Liskov Substitution Principle

This principle is just an extension of the Open Close Principle.

It means that we must make sure that new derived classes are extending the base classes without changing their behavior.

Functions that use pointers or references to base classes must be able to use objects of derived classes without knowing it.

 Or

If any module is using a Base class, then the reference to that Base class can be replaced with a Derived class without affecting the functionality of the module.

Or

While implementing derived classes, need to ensure that the derived classes just extend the functionality of base classes without replacing the functionality of base classes

E.g.

If we are calling a method defined at a base class upon an abstracted class, the function must be implemented properly on the subtype class.

// If any module is using a Base class then the reference to that Base class can be replaced with a Derived class without affecting the functionality of the module.

// Or

// While implementing derived classes, one needs to ensure that, derived classes just extend the functionality of base classes without replacing the functionality of base classes.

**class** Rectangle

{

**protected** **int** mWidth = 0 ;

**protected** **int** mHeight = 0;

**public** **virtual** **void** SetWidth(**int** width)

    {

        mWidth = width;

    }

**public** **virtual** **void** SetHeight(**int** height)

    {

        mHeight = height;

    }

**public** **virtual** **int** GetArea()

    {

**return** mWidth \* mHeight;

    }

}

// While implementing derived class if one replaces the functionality of base class then,

// it might results into undesired side effects when such derived classes are used in existing program modules.

**class** Square : Rectangle

{

    // This class modifies the base class functionality instead of extending the base class functionality

    // Now below methods implementation will impact base class functionality.

**public** **override** **void** SetWidth(**int** width)

    {

        mWidth = width;

        mHeight = width;

    }

**public** **override** **void** SetHeight(**int** height)

    {

        mWidth = height;

        mHeight = height;

    }

}

**class** LiskovSubstitutionPrincipleDemo

{

**private** **static** Rectangle CreateInstance()

    {

        // As per Liskov Substitution Principle "Derived types must be completely substitutable for their base types".

**bool** SomeCondition = **false**;

**if** (SomeCondition == **true**)

        {

**return** **new** Rectangle();

        }

**else**

        {

**return** **new** Square();

        }

    }

**public** **static** **void** LSPDemo()

    {

        Console.WriteLine("\n\nLiskov Substitution Principle Demo ");

        Rectangle RectangleObject = CreateInstance();

        // User assumes that RectangleObject is a rectangle and (s)he is able to set the width and height as for the base class

        RectangleObject.SetWidth(5);

        RectangleObject.SetHeight(10);

        // Now this results into the area 100 (10 \* 10 ) instead of 50 (10 \* 5).

        Console.WriteLine("Liskov Substitution Principle has been violated and returned wrong result : " + RectangleObject.GetArea());

        // So once again I repaet that sub classes should extend the functionality, sub classes functionality should not impact base class functionality.

    }

}