Beckhoff PLC (TwinCAT 3)

PLC programming by using OOP approach

Most PLC programmers use traditional programming style (at least around me) instead of OOP style but sometimes we have to do that because of devices which we are using. OOP is easily applied to Beckhoff PLC and this gives us significant features , such as flexibility , avoiding repeating code or solid programming standards.

In this paper , an example which is a database example is done to demonstrate OOP in the PLC .In addition , abstract function block (class) , extend, inheritance or polymorphism were used and if you are not familiar with them , I highly recommend alltwincat.com and plccoder.com because you can find more information.

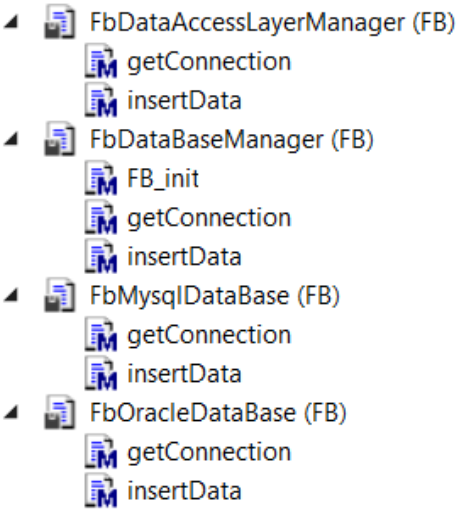

Figure 1

DataAccessLayerManager is an abstract block which MysqlDateBase and OracleDateBase inherit and DataBaseManager is used to keep the address of database blocks in order to create a polymorphic system.

while using abstract block , we don't implement any piece of code if a method labeled as an abstract , so we just define and user of that block has to implement abstract method but in the abstract block you can also define non-abstract method and it is not needed to implement when others inherit it however it can be overwritten.

DataAccessLayerManagerBlock

You can see Figure 2,3 and 4 that nothing was implemented in the DataAccessLayerManager block and methods, they were just defined for being extended by others and please pay attention to the "ABSTRACT" label because it is the most important thing . I think that naming is also so essential

because you don’t pay attention naming while you are working on a program , then you might be in trouble with the names and in this example, pascal case was used for blocks and camel naming is used for others ,such as variable or method .For further information, please check https://infosys.beckhoff.com/english.php and https://en.wikipedia.org/wiki/Naming_convention_(programming)

