HomeWork 3 - CS 5007

Thierry Petit

`tpetit@WPI.edu`

Learning objectives

- Object programming
- Files

Dates

- Assigned: Monday June 22, 2020.
- Due: Wednesday July 1, 2020, 8PM.

The basic assignment is worth 100 points and the extra credit option is worth up to 5 additional points (last question). This is an individual assignment. You may discuss any aspect of this assignment with anyone, but you must type everything into an IDLE window yourself. Except where specified, you may never copy and paste from any electronic source.

Write your code and comments on a file `FIRSTNAME-LASTNAME-HW3.py` that you will create from scratch (there is no template). Upload it on the course website (no delay accepted).

1 Point (30 points)

State a class `Point` that has two instance variables, its rational coordinates `self.x` and `self.y`.

- Implement in this class a constructor with two arguments, the values of the coordinates. Each argument should have a default value 0.0, allowing to call this method without arguments.
- Write two methods named `getX` and `getY` that respecitvely return the value of `self.x` and the value of `self.y`.
- Write a method named `setX` that takes a rational value as argument and modifies `self.x` so as it becomes equal to this value.
- Write a method named `setY` that takes a rational value as argument and modifies `self.y` so as it becomes equal to this value.
- Write a method named `toString` that returns a string containing the two coordinates of the point `self`, within parenthesis and separated by a comma.
- Write a method named `equals` that takes another object of class `Point` as argument and returns `True` if the x and y coordinates of this points are equal to the x and y coordinates of `self`, and returns `False` otherwise.

Outside the class, write the following statements: Create a point p_1 of coordinates $(0, 0)$ and un point p_2 of coordinates $(1, 2)$. Print out the coordinates of the two points on the same line, by calling `toString` on the two points. Print the result of applying the method `equals` on point p_1, using p_2 as argument. Set the x coordinate of p_2 equal to the x coordinate of p_1, using the methods `setX` and `getX`. Set the y coordinate of p_2 equal to the y coordinate of p_1, using the method `setY` and `getY`. Print again the result of applying the method `equals` on point p_1, using p_2 as argument. Increment by 1 the x coordinate of p_2, using the methods `setX` and `getX`. Print again the result of applying the method `equals` on point p_1, using p_2 as argument.

2 Rectangle (30 points)

State a class `Rectangle` that has three instance variables: a point (an object of the class `Point` previously defined, corresponding to the left bottom corner), a rational width and a rational height.

- Implement in this class a constructor with three arguments, a point and two rational values, to be assigned to the three instance variables. No default values should be stated (the constructor must be called with three arguments).
- Write a method named `perimeter` that returns the perimeter of the rectangle.
- Write a method named `area` that returns the area of the rectangle.
- Write a method named `getOrigin` that returns a reference to the point representing the left bottom corner of the rectangle.
- Write a method named `toString` that returns a string containing, on the same line, the string representing the bottom left corner, followed by the rectangle width and height, formatted as follows (the values will differ depending on each object):
  `(0.0,0.0), L=2, H=5.5`.

Outside the class, write the following statements: Create a rectangle r_1 whose bottom left corner is p_1, width is 2 and height is 5.5. Create a rectangle r_2 whose bottom left corner is p_2, width is 3 and height is 6. Print out the bottom left corner of r_1 and the perimeter of r_1, by calling the appropriate methods. Print out the bottom left corner of r_2 and the area of r_2, by calling the appropriate methods. Print the result of the call to the method `toString` respectively applied to the two rectangles r_1 and r_2.

3 Movable rectangle (40 points)

State a class `MovableRectangle` that inherits from the class `Rectangle`. This class has a supplementary instance variable, assumed to be boolean, called `self.move`.

- Implement in this class a constructor with three arguments, a point and two rational values, to be assigned to the three instance variables. `self.move` is initially always set to `False`. Your code MUST call the constructor of the superclass.
- Write a method named `unlock` that sets `self.move` to `True`.
- Write a method named `lock` that sets `self.move` to `False`.
- Write a method named `moveTo` that takes an object of class `Point` as argument, and: If the rectangle is unlocked, moves it so as its new bottom left is the point given as argument. Otherwise, the method prints the following message: `Warning: locked`.
- Write a method named `toString` that returns a string containing, on the same line, the string representing the bottom left corner, followed by the rectangle width and height, followed by the current status (locked or unlocked) formatted as follows (the values will differ depending on each object):
  `(5,15), W=2, H=2, Movable? True`
  (if it is locked, the end of string should be `...Movable? False`).

Outside the class, write the following statements: Create a movable rectangle r_3 whose bottom left corner is p_1, width is 2 and height is 2. Print the rectangle r_3, using `toString`. Create a point p_3 of coordinates $(5, 15)$. Call the method `moveTo` on rectangle r_3 in order to try moving its origin to p_3. Print again the rectangle r_3, using `toString`. Unlock r_3. Call again the method `moveTo` on rectangle r_3 in order to try moving its origin to p_3. Print again the rectangle r_3, using `toString`.

4 Extra-credit : Read File (5 points)

Create a class `Tour` with a single instance variable `self.MyList` that will refer to a list object. Create a constructor that takes one string parameter, the name of a file, `filename`. This class deals with files representing sets of cities. You will find on the course website some data files `tourX.csv`, where `X` is the number of cities in the set. If you open such a file with a *text* editor, you will see that the file states one city per row: name, x coordinate, y coordinate. The constructor must assign the instance variable `self.MyList` using the file whose name is given as argument (string parameter `filename`, assumed to be one of the files `tourX.csv`). When called, the argument is directly a file name, not a path. The constructor assigns to `self.MyList` a list of the following form (the above text is just an example used to show the expected format):

```
[['Atlanta', '22', '3'], ['Augusta', '46', '28'], ...]
```

▷ *Note: in such a list* `[['Atlanta', '22', '3'], ['Augusta', '46', '28'], ...]`, *all elements in sublists are strings (not integers).*