

# **Tema 1 – Sessió 3**

## **Programació orientada a objecte**

---

# Exercici

Implementeu una calculadora de nombres complexos seguint els passos següents:

1. Definiu una classe `NumeroComplex` amb els atributs necessaris per guardar un nombre complex i mètodes que permetin:
  - Mostrar el nombre complex per pantalla en format  $a + bi$
  - Llegir per teclat els valors d'un nombre complex
  - Obtenir el conjugat d'un nombre complex
  - Sumar dos nombres complexos
  - Restar dos nombres complexos
  - Multiplicar dos nombres complexos
2. Implementeu el programa principal. Ha d'anar demanant operacions (suma, resta, multiplicació o obtenir el conjugat) per fer fins que es seleccioni l'opció de sortir. En cada operació es demanaran per teclat els nombres complexos d'entrada i es mostrarà el resultat final per pantalla.

# Exercici: solució

```
class NumeroComplex:
    def __init__(self, pReal = 0, pImag = 0):
        self.real = pReal
        self.img = pImag

    def llegeix(self):
        self.real = float(input('Part real: '))
        self.img = float(input('Part imaginaria: '))

    def conjugat(self):
        return NumeroComplex(self.real, -self.img)

    def suma(self, c):
        return NumeroComplex(self.real + c.real, self.img + c.img)

    def resta(self, c):
        return NumeroComplex(self.real - c.real, self.img - c.img)

    def multiplica(self, c):
        real = self.real*c.real - self.img*c.img
        img = self.real*c.img + self.img*c.real
        return NumeroComplex(real, img)

    def __str__(self):
        return str(self.real) + '+' + str(self.img) + 'i'
```

# Exercici: solució

```
import complex

def principal():
    opcio = '0'
    c1 = complex.NumeroComplex()
    c2 = complex.NumeroComplex()
    while (opcio != '5'):
        opcio = input ("Introdueix una opcio (1. Suma, 2. Resta, 3. ...)")
        if (opcio == '1'):
            c1.llegeix()
            c2.llegeix()
            print ("Resultat de la suma: ", c1.suma(c2))
        elif (opcio == '2'):
            c1.llegeix()
            c2.llegeix()
            print ("Resultat de la resta: ", c1.resta(c2))
        elif (opcio == '3'):
            c1.llegeix()
            c2.llegeix()
            print ("Resultat de la multiplicacio: ", c1.multiplica(c2))
        elif (opcio == '4'):
            c1.llegeix()
            print ("Conjugat", c1.conjugat())
        elif (opcio != '5'):
            print ("Opcio incorrecta")

if __name__ == "__main__":
    principal()
```

# Recordem: classes i objectes

- Un objecte és una col·lecció de dades (propietats) i el seu comportament (mètodes o accions) associat.
- Les classes descriuen tipus d'objectes: conjunt d'objectes amb les mateixes propietats i comportament.
- Cada objecte és una instància específica i diferent d'una classe. Cada objecte tindrà valors diferents per les propietats, però el mateix comportament (accions) que la resta d'objectes de la mateixa classe.

```
class Point:
    def __init__(self, x = 0, y = 0):
        self.x = x
        self.y = y
    def distance_origin(self):
        return math.sqrt(self.x**2 + self.y**2)
    def distance(self, p2):
        return math.sqrt((self.x - p2.x)**2 + (self.y - p2.y)**2)
    def midpoint(self, p2):
        return Point((self.x + p2.x)/2, (self.y + p2.y)/2)
    def __str__(self):
        return "(" + str(self.x) + ", " + str(self.y) + ")"
```

# Recordem: classes i objectes

```
class Point:
```

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

**x, y** : atributs ( propietats de la classe )

**\_\_init\_\_** : constructor

- Mètode que es crida quan es crea un objecte de la classe
- Serveix per inicialitzar l'estat de l'objecte (el valor de tots els seus atributs)

p = Point(1,0) -----> \_\_init\_\_(self,x, y)

