**Extract Class**

**Motivation**

Classes can have methods that do too many different things. These methods may not relate to the original purpose of the class. This is problematic as it is not immediately obvious as to what a class does if the methods do not serve the same purpose. This makes the code less readable if future work is done on it. To achieve this, methods can be grouped according to their functionality and assigned to their appropriate class. This ensures each class has its own functionality for communicating its intended purpose. This refactoring method also simplifies the code base, allowing developers to quickly determine the functionality of a class by looking at its name. The Abstract Factory pattern can be used in this case as it instantiates multiple classes from one class.

**Methods**

1. Identify class with methods unrelated to its purpose.
2. Create a new class to hold these methods and name it after its intended purpose
3. Move the appropriate methods to the new class
4. Change callers and references to the new class as needed.

**Sample Code to refactor**

A class draws a shape when a user enters the proper coordinates. If an error occurs, it is logged into a text file. These functionalities are unrelated and can be separated into their own classes.

abstract class Shape

{

public void Draw()

{

try

{

// draw

}

catch (Exception e)

{

LogError(e);

}

}

public static void LogError(Exception e)

{

File.WriteAllText(**@"c:\Errors\Exception.txt"**, e.ToString());

}

}