Міністерство освіти і науки, молоді та спорту України

Національний технічний університет України

«Київський політехнічний інститут»

Фізико-технічний інститут

**Лабораторна робота з програмування № 9**

**Виконав:**

Студент 2 курсу

групи ФЕ-81

Адамантіс М. В.

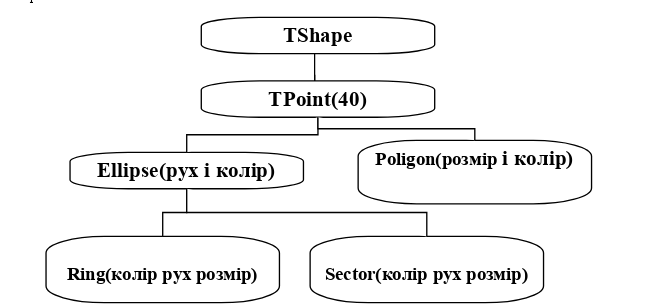
**Перевірив:**

Прогонов Д. О.

Київ 2019

**Формулювання завдання**

1. Реалізуйте програмно приклади приведені в тексті
2. Виконайте свій варіант побудови геометричних фігур за вказівками викладача



**Програмний код**

# main.py

import sys, os

from shapes\_c import \*

sys.path.append(os.path.abspath(os.path.join(os.path.dirname(\_\_file\_\_), "..")))

from inc.errorhandling import handle

ids = {

# 'id': obj,

}

shapes = {

'TShape': TShape,

'TPoint': TPoint,

'Polygon': Polygon,

'Ellipse': Ellipse,

'Ring': Ring,

'Sector': Sector

}

def print\_help():

print('Command help:')

print('\t[list / l]')

print('\t[create / c] <type> [<n>]')

print('\t[delete / d] <id>')

print('\t[inspect / i] <id>')

print('\t[move / m] <id> <x> <y>')

print('\t[paint / p] <id> <R> <G> <B>')

print('\t[scale / s] <id> <size>')

print('Available types:')

print(str([s + '\n' for s in shapes.keys()]))

print ("""

Example:

> list

>>> ...

> create TPoint

>>> Created new TPoint with id 9982

> move 9982 20 30

> inspect 9982

>>> TPoint <20, 30>

> paint 9982 255 255 255

>>> This shape does not support this method!

""")

def list\_obj():

return '\n'.join(['{0}:{1}'.format(k, v) for k, v in ids.items()])

def create\_obj(obj, n):

ans = []

if n <= 0:

raise ValueError('n must be greater than 1!')

if obj not in shapes.keys():

raise ValueError('Incorrect object name')

base = len(ids)

for i, \_ in enumerate(range(n)):

ids[str(i + base)] = shapes[obj]()

ans.append('Created new ' + obj + ' with id ' + str(i+base))

return '\n'.join(ans)

def move\_obj(id, x, y):

ids[id].set\_position(Position(x, y))

return 'New position for ' + id + ': ' + repr(ids[id].get\_position())

def paint\_obj(id, clr):

r, g, b = [x if x in range(255) else 0 for x in clr]

c = Color(r, g, b)

ids[id].set\_color(c)

return 'Painted {0} with id {1} to color {2}'.format(repr(ids[id]), id, c)

def delete\_obj(id : str):

if int(id) not in range(len(ids) - 1):

raise IndexError

ids.pop(id)

return 'Removed object with id ' + id

def scale\_obj(id, size):

ids[id].set\_size(size)

return 'The size of object ' + id + ' was succesfully changed!'

def inspect\_obj(id):

return repr(ids[id])

def interpret(cmd):

try:

action, \*params = cmd.split(' ')

obj = params.pop(0) if len(params) != 0 else None

if action in ['c', 'create']:

if len(params) == 0:

params = [1]

return create\_obj(obj, int(params[0]))

elif action in ['d', 'delete']:

return delete\_obj(obj)

elif action in ['m', 'move']:

x, y = [int(x) for x in params]

return move\_obj(obj, x, y)

elif action in ['i', 'inspect']:

return inspect\_obj(obj)

elif action in ['l', 'list']:

return list\_obj()

elif action in ['p', 'paint']:

if len(params) == 0:

params = ['0', '0', '0']

return paint\_obj(obj, (int(c) for c in params))

elif action in ['s', 'scale']:

return scale\_obj(obj, int(params[0]))

else:

raise ValueError('Incorrect keyword')

except Exception as e:

handle(e)

def main():

print\_help()

create\_obj('TPoint', 40)

print(list\_obj())

while True:

try:

cmd = input("lab9> ").strip()

if cmd != '':

result = interpret(cmd)

if result:

print(result)

except EOFError or KeyboardInterrupt:

print('Bye!')