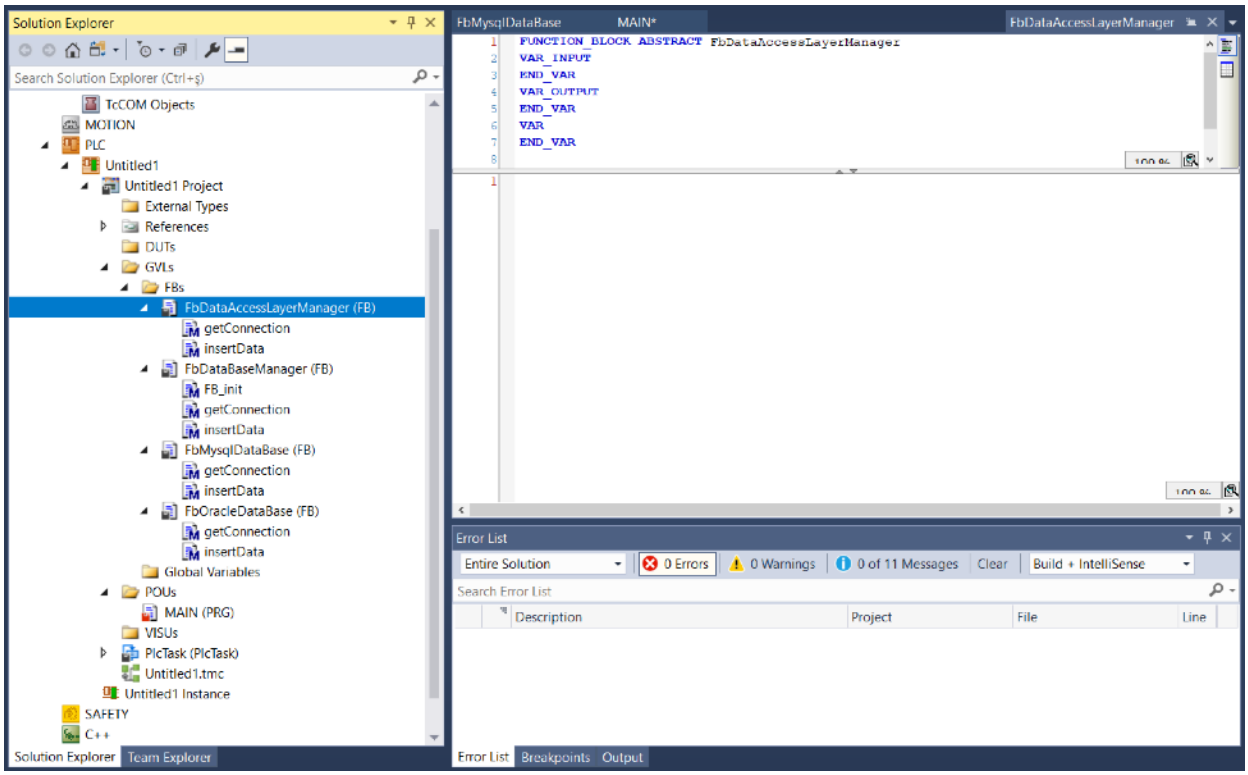

Figure 2

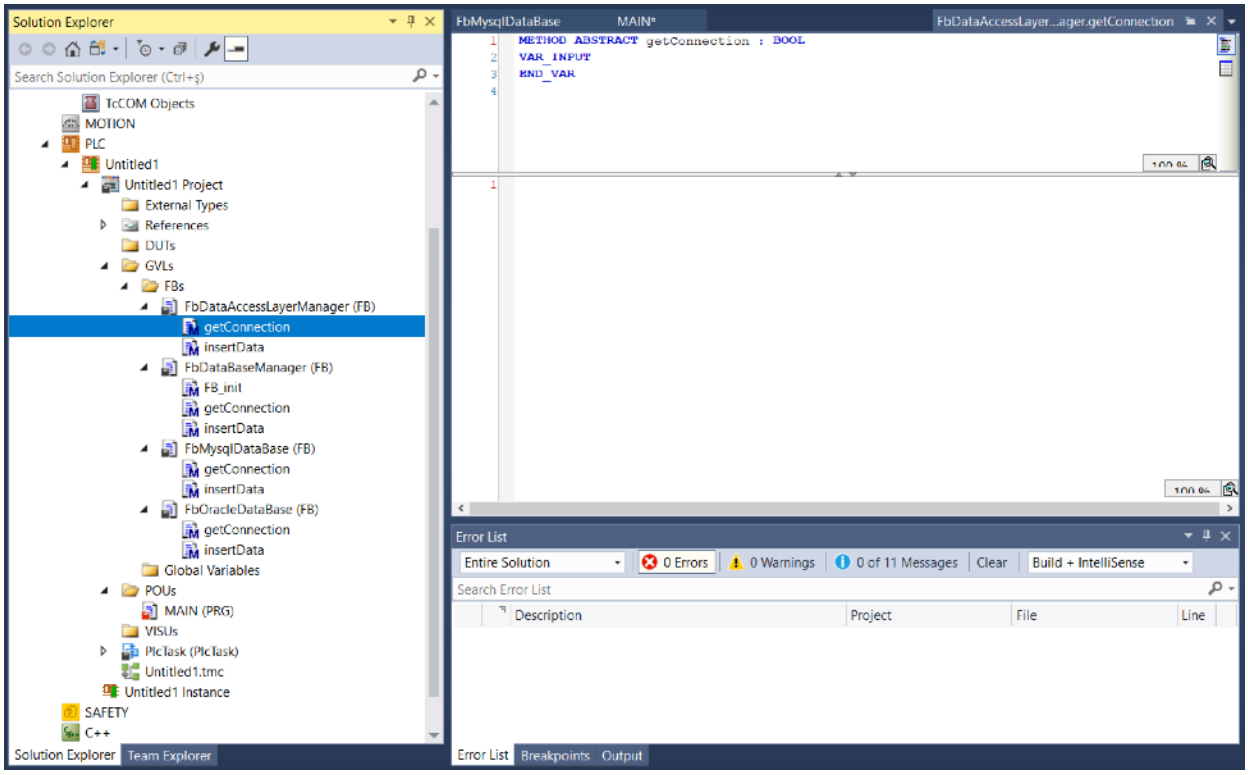

Figure 3

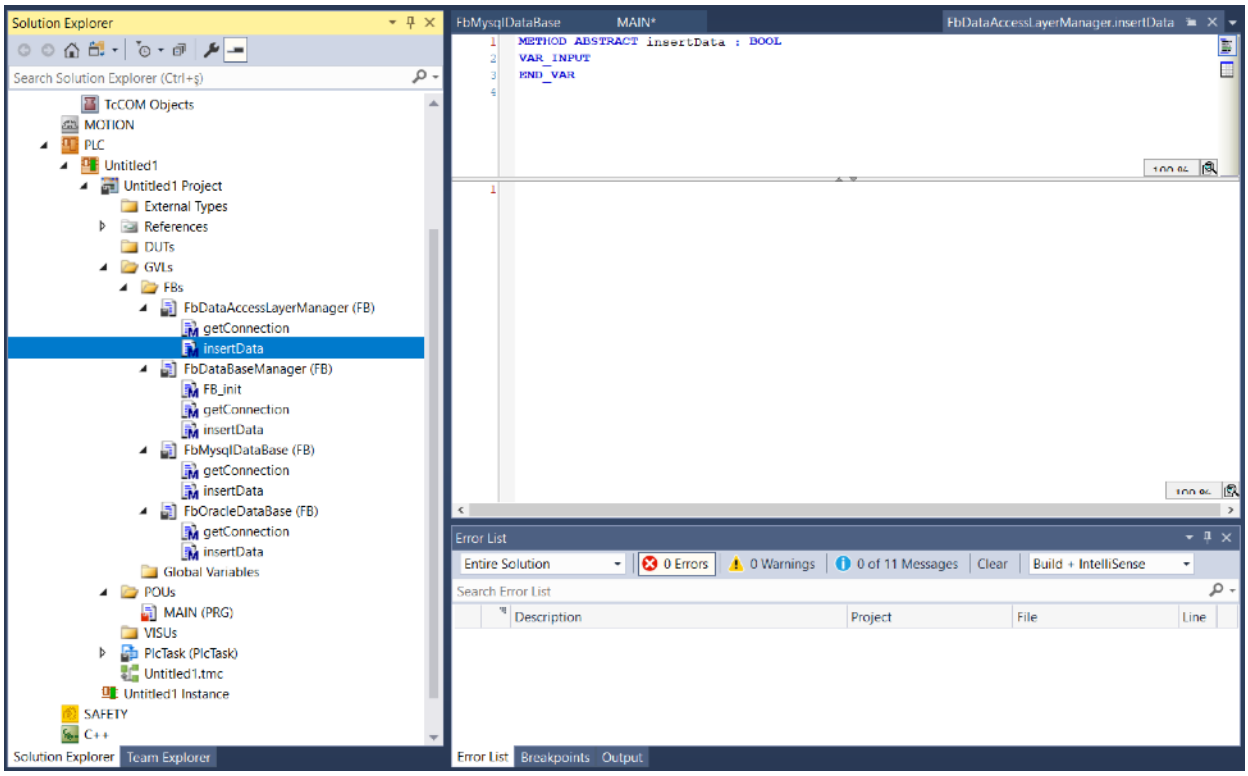

Figure 4

DataBaseManagerBlock

This example could run without a DataBaseManager block but we lose polymorphic structure and flexibility because these blocks keep which database is being used and it also calls methods.

