ACIT 2515 – Object Oriented Programming - Lab 6
Python Class Built-In Methods and Observer Design Pattern

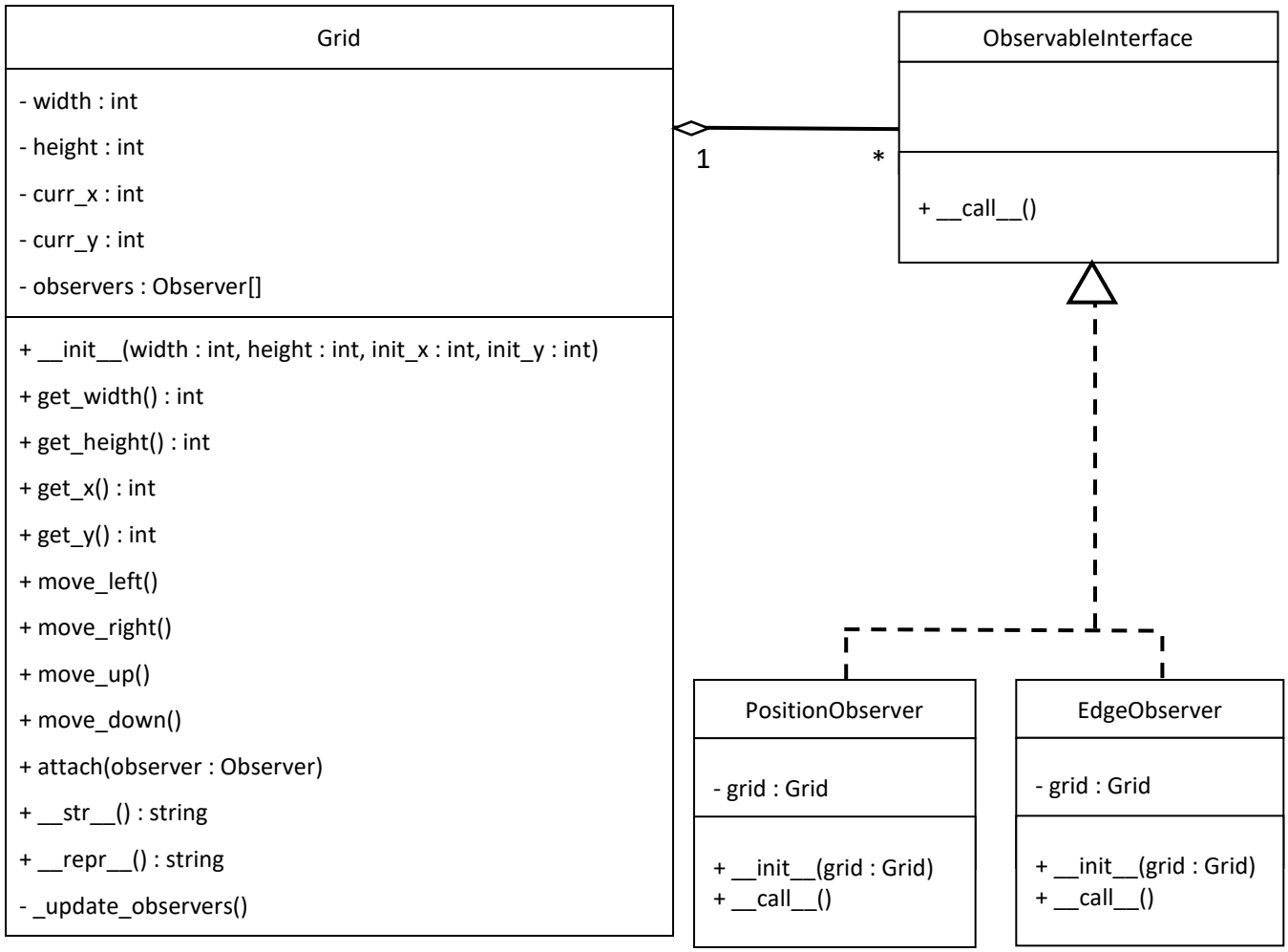
Instructor	Mike Mulder (mmulder10@bcit.ca) Also available on Slack.
Total Marks	10
Due Date	Recommend By the End of Class. No later than 9pm today (Feb. 15, 2019)

Goals

- Override several of the built-in methods in Python classes.
- Implement the Observer Design Pattern.

