Rectangle Exercise Solution

Hyungmo Gu

April 4, 2020

Part 1

1. **Class Name:** Rectangle

   One Line Summary: A rectangle is defined by its top-left coordinates as well as its width and height.

```
Rectangle(10,20,300,400)

```

3. **Headers:**

   - translate_left(self, num):
   - translate_right(self, num):
   - translate_up(self, num):
   - translate_down(self, num):
   - is_equal(self, rect):
   - is_falling_within_another_rectangle(self, rect):
   - is_overlapping(self, rect):

```
class Rectangle:
    """A rectangle is defined by its top-left coordinates as well
 as its width and height.

    @type x: int
        The x coordinate of top-left corner of this rectangle
    @type y: int
        The y coordinate of top-left corner of this rectangle
    @type width: int
        The width of this rectangle
    @type height: int
        The height of this rectangle
    """

```

```
14          def translate_left(self, num):
15              """Translate Rectangle to left by <num>
16              @type self: Rectangle
17              @type num: int
18              @rtype: None
19              >>> rect = Rectangle(10,20,300,400)
20              >>> rect.translate_left(10)
21              """
22          def translate_right(self, num):
23              """Translate Rectangle to right by <num>
24              @type self: Rectangle
25              @type num: int
26              @rtype: None
27              >>> rect = Rectangle(10,20,300,400)
28              >>> rect.translate_right(10)
29              """
30
31          def translate_up(self, num):
32              """Translate Rectangle to up by <num>
33              @type self: Rectangle
34              @type num: int
35              @rtype: None
36              >>> rect = Rectangle(10,20,300,400)
37              >>> rect.translate_up(10)
38              """
39
40          def translate_down(self, num):
41              """Translate Rectangle to down by <num>
42              @type self: Rectangle
43              @type num: int
44              @rtype: None
45              >>> rect = Rectangle(10,20,300,400)
46              >>> rect.translate_down(10)
47              """
48
49          def is_equal(self, rect):
50              """Return whether <rect> and <self> have the same
    coordinate and size
51              @type self: Rectangle
52              @type rect: Rectangle
53              @rtype: bool
54              >>> rect_1 = Rectangle(10,20,300,400)
55              >>> rect_2 = Rectangle(10,20,300,400)
56              >>> rect_3 = Rectangle(15,25,300,400)
57              >>> rect_1.is_equal(rect_2)
58              True
59              >>> rect_1.is_equal(rect_3)
60              False
61              """
62
63          def is_falling_within_another_rectangle(self, rect):
64              """Return whether <self> is inside <rect>
65              @type self: Rectangle
66              @type rect: Rectangle
```

```
67              @rtype: bool
68              >>> rect_1 = Rectangle(10,20,300,400)
69              >>> rect_2 = Rectangle(15,15,100,50)
70              >>> rect_2.is_falling_within_another_rectangle(rect_1)
71              True
72              """
73
74          def is_overlapping(self, rect):
75              """Returns whether <self> has overlapping region with <
    rect>
76              @type self: Rectangle
77              @type rect: Rectangle
78              @rtype: bool
79              >>> rect_1 = Rectangle(10,20,300,400)
80              >>> rect_2 = Rectangle(0,0,300,400)
81              >>> rect_1.is_overlapping(rect_2)
82              True
83              """
84
```

Notes:

- What should be written for **@rtype** if nothing is returned?
- What should be written for **@type** if a parameter is of type class?
- What should be written for **@type** if a parameter is **self**?
- When writing an example, should class instantiation also be included like below?

```
1           def is_overlapping(self, rect):
2               """
3               ...
4               >>> rect_1 = Rectangle(10,20,300,400)
5               >>> rect_2 = Rectangle(0,0,300,400)
6               ...
7               """
8
```

Part 2