Association

OOP – Spring 2022 (Python)

Association

Its a relationship between two classes and that relationship is established through their objects. Each object has its own life cycle and there is no owner object. It is a weak type of relationship. It can be one-to-one, one-to-many, many-to-one, or many-to-many.

For example students and teachers, both classes are associated with each other. The objects of each class have their own life cycle and there is no owner.

Ref: https://faun.pub/association-aggregation-composition-python-ec9947832cbd

Association – Example I

```
class A(object):
    def __init__(self, n, x1, x2):
        self.n = n
        self.x1 = x1
        self.x2 = x2

class B(object):
    def __init__(self, y1, y2):
        self.y1 = y1
        self.y2 = y2

    def addAllNums(self, num1, num2):
        return self.y1 + self.y2 + num1 + num2

objA = A('1', 2, 6)
objB = B(5, 9)

sum = objB.addAllNums(objA.x1, objA.x2)
print (f'Sum of all values: {sum}')
```

Output:

```
Sum of all values: 22
```

Association – Example II

```
class Circle(Shape):
    def __init__(self, screen, center_x, center_y, radius, color='BLUE'):
        super().__init__(screen, color)
        self.__center_x = center_x
        self.__center_y = center_y
        self.__radius = radius

    def get_center(self):
        return self.__center_x, self.__center_y

    def set_center(self, x, y):
        self.__center_x = x
        self.__center_y = y

    def move(self, x, y):
        py.draw.circle(self.screen, 'WHITE', (self.__center_x, self.__center_y), self.__radius)
        sleep(0.5)
        self.set_center(x, y)
        py.draw.circle(self.screen, self.color, (self.__center_x, self.__center_y), self.__radius)
        py.display.update()
```

Association – Example II

```
class Shape_Mover():
    def __init__(self, width, height):
        self.width = width
        self.height = height

    def get_points_after_rotation(x, y, angle):
        x_new = x * math.cos(angle) - y * math.sin(angle)
        y_new = y * math.cos(angle) + x * math.sin(angle)
        return x_new, y_new

    def move(self, circle):
        rx = self.width // 2                ry = self.height // 2
        fac = math.pi / 32                  angle = math.pi
        angle_end = 0
        x, y = circle.get_center()
        while angle >= angle_end :
            x_n, y_n = Shape_Mover.get_points_after_rotation(x - rx , y - ry, angle)
            x_n += rx                     y_n += ry
            circle.move(x_n, y_n)          angle -= fac
```

Association – Example II

```
def main():
    screen = py.display.set_mode((1200, 800))
    screen.fill('white')
    py.display.update()
    circle = Circle(screen, 500, 300, 40)
    sm = Shape_Mover(1200, 800)
    sm.move(circle)
```