* EditorController/CanvasController is the center processing controller.
* all primitive and composite(group) components extends IDrawable (for runtime polymorphism)
* To add more Drawable shapes, we just need to implement IDrawable, so design is extendable
* Each Component class will have only properties related to that specific component, for example in case of a circle all we want to know is where the center is, radius and its color. Other operations like save, edit, delete, move should not be a part of Circle class. Of course while performing such operations on circle, its properties will be required (accessed), but these operations will not be in the circle class itself.

(The circle object will be passed to such operation class via central controller to access its properties for such calculation)

* Editing Object will directly change the property value for that component’s corresponding property.(e.g. right click and change the color of circle, will directly change the color property of that object of circle class, similarly corners for rectangle can also be set.
* As we had discussed, Group is composed of other primitive objects or another Group itself.
* A group object will have a collection of IDrawable Objects, which are part of that group.
* Each primitive object, if it is a part of a group, will have a reference to its group object. So when that object is selected/deleted/moved/ copied, using its group object reference, which in turn has reference to other primitive objects in that group, all other objects will be selected/deleted/moved/ copied respectively, or else this reference will be null.
* For ungroup function(removes one level of grouping), traverse all the primitive objects in the IDrawable collection of Group class, set their parent group reference to null(perform only for primitive, as it is given only one layer of grouping is removed)
* For ungroup all, perform the above operation recursively (i.e. for each sub-group too), when done set the IDrawable collection object to null and release the Group Object for garbage collection.
* All the operation/functionalities are divided into two parts, based on what they work on, as file based operations and component/object based operations.
* Again to used runtime polymorphism plus for future extensibility, all these operations are realizing IFileOperations interface or IComponentOperations Interface, as shown in the diagram.
* Each of these operations, while implementing will be quite complex also their purpose is completely different from the other operations hence a different class. This design also supports extensibility. (Open close principle, the class should be closed for modification but the design should be open for extension.)
* AbstractController is not an interface, but an abstract class, which can have all common attributes and implementations
* In case of a group, we are registering/adding primitive type to group, an changes to the group is pushed to all the registered primitive components, it’s a kind of a Observer design pattern