SOLID

S – Single Responsibility Principle

A class should have only a single responsibility

An object should have:

- one primary responsibility
- one reason to exist
- one reason to change and that reason entirely encapsulated within one class

Never one object doing the job of many entities
Never split one entity across several objects

A class can have multiple behaviors, but all those behaviors are oriented around that core reason for existence and everything it does should support that.

Bad classes could make it possible for objects to print or save themselves

Implementation: Division of code

Example – Bad Class (Print/Save)

```
class Book {
   function getTitle() {
      return "A Great Book";
   }

   function getAuthor() {
      return "John Doe";
   }

   function turnPage() {
      // Reference to next page
   }

   function printCurrentPage() {
      System.out.println( "current page content" );
   }

   function saveBook() {
      // Save to a file
   }
}
```

O – Open / Closed Principle

Open for extension, but closed for modification.

Do not change existing working code, but add new code for new requirements.
Can add, but try not to change code that works already.

Implementation: Use inheritance or interfaces

Example – Bad method

```
public double Area(object[] shapes)
{
```

```
    double area = 0;

    foreach (var shape in shapes)
    {
        if (shape is Rectangle)
        {
            Rectangle rectangle = (Rectangle) shape;
            area += rectangle.Width*rectangle.Height;
        }
        else
        {
            Circle circle = (Circle)shape;
            area += circle.Radius * circle.Radius * Math.PI;
        }
    }

    return area;
}
```