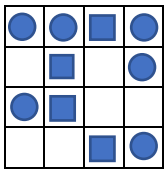
Algorithms and Data structures

OO review, ADT

You need to develop a basic system that allows geometrical shapes to be contained in a board. Imagine something like this:



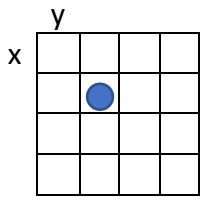
The requirements are that a given board can only hold any type of shape, for example, circles and rectangles. Other shapes (sub-classes of Shape) may be developed in the future and these should also work with the system. The board is essentially a grid and shapes are held in positions on the grid. There can be empty positions. You are given a project containing a Shape superclass and a tester class. You must write a Circle, Rectangle and Board class to pass the tests.
These are more detailed specifications of the classes. The tests and Shape class also contain information that you need.
Classes:
Circle:

subclass of Shape
has a radius of type double (and of course a colour)
equals: two circles are equal if their areas and colours are equal (see slides in w1-pr-introduction.pdf)

Rectangle:

Subclass of Shape
Has a width and length (dimensions)
Equals: two Rectangles are equal if their dimensions and colours are equal

Board: is able to store any subclass of Shape in a two-dimensional (2D), square array. The 2D array below is a representation of a circle in position x = 1, y = 1 (remember the array is zero-based). This is just to help you imagine the board – your system will not do any drawing or show a board and shapes on a screen. It purely stores the Shape objects in a data structure.



Board fields:
A Board will be defined by its size. Knowing the size you can initialize a 2D array. A board is always square. Tip: in the constructor you only need to pass the size and then you can initialize the data structure to store the shapes.
You should use a 2D array of Shape that contains the actual shapes. You will need other fields to maintain state and iterate over the board.
Board methods:

add: you need to be able to add a Shape (subclass) to a given (x,y) grid position on the Board. When the specified grid position is empty, the object shall be added and "true" should be returned from the method. When the position was occupied already, nothing should be added, and false should be returned. If an illegal (x,y) grid position is specified, an `IndexOutOfBoundsException` shall be thrown with an appropriate message:

"Position ?,? is not available on a board of size ?"

(fill out the question marks with appropriate numbers.)

remove: this method should return the Shape found at the supplied position (x, y) or null if the object at that position is null. If an illegal (x,y) grid position is specified the same exception shall be thrown as for the add method.

getGrid(): this returns a 2D array of Shapes.

getArea(): returns the cumulative total area of all shapes on the board.

Other comments and recommendations:
You should not change ShapeTest or Shape.