Rectangle Exercise Solution

Hyungmo Gu

April 5, 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
```

```
            @rtype: bool
            >>> rect_1 = Rectangle(10,20,300,400)
            >>> rect_2 = Rectangle(15,15,100,50)
            >>> rect_2.is_falling_within_another_rectangle(rect_1)
            True
            """

        def is_overlapping(self, rect):
            """Returns whether <self> has overlapping region with <
    rect>
            @type self: Rectangle
            @type rect: Rectangle
            @rtype: bool
            >>> rect_1 = Rectangle(10,20,300,400)
            >>> rect_2 = Rectangle(0,0,300,400)
            >>> rect_1.is_overlapping(rect_2)
            True
            """

```

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?

```
        def is_overlapping(self, rect):
            """
            ...
            >>> rect_1 = Rectangle(10,20,300,400)
            >>> rect_2 = Rectangle(0,0,300,400)
            ...
            """

```

Part 2

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."""

      def __init__(self, x, y, width, height):
          """ Initialize this Rectangle

          @type self: Rectangle
          @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
          """

          self.x = x
          self.y = y
          self.width = width
          self.height = height

      def translate_left(self, num):
          """Translate Rectangle to left by <num>
          @type self: Rectangle
          @type num: int
          @rtype: None
          >>> rect = Rectangle(10,20,300,400)
          >>> rect.translate_left(10)
          """

          self.x -= num

      def translate_right(self, num):
          """Translate Rectangle to right by <num>
          @type self: Rectangle
          @type num: int
          @rtype: None
          >>> rect = Rectangle(10,20,300,400)
          >>> rect.translate_right(10)
          """
```

```
42
43              self.x += num
44
45          def translate_up(self, num):
46              """Translate Rectangle to up by <num>
47              @type self: Rectangle
48              @type num: int
49              @rtype: None
50              >>> rect = Rectangle(10,20,300,400)
51              >>> rect.translate_up(10)
52              """
53
54              self.y += num
55
56          def translate_down(self, num):
57              """Translate Rectangle to down by <num>
58              @type self: Rectangle
59              @type num: int
60              @rtype: None
61              >>> rect = Rectangle(10,20,300,400)
62              >>> rect.translate_down(10)
63              """
64
65              self.y -= num
66
67          def is_equal(self, rect):
68              """Return whether <rect> and <self> have the same
    coordinate and size
69              @type self: Rectangle
70              @type rect: Rectangle
71              @rtype: bool
72              >>> rect_1 = Rectangle(10,20,300,400)
73              >>> rect_2 = Rectangle(10,20,300,400)
74              >>> rect_3 = Rectangle(15,25,300,400)
75              >>> rect_1.is_equal(rect_2)
76              True
77              >>> rect_1.is_equal(rect_3)
78              False
79              """
80
81              if (self.x == rect.x and
82                  self.y == rect.y and
83                  self.width == rect.width and
84                  self.height == rect.height):
85                  return True
86
87              return False
88
89
90          def is_falling_within_another_rectangle(self, rect):
91              """Return whether <self> is inside <rect>
92              @type self: Rectangle
93              @type rect: Rectangle
94              @rtype: bool
```

```
            >>> rect_1 = Rectangle(10,20,300,400)
            >>> rect_2 = Rectangle(15,15,100,50)
            >>> rect_2.is_falling_within_another_rectangle(rect_1)
            True
            """

            if (self.x < rect.x and
                rect.x + rect.width < self.x + self.width and
                self.y < rect.y and
                rect.y + rect.height < self.y + self.height):

                return True

            if (rect.x < self.x and
                self.x + self.width < rect.x + rect.width and
                rect.y < self.y and
                self.y + self.height < rect.y + rect.height):

                return True

            return False

        def is_overlapping(self, rect):
            """Returns whether <self> has overlapping region with <
    rect>
            @type self: Rectangle
            @type rect: Rectangle
            @rtype: bool
            >>> rect_1 = Rectangle(10,20,300,400)
            >>> rect_2 = Rectangle(0,0,300,400)
            >>> rect_1.is_overlapping(rect_2)
            True
            """

            overlaps_in_x = (
                (rect.x <= self.x) and
                (self.x <= rect.x + rect.width) or
                (self.x <= rect.x) and
                (rect.x <= self.x + self.width))

            overlaps_in_y = (
                (rect.y <= self.y) and
                (self.y <= rect.y + rect.height) or
                (self.y <= rect.y) and
                (rect.y <= self.y + self.height))

            if overlaps_in_x and overlaps_in_y:
                return True

            return False
```

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?

```
        def is_overlapping(self, rect):
            """
            ...
            >>> rect_1 = Rectangle(10,20,300,400)
            >>> rect_2 = Rectangle(0,0,300,400)
            ...
            """

```