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 # ---------------- class 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 # ---------------- class Pet ----------------
17 class Pet(Animal):   # Derived from Animal
18
19     def __init__(self, imgPath, name):
20         super().__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

```
 1 from gamegrid import *
 2 from java.awt import Point
 3
 4 # ---------------- class Animal ----------------
 5 class Animal():
 6
 7     def __init__(self, imgPath):
 8         self.imagePath = imgPath
 9
10
11     def showMe(self, x, y):
12          bg.drawImage(self.imagePath, x, y)
13
14 # ---------------- class Pet ----------------
15 class Pet(Animal):
16
17     def __init__(self, imgPath, name):
18         self.imagePath = imgPath
19         self.name = name
20
21     def tell(self, x, y):
22         bg.drawText(self.name, Point(x, y))
23
24 # ---------------- class Dog ----------------
25 class Dog(Pet):
26
27     def __init__(self, imgPath, name):
28         self.imagePath = imgPath
29         self.name = name
30
31     def tell(self, x, y): # Overriding
32         bg.setPaintColor(Color.blue)
33         bg.drawText(self.name + " tells 'Waoh'", Point(x, y))
34
35 # ---------------- class Cat ----------------
36 class Cat(Pet):
37
38     def __init__(self, imgPath, name):
39         self.imagePath = imgPath
40         self.name = name
41
42     def tell(self, x, y): # Overriding
43         bg.setPaintColor(Color.gray)
44         bg.drawText(self.name + "  tells 'Meow'", Point(x, y))
45
46 makeGameGrid(600, 600, 1, False)
47 setBgColor(Color.green)
48 show()
49 doRun()
50 bg = getBg()
51
52 alex = Dog("sprites/dog.gif", "Alex")
53 alex.showMe(100, 100)
```

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

Polymorphisme

```
1  from gamegrid import *
2  from soundsystem import *
3
4  # ---------------- class Animal ----------------
5  class Animal():
6
7      def __init__(self, imgPath):
8          self.imagePath = imgPath
9
10
11     def showMe(self, x, y):
12          bg.drawImage(self.imagePath, x, y)
13
14 # ---------------- class Pet ----------------
15 class Pet(Animal):
16
17     def __init__(self, imgPath, name):
18         self.imagePath = imgPath
19         self.name = name
20
21     def tell(self, x, y):
22         bg.drawText(self.name, Point(x, y))
23
24 # ---------------- class Dog ----------------
25 class Dog(Pet):
26
27     def __init__(self, imgPath, name):
28          self.imagePath = imgPath
29          self.name = name
30
31     def tell(self, x, y): # Overridden
32          Pet.tell(self, x, y)
33          openSoundPlayer("wav/dog.wav")
34          play()
35
36 # ---------------- class Cat ----------------
37 class Cat(Pet):
38
39     def __init__(self, imgPath, name):
40         self.imagePath = imgPath
41         self.name = name
42
43     def tell(self, x, y): # Overridden
44         Pet.tell(self, x, y)
45         openSoundPlayer("wav/cat.wav")
46         play()
47
48
49 makeGameGrid(600, 600, 1, False)
50 setBgColor(Color.green)
51 show()
52 doRun()
```

```
53 bg = getBg()
54
55 animals =
56     [Dog("sprites/dog.gif", "Alex"),
57      Dog("sprites/dog.gif", "Rex"),
58      Cat("sprites/cat.gif", "Xara")]
59
60 y = 100
61 for animal in animals:
62     animal.showMe(100, y)
63     animal.tell(200, y + 30)    # Which tell()????
64     pet.show())
65     y = y + 200
66     delay(1000)
```