**Single Responsibility Principle**

In programming, the Single Responsibility Principle states that every module or class should have responsibility over a single part of the functionality provided by the software. There is a saying in *Japanese* ‘one thing at a time’. So basically Single Responsibility Principle follows the same thing.

Following example violates the single responsibility principle so let’s observe it:

class User

{

void CreatePost(Database db, string postMessage)

{

try

{

db.Add(postMessage);

}

catch (Exception ex)

{

db.LogError("An error occured: ", ex.ToString());

File.WriteAllText("\LocalErrors.txt", ex.ToString());

}

}

}

We notice how the CreatePost() method has too much responsibility, given that it can both create a new post, log an error in the database, and log an error in a local file.

This violates the single responsibility principle.

Let’s try to fix it.

class Post

{

private ErrorLogger errorLogger = new ErrorLogger();

void CreatePost(Database db, string postMessage)

{

try

{

db.Add(postMessage);

}

catch (Exception ex)

{

errorLogger.log(ex.ToString())

}

}

}

class ErrorLogger

{

void log(string error)

{

db.LogError("An error occured: ", error);

File.WriteAllText("\LocalErrors.txt", error);

}

}

By abstracting the functionality that handles the error logging, we no longer violate the single responsibility principle.

Now we have two classes that each has one responsibility; to create a post and to log an error, respectively.