Furthermore, in the block description only one variable is created ,nothing more and it keep reference of DataAccessLayerManager (it must be reference type because you can not initialize abstract block) because DataAccessLayerManager will extended by database blocks , like Mysql or Oracle . One feature of “INHERETENCE” is that if block A and C inherit Block B , then Block B can keep address of Block A and C . In another word , Block B is mother of Block A and C . In order to initialize the block variable , FB_init method is used and this method is called only one time. Inside getConnection and insertData , methods of DataAccessLayerManager are called but actually which database’s address it keeps ,so it calls that database’s methods.

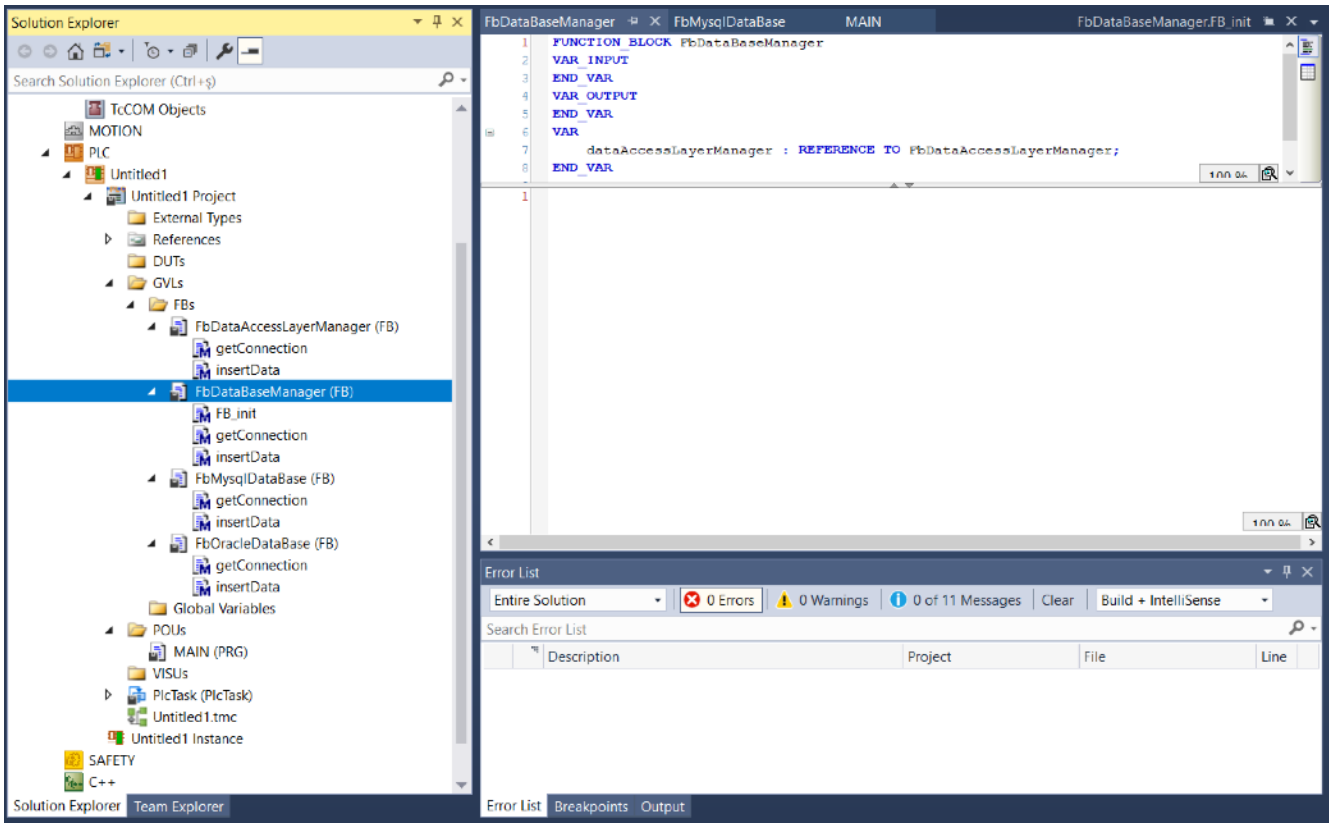

Figure 5

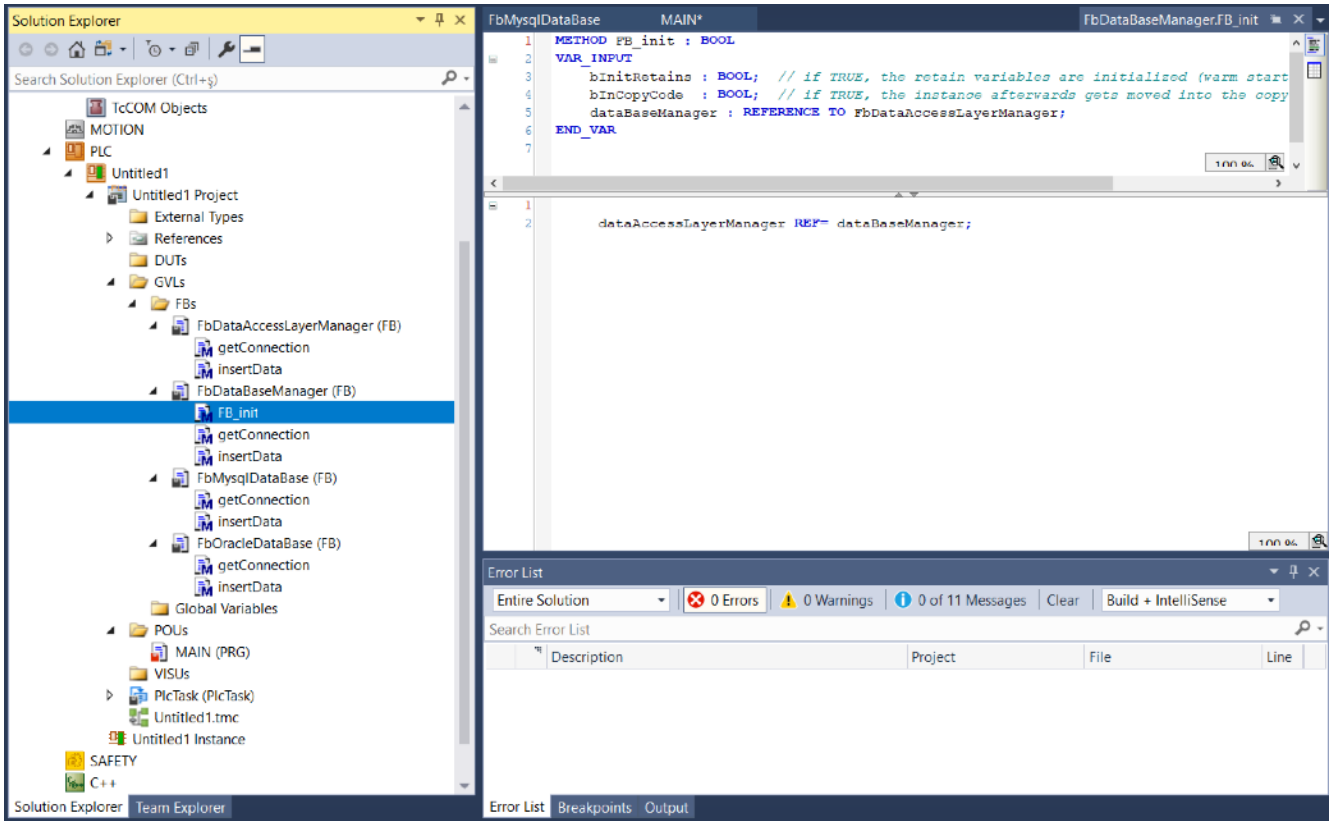

Figure 6

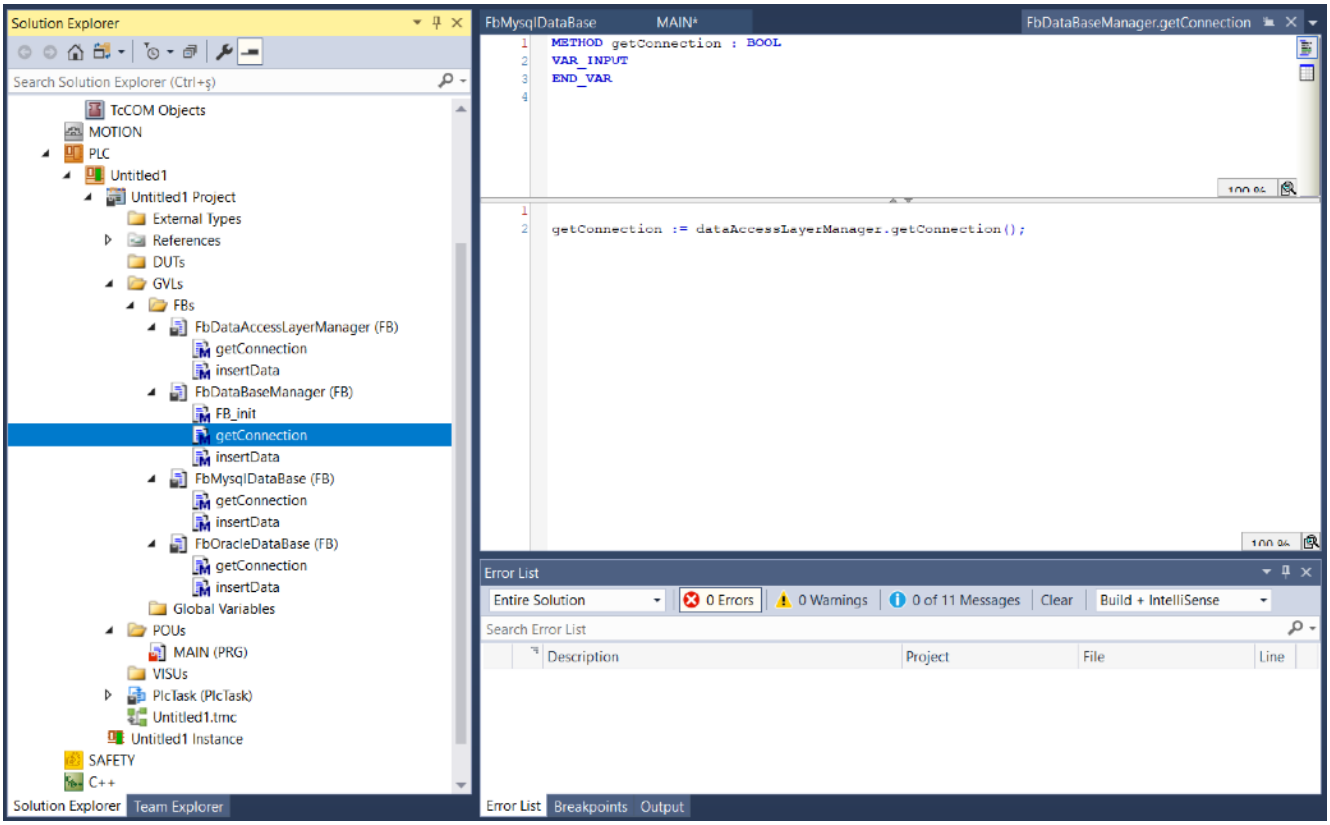

Figure 7

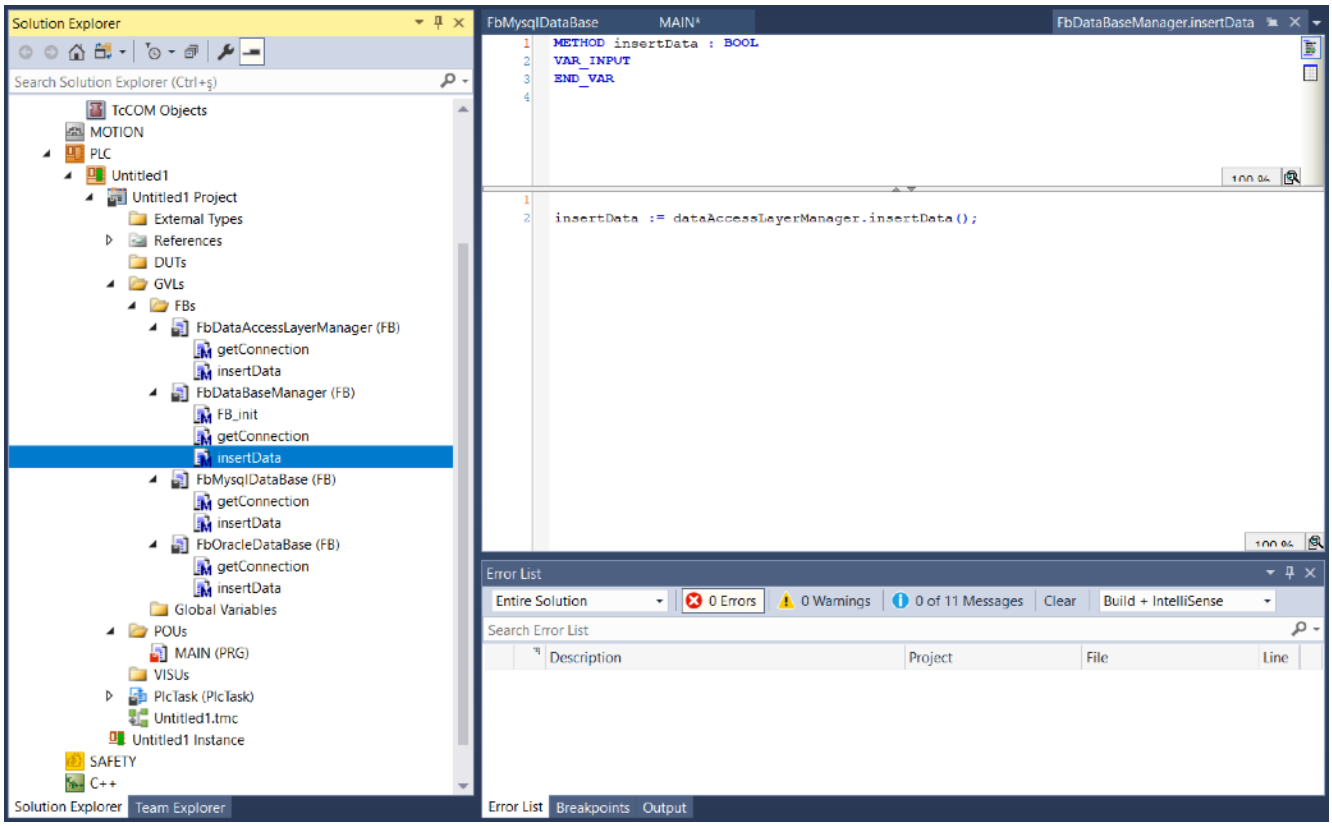

Figure 8

DataBaseBlock

In the block description , there is nothing implemented for this example but you can define and implement your code depending on your requirements and don't forget , this block has to extend DataAccessLayerManager. In this example , I will not show how data is inserted to the database because our point is how we use OOP in the PLC but if you want to know , please look at this example github.com/vancassa/TwinCAT_Database_Simple . Therefore , I simply wrote "insert your code here" to get a connection and insert data and return only a bool variable as a done signal. Figures show Mysql database block but Oracle is defined exactly the same.

