**OOPDraw  
Learn the principles of OOP by writing a simple drawing program**

TEACHER’S GUIDE

Created by Richard Pawson

[Creative Commons License](https://creativecommons.org/licenses/by-sa/4.0/)  
This work is licensed under a [Creative Commons Attribution-ShareAlike 4.0 International License](https://creativecommons.org/licenses/by-sa/4.0/).

This Teacher Guide is to be used in conjunction with the Student Workbook of the same title.

## What this resource helps to teach

## How to use this resource

# Complete code after Exercise 1

### MyDrawing.cs

using Nakov.TurtleGraphics;

using System;

using System.Drawing;

namespace OOPDraw

{

public class MyDrawing

{

public static void Draw()

{

DrawSquare(0,0,Color.Blue, 100);

DrawSquare (100, 0, Color.Red, 50);

DrawCircle(125, -10, Color.Black, 10);

DrawCircle(25, -10, Color.Black, 10);

}

private static void DrawSquare(float positionX, float positionY,   
 Color lineColor, float sideLength)

{

Turtle.X = positionX;

Turtle.Y = positionY;

Turtle.PenColor = lineColor;

for (int i = 0; i < 4; i++)

{

Turtle.Forward(sideLength);

Turtle.Rotate(90);

}

}

private static void DrawCircle(float positionX, float positionY,   
 Color lineColor, float radius)

{

Turtle.X = positionX;

Turtle.Y = positionY;

Turtle.PenColor = lineColor;

for (int i = 0; i < 360; i++)

{

Turtle.Forward((float) (2 \* Math.PI \* radius /360));

Turtle.Rotate(1);

}

}

}

}

# Complete code after Exercise 2

### MyDrawing.cs

using Nakov.TurtleGraphics;

using System;

using System.Drawing;

namespace OOPDraw

{

public class MyDrawing

{

public static void Draw()

{

var body = new Square(0,0,Color.Blue, 100);

DrawSquare(body);

var cab = new Square(100, 0, Color.Red, 50);

DrawSquare(cab);

var frontWheel = new Circle(125, -10, Color.Black, 10);

DrawCircle(frontWheel);

var rearWheel = new Circle(25, -10, Color.Black, 10);

DrawCircle(rearWheel);

}

private static void DrawSquare(Square sq)

{

Turtle.X = sq.PositionX;

Turtle.Y = sq.PositionY;

Turtle.PenColor = sq.LineColor;

for (int i = 0; i < 4; i++)

{

Turtle.Forward(sq.SideLength);

Turtle.Rotate(90);

}

}

private static void DrawCircle(Circle c)

{

Turtle.X =c.PositionX;

Turtle.Y = c.PositionY;

Turtle.PenColor = c.LineColor;

for (int i = 0; i < 360; i++)

{

Turtle.Forward((float) (2 \* Math.PI \* c.Radius /360));

Turtle.Rotate(1);

}

}

}

}

### Square.cs

using System.Drawing;

namespace OOPDraw

{

public class Square

{

//Properties

public float PositionX { get; set; }

public float PositionY { get; set; }

public Color LineColor { get; set; }

public float SideLength { get; set; }

//The 'Constructor

public Square(float x, float y, Color lineColor, float sideLength)

{

LineColor = lineColor;

PositionX = x;

PositionY = y;

SideLength = sideLength;

}

}

}

### Circle.cs

using System.Drawing;

namespace OOPDraw

{

public class Circle

{

//Properties

public float PositionX { get; set; }

public float PositionY { get; set; }

public Color LineColor { get; set; }

public float Radius { get; set; }

//The 'Constructor

public Circle(float x, float y, Color lineColor, float radius)

{

LineColor = lineColor;

PositionX = x;

PositionY = y;

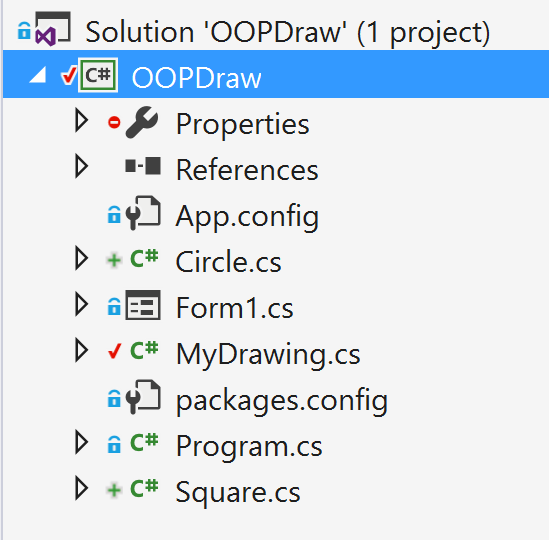
Radius = radius;

