Classes et objets

Récapitulation des notions de base de la POO

Observez attentivement le code ci-dessous et répondre aux questions. Toutes les questions posées sont vraiment essentielles et dont des questions types qui peuvent être posées lors d'un oral de BAC.

```
 1 from gamegrid import *
 2
 3 # ---------------- classe Animal ----------------
 4 class Animal():
 5
 6     def __init__(self, imgPath):
 7         self.imagePath = imgPath
 8
 9
10     def showMe(self, x, y):
11          bg.drawImage(self.imagePath, x, y)
12
13
14 def pressCallback(e):
15     myAnimal = Animal("sprites/animal.gif")
16     myAnimal.showMe(e.getX(), e.getY())
17
18 makeGameGrid(600, 600, 1, False, mousePressed = pressCallback)
19 setBgColor(Color.green)
20 show()
21 doRun()
22 bg = getBg()
```

Analyse de code

1. Décrire le rôle de la fonction `__init__` aux lignes 6 et 7

2. Décrire précisément ce qui se passe à la ligne 7

3. Que représente le premier paramètre `self` dans la définition des méthodes d'instance ?

4. À quoi sert la fonction `pressCallback(e)` définie aux lignes 14 à 16 ?

5. Que représente le paramètre `e` de la fonctoin `pressCallback(e)` ?

6. Décrire précisément ce qui se passe à la ligne 16 ?

7. Expliquer ce que fait globalement ce code Python?

Héritage

L'héritage est une des propriétés les plus utiles et fondamentales dans la POO. Ce mécanisme permet de réutiliser du code défini dans d'autres classes par dérivation. Observer ce code en répondre aux questions posées :

```
1 from gamegrid import *
2 # Une des forces de TigerJython est qu'il permet d'utiliser
3 # les bibliothèques Java
4 from java.awt import Point
5
6 # ---------------- classe Animal ----------------
7 class Animal():
8
9     def __init__(self, imgPath):
10        self.imagePath = imgPath
11
12
13    def showMe(self, x, y):
14         bg.drawImage(self.imagePath, x, y)
15
16 # ---------------- classe Pet ----------------
17 class Pet(Animal):   # Derived from Animal
18
19    def __init__(self, imgPath, name):
20        Animal.__init__(self, imgPath)
21        self.name = name
22
23    def tell(self, x, y): # Additional method
24        bg.drawText(self.name, Point(x, y))
25
26 makeGameGrid(600, 600, 1, False)
27 setBgColor(Color.green)
28 show()
29 doRun()
30 bg = getBg()
31 bg.setPaintColor(Color.black)
32
33 for i in range(5):
34    myPet = Pet("sprites/pet.gif", "Trixi")
35    myPet.showMe(50 + 100 * i, 100)
36    myPet.tell(72 + 100 * i, 145)
```

Questions

1. Pourquoi met-on `Animal` entre parenthèses après `class Pet` dans la définition de la casse `Pet` ?

2. Décrire précisément ce que fait la ligne 20

3. Décrire ce que fait le programme globalement

4. Dessiner le diagramme de classes de `Animals` et `Pet`

Hiérarchie de classes

Étudier attentivement le code suivant et répondre aux questions :

```
from gamegrid import *
from java.awt import Point

# ---------------- classe Animal ----------------
class Animal():

    def __init__(self, imgPath):
        self.imagePath = imgPath


    def showMe(self, x, y):
         bg.drawImage(self.imagePath, x, y)

# ---------------- classe Pet ----------------
class Pet(Animal):

    def __init__(self, imgPath, name):
        Animal.__init__(self, imgPath)
        self.name = name

    def tell(self, x, y):
        bg.drawText(self.name, Point(x, y))

# ---------------- classe Dog ----------------
class Dog(Pet):

    def __init__(self, imgPath, name):
        Pet.__init__(self, imgPath, name)

    def tell(self, x, y): # Overriding
        bg.setPaintColor(Color.blue)
        bg.drawText(self.name + " tells 'Waoh'", Point(x, y))

# ---------------- classe Cat ----------------
class Cat(Pet):

    def __init__(self, imgPath, name):
        Pet.__init__(self, imgPath, name)
        self.name = name

    def tell(self, x, y): # Overriding
        bg.setPaintColor(Color.gray)
        bg.drawText(self.name + "  tells 'Meow'", Point(x, y))

makeGameGrid(600, 600, 1, False)
setBgColor(Color.green)
show()
doRun()
bg = getBg()

alex = Dog("sprites/dog.gif", "Alex")
```

```
52 alex.showMe(100, 100)
53 alex.tell(200, 130)
54
55 rex = Dog("sprites/dog.gif", "Rex")
56 rex.showMe(100, 300)
57 rex.tell(200, 330)
58
59 xara = Cat("sprites/cat.gif", "Xara")
60 xara.showMe(100, 500)
61 xara.tell(200, 530)
```

Questions

1. Dessiner le diagramme de classes de `Animal`, `Pet`, `Cat`, `Dog`

2. Modifier les classes `Dog` et `Cat` pour qu'elles chargent automatiquement le bon sprite (la bonne image représentative) sans devoir le spécifier dans le construteur