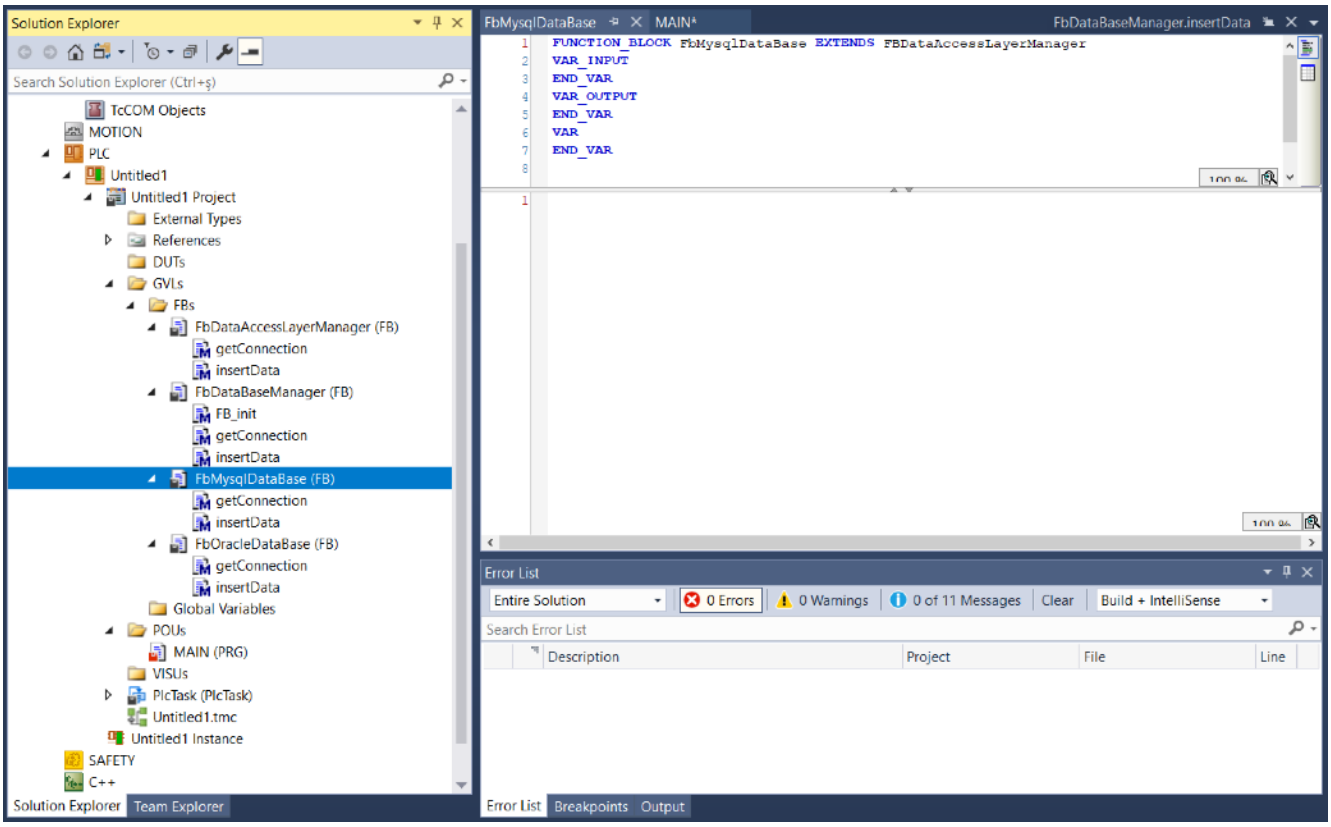

Figure 9

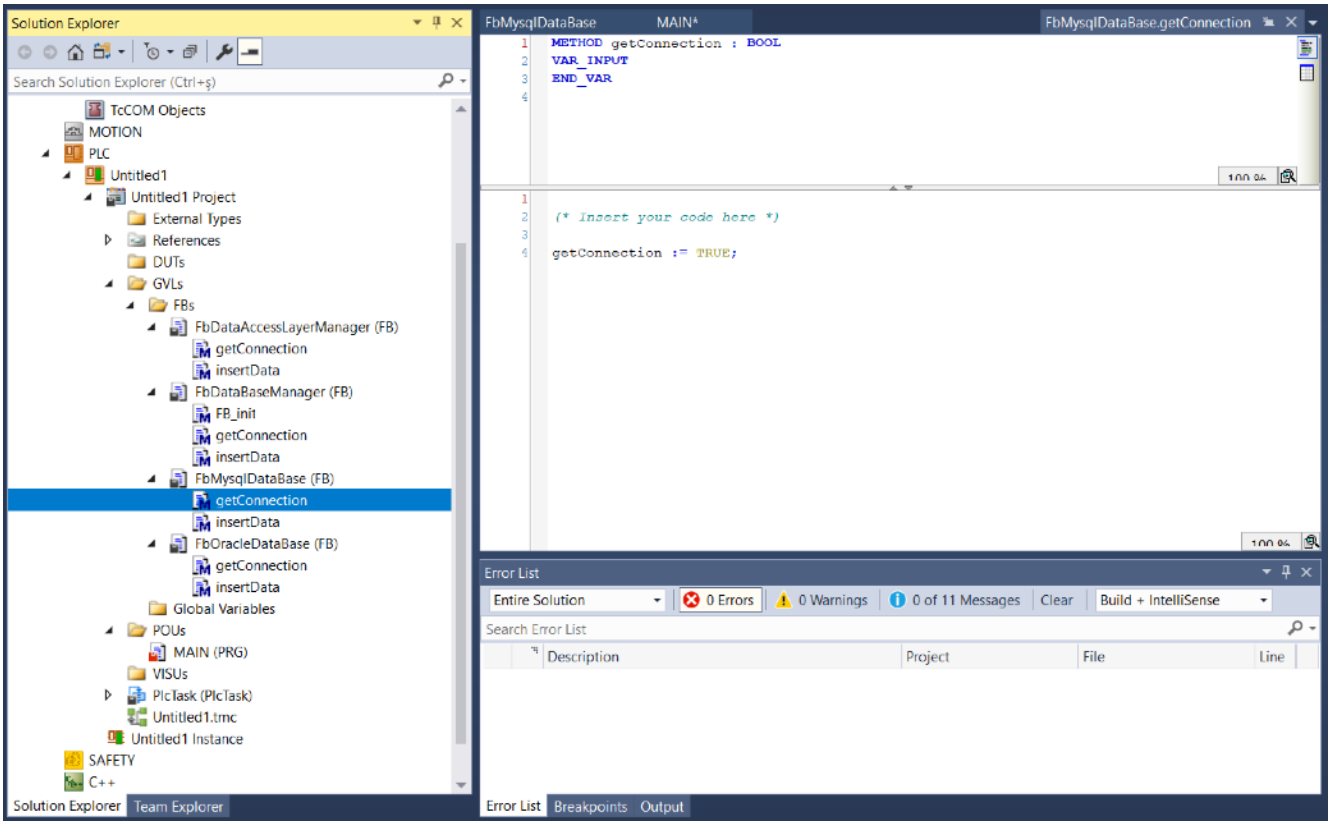

Figure 10

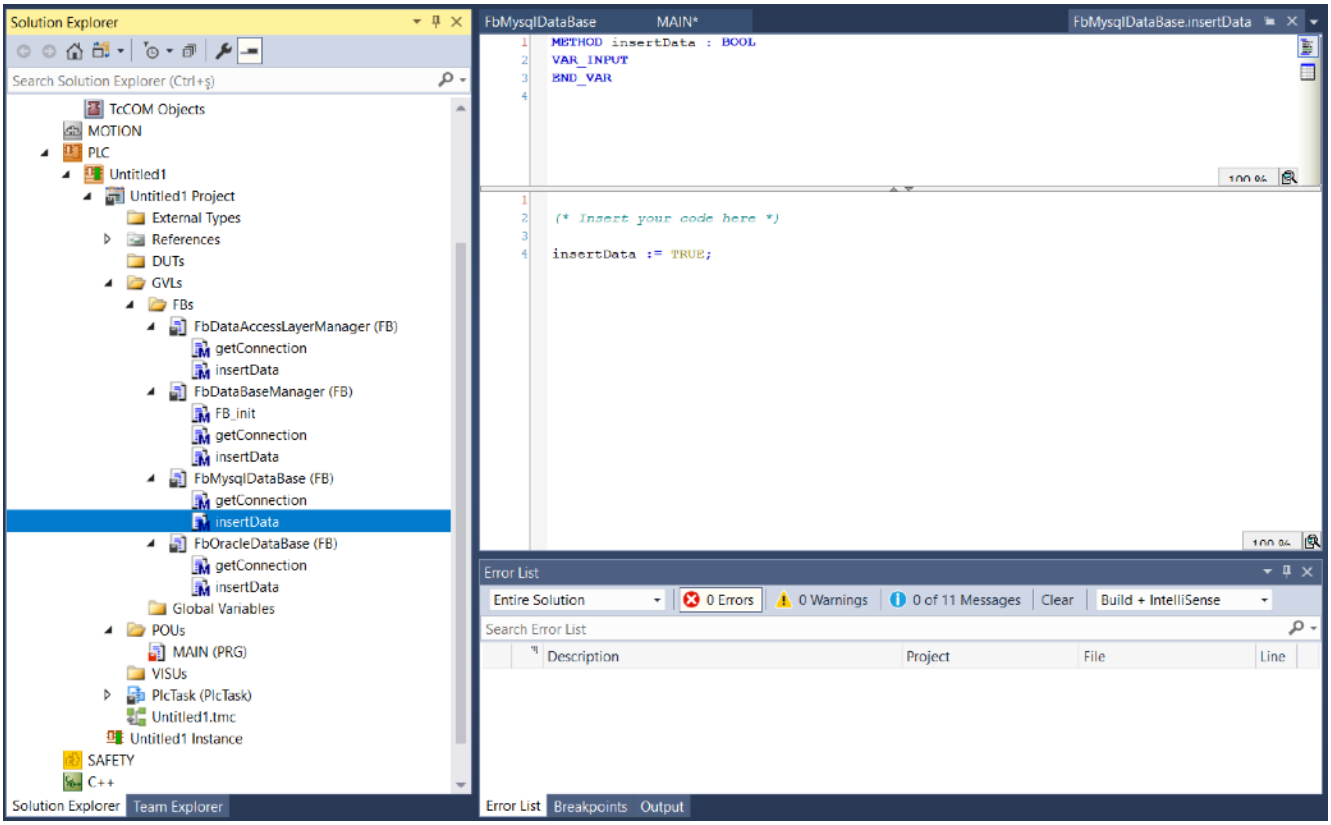

Figure 11

Main

In the main , instances of Oracle ,Mysql and DataBaseManager blocks are created and Mysql block is given to be initialized as a FB_init input of DataBaseManager and Do you remember we create reference of DataAccessLayerManager block ,not dataBase block ? but you know that DataAccessLayerManager is the mother of the database , therefore it is working without any problem. In addition , 4 bools variables are created to test and suppose that when we give getConnection bool , connection is established and return done signal by calling dataBaseManager.getConnection().

