Programmation Orientée Objets : examen écrit

Consignes générales

- Barème : ..... / 45 points
- Les temps indicatifs pour chaque question sont indiqués
- Respecter la syntaxe Python au mieux
- Si vous ne parvenez pas à voir comment coder quelque chose en Python, expliquez au mieux les étapes de résolution en pseudo-code et/ou en français.

Partie 1 : une bibliothèque de classe

Vous êtes développeur chez un éditeur de jeux vidéos qui veut développer son propre moteur graphique et vous êtes mandaté(e) pour développer une bibliothèque de classes d'après le diagramme de classe suivant

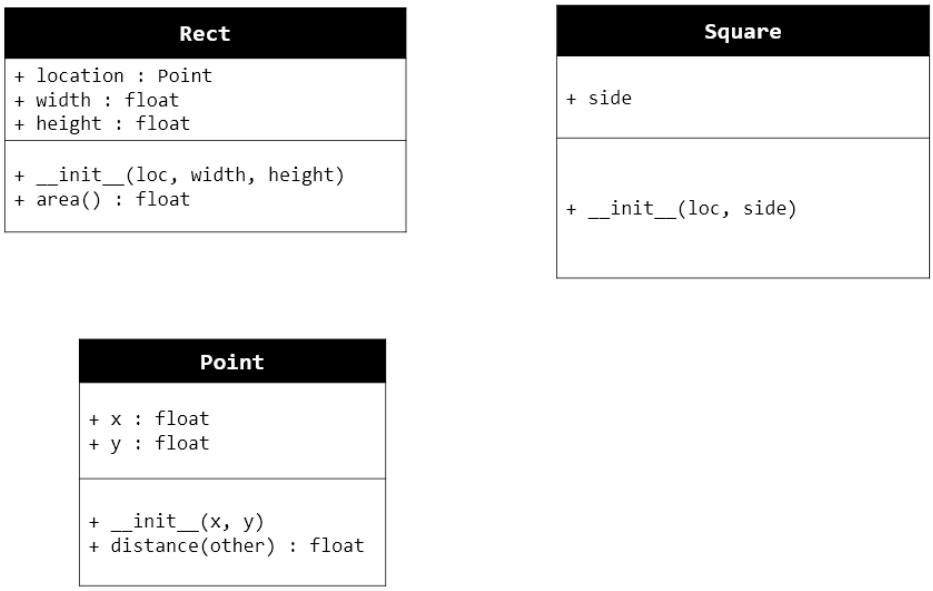

Diagramme de classes à compléter

Question 1 (... / 4 points)

Compléter le diagramme de classes ci-dessus en veillant à y intégrer les éléments suivants :

a. Héritage entre les classes `Rect` et `Square`

b. Composition entre `Rect` et `Point`. L'existence de l'attribut d'instance `Rect` est dépendante de l'existence de `Point`.

Question 2 (... / 15 points)

Fonctionnement

Vous devrez rendre la question 2 avant de pouvoir passer à la question 3 (Temps conseillé : 10 minutes)

Définir une classe `Point` représentant un point du plan cartésien dont l'utilisation est la suivante dans une session interactive IDLE :

```
>>> origine = Point(0,0)
# indique le nombre d'instances créées
>>> Point.count()
1
>>> p1 = Point(x=3, y=4)
# indique le nombre d'instances créées
>>> Point.count()
2
>>> p1.x
10
>>> p1.distance(origine)
5
>>> p1
'Point (3 ; 4)'
>>> str(p1)
'Point (3 ; 4)'
>>> print(p1)
Point (3 ; 4)
```

Indications

- Il faut utiliser le théorème de Pythagore pour calculer la distance entre deux points du plan.
- Le module `math` contient une fonction `sqrt` permettant de calculer la racine carrée d'une nombre réel. Veillez à importer le module `math` correctement !
- Définir une variable de classe `count` et faire en sorte qu'elle indique toujours le nombre d'instances créées, comme le montre l'exemple.

Classe `Point`

Écrire le code de la classe `Point` ci-dessous :

Question 3 (... / 10 points)

On veut définir une classe `Rect` pour représenter une rectangle et une classe `Square` pour représenter un carré selon le diagramme de classe ci-dessous

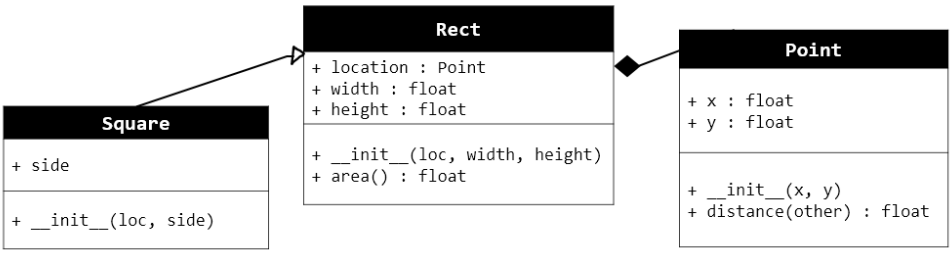

Diagramme de classes de la situation

b. Compléter les classes ci-dessous pour correspondre au diagramme de classe ci-dessus :

```
class Rect(..............):

    def __init__(self, origin, width, height):
    # compléter le code ici




    def area(self):
        # compléter le code ici



class Square(..................):

    def __init__(self, side):
        # compléter le code ici



    # rajouter des méthodes si nécessaire



    #
```