exit(0)

except Exception as e:

handle(e)

if \_\_name\_\_ == '\_\_main\_\_':

main()

# shapes.py

class Position:

def \_\_init\_\_(self, x = 0, y = 0):

if x < 0:

x = 0

if y < 0:

y = 0

self.x = x

self.y = y

def \_\_repr\_\_(self):

return ' pos: ({0}, {1}) '.format(self.x, self.y)

class Color:

def \_\_init\_\_(self, red = 0, green = 0, blue = 0):

self.red = red

self.green = green

self.blue = blue

def setRGB(self, r, g, b):

self.red, self.green, self.blue = r, g, b

@classmethod

def from\_hex(cls, hex):

c = cls()

c.red, c.green, c.blue = tuple(int(hex[i:i+2], 16) for i in (0, 2, 4))

return c

def \_\_repr\_\_(self):

return ' color: {0} {1} {2} '.format(self.red, self.green, self.blue)

class TShape:

def \_\_init\_\_(self, position : Position = Position()):

self.position = position

def get\_position(self):

return self.position

def set\_position(self, p: Position):

self.position = p

def \_\_repr\_\_(self):

return 'TShape <' + repr(self.position) + '>'

def paint(self):

return 'This shape does not support this method!'

class TPoint(TShape):

def \_\_init\_\_(self, position : Position = Position(), size = 0):

super().\_\_init\_\_(position)

self.size = size

def get\_size(self):

return self.size

def set\_size(self, size):

self.size = abs(size)

def get\_position(self):

return super().get\_position()

def set\_position(self, p):

return super().set\_position(p)

def \_\_repr\_\_(self):

return 'TPoint < {0} size: {1} >'\

.format(repr(self.get\_position()),

str(self.get\_size())

)

class Polygon(TPoint):

def \_\_init\_\_(self, position : Position = Position(), size = 0, color : Color = Color()):

super().\_\_init\_\_(position, size)

self.color = color

def get\_color(self):

return self.color

def set\_color(self, color : Color):

self.color = color

def \_\_repr\_\_(self):

return 'Polygon < {0} size: {1} {2} >'\

.format(repr(self.get\_position()),

str(self.get\_size()),

repr(self.get\_color())

)

class Ellipse(TPoint):

def \_\_init\_\_(self, position : Position = Position(), size = 0):

super().\_\_init\_\_(position, size)

def move(self, side, distance):

x, y = self.position.cartesian()

if side == 'up':

y += distance

elif side == 'down':

y -= distance

elif side == 'left':

x -= distance

elif side == 'right':

x += distance

self.position = Position(x, y)

def \_\_repr\_\_(self):

return 'Ellipse <' + repr(self.get\_position()) + \

str(self.get\_size()) + '>'

class Ring(Ellipse):

def \_\_init\_\_(self, position : Position = Position(), size = 0, color : Color = Color()):

super().\_\_init\_\_(position, size)

self.color = color

def get\_color(self):

return self.color

def set\_color(self, color : Color):

self.color = color

def \_\_repr\_\_(self):

return 'Ring <' + repr(self.get\_position()) + \

str(self.get\_size()) + repr(self.get\_color()) + '>'

class Sector(Ellipse):

def \_\_init\_\_(self, position : Position = Position(), size = 0, color : Color = Color()):

super().\_\_init\_\_(position, size)

self.color = color

def get\_color(self):

return self.color

def set\_color(self, color : Color):

self.color = color

def \_\_repr\_\_(self):

return 'Ellipse <' + repr(self.get\_position()) + \

str(self.get\_size()) + repr(self.get\_color()) + '>'

**Приклад виконання**