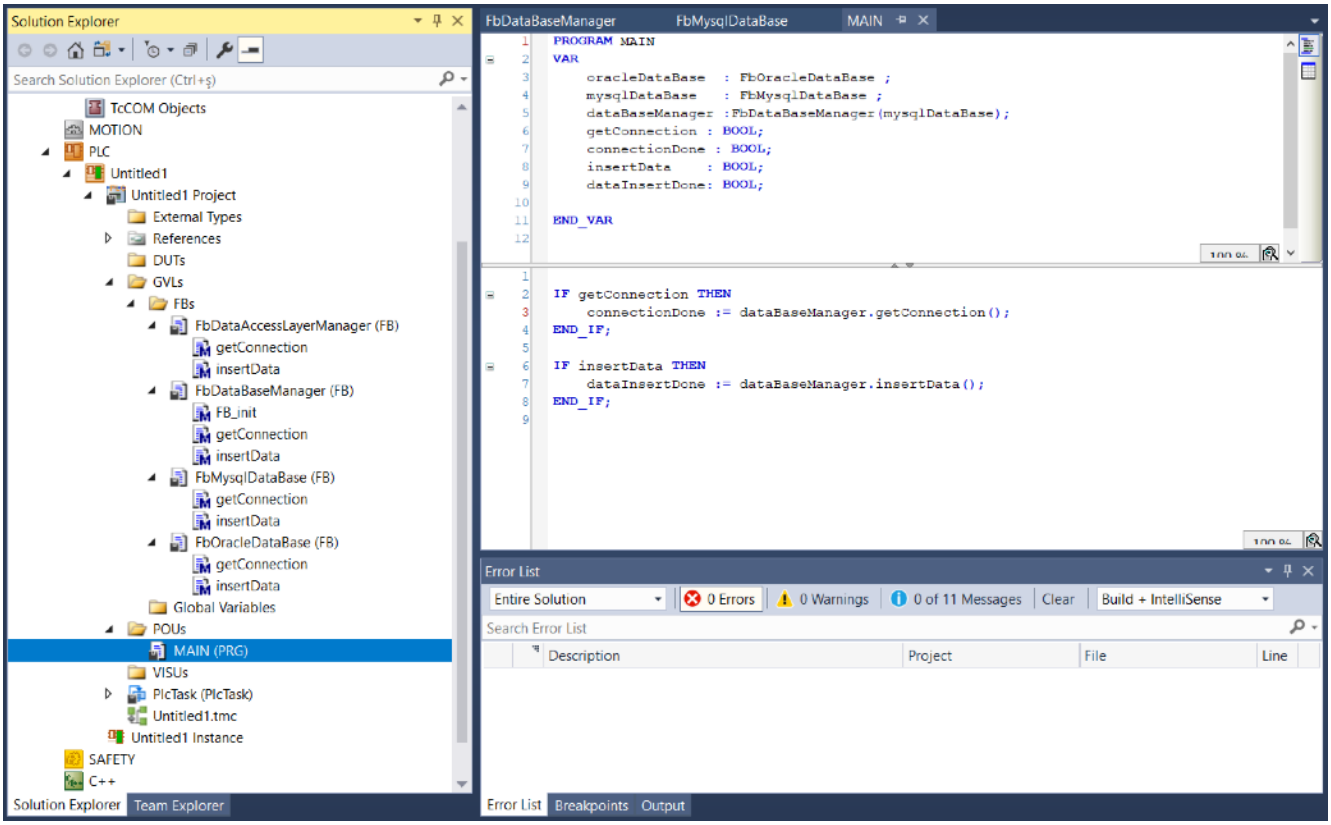

Figure 12

Imagine that we didn't create anything related to the Oracle database and after meanwhile , your system needs to work with Oracle instead of Mysql . Therefore, only thing we need to do that Mysql database is changed with Oracle to be initialized as a FB_init input of DataBaseManager after creating OracleDataBase block . It is so easy , isn't it ? and this can be happening thanks to OOP and it is power of polymorphism. In this way , we did solid programming and we didn't touch existing codes. Now Imagine that we don't use OOP approach , so we need to implement lots of codes and do lots of repeating work.

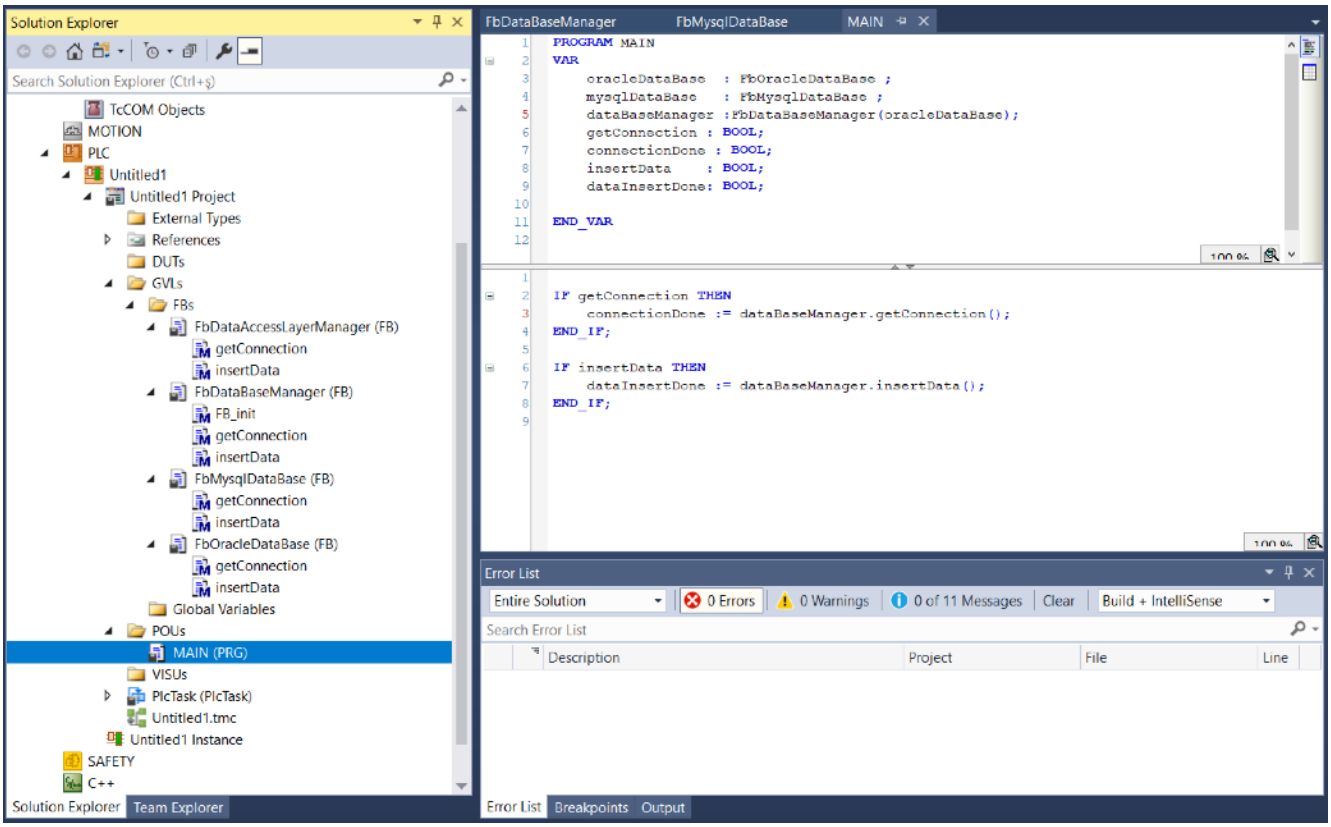

Figure 13

Consequently , I think this type of PLC programming style is fascinating and gives us the opportunity to design more quality systems full of flexibility by using OOP features but for now not all PLC producers support such a feature . I do expect that it will be happened very soon because conventional PLC programming style can not keep up with improvement of technology.

I wish you all the best

Murat TOPRAK