Point

p:	x: 1
	y: 0

p.x → 1

p.y → 0

# Recordem: classes i objectes

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

```
def distance_origin(self):  
    return math.sqrt(self.x**2 + self.y**2)  
  
def distance(self, p2):  
    ...  
def midpoint(self, p2):  
    ...  
def __str__(self):  
    ...
```

**self.xxxx:** accés als atributs de la classe dins dels mètodes

**Mètodes** de la classe: accions associades als objectes de la classe: distance:origin, distance, midpoint, \_\_str\_\_

p1.distance\_origin() -----> distance\_origin(self)  
p1.distance(p2) -----> distance(self, c)

# Sobrecàrrega d'operadors

```
class NumeroComplex:
```

```
...
```

```
def __str__(self):  
    return str(self.real) + '+' + str(self.img) + 'i'
```

`__str__`: sobrecàrrega de l'operador `str` per convertir els objectes a un `string`: S'utilitza per mostrar per pantalla els objectes de la classe

Altres operadors que es poden sobrecarregar

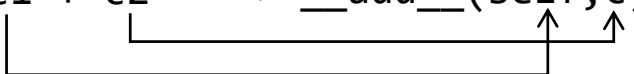
- Operadors de comparació:
  - `__lt__` (a, b): Més petit que ( $a < b$ )
  - `__ne__` (a, b): Diferent ( $a \neq b$ )
  - `__le__` (a, b): Més petit igual ( $a \leq b$ )
  - `__ge__` (a, b): Més gran igual ( $a \geq b$ )
  - `__eq__` (a, b): Iguals ( $a == b$ )
  - `__gt__` (a, b): Més gran que ( $a > b$ )
- Operadors aritmètics:
  - `__add__` (a, b): Suma ( $a + b$ )
  - `__mul__` (a, b): Multiplicació ( $a * b$ )
  - `__div__` (a, b): Divisió ( $a / b$ )
  - `__sub__` (a, b): Resta ( $a - b$ )

`print(c1) --> print(c1.__str())`  
`print(str(c1))`

`c1 = NumeroComplex(1,1)`

`c2 = NumeroComplex(2,2)`

`c1 + c2 -----> __add__(self, c)`





## Exercici

Modifiqueu la classe `NumeroComplex` i el programa principal perquè les operacions de suma, resta i multiplicació es facin sobrecarregant els operadors aritmètics habituals: `+`, `-` i `*`

# Exercici: solució

```
class NumeroComplex:
    def __init__(self, pReal = 0, pImg = 0):
        self.real = pReal
        self.img = pImg

    def llegeix(self):
        self.real = float(input('Part real: '))
        self.img = float(input('Part imaginaria: '))

    def conjugat(self):
        return NumeroComplex(self.real, -self.img)

    def __add__(self, c):
        return NumeroComplex(self.real + c.real, self.img + c.img)

    def __sub__(self, c):
        return NumeroComplex(self.real - c.real, self.img - c.img)

    def __mul__(self, c):
        real = self.real*c.real - self.img*c.img
        img = self.real*c.img + self.img*c.real
        return NumeroComplex(real, img)

    def __str__(self):
        return str(self.real) + '+' + str(self.img) + 'i'
```

# Exercici: solució

```
import complex

def principal():
    opcio = '0'
    c1 = complex.NumeroComplex()
    c2 = complex.NumeroComplex()
    while (opcio != '5'):
        opcio = input ("Introdueix una opcio (1. Suma, 2. Resta, 3. ...
        if (opcio == '1'):
            c1.llegeix()
            c2.llegeix()
            print ("Resultat de la suma: ", c1 + c2)
        elif (opcio == '2'):
            c1.llegeix()
            c2.llegeix()
            print ("Resultat de la resta: ", c1 - c2)
        elif (opcio == '3'):
            c1.llegeix()
            c2.llegeix()
            print ("Resultat de la multiplicacio: ", c1 * c2)
        elif (opcio == '4'):
            c1.llegeix()
            print ("Conjugat", c1.conjugat())
        elif (opcio != '5'):
            print ("Opcio incorrecta")

if __name__ == "__main__":
    principal()
```