	10
TOTAL	35

Part B

FEATURE	POINTS
Part B-1: Line class	10
Part B-2: Rectangle class	15
Part B-3: Triangle class	15
Part B-4: Circle class	15
TOTAL	55

Part C

FEATURE	POINTS
Part C-1: Random Shapes	10
TOTAL	10

FULL SAIL UNIVERSITY

COP2334

PROGRAMMER'S CHALLENGE

As with every programmer's challenge, remember the following…

1. Do the rubric first. Make sure you have something to turn in for the assignment.
2. When attempting the challenge, don't break your other code.
3. You have other assignments so don't sacrifice them to work on the challenges.

Factory Challenge

Create a static Factory class. Add methods to randomly create each kind of shape (EX: CreateRandomShape, CreateRandomLine, CreateRandomRectangle, CreateRandomCircle). Use these methods in each of the case statements for the menu.

Extension Method Challenge

Add an extension method to the ConcoleColor to randomly generate a random color. Accept an optional bool to exclude Black as a random color option. If true, then the method should not return if the random color picked is black. Keep picking a random color until it is not black. If the optional parameter is false, then accept any random color.