Αντικειμενοστραφής Προγραμματισμός - Μέρος 1ο

Μιχάλης Δρακόπουλος

Ιούνιος 2020

Διαδικαστικός και Αντικειμενοστραφής Προγραμματισμός

Διαδικαστικός προγραμματισμός

- Το πρόβλημα αναλύεται σε υποπροβλήματα (σχεδιασμός top - down)
- Κάθε υποπρόβλημα υλοποιείται προγραμματιστικά ως μια **διαδικασία** (= συνάρτηση)
- Οι διαδικασίες δέχονται δεδομένα και παράγουν αποτελέσματα τα οποία τροφοδοτούν ως δεδομένα σε άλλες διαδικασίες.
- Οι διαδικασίες χρησιμοποιούν τους βασικούς τύπους δεδομένων της ΓΠ.
- Τα δεδομένα είναι ανεξάρτητα από τις διαδικασίες.

Αντικειμενοστραφής προγραμματισμός

- Το πρόβλημα προσομοιώνεται από ενέργειες και αλληλεπιδράσεις **αντικειμένων** (σχεδιασμός bottom - up)
- Τα αντικείμενα αλληλεπιδρούν μεταξύ τους, όπως και στον πραγματικό κόσμο.
- Τα αντικείμενα είναι εξειδικευμένοι τύποι δεδομένων που ενσωματώνουν δεδομένα και συναρτήσεις.

Αντικείμενα και κλάσεις

Αφαίρεση

- Αντικείμενα του ιδίου τύπου ανήκουν στην ίδια **κλάση** και **κατασκευάζονται** από αυτήν.
- Οι κλάσεις είναι αφηρημένοι τύποι δεδομένων για τον ορισμό αντικειμένων.

Ενθυλάκωση

- Σε κάθε αντικείμενο ενσωματώνονται τα δεδομένα του **και** οι συναρτήσεις που δρουν πάνω σε αυτά.
  - Τα δεδομένα αντιστοιχούν σε *ιδιοχαρακτηριστικά* του αντικειμένου και ονομάζονται *πεδία*
  - Οι ενσωματωμένες συναρτήσεις λέγονται *μέθοδοι* και αντιστοιχούν στις ενέργειες που μπορούμε να κάνουμε **με** το αντικείμενο ή **στο** αντικείμενο.

Δομή μιας κλάσης

[1]:
```
import random

class Coin:

    def __init__(self):
        self.sideup = 'Heads'
```

```
    def toss(self):
        self.sideup = random.choice(['Heads', 'Tails'])

    def get_sideup(self):
        return self.sideup
```

Κατασκευή αντικειμένων

Δημιουργία 2 αντικειμένων τύπου `Coin`

[2]:
```
c1 = Coin()
c2 = Coin()
```

Ρίψη των κερμάτων

[3]:
```
c1.toss()
c2.toss()
```

Ελεγχος αποτελεσμάτων

[4]:
```
if c1.get_sideup() == c2.get_sideup():
    print('Both coins are ' + c1.get_sideup())
else:
    print('Coin 1 is ' + c1.get_sideup())
    print('Coin 2 is ' + c2.get_sideup())
```

```
Both coins are Tails
```

Μέθοδοι πρόσβασης και τροποποιητικές μέθοδοι

Κακός ΑΣΠ: Αλλαγές ιδιοχαρακτηρίστικων με άμεσο τρόπο!

[5]:
```
class Employee:

    def __init__(self, name, salary):
        self.name = name.upper()
        self.salary = salary

    def give_raise(self, percent):
        self.salary *= (1 + percent)
```

[6]:
```
e = Employee('Bob', 1000)
e.salary = 2000
print(e.name + "'s salary is", e.salary)
```

```
BOB's salary is 2000
```

[7]:
```
e.name = 'bob'
e.give_raise(0.2)
```

```
print(e.name, 'changed his name and his salary is now', e.salary)
```

```
bob changed his name and his salary is now 2400.0
```

Καλός ΑΣΠ: Επεξεργασία ιδιοχαρακτηριστικών μέσω μεθόδων!

