ISSN 2348-1196 (print)
International Journal of Computer Science and Information Technology Research ISSN 2348-120X (online)
Vol. 3, Issue 2, pp: (120-123), Month: April - June 2015, Available at: www.researchpublish.com

Compile-Time and Run-Time Polymorphism in Design Patterns

[1]MOUNIKA MANNE, [2]TARIK ELTAEIB

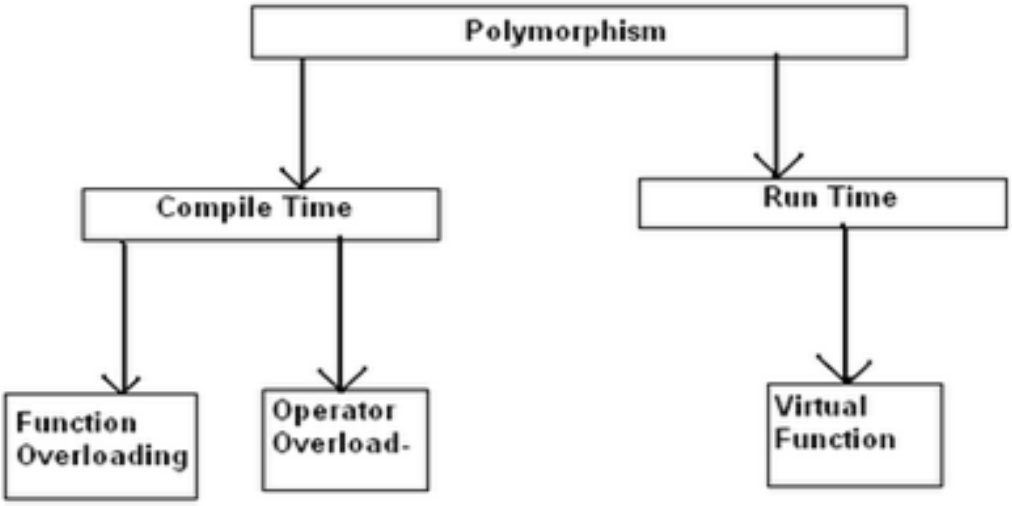
[1, 2] Department of Computer Science, University of Bridgeport, Bridgeport, United States Of America

***Abstract*: Polymorphism plays an important role in design patterns programming and it is used in different aspects. In this paper different types of polymorphism is explained, and coding is given as an example for every polymorphism, function overloading and function overriding. The significance of compile-time polymorphism and run-time polymorphism is explained briefly. We also discuss about the relationship between operator overloading, function overloading and virtual function. We also present the design patterns with an example which helps us to understand the patterns more easily.**

Keywords: **Design patterns, Polymorphism, Compile-time, Run-time, Chain of Responsibility, Evolution, Applicability, Object Oriented Programming, and Model.**

1. INTRODUCTION

This research paper explains Polymorphism which is used to reuse different objects in one situation. Polymorphism is implemented by using method overloading and method overriding. The pure virtual and non-virtual functions are used in static and dynamic binding. These virtual function is used by "Virtual" keyword before base class method and "Override" keyword before derive class method. The Destructors in base class can be made Virtual and known as Virtual Destructors. Polymorphism is flexible as the multiple objects can be used in single aspect.



Polymorphism:

Polymorphism can be defined as same function or operator will show different behaviors when different types of values or number of values passed.[1] There are two types of polymorphisms are available. They are static or compile time or early binding is achieved using method overloading. Dynamic polymorphism is achieved by method overriding.[2]

ISSN 2348-1196 (print)
International Journal of Computer Science and Information Technology Research ISSN 2348-120X (online)
Vol. 3, Issue 2, pp: (120-123), Month: April - June 2015, Available at: www.researchpublish.com

Compile-time Polymorphism:

