Holly Wood Principle in Coding from Traditional

Mr S. MOHAN[1], Mrs V. RADHA[2], Dr M. VARATHARAJ[3]
Professors of Computer Science and Engineering Department & Electrical and Electronics Engineering Department, VSB College of Engineering Technical Campus, (An Autonomous Institution), Ealur Pirivu, Solavampalayam PO, Kinathukadavu, Coimbatore, Tamailnadu, India.

***Abstract*—In this paper we will discuss about holly wood principle in coding that means inverse of control (IoC). Because it is just opposite of traditional programs. while write a program in traditional which more complex in write .in holly wood or inverse of control (IoC) is the dependencies injection write down the application. the program will appear suitable program to that application. It is more flexible and reusable about the object and source, that means from input to output in inverse of control (IoC) from output to input like as .inverse of control is the use interface or UI are easily under stood by the user because when the frame work dedicated the flow of control and call you code appropriate time rather than code initializing calls to the frame works .because It has the flexibility and modularity of entered frame work for user handling that code from the object to source . but in traditional means source to object.**

***Index Terms*—IoC.,DI,DIP.,**

I. INTRODUCTION

Discuss about what is dependency injection (DI). from the frame work code the user can create object of the screen and automatically appeared suitable code to the object. that source code no need to understand the code only object create on the screen what style? what type? They will give suitable code are program to that object, that also frame work or dependency injection.

In software development means ("don't call us, we will call you"). This is principles of inverse of control (IoC). Where a high level frame work or control or container control flow of execution calling you custom code where it is need instead of code initialing call to library the frame work handle the control rather than your code control the frame work. that means holly wood principle in the software development

Unlike traditional integrated development environment (IDE'S)visual studio code priority swift code editing you can personalize it through variety of extension to meet your specific requirements.

II. DEPENDENCY INJECTION

Dependency injection is a software design pattern that promote loose coupling between class and dependency it is form of inversion of control (IoC) where the responsibility of creating and managing dependency is shifted from the different classes
to an external entity often an (IoC) container or DI frame work.

Loose coupling:

Instead of class directly creating and dependency. the dependency is provided into the class from the outside. The reduce tight coupling making the code more flexible, testable and maintainable.

Inverse of control:

The control over the dependency creates and management is inverted the client class no longer control to create its services instead of an external entity (the rejection) handle it.

Interface over the concrete implementation:

DI Encourager programming against interface rather than concrete classes this allows for easier swapping of implementation without modifying the client code.

Types of dependency injection:

Dependency is provided through the class constructor as parameters. this is the most common and recommended type of DI as it ensures that all required dependency are available when the object is created.

Code:

```
Public class myservice
{
Print read only Ilogger- logger
Public myservise (Ilogger-logger)
```

```
{
-logger=logger;
}
Public void do something ()
{
-logger. Log information ("do something …")
}
}
```

Prosperity injection or setter injection

Dependency is provided through the public property of the class. This allows from the optional dependency that can be aligned often object creation.

Code:

```
Public class myservice
{
Public Ilogger logger {get;set;}
Public void dosomething ()
{
Logger? log information ("dosomething")
}
}
```

Method injection

Dependency are passed as parameter to specific method that required them this is useful where a dependency is only needed for a particular method call.

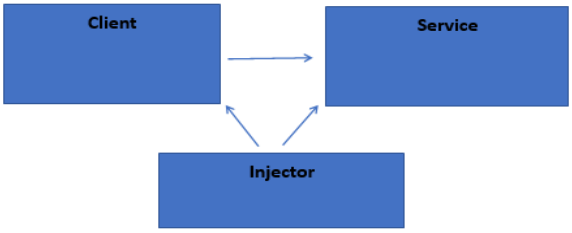
Code:

```
Public class myservice
{
Public void dosomething (Ilogger Logger)
{
Logger. Log information ("dosomthing…")
}
}
```

Implement in c#:

C# application especially ASP.NET core after language built in DI framework or third party of IoC container autofill injection or unity to manage the dependency registration and restart. This frame work allow you to configure then interface and then concrete implementation and mapped and automatically injected the concert dependency when object are initiated

Dependency injection:



The most common to achieve IoC where dependency and injected into object by an external entity rather than objects creating the own dependency.

