# Shape Exercise

'''

lab5.py

This program has several sub-classes that when used properly can have their respective draw functions called to draw the coresponding shapes with particular elements specified by the user.

'''

import turtle

import math

# Other imports, such as math, random, etc., as needed

class Shape(object):

'''This class is the parents class for basic elements of a shape, such as a color.'''

def \_\_init\_\_(self, t, color):

'''Initializes the color and turtle.'''

self.t = t

self.color = color

def draw(self): #Draw method does nothing for this class.

return

''' Circle class inherits Shape class'''

class Circle(Shape):

def \_\_init\_\_(self, t, radius, color, centerX, centerY):

Shape.\_\_init\_\_(self, t, color)

'''Initializes the needed remaining instance variables needed.'''

self.radius = radius

self.centerx = centerX

self.centery = centerY

def draw(self): #Overrides parent's draw method and draws a circle instead. Typical child ignoring their parents...

self.t.down()

self.t.pencolor(self.color)

self.t.up()

self.t.goto(self.centerx, self.centery)

self.t.right(180)

self.t.forward(self.radius)

self.t.right(90)

self.t.down()

distance = 2 \* math.pi \* self.radius / 120

for count in range(120):

self.t.forward(distance)

self.t.right(3)

class Rectangle(Shape):

'''Rectangle class inherits Shape class'''

def \_\_init\_\_(self, t, color, longS, shortS):

Shape.\_\_init\_\_(self, t, color)

'''Initializes the remaining instance variables needed for a sqaure'''

self.long = longS

self.short = shortS

def draw(self): #Overrides parent's draw method and draws a circle instead. Two child classes disobeying must be hard for the parent class.

self.t.pencolor(self.color)

self.t.down()

self.t.setheading(0)

self.topleftcorner = self.t.pos()

for i in range(2):

self.t.forward(self.long)

self.t.right(90)

self.t.forward(self.short)

self.t.right(90)

class Line(Shape):

'''Line calss inherits Shape class'''

def \_\_init\_\_(self, t, color, length, width, angle):

Shape.\_\_init\_\_(self, t, color)

self.length = length

self.width = width

self.angle = angle

def draw(self): #Overrides parent's draw method and draws a circle instead. This rebellion is getting ridiculous...

self.t.down()

self.t.pencolor(self.color)

self.t.width(self.width)

self.t.setheading(self.angle)

self.t.forward(self.length)

class Triangle(Shape):

'''Triangle class inherits Shape class'''

def \_\_init\_\_(self, t, color, length):

Shape.\_\_init\_\_(self, t, color)

self.length = length

def draw(self): #Overrides parent's draw method and draws a circle instead. At this point it's on the parent...

self.t.down()

self.t.setheading(60)

self.t.forward(self.length)

self.t.right(120)

self.t.forward(self.length)

self.t.right(120)

self.t.forward(self.length)

def main(): #Main function of the program

t = turtle.Turtle()

drawStickman(t)

drawHouse(t)

t.hideturtle()

turtle.done()

def drawStickman(t): #Calls the classes and functions of classes needed to draw a handsome stickman.

head = Circle(t, 20, "black", 20, 15)

body = Line(t, "black", 40, 1, 270)

arm1 = Line(t, "black", 10, 1, 30)

arm2 = Line(t, "black", 10, 1, 140)

leg1 = Line(t, "black", 20, 1, 300)

leg2 = Line(t, "black", 20, 1, 230)

head.draw()

t.up()

t.goto(20, 15)

t.setheading(270)

t.forward(20)

neck = t.pos()

body.draw()

t.up()

legpos = t.pos()

t.goto(neck)

t.forward(10)

arm1.draw()

t.up()

t.setheading(270)

t.goto(neck)

t.forward(10)

arm2.draw()

t.up()

t.goto(legpos)

leg1.draw()

t.up()

t.goto(legpos)

leg2.draw()

def drawHouse(t): #Draws a house, or as some might consider it, a countryside church building minus the cross.

t.up()

t.goto(100,0)

body = Rectangle(t, "black", 50, 40)

roof = Triangle(t, "black", 50)

body.draw()

t.up()

t.goto(100,0)

roof.draw()

t.up()

t.goto(100,0)

t.setheading(0)

t.forward(50)

t.right(90)

t.forward(40)

t.right(90)

t.forward(20)

t.right(90)

t.forward(20)

doorpos = t.pos()

door = Rectangle(t, "black", 10, 20)

t.right(90)

door.draw()

t.up()

t.forward(10)

t.right(90)

t.forward(12)

t.right(90)

t.forward(3)

circ = t.pos()

knob = Circle(t, 2, "black", circ[0], circ[1])

knob.draw()

t.up()

t.goto(doorpos)

t.setheading(180)

t.forward(20)

t.setheading(0)

winpos = t.pos()

window = Rectangle(t, "black", 5, 5)

window.draw()

t.up()

t.forward(5)

window.draw()

t.up()

t.goto(winpos)

t.setheading(270)

t.forward(5)

t.setheading(0)

window.draw()

t.up()

t.forward(5)

window.draw()

if \_\_name\_\_ == "\_\_main\_\_":

main()

1.

Graphical user interface

Description automatically generated with medium confidence

2.

Graphical user interface

Description automatically generated with medium confidence

3.

Graphical user interface

Description automatically generated with medium confidence

4.

Graphical user interface

Description automatically generated with medium confidence