Міністерство освіти і науки України

НТУУ «Київський політехнічний інститут ім. Ігоря Сікорського»

Фізико-технічний інститут

# Програмування 4

# Лабораторна робота №9

«Побудова об’єктної ієрархії геометричних фігур»

**Виконала:**

Студентка II курсу ФТІ групи ФЕ-81

Меделян Валерія

Київ 2020

1. Завдання лабораторної роботи

**TShape**

**TPoint (50 точок змінюють колір)**

**Лінія(рух і колір, маштаб)**

**Ламана (колір рух)**

**Багатокутник(масштаб і колір)**

1. Код реалізації

|  |
| --- |
| Lab\_9.py |
| from tkinter import \*  from random import randint, choice  class TShape:  def \_\_init\_\_(self, canvas, coords):  self.coords = coords  self.canvas = canvas  self.ID = None  class TPoint(TShape):  def \_\_init\_\_(self, canvas, coords, point\_size = 1, color = 'black'):  TShape.\_\_init\_\_(self, canvas, coords)  self.color = color  self.point\_size = point\_size  def draw(self):  x, y = self.coords  self.ID = self.canvas.create\_rectangle((x, y, x+self.point\_size, y+self.point\_size),fill = self.color, outline = self.color)  def delete(self):  canvas.delete(self.ID)  def change\_color(self):  self.color = choice(['black', 'blue', 'green', 'red', 'yellow'])  self.delete()  self.draw()  class Line(TPoint):  def \_\_init\_\_(self, coords1, coords2, canvas, scale, color):  self.coords1 = coords1  self.coords2 = coords2  self.canvas = canvas  self.scale = scale  self.color = color  self.points = list()  self.get\_points()  def get\_coords(self):  dx = self.coords2[0] - self.coords1[0]  dy = self.coords2[1] - self.coords1[1]  xsign = 1 if dx > 0 else -1  ysign = 1 if dy > 0 else -1  dx = abs(dx)  dy = abs(dy)  if dx > dy:  xx, xy, yx, yy = xsign, 0, 0, ysign  else:  dx, dy = dy, dx  xx, xy, yx, yy = 0, ysign, xsign, 0  D = 2\*dy - dx  y = 0  for x in range(dx + 1):  yield self.coords1[0] + x\*xx + y\*yx, self.coords1[1] + x\*xy + y\*yy  if D >= 0:  y += 1  D -= 2\*dx  D += 2\*dy  def get\_points(self):  coords = self.get\_coords()  for coor in coords:  point = TPoint(canvas, coor, self.scale, self.color)  self.points.append(point)  def draw(self):  for p in self.points:  p.draw()  def delete(self):  for p in self.points:  p.delete()  def change\_move(self):  self.color = choice(['black', 'blue', 'green', 'red', 'yellow'])  self.coords1 = (randint(0, 500), randint(0, 500))  self.coords2 = (randint(0, 500), randint(0, 500))  self.delete()  self.points = list()  self.get\_coords()  self.get\_points()  self.draw()  def change\_scale(self):  self.scale = randint(1, 10)  self.delete()  self.points = list()  self.get\_coords()  self.get\_points()  self.draw()  class Curveline(Line):  def \_\_init\_\_(self, canvas, coords, color, scale = 1): #((x0,y0),(x1,y1)) - coords element  self.lines = list()  self.coords = coords  self.color = color  self.canvas = canvas  self.scale = scale  self.get\_lines()  def get\_lines(self):  cor2 = self.coords[0]  for i in range(1, len(self.coords)):  cor1 = cor2  cor2 = self.coords[i]  line = Line(cor1, cor2, self.canvas, self.scale, self.color)  self.lines.append(line)  def draw(self):  for line in self.lines:  line.draw()  def delete(self):  for l in self.lines:  l.delete()  def move(self):  p\_num = len(self.coords)  self.delete()  self.lines = list()  self.coords = list()  for i in range(p\_num):  new\_x, new\_y = (randint(0, 500), randint(0, 500))  self.coords.append((new\_x, new\_y))  self.coords = tuple(self.coords)  self.get\_lines()  self.draw()  def change\_color(self):  self.color = choice(['black', 'blue', 'green', 'red', 'yellow'])  self.delete()  self.lines = list()  self.get\_lines()  self.draw()  class Polygon(Curveline):  def \_\_init\_\_(self, canvas, scale, color):  self.color = color  self.canvas = canvas  self.scale = scale  self.me = None  def draw(self):  pl\_coords = ( (50, 50), (100, 50), (120, 100), (100, 120), (50, 120), (20, 70), (50, 50) )  self.me = Curveline(self.canvas, pl\_coords, self.color, self.scale)  self.me.draw()  def delete(self):  if bool(self.me):  self.me.delete()  def change\_move(self):  self.scale = randint(1, 12)  self.color = choice(['black', 'blue', 'green', 'red', 'yellow'])  self.delete()  self.draw()  root = Tk()  canvas = Canvas(root, width = 500, height = 500)  canvas.pack(fill = BOTH)  class m50\_points:  def \_\_init\_\_(self, canvas):  self.canvas = canvas  self.points = self.get\_points()  def get\_points(self):  points = list()  for i in range(0, 50):  cor = (randint(0, 500), randint(0, 500))  p = TPoint(self.canvas, cor)  points.append(p)  return points  def draw(self):  for p in self.points:  p.draw()  def color(self):  if len(self.points) == 0:  self.points = self.get\_points()  self.draw()  else:  for p in self.points:  p.change\_color()  def delete(self):  for p in self.points:  p.delete()  mp = m50\_points(canvas)  p\_b1 = Button(text = 'Points', command = mp.color)  p\_b1.pack(side = LEFT)  l = Line((10, 10), (50, 50), canvas, 1, 'blue')  l\_b1 = Button(text = 'Line move&color', command = l.change\_move)  l\_b1.pack(side = LEFT)  l\_b2 = Button(text = 'Line scale', command = l.change\_scale)  l\_b2.pack(side = LEFT)  cl\_cor = ( (10, 10), (100, 20), (40, 70), (40, 10))  cl = Curveline(canvas, cl\_cor, 'blue')  cl\_b1 = Button(text = 'Broken line move', command = cl.move)  cl\_b1.pack(side = LEFT)  cl\_b2 = Button(text = 'Broken line color', command = cl.change\_color)  cl\_b2.pack(side = LEFT)  pl = Polygon(canvas, 2, 'black')  pl\_b = Button(text = 'Polygon', command = pl.change\_move)  pl\_b.pack(side = LEFT)  def clear():  mp.delete()  l.delete()  cl.delete()  pl.delete()  clear\_b = Button(text = 'Hide all', command = clear)  clear\_b.pack(side = LEFT)  root.mainloop() |

1. Виконання програми







