

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.

```
21 Rectangle(10,20,300,400)
2
```

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):

```
1 class Rectangle:
2     """A rectangle is defined by its top-left coordinates as well
3     as its width and height."""
4
5     def translate_left(self, num):
6         """Translate Rectangle to left by <num>
7         @type self: Rectangle
8         @type num: int
9         @rtype: None
10        >>> rect = Rectangle(10,20,300,400)
11        >>> rect.translate_left(10)
12        """
13
14    def translate_right(self, num):
15        """Translate Rectangle to right by <num>
```

```

14         @type self: Rectangle
15         @type num: int
16         @rtype: None
17         >>> rect = Rectangle(10,20,300,400)
18         >>> rect.translate_right(10)
19         """
20
21     def translate_up(self, num):
22         """Translate Rectangle to up by <num>
23         @type self: Rectangle
24         @type num: int
25         @rtype: None
26         >>> rect = Rectangle(10,20,300,400)
27         >>> rect.translate_up(10)
28         """
29
30     def translate_down(self, num):
31         """Translate Rectangle to down by <num>
32         @type self: Rectangle
33         @type num: int
34         @rtype: None
35         >>> rect = Rectangle(10,20,300,400)
36         >>> rect.translate_down(10)
37         """
38
39     def is_equal(self, rect):
40         """Return whether <rect> and <self> have the same
41         coordinate and size
42         @type self: Rectangle
43         @type rect: Rectangle
44         @rtype: bool
45         >>> rect_1 = Rectangle(10,20,300,400)
46         >>> rect_2 = Rectangle(10,20,300,400)
47         >>> rect_3 = Rectangle(15,25,300,400)
48         >>> rect_1.is_equal(rect_2)
49         True
50         >>> rect_1.is_equal(rect_3)
51         False
52         """
53
54     def is_falling_within_another_rectangle(self, rect):
55         """Return whether <self> is inside <rect>
56         @type self: Rectangle
57         @type rect: Rectangle
58         @rtype: bool
59         >>> rect_1 = Rectangle(10,20,300,400)
60         >>> rect_2 = Rectangle(15,15,100,50)
61         >>> rect_2.is_falling_within_another_rectangle(rect_1)
62         True
63         """
64
65     def is_overlapping(self, rect):
66         """Returns whether <self> has overlapping region with <
67         rect>

```

```

66         @type self: Rectangle
67         @type rect: Rectangle
68         @rtype: bool
69         >>> rect_1 = Rectangle(10,20,300,400)
70         >>> rect_2 = Rectangle(0,0,300,400)
71         >>> rect_1.is_overlapping(rect_2)
72         True
73         """
74

```

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