American University of Beirut
Department of Computer Science
CMPS 200 – Introduction to Programming
Lab 10: April 11 – April 15

Exercise 1. Intro to Classes - Rectangle

Write a class called *Rectangle* that represents a rectangular two-dimensional region. Your Rectangle objects should have the following methods:

- ***public Rectangle(int x, int y, int width, int height)*** constructor to construct a new rectangle whose top-left corner is specified by the given coordinates and with the given width and height.
- ***public int getHeight()*** a function that returns this rectangle's height.
- ***public int getWidth()*** a function that returns the rectangle's width.
- ***public int getX()*** a function that returns the rectangle's x-coordinate.
- ***public int getY()*** a function that returns the rectangle's y-coordinate.
- ***public void setHeight(int height)*** a function that changes the rectangle's height attribute.
- ***public void setWidth(int width)*** a function that changes the rectangle's width attribute.
- ***public void setX(int x)*** a function that changes the rectangle's x-coordinate attribute.
- ***public void setY(int y)*** a function that changes the rectangle's y-coordinate attribute.
- ***public String toString()*** a function that returns a string representation of the rectangle, such as "Rectangle[x=1,y=2,width=3,height=4]".
- ***public Rectangle union(Rectangle rect)*** a function that returns a new Rectangle that represents the area occupied by the tightest bounding box that contains both this Rectangle and the given other Rectangle. Your method should not modify the current Rectangle or the one that is passed in as a parameter; you should create and return a new rectangle.
- ***public Rectangle intersection(Rectangle rect)*** a function that returns a new rectangle that represents the largest rectangular region completely contained within both this rectangle and the given other rectangle. If the rectangles do not intersect at all, returns null. Your method should not modify the current Rectangle or the one that is passed in as a parameter; you should create and return a new rectangle.

Upon a class gets instantiated into an object, ie {Rectangle rec1 = new Rectangle(0,0,3,3);} , the first method that will be automatically called is the constructor. We will use the constructor to declare and instantiate all the object's variables:

- Start by defining the attributes of the class (x,y,height,width) as private fields.
- In the constructor body, you will pass the values of the parameters into the attributes ex: Public Rectangle(double x, double y, ...) { this.x = x; this.y = y; ... }. The *this* keyword refers to the object attributes to avoid referring to the same local attributes.
- Next you will create 4 getter methods: a getter method is a gateway to access the value of a private attribute with read-only privileges. In the body of the method, you will simply return an attribute's value.
- Next you will create 4 setter methods: a setter method is a gateway to modify the value of a private attribute with write-only privileges. In the body of the method, you will only change attribute's value.
- Next you will implement the toString() method which returns a String representation of the rectangle.
- While implementing the union method, and figuring out the coordinates and dimensions of the new rectangle, you can instantiate a rectangle object inside the method using new and pass the calculated attributes to it and then return the whole object.

Write a driver program to test all your methods.

Exercise 2. Sorting Algorithm

Write a program `BubbleSort.java` that implements the *Bubble Sort* algorithm to sort an array but in the **descending** order. The Bubble Sort algorithm is a simple sorting algorithm that works by repeatedly stepping through the array to be sorted, comparing each pair of adjacent elements and swapping them if they are in the wrong order. The pass through the array is repeated until no swaps are needed, which indicates that the list is sorted.

You should create an array of size 10, such that the elements are randomly generated (between 0 and 1000). The program should take as command line argument the number of times to perform bubble sort (for new random values in the array). *PS*: You are NOT allowed and should NOT use the method `Arrays.sort`. <u>Example:</u>

```
> java BubbleSort 3

before: 45 87 39 32 93 86 12 44 75 50
after : 93 87 86 75 50 45 44 39 32 12


before: 450 871 392 321 932 865 121 441 755 500
after : 932 871 865 755 500 450 441 392 321 121


before: 145 187 139 132 193 186 112 144 175 150
after : 193 187 186 175 150 145 144 139 132 112
```