III. SERVICE LOCATOR PATTERN

Another approach to the general service location object provided access to the request service.

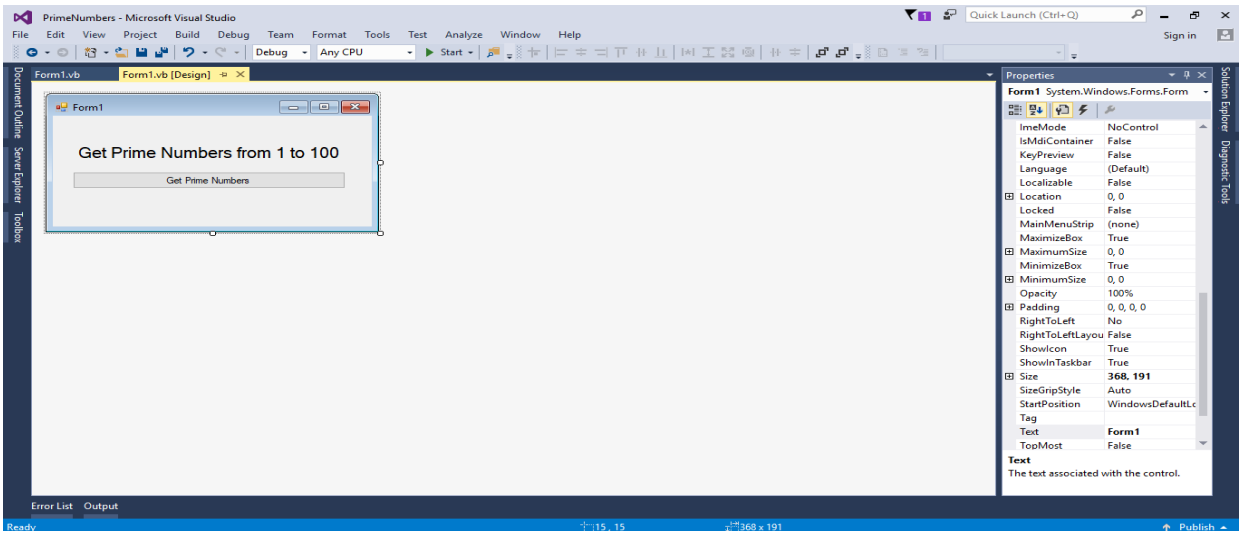
The design principle and design pattern and often which understood by the test of inverse of control (IoC). DIP (dependency inversion principle). Then all are design pattern and IoC container are framework.

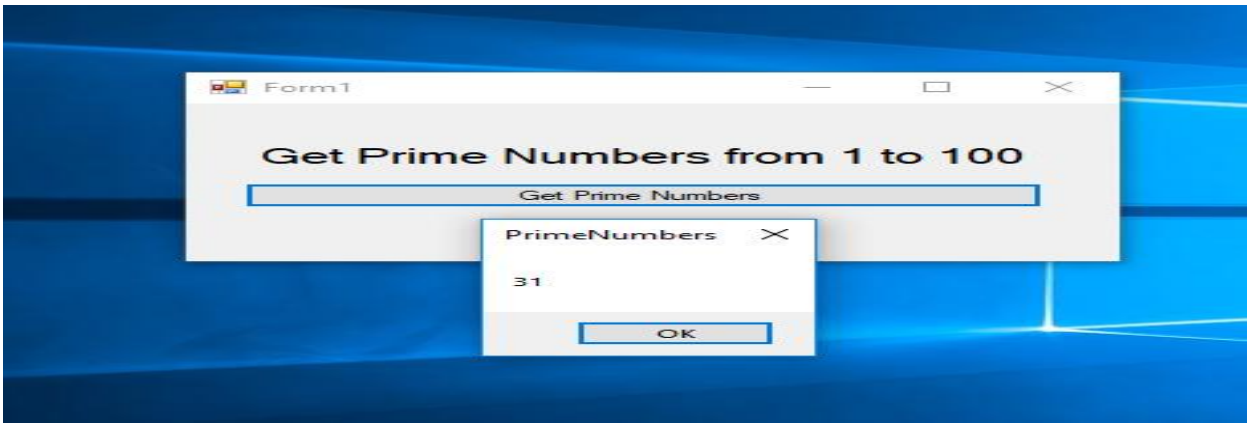
Example for traditional in c program:

```
#include<stdio.h>
Int main()
{
Int a,b,c;
Printf ("/n enter a =");
Scanf ("%d ",&a);
Printf ('/n enter b=");
Scanf("%d"'&b);
C=a+b;
Printf('/n sum aand b=",c);
Return 0;
}
```

