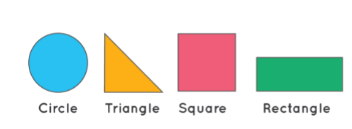
# C5- S2 – PRACTICE



 Your project must include a tsconfig.json file and build JS files in /dist folder

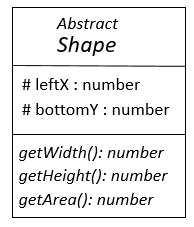
 Each class must be in a separate file *(example: Rectangle.ts)*

 You also need to create a Main.ts file to test all your shapes

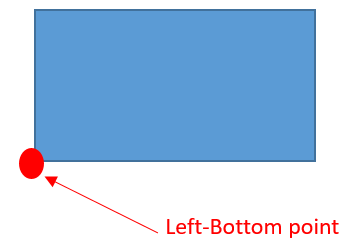
**A) The abstract shape**

We want to manage shapes such as triangles, squares, rectangles, circles…

We first define an **abstract** class Shape as follows:



* leftX and bottomY are the position of the left bottom point of shapes.



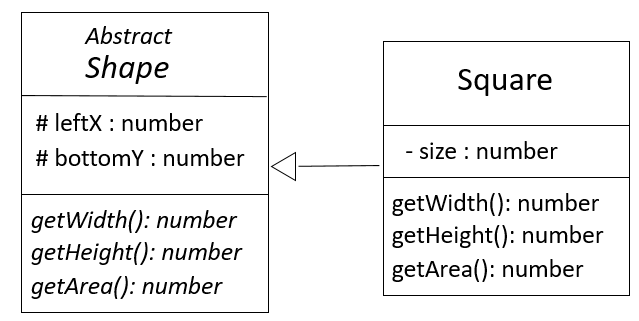
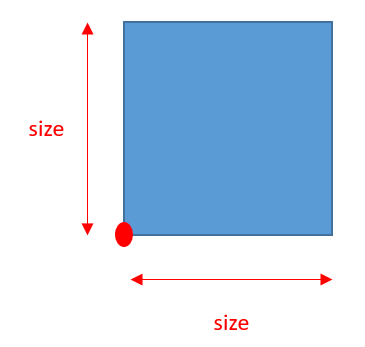
* getWidth, getHeight, getArea are abstract, because they depends on the specific shape.

**Q1** Implement this class Shape

**B) The square**

A square:

* Inherits from the abstract Shape class
* Is defined by its **left-bottom point** and size

**Q2** Implement the class Square

*Note: the constructor has following parameters: leftX, bottomY and size.*

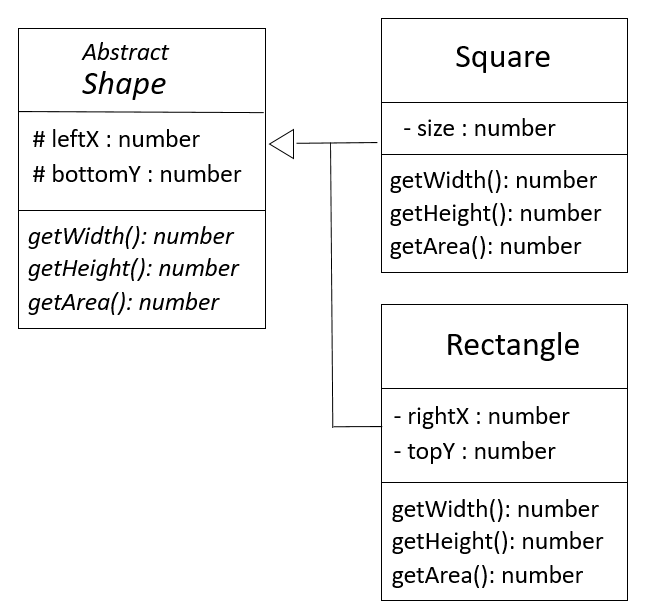
**Q3** Implement the required methods (required by the abstract class):

