Open Closed Principle (OCP)

Classes, modules, functions, etc. should be open for extension, but closed for modification. that is, such an entity can allow its behavior to be extended without modifying its source code.

*Software entities should be open for extension, but closed for modification.*

*—Bertrand Meyer*

Robert C. Martin has defined the OCP in many different writings over the years. A more verbose version has been chosen here to contrast with the brief original:

*”Open for extension*.*” This means that the behavior of the module can be extended*.*As the requirements of the application change, we are able to extend the module with new behaviors that satisfy those changes*.*In other words, we are able to change what the module does.*

*“Closed for modification.” Extending the behavior of a module does not result in changes to the source or binary code of the module*.*The binary executable version of the module, whether in a linkable library, a DLL, or a Java .jar, remains untouched.*

*—Robert C. Martin, Agile Software Development: Principles, Patterns, and Practices*

*(Prentice Hall, 2003)*

Create a Base class with Required functionality, and ensure we will not modify that class. (Closed for modification)

Create a Derived class by inheriting the Base class for extension (Open for modification)

The

goal is a very useful guideline that will enable you to create code that is more adaptive to future

changes.

// Here DataProvder is open for extension (extends to Sql, Oracle, Oledb Providers and so on..) and closed for manipulation

**abstract** **class** DataProvider

{

**public** **abstract**  **int** OpenConnection();

**public** **abstract**  **int** CloseConnection();

**public** **abstract** **int** ExecuteCommand();

}

**class** SqlDataProvider : DataProvider

{

**public** **override** **int** OpenConnection()

    {

        Console.WriteLine("\nSql Connection opened successfully");

**return** 1;

    }

**public** **override** **int** CloseConnection()

    {

        Console.WriteLine("Sql Connection closed successfully");

**return** 1;

    }

**public** **override** **int** ExecuteCommand()

    {

        Console.WriteLine("Sql Command Executed successfully");

**return** 1;

    }

}

**class** OracleDataProvider : DataProvider

{

**public** **override** **int** OpenConnection()

    {

        Console.WriteLine("Oracle Connection opened successfully");

**return** 1;

    }

**public** **override** **int** CloseConnection()

    {

        Console.WriteLine("Oracle Connection closed successfully");

**return** 1;

    }

**public** **override** **int** ExecuteCommand()

    {

        Console.WriteLine("Oracle Command Executed successfully");

**return** 1;

    }

}

**class** OledbDataProvider : DataProvider

{

**public** **override** **int** OpenConnection()

    {

        Console.WriteLine("OLEDB Connection opened successfully");

**return** 1;

    }

**public** **override** **int** CloseConnection()

    {

        Console.WriteLine("OLEDB Connection closed successfully");

**return** 1;

    }

**public** **override** **int** ExecuteCommand()

    {

        Console.WriteLine("OEDB Command Executed successfully");

**return** 1;

    }

}

**class** OpenClosePrincipleDemo

{

**public** **static** **void** OSPDemo()

    {

        Console.WriteLine("\n\nOpen Close Principle Demo ");

        DataProvider DataProviderObject = **new** SqlDataProvider();

        DataProviderObject.OpenConnection();

        DataProviderObject.ExecuteCommand();

        DataProviderObject.CloseConnection();

        DataProviderObject = **new** OracleDataProvider();

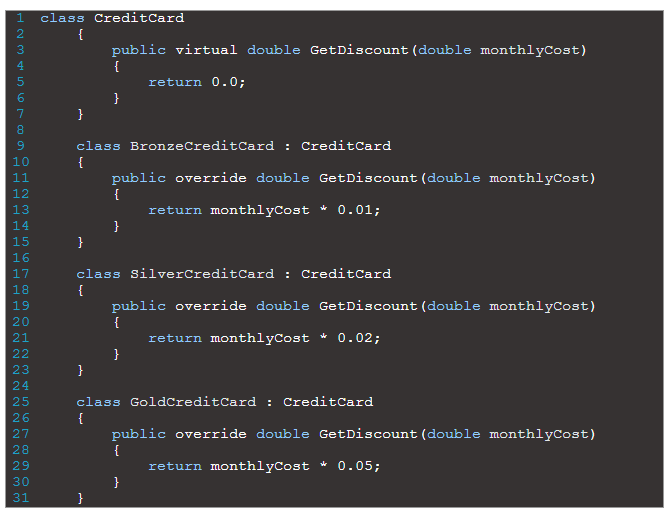
        DataProviderObject.OpenConnection();

        DataProviderObject.ExecuteCommand();

        DataProviderObject.CloseConnection();

    }

}



**The code above might seem to be good. However, if we wanted to offer discounts to other type of creditcard, we’ll have to modify the existing code to accomplish it. For example, if we wanted to offer specific discounts to other creditcard types, we’ll have to do so by adding more IF statements to the GetDiscount method in the example above. By modifying the existing code we’ll be breaking the Open/Closed principle.**

