CS417 Lab 09

Getting Started

Begin the lab by downloading these starting files:

- `img_viewer.py`
- `img_viewer_oo.py`
- `viewport.py`
- `usa.txt`

Overview

You will take a program that doesn't use objects, and rewrite it to use object-oriented programming instead.

Your Tasks

1. Run the program `img_viewer.py`. It displays an image (by default, it opens `usa.txt`, unless the user indicates another file, in the command line).

   The `main()` function runs this loop:
   - shows you a viewport into part of the image
   - asks for an action

   Your action can be 'up', 'down', 'left', 'right', or 'quit' (you can simply use the first letter: `u`, `d`, `l`, `r`, or `q`).

   Your task: `usa.txt` is a map of the USA. Your viewport is initially in the middle of the map. Move the viewport until you can see New Hampshire *(hint: it's on the upper-right part of the country!)*.

2. Edit `img_viewer.py`, and notice that the following information is needed to move the viewport:
   - `x` and `y`, the bottom-right corner of the viewport
   - `width` and `height`, the viewport's dimensions.
   - `img_width` and `img_height`, the image's dimensions

   There are four functions to move the viewport:
   - `move_left`
   - `move_right`
   - `move_up`
   - `move_down`

Notice that each of them is passed `x`, `y`, `width` and `height`. These four variables describe the viewport.

Also notice that each of them modifies some of these variables, and returns them to the caller:

```
return (x, y, width, height)
```

3. Now open `viewport.py` and `img_viewer_oo.py` (oo stands for *Object-Oriented*).

   Notice how much simpler `img_viewer_oo.py` is. All of the work will now be done in `viewport.py`.

   In this lab, you will modify `viewport.py`.

4. Look at the constructor method, `__init__`, in the `Viewport` class. It creates the four instance variables for the viewport, `self.x`, `self.y`, `self.width` and `self.height`.

5. Implement the `set_img` method:

   Create two instance variables, `self.img_width` and `self.img_height`, from the two variables passed in, `width` and `height`.

   Notice that the constructor initially positioned the viewport at the top-left corner of the image: `self.x` and `self.y` are both 0 to begin with.

   Modify `self.x` and `self.y` inside the set_img method:

   ```
   self.x = self.img_width // 2  -  self.width // 2
   self.y = self.img_height // 2  -  self.height // 2
   ```

   Look at the original `img_viewer`. There are two lines very much like these, just before the `while True` loop.

6. Implement the other three 'move' methods: `move_down`, `move_left`, and `move_right`.

   To do this, look at how `move_up` changes the instance variables `self.x` or `self.y`. Compare this to the corresponding functions in `img_viewer.py`, and reproduce their behavior.

   Three things to watch for:
   - you only need to change `self.x` or `self.y`
   - you don't have to return anything, from these methods
   - *Important*: don't add any parameters passed into any of these methods.

They should only expect the `self` parameter.

7. Now, implement the display method. Adapt the code from the display function in `img_viewer.py`.

   Notice that this method has the usual `self` parameter, and also an ordinary parameter `img`. Look at the `main()` program in `img_viewer_oo.py`, and notice the call:

   ```
   port.display(img)
   ```

   Methods are just functions with a `self` argument. When python calls a method, it passes the object as the `self` argument. So the above call is equivalent to

   ```
   Viewport.display(port, img)
   ```

8. Finally, notice that the program keeps printing something like this:

   ```
   <viewport.Viewport instance at 0x02319558>
   ```

   This is the result of the main program's line:

   ```
   print (port)
   ```

   Why? `print()` tries to convert its arguments to string. The default string conversion is not very useful. Let's tell python to produce a more informative string.

   Add a new method, `__str__(self)`. Indent it carefully: line it up with the other methods in `viewport.py`.

   Put this in the `__str__` method:

   ```
   return 'viewport (x y w h)=({} {} {} {})'.format(\
             self.x, self.y,self.width, self.height)
   ```

   that back-slash `\` should be the LAST character in its line!

Turning in your work

To submit your work, go to mycourses.unh.edu, find cs417, and the lab, and upload

`viewport.py`. Submit whatever you have completed, at the end of the lab session. You can submit again until midnight, with no lateness penalty.