}

}

}

## Project view



# Complete code after Exercise 3

### MyDrawing

using System.Drawing;

namespace OOPDraw

{

public class MyDrawing

{

public static void Draw()

{

var body = new Square(0,0,Color.Blue, 100);

body.Draw();

var cab = new Square(100, 0, Color.Red, 50);

cab.Draw();

var frontWheel = new Circle(125, -10, Color.Black, 10);

frontWheel.Draw();

var rearWheel = new Circle(25, -10, Color.Black, 10);

rearWheel.Draw();

}

}

}

### Square

using Nakov.TurtleGraphics;

using System.Drawing;

namespace OOPDraw

{

public class Square

{

//Properties

public float PositionX { get; set; }

public float PositionY { get; set; }

public Color LineColor { get; set; }

public float SideLength { get; set; }

//The 'Constructor

public Square(float x, float y, Color lineColor, float sideLength)

{

LineColor = lineColor;

PositionX = x;

PositionY = y;

SideLength = sideLength;

}

public void Draw()

{

Turtle.X = PositionX;

Turtle.Y = PositionY;

Turtle.PenColor = LineColor;

for (int i = 0; i < 4; i++)

{

Turtle.Forward(SideLength);

Turtle.Rotate(90);

}

}

}

}

### Circle

using System.Drawing;

using Nakov.TurtleGraphics;

using System;

namespace OOPDraw

{

public class Circle

{

//Properties

public float PositionX { get; set; }

public float PositionY { get; set; }

public Color LineColor { get; set; }

public float Radius { get; set; }

//The 'Constructor

public Circle(float x, float y, Color lineColor, float radius)

{

LineColor = lineColor;

PositionX = x;

PositionY = y;

Radius = radius;

}

public void Draw()

{

Turtle.X = PositionX;

Turtle.Y = PositionY;

Turtle.PenColor = LineColor;

for (int i = 0; i < 360; i++)

{

Turtle.Forward((float)(2 \* Math.PI \* Radius / 360));

Turtle.Rotate(1);

}

}

}

}

# Square

using Nakov.TurtleGraphics;

using System.Drawing;

namespace OOPDraw

{

public class Square

{

public float PositionX { get; set; }

public float PositionY { get; set; }

public Color LineColor { get; set; }

public float SideLength { get; set; }

public Square(float x, float y, Color lineColor, float sideLength)

{

LineColor = lineColor;

PositionX = x;

PositionY = y;

SideLength = sideLength;

}

public void Draw()

{

Turtle.PenColor = LineColor;

Turtle.X = PositionX;

Turtle.Y = PositionY;

for (int i = 0; i < 4; i++)

{

Turtle.Forward(SideLength);

Turtle.Rotate(90);

}

}

}

}

# Circle

using Nakov.TurtleGraphics;

using System;

using System.Drawing;

namespace OOPDraw

{

public class Circle

{

public float PositionX { get; set; }

public float PositionY { get; set; }

public Color LineColor { get; set; }

public float Radius { get; set; }

public Circle(float x, float y, float radius, Color lineColor)

{

LineColor = lineColor;

PositionX = x;

PositionY = y;

Radius = radius;

}

public void Draw()

{

Turtle.PenColor = LineColor;

Turtle.X = PositionX;

Turtle.Y = PositionY;

for (int i = 0; i < 360; i++)

{

Turtle.Forward((float) Math.PI \* 2 \* Radius / 360);

Turtle.Rotate(1);

}

}

}

}

# Shape

using System;

using System.Collections.Generic;

using System.Drawing;

using System.Linq;

using System.Text;

namespace DrawingObjects

{

public abstract class Shape

{

public float PositionX { get; set; }

public float PositionY { get; set; }

public Color LineColor { get; set; }

public abstract void Draw();

public void MoveTo(float newX, float newY)

{

PositionX = newX;

PositionY = newY;

}

public abstract void GrowBy(float percent);

}

}