LE GOUVERNEMENT
DU GRAND-DUCHÉ DE LUXEMBOURG
Ministère de l'Éducation nationale,
de l'Enfance et de la Jeunesse

EXAMEN DE FIN D'ÉTUDES SECONDAIRES CLASSIQUES

2020

CORRIGÉ – BARÈME

BRANCHE	SECTION(S)	ÉPREUVE ÉCRITE
INFORMATIQUE	**1B**	*Durée de l'épreuve :* 3h *Date de l'épreuve :*

```python
# Créé par Manch, le 12.07.2019 en Python 3.4
# imports & variables                                   2
import pygame, sys
from pygame.locals import *
from random import randint
from math import sqrt


# class Shape                                           15
class Shape:
    # 1
    def __init__(self, x, y, x_step, y_step, size, kind):
        self.x = x
        self.y = y
        self.x_step = x_step
        self.y_step = y_step
        self.size = size
        self.kind = kind

    # 3,5
    def is_inside(self, point):
        if (self.kind == 0):
            d = sqrt((self.x - point[0]) ** 2 + (self.y - point[1]) ** 2)
            return d <= self.size
        else:
            return (self.x - self.size <= point[0] <= self.x + self.size) \
                   and (self.y - self.size <= point[1] <= self.y + self.size)

    # 3
    def is_touching(self, other):
        if (self.kind == 1) and (other.kind == 1):
            return (abs(self.x - other.x) <= self.size + other.size) and (
                        abs(self.y - other.y) <= self.size + other.size)
        else:
            d = sqrt((self.x - other.x) ** 2 + (self.y - other.y) ** 2)
            return d <= self.size + other.size

    # 5
    def move(self):
        # 2
        new_size = self.size + randint(-2, 2)
        if min_size <= new_size <= max_size:
            self.size = new_size
        # 2
        new_x = self.x + self.x_step
        new_y = self.y + self.y_step
        if (new_x + self.size >= width) or (new_x - self.size < 0):
            self.x_step = -self.x_step
        if (new_y + self.size >= height) or (new_y - self.size < 0):
            self.y_step = -self.y_step
```

```
        # 1
        self.x = self.x + self.x_step
        self.y = self.y + self.y_step

    # 2,5
    def draw(self):
        x = (int)(self.x)
        y = (int)(self.y)
        r = (int)(self.size)
        if (self.kind == 0):
            pygame.draw.circle(screen, Color("Blue"), (x, y), r)
            pygame.draw.circle(screen, Color("Black"), (x, y), r, 1)
        else:
            pygame.draw.rect(screen, Color("Red"), (x - r, y - r, 2 * r, 2 * r))
            pygame.draw.rect(screen, Color("Black"), (x - r, y - r, 2 * r, 2 * r), 1)


# class Shapes                                          19
class Shapes:

    # 1
    def __init__(self):
        self.shapes_list = []

    def size(self):
        return len(self.shapes_list)

    # 3,5
    def add_random(self, x, y, radius):
        x_step = randint(-500, 500) / 100
        y_step = randint(-500, 500) / 100
        kind = randint(0, 1)
        self.shapes_list.append(Shape(x, y, x_step, y_step, radius, kind))

    # 2,5
    def reset(self):
        self.shapes_list = []
        x = randint(start_size, width - start_size)
        y = randint(start_size, height - start_size)
        self.add_random(x, y, start_size)

    # 3
    def split(self, i):
        b = self.shapes_list[i]
        r = b.size / 2
        x = b.x
        if (r >= split_size):
            self.add_random(x - r, b.y, r)
            self.add_random(x + r, b.y, r)
        del (self.shapes_list[i])

    # 6
    def get_clicked_shape(self, p):
        i = 0
        result = -1
        while (i < len(self.shapes_list)) and result == -1:
            if self.shapes_list[i].is_inside(p):
                result = i
            i = i + 1
        return result
```

```
    # 3
    def move(self):
        for shape in self.shapes_list:
            shape.move()

    def draw(self):
        for shape in self.shapes_list:
            shape.draw()


min_size = 10
max_size = 30
start_size = 40
split_size = 6
width = 500
height = 400
done = False
game_over = False
time = 60
points = 0
FPS = 25

# initialisations                                   2
pygame.init()
size = (width, height)
screen = pygame.display.set_mode(size)
fpsClock = pygame.time.Clock()
my_shapes = Shapes()
my_shapes.reset()

# boucle principale                                 22
while not done:
    # 7
    for event in pygame.event.get():
        # 1
        if event.type == QUIT:
            done = True
        # 3
        if event.type == KEYDOWN:
            if event.key == K_n:
                my_shapes.reset()
                game_over = False
                time = 60
                points = 0
        # 3
        if event.type == MOUSEBUTTONDOWN and not game_over:
            p = pygame.mouse.get_pos()
            numball = my_shapes.get_clicked_shape(p)
            if numball != -1:
                my_shapes.split(numball)
                points = points + 1

    # 6
    for i in range(my_shapes.size()):
        for j in range(i + 1, my_shapes.size()):
            shape1 = my_shapes.shapes_list[i]
            shape2 = my_shapes.shapes_list[j]
            if shape1.is_touching(shape2):
                shape1.x_step = -shape1.x_step
```

```
                shape2.x_step = -shape2.x_step
                while shape1.is_touching(shape2):
                    shape1.move()
                    shape2.move()
    # 4
    if not game_over:
        screen.fill(Color("White"))
        pygame.display.set_caption(f"Time : {(int)(time)} Score: {points}")
        my_shapes.draw()
        pygame.display.update()
        my_shapes.move()
        fpsClock.tick(FPS)
    # 3
    if (time >= 0):
        time = time - 1 / 25
    else:
        game_over = True
        pygame.display.set_caption("You lost!")

    # 2
    if my_shapes.size() == 0:
        game_over = True
        pygame.display.set_caption("You won!")

pygame.quit()
sys.exit()
```