Holly wood principles in dot-net.

```
Dim prime, numbers, i As Integer
      prime = 1
MsgBox("The prime Numbers are : ")
    I For numbers = 1 To 100
        For i = 2 To numbers - 1
          If numbers Mod i = 0 Then
             prime = 0
             Exit For
          Else
             prime = 1
          End If
        Next
        If prime = 1 Then
          MsgBox(numbers)
        End If
      Next
```





IV. CONCLUSION

Holly wood principle means provided the script that the frame work call when it is time to execute (‘don’t call us, we will call you”) the concept control gives the design pattern and IoC call the control the frame work continues. Which is also called object program and source program.

The object is the design pattern and source program is frame work and time to execute because it means multi way continue code appeared in the object and source compare to the traditional code.in traditional means when the output created their own dependency. IoC shift there responsibly to an external frame work but in IoC allowing for each testing via mock object dynamic swapping of implement and more organized, organized code based.

Because its flexibly reusability from the object mock create by that. created by the source program execute the source type accepted created to flexibility and reusability and maintainability from the object. but traditional is one-way each and every time created the program obey the object. but in holly wood principles not that single type its principles to flexible to create responsibility source code from the object so that is flexible compare to the traditional.

REFERENCE

[1] Shubham Dubey, et al. “Optimized C# Open Authentication System In.Net”, Conference of Omni Spectrum, Lambert Academic Publication, Germany, 2018.

[2] H. Komara, et al. “Dynamic generic web pattern for multi-platform,” International Conference on Data and Software Engineering (ICoDSE), 2016, pp. 1 - 5.

[3] Brian Hall, et al. “Advantages and Uses of ASP.NET development services.”, *Journal of Computer and Actuator Networks.* Vol. 9. 10.3390/jsan9040055, 2020.

[4] Karim Hadjar, et al. "C# Language Specification," *Addison-Wesley Longman Publishing Co., Inc.* pp. 7209, 2020.
[5] Neelamadhab Padhya, et al. "Utility of an object-oriented metrics component: examining the feasibility of.Net and C #", *International Journal of Mobile Learning and Organization* vol. 12, no. 3, 2018, pp. 263 - 279.
[6] Naman M. et al. "C# and the.NET Framework", *Neurocomputing*, 2020.
[7] David Roy Hanson, et al. "A research C# compiler," 14th Conference of Software Practice and Experience, vol. 34 (13): 1211 - 1224, 2011.
[8] Don Syme, et al. "Design and Implementation of Generics for the.NET Common Language Runtime", Conference of Microsoft Research, Cambridge, U.K
[9] Andrew Kennedy, et al. "Securing the.NET Programming Model (Industrial Application)", Conference of Microsoft Research, Cambridge, U.K
[10] Bhattacharya, et al. "Predicting Emergent Properties of Component Based Systems," International Conference on Commercial-off-the-Shelf (COTS)-Based Software Systems (ICCBSS'07), 2007, pp. 41 - 50.
[11] Barmase, Gaurav V et al. "GUI based Industrial Monitoring and Control System" *Journal of Information Technology and Digital World 3*, no. 2 (2021): 108 - 117.