Overview

You will be implementing the following design for this lab, which follows the Observer design pattern.



Instructions

Grid (Core Class)

Create a Grid class (in a file called grid.py), based on the UML in the Overview, that holds a grid of a given width and height with a current x/y position in the grid. Conceptually, the coordinates for each cell in a grid of width 4 and height 5 is as follows.

(0,0)	(1,0)	(2,0)	(3,0)
(0,1)	(1,1)	(2,1)	(3,1)
(0,2)	(1,2)	(2,2)	(3,2)
(0,3)	(1,3)	(2,3)	(3,3)
(0,4)	(1,4)	(2,4)	(3,4)

- The size and initial position of the grid are set by the constructor.
    - Note that the x/y positions are zero-based, with 0/0 in the top left corner.
- There are accessor methods to get the width and height (get_width, get_height).
- There are accessor methods to get the current x/y position in the grid (get_x, get_y).
- There are methods to modify the current position:
    - move_left – Subtracts one from the x position but CANNOT be negative.
    - move_right – Adds one to the x position but CANNOT be greater than (width – 1).
    - move_up – Subtracts one from the y position but CANNOT be negative.
    - move_down – Adds one to the y position but CANNOT be greater than (height -1).
- The attach method adds an Observer object to the list of observers.
- The _update_observers method "calls" each Observer object in the list of observers. It should be called whenever the current x or y coordinate changes (i.e., from the move methods).

Also implement the following two built-in Python class methods in the Grid class:

- __str__ - Returns a string with the width and height of the grid and the current x/y coordinates.
- __repr__ - Returns a string with the width and height of the grid, the current x/y coordinates, the number of attached observers and the ID of the object (use the id function in Python).

PositionObserver and EdgeObserver

Create the PositionObserver observer class (in a file called position_observer.py), based on the UML in the Overview, which prints the current x/y coordinates of a Grid object when called. The format should look like – "X = 4, Y = 6"

Create the EdgeObserver observer class (in a file called edge_observer.py), based on the UML in the Overview, which print out one of the following if the current x/y coordinate of the Grid is on the edge:

- x coordinate is zero – "X is at the left edge of the grid!"
- x coordinate is at (width - 1) – "X is at the right edge of the grid!"
- y coordinate is zero – "Y is at the top edge of the grid!"
- y coordinate is at (height - 1) – "Y is at the bottom edge of the grid!"

main.py

In a main() function in main.py:

- Create a Grid object with a width of 7, height of 5 and initial x/y position at the centre (i.e., x=3/y=2 for zero based indices).
- Create PositionObserver and EdgeObserver objects (passing in the Grid object to the constructor).
- Attach the PositionObserver and EdgeObserver objects to the Grid object.
- Move the current x/y position of the Grid object to each of the edges of the grid (left, right, top, bottom) using the move methods:
    - The PositionObserver should print to the console each time the x/y position is updated in the grid.
    - The EdgeObserver should print to the console when the x/y position is at the edge of the grid.
- Print your Grid object (i.e., print the variable that holds your Grid object).
- Print the output from __str__ for the Grid object (i.e., str(grid)).
- Print the output from __repr__ for the Grid object (i.e., repr(grid)).

Demonstrate the output of your main function to your Instructor.

Submission

Once your output has been reviewed by your Instructor, upload the following files to the Lab 6 dropbox on D2L as a **zipfile** called **lab6.zip**:

- grid.py
- position_observer.py
- edge_observer.py
- main.py
- A screenshot of your console output