[8]:
```
class Employee:

    def __init__(self, name, salary):
        self.name = name.upper()
        self.salary = salary

    # Getters

    def get_name(self): return self.name

    def get_salary(self): return self.salary

    # Setters

    def set_name(self, new_name):
        self.name = new_name.upper()

    def set_salary(self, money):
        self.salary = money

    def give_raise(self, percent):
        self.salary *= (1 + percent)
```

[9]:
```
e = Employee('Bob', 1200)
print(e.get_name(), e.get_salary())

e.set_name('Babis')
e.give_raise(0.1)
print(e.get_name(), e.get_salary())
```

```
BOB 1200
BABIS 1320.0
```

Παράδειγμα: κλάση κατασκευής σημείων στο επίπεδο

[10]:
```
import math

class Point:

    def __init__(self, x=0, y=0):
        self.x = x
        self.y = y
```

```
    def getX(self): return self.x

    def getY(self): return self.y

    def dist(self, other):
        return math.sqrt((self.x - other.x)**2 + (self.y - other.y)**2)

    def move(self, dx, dy):
        self.x += dx
        self.y += dy

    def equals(self, other):
        return self.x == other.x and self.y == other.y
```

[11]:
```
pt = Point(1/3, math.pi)
print(pt)
```

```
<__main__.Point object at 0x7f40cc85e850>
```

Η μέθοδος __str__

[12]:
```
dir(int)
```

[12]:
```
['__abs__',
 '__add__',
 '__and__',
 '__bool__',
 '__ceil__',
 '__class__',
 '__delattr__',
 '__dir__',
 '__divmod__',
 '__doc__',
 '__eq__',
 '__float__',
 '__floor__',
 '__floordiv__',
 '__format__',
 '__ge__',
 '__getattribute__',
 '__getnewargs__',
 '__gt__',
 '__hash__',
 '__index__',
 '__init__',
 '__init_subclass__',
 '__int__',
```

```
'__invert__',
'__le__',
'__lshift__',
'__lt__',
'__mod__',
'__mul__',
'__ne__',
'__neg__',
'__new__',
'__or__',
'__pos__',
'__pow__',
'__radd__',
'__rand__',
'__rdivmod__',
'__reduce__',
'__reduce_ex__',
'__repr__',
'__rfloordiv__',
'__rlshift__',
'__rmod__',
'__rmul__',
'__ror__',
'__round__',
'__rpow__',
'__rrshift__',
'__rshift__',
'__rsub__',
'__rtruediv__',
'__rxor__',
'__setattr__',
'__sizeof__',
'__str__',
'__sub__',
'__subclasshook__',
'__truediv__',
'__trunc__',
'__xor__',
'as_integer_ratio',
'bit_length',
'conjugate',
'denominator',
'from_bytes',
'imag',
'numerator',
'real',
'to_bytes']
```

[13]:
```
import math

class Point:

    def __init__(self, x=0, y=0):
        self.x = x
        self.y = y

    def getX(self): return self.x

    def getY(self): return self.y

    def dist(self, other):
        return math.sqrt((self.x - other.x)**2 + (self.y - other.y)**2)

    def move(self, dx, dy):
        self.x += dx
        self.y += dy

    def equals(self, other):
        return self.x == other.x and self.y == other.y

    def __str__(self):
        return '('+ format(self.x,'.3f') + ',' + format(self.y,'.3f') + ')'
```

[14]:
```
pt = Point(1/3, math.pi)
print(pt)
```

```
(0.333,3.142)
```

Εφαρμογή - Αποθήκευση αντικειμένων σε λίστα

Μέγιστη απόσταση ανάμεσα όλα τα δυνατά ζεύγη 100 τυχαίων σημείων στο επίπεδο.

[15]:
```
import random
# from geom import Point

def main():
    n = 100
    points = []
    for i in range(100):
        x = random.random()
        y = random.random()
        points.append(Point(x,y))
    d, Q1, Q2 = diameter(points)
    print("M         : " + format(d,'.3f'))
    print("            "+ str(Q1) + "    "+ str(Q2))
```

```
def diameter(pt):
    n = len(pt)
    dmax = 0
    for i in range(n-1):
        for j in range(i+1,n):
            dij = pt[i].dist(pt[j])
            if dij > dmax:
                dmax = dij
                Q1 = pt[i]
                Q2 = pt[j]
    return dmax, Q1, Q2

main()
```

```
M         : 1.210
            (0.049,0.999)    (0.846,0.088)
```

Εφαρμογή - Σύνθεση, κοινά ιδιοχαρακτηριστικά

Δυναμικό εξαιτίας σημειακού φορτίου q σε απόσταση r

$$V = k\frac{q}{r}$$

[16]:
```
# from geom import Point

class Charge:

    k = 8.99e9

    def __init__(self, q, pt):
        self.pos = pt
        self.q = q

    def potential_at(self, pt):
        return Charge.k*self.q/self.pos.dist(pt)

    def __str__(self): return str(self.q)+' at '+ str(self.pos)


def main():
    t = Point(0.51, 0.63)
    p = Point(1, 1)
    c = Charge(21.3, t)
    print(c)
    print(c.potential_at(p))

main()
```

```
21.3 at (0.510,0.630)
311866423698.9439
```