# Exercises for Chapter 8 – The C# Classes

## Exercise 1

***Shapes Exercise***

* In this exercise we’ll enhance the **Rectangle** class.
* You should create a graphical object hierarchy, including **Rectangle** and **Circle** classes, with a common base class: **Shape**.
* Different shapes should support their specific functionality and properties (e.g. Width, Radius).
* However, all shapes should provide the following common functionality:
  + All shapes should have a center point, X,Y (private member, might have access methods/properties)
  + **Resize** - Changing size by a given percentage. (s1.resize(5) should resize rectangle Width and Height by 5 precents, and Circle Radius by 5 percnet. What should it do in base class Shape?)
  + **Move** - Moving the center of the object in the plane.  **(move the center point to a new X,Y)**
  + **Print** - Printing the following shape details:
    - geometric form ("I am a rectangle"…. Etc)
    - spatial location (The X and Y coordinates)
    - Information specific to that shape (e.g. radius for a circle).
    - Note:
* The main program should create and manipulate different graphical objects.

## Exercise 2

***Enhanced Shapes Exercise***

* To enhance our graphical system, you should add a compound shape.
* A compound shape is created as a result of selecting a few shapes together on a graphical system.
  + What should this compound shape interface provide?
  + How should it be implemented?
  + Suggestions: use either an internal array of shapes, or a List<shape>. List is a simple array like interface, that allows adding shapes and iterating using foreach. See documentation if desired.
* Enhance your main program to create and manipulate all graphical system components

For compound shape (picture) please add:

* Indexer [] (getting the ith shape in the compound shape)
* If the user gives an index out of bounds, throw exception
* Try and add a "Sort" method that sorts the internal shapes array/list. Why do you get an exception during run time? (Will be resolved when we deal with interfaces).