Collaborate the function label with the entry point of the function during the compile time is defined as the compile-time polymorphism. This is achieved by using method overloading. A method overloading can be compared with a function will have already same work to do and if we assign different work to the same function then we say function is overloading. A function will be overloading in two situations. When datatypes of arguments are changed and number of arguments are changed the function will be overloading.[3]

Method Overloading:

Providing a new implementation to same function with different signatures is known as function overloading. In general, function overloading is implemented within the same class. When a binary operator is overloaded, then such method function exactly one parameter. Since a binary operator require two operands, and default objects is the first operand, so that the parameter which is passed, become second operator. When a unary operator is overloaded then such method function, will not take any parameter. Any valid operator of C++ can be overloaded except the following:[4]

- :: Scope resolution operator
- ?: Ternary or conditional operator
- Sizeof() operator
- . Member operator
- ::* Pointer to member operator

Code:

```
#include <iostream>
using namespace std;
/* Number of arguments are different */
void display(char []);  // print the string passed as argument
void display(char [], char []);
int main()
{
char first[] = "C programming";
char second[] = "C++ programming";
display(first);
display(first, second);
return 0;
}
void display(char s[])
{
cout << s << endl;
}
void display(char s[], char t[])
{
cout << s << endl << t << endl;
}
```

Function Overriding:

Function overriding is providing is providing a new implementation to a function with same signature**.** In general function overriding will be implemented within the derived class. To make any function as virtual use virtual keyword, to override

any virtual function in derived class use override keyword. Overriding of virtual functions are not compulsory, it is optional.[5]

Code:

```
#include <iostream>
using namespace std;
Class Base
{
  Public:
Virtual void myfunc()
{
cout << "Base::myfunc .." << end1;
}
};
Class Derived: public Base {
Public:
Void myfunc()
{
cout << "Derived::myfunc .."<< endl;
}
};
Void main()
{
Derived d:
d.myfunc();
}
```

Virtual function:

Virtual function is an empty block which is assigned by null value(zero). It must be overridden by all the derived classes. A class which contains atleast one pure virtual function is known as an Abstract base class. The objects cannot be created but pointers can be created. It also contains the dynamic binding.[6]

Code:

```
#include <iostream>
using namespace std;
class parent
{
Protected:
Int a;
Public:
Void disp()
{
Cout << " This is parent class Member Function.\n";
}
};
```

```
Class child: public parent
{
Int b;
public:
Void disp()
{
Cout << " This is child class Member Function.\n";
}
};
Void main()
{
Parent*P,X;
Child Y;
Clrscr();
P=&X;
P->disp();
getch();
}
```

2. CONCLUSION

In conclusion method overloading is a refinement technique and method overriding is a replacement technique. A virtual function can also be declared and defined like any other method function, but should be preceded by keyword virtual. Run-time polymorphism can also be done without virtual function.

REFERENCES

[1] Nath, S.P., S. Vikram, and K. Anand. Better alternative Run-Time Polymorphism in C++: A practical approach in OO design for machine intelligence. in Computer Communication and Informatics (ICCCI), 2012 International Conference on. 2012

[2] Su, H. A new polymorphism testing method for C++ software. in Electric Information and Control Engineering (ICEICE), 2011 International Conference on. 2011.

[3] Tonella, P., et al. Flow Insensitive C++ Pointers and Polymorphism Analysis and its application to slicing. in Software Engineering, 1997., Proceedings of the 1997 International Conference on. 1997.

[4] Mei-Hwa, C. and H.M. Kao. Effect of class testing on the reliability of object-oriented programs. in Software Reliability Engineering, 1997. Proceedings., The Eighth International Symposium on. 1997.

[5] Jarvi, J. Compile time recursive objects in C++. in Technology of Object-Oriented Languages, 1998. TOOLS 27. Proceedings. 1998.

[6] Shopyrin, D., Multimethods in C++ using recursive deferred dispatching. Software, IEEE, 2006. 23(3): p. 62-73.