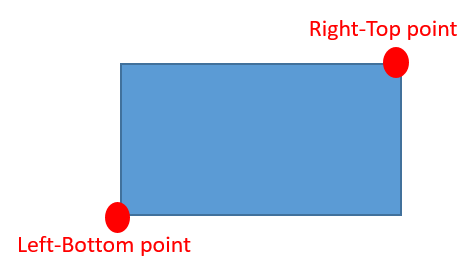
* getWidth
* getHeight
* getArea

**C) The rectangle**

A rectangle:

* Inherits from the abstract Shape class
* Is defined by its **left-bottom point** and its **right-top points**





**Q4** Implement the class Rectangle

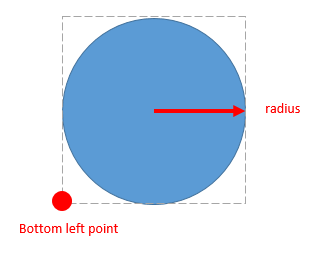
*Note: the constructor has following parameters: leftX, bottomY and rigthX, topY.*

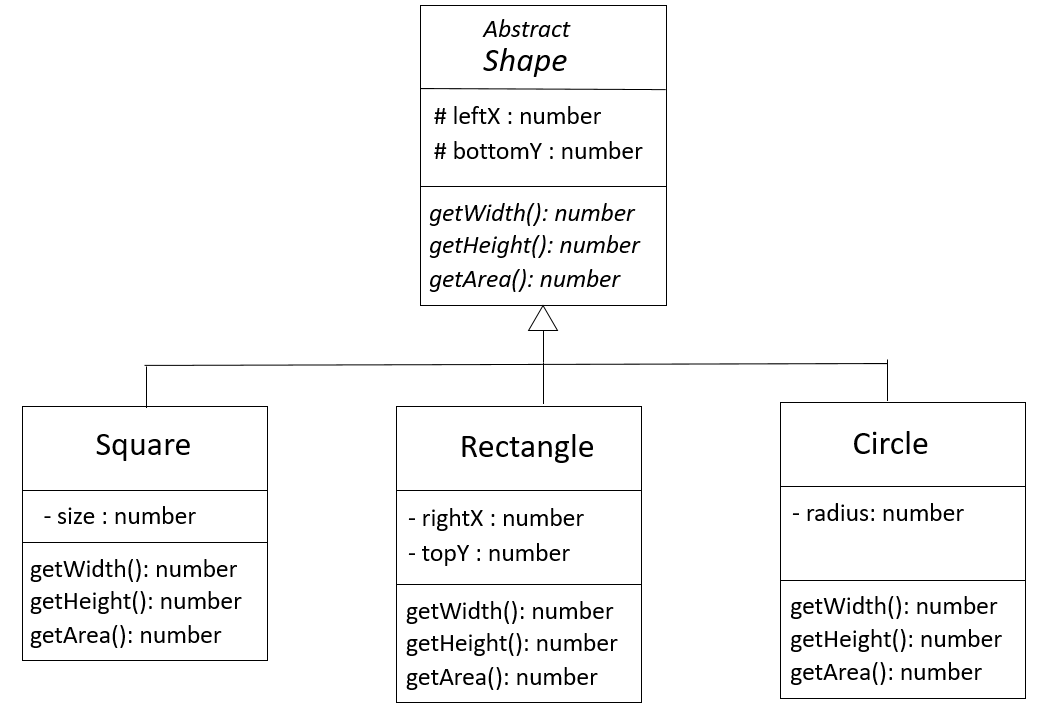
**Q5** Implement the required methods (required by the abstract class):

* getWidth
* getHeight
* getArea

**C) The circle**

A circle:

* Inherits from the abstract Shape class
* Is defined by its **left-bottom point** and its **radius**
* 



**Q4** Implement this class Circle

*Note: the constructor has following parameters: leftX, bottomY and radius*

**Q5** Implement the required methods (required by the abstract class):

* getWidth
* getHeight
* getArea