**Subject 1 - OOP practical test**

Time: 1h 30min

29 May 2020

1. Define 2 classes: **Rectangle** and **Point**.

A **Point** contains two member variables, its x and y coordinates. A **Rectangle** contains two member variables of type **Point**, representing type top-left and bottom-right corners coordinates.

1. Overload the insertion (operator <<) and extraction (operator >>) operators for these two classes. When reading a Point from the stream, you can read its coordinates separated by spaces. Similarly, for reading a Rectangle you can read the coordinates of its top left and bottom right coordinates.
2. Overload the bitwise and operator (operator &) for the Rectangle class. The operator will return *true* if the two rectangles intersect (overlap) and *false* otherwise.
3. Now design a new class **RectangleDrawing** to simulate how several **Rectangle** objects are drawn on the screen.

The class should have at least the following methods:

* void addRectangle(Rectangle r) - this method adds a rectangle to be drawn on the screen. You need to store the parameter *r* in some sort of container inside the **RectangleDrawing** class.
* void displayRectangles() - this method will display the **Rectangle**s which will be drawn on the screen. The following rule must be ensured: if two rectangles intersect, only the latter rectangle should be drawn on the screen (i.e. the rectangle which was added last using the addRectangle(Rectangle r) function).

1. Display how many times the copy constructor was called for the **Rectangle** class..   
   Hint: You might need to use static class variables for this.

Example

**If the user adds the following rectangles to be drawn:**

Rectangle(Point(10, 10), Point(20, 20))

Rectangle(Point(100, 200), Point(200, 300))

Rectangle(Point(150, 200), Point(300, 300)) -- overlaps with Rectangle(Point(100, 200), Point(200, 300))

Rectangle(Point(30, 30), Point(75, 75))

Rectangle(Point(400, 600), Point(450, 650))

**then when calling the *displayRectangles()*****function*,* the following rectangles will be displayed:**

Rectangle(Point(10, 10), Point(20, 20)), Rectangle(Point(150, 200), Point(300, 300)) , Rectangle(Point(30, 30), Point(75, 75)), Rectangle(Point(400, 600), Point(450, 650))

**Grading**

Ways to lose points (max 1 point), even if your program work perfectly:

* Your class ignores encapsulation, for example you define everything as public. **(- 0.2 p)**
* You don’t use meaningful names for your class members and methods: for example, your methods are called void m1(), void m2(char \* s) **(-0.2 p)**
* You don’t use indentation and you don’t separate the implementation into a *header* and a *source* file. For example you write the class MyString in a header file on a single line. **(-0.2 p)**
* You don`t mark the required methods as *const* (for example the getters) in their declaration, to ensure that you cannot change any member data through that method. **(-0.2 p)**
* You don’t use a consistent coding style. **(-0.2p)**

**If your project does not compile, it will not be graded!**

**As a rule of thumb, don’t write more than 10 lines of code without compiling your project.**

**If your project does not compile, fix the errors and only when your project is error free you should move on to the next point.**

Grade

|  |  |
| --- | --- |
| Class declaration, constructors, destructor, & operator overloading | 2 |
| Class association, correct access modifiers for variables | 1 |
| Static members (number of totally created and living instances) | 1 |
| Insertion and extraction operator overloading (operator<< and operator>>) | 1 |
| addRectangle(Rectangle r) function + storing the rectangle in a container | 2 |
| Display the Rectangles which will be displayed on the screen | 1 |
| Coding style (see Ways to lose points) | 1 |
| Default | 1 |
| **Total** | **10** |

Overload stream operators:

<https://www.tutorialspoint.com/cplusplus/input_output_operators_overloading.htm>

Overload & operator:

bool operator&(const Rectangle &r)

Suggestion: Two rectangles don’t intersect if one of the rectangles is above, below, to the left or to the right of the other rectangle.