Exemple d'utilisation

```
>>> r = Rect(Point(0,0), 10, 20)
>>> r.area
200
>>> s = Square(Point(15, 15), 30)
>>> s.area()
225
```

Partie 2 : Compréhension et analyse de code

Question 5 (... / 15 points)

1. On veut augmenter à 36 le nombre de frames par secondes (fps) sans se soucier de l'accélération du jeu.
    a. Indiquer la période de raffraichissement qu'il faut adopter
    b. Indiquer le numéro de la ligne à changer et effectuer le changement nécessaire
2. Définir la notion de variable de classe et mettre en évidence un exemplaire dans le code
3. Définir la notion de variable d'instance et mettre en évidence un exemplaire dans le code
4. Définir la notion de méthode de classe et indiquer les différences avec une méthode d'instance
5. Expliquer en quoi consiste le mécanisme de polymorphisme et indiquer comment le code du Frogger y fait recours
6. Expliquer à quoi sert la méthode `act` de la classe `Car` et comment / quand cette méthode est appelée
7. Expliquer à quoi sert l'instruction `delay(100)` à la ligne 46.
8. Pourrait-on toujours jouer à Frogger si la ligne n'était pas présente? Justifier!

Code à analyser

```
from gamegrid import *
from soundsystem import *

import time
import random

class FroggerGame(object):

    K_LEFT      = 37
    K_UP        = 38
    K_RIGHT     = 39
    K_DOWN      = 40

    def __init__(self):

        self.nb_succes = 0
        self.max_temps = 15
        self.temps_restant = self.max_temps
        self.is_game_over = False

        makeGameGrid(800, 600, 1, None, "sprites/lane.gif", False,
             keyRepeated = self.keyCallback)
        setSimulationPeriod(50);
        self.frog = Frog()
        addActor(self.frog, Location(400, 560), 90)
        self.initCars()
        show()
        doRun()
        self.manager()

    def manager(self):
        while not isDisposed() and not self.is_game_over:
            t0 = time.time()
            if self.frog.vies == 0:
                self.gameover()

            infos = '#Vies : {vies} // ' + \
                            '#Succès : {success} // ' + \
                            '#Points : {points} // ' + \
                            'Temps restant : {temps}'
            infos = infos.format(vies=self.frog.vies,
                                 success=self.nb_succes,
                                 points=self.frog.points,
                                 temps=self.frog.temps_restant)
            setTitle(infos)
            delay(100)

            t1 = time.time()
            print(t1-t0)
            self.frog.temps_restant -= (t1 - t0)

            if self.frog.temps_restant <= 0:
                self.frog.reset()
                self.frog.points -= 10
```

```
 55
 56     def initCars(self):
 57         speeds = [9, 10, 11, 12]
 58         random.shuffle(speeds)
 59         for i in range(20):
 60             car = Car("sprites/car" + str(i) + ".gif")
 61             self.frog.addCollisionActor(car)
 62 
 63             if i < 5:
 64                 addActor(car, Location(350 * i, 100))
 65                 car.speed = speeds[0]
 66             if i >= 5 and i < 10:
 67                 addActor(car, Location(350 * (i - 5), 220))
 68                 car.speed = -speeds[1]
 69             if i >= 10 and i < 15:
 70                 addActor(car, Location(350 * (i - 10), 350))
 71                 car.speed = speeds[2]
 72             if i >= 15:
 73                 addActor(car, Location(350 * (i - 15), 470))
 74                 car.speed = -speeds[3]
 75 
 76     def keyCallback(self, keyCode):
 77         if keyCode == FroggerGame.K_LEFT:
 78             self.frog.setX(self.frog.getX() - 5)
 79         elif keyCode == FroggerGame.K_UP:
 80             self.frog.setY(self.frog.getY() - 5)
 81         elif keyCode == FroggerGame.K_RIGHT:
 82             self.frog.setX(self.frog.getX() + 5)
 83         elif keyCode == FroggerGame.K_DOWN:
 84             self.frog.setY(self.frog.getY() + 5)
 85 
 86         if self.frog.hasSucceeded():
 87             self.nb_succes += 1
 88             self.frog.points += 5
 89             self.frog.max_temps -= 1
 90             openSoundPlayer("wav/notify.wav")
 91             play()
 92             self.frog.reset()
 93 
 94 
 95     def gameover(self):
 96         addActor(Actor("sprites/gameover.gif"), Location(400, 285))
 97         removeActor(self.frog)
 98         self.is_game_over = True
 99         doPause()
100 
101 
102 class Frog(Actor):
103     def __init__(self, vies = 3):
104         Actor.__init__(self, "sprites/frog.gif")
105         self.vies = vies
106         self.points = 0
107         self.max_temps = 15
108         self.temps_restant = self.max_temps
109 
110     def collide(self, actor1, actor2):
```

```
111         openSoundPlayer("wav/boing.wav")
112         play()
113         self.reset()
114         self.vies -= 1
115         self.points -= 5
116         return 0
117 
118     def hasSucceeded(self):
119         return self.getY() < 20
120 
121     def reset(self):
122         self.setLocation(Location(400, 560))
123         self.temps_restant = self.max_temps
124 
125 class Car(Actor):
126     def __init__(self, path, speed=10):
127         Actor.__init__(self, path)
128         self.speed = speed
129 
130     def act(self):
131         self.move()
132         if self.getX() < -100:
133             self.setX(1650)
134         if self.getX() > 1650:
135             self.setX(-100)
136 
137     def move(self):
138         self.setX(self.getX() + self.speed)
139 
140 game = FroggerGame()
```