Review on Concepts Related to Object Oriented Programming System

VISHAL V. MEHTRE[1], YASH NIGAM[2]
[1,2] *Department of Electrical Engineering, Bharati Vidyapeeth Deemed University, College of Engineering, Pune*

Abstract - This review paper states that the concept of object-oriented programming and its characteristics such as polymorphism, inheritance and encapsulation. Object oriented programming tell us about the concept of class, object and several other concepts which defines the characteristics of object-oriented programming. In this, inheritance plays a major role in enhancing the characteristics of OOPs.

I. INTRODUCTION

Object Oriented Programming is an applied language which works on the basic concept of objects, used to store data which can be in fields, code or procedure.
Object oriented programming language is supported by worlds most widely used multi-paradigm languages such as (C++, Python, Java, PHP, Dart, and Ruby).
By using oops we can design computer programmers by casting them out of objects which can interact with one another. There is a feature in objects- object's procedure, that can ingress and can modify the data fields of the objects from which they are linked.[1]
While using object-oriented programming we have advantage over procedural programming that is this enable helps to enable programmers to create modules and there is no need to change when there is an addition of new objects. By this advantage a programmer can create a new object which contains many features from existing software. This advantage of object-oriented programs over procedural program makes it easier to adapt.

Object oriented programming gives the programmers an ease in structuring the various software programs. The primary step in OOP is to spot the entire object which a programmer needs to operate and the way they connect from each other, this is also known as data modelling. Primary, OOPS focus on the object which a programmer has to operate. In OOPS data is accessed, than it is processed.

II. DATA ABSTRACTION

Abstraction is a basic concept in software developing. This process of abstraction can also be said to be connected to concepts of theory and design. One of the main abstractions in computing is language abstraction. The process of the selecting the important data for an object and leaving which is not necessary is known as abstraction. Once you have modified your object with the help of abstraction in different applications same data can be used.[2]

The advantage of abstraction is that it reduces the efforts in complexity of programming. Abstraction helps you to group different classes as one. We can understand abstraction concept by using an example:
Like if you want to develop a banking application and there you have to collect information about the customers.

You will get much information about your customers, but you only need to keep to important ones, like Names, Address, etc. for this you have to develop a banking application. As we have extracted the information about the customer from a larger database, this process is known as abstraction.

Example:

```
Class bankacc
{
private:
    Float currentbal;
Protected:
Virtual float minimum dpst () =0;
Virtual float minimum wdw () =0;
Public:
bankacc(){currentbal=0;}
Virtual ~bankacc(){};
Void deposit(float amt)
```

```
{If ( amt>=minimum dpst()) currentbal +=amt;}
Float wdw(float amt = 100.0)
{
If ((amt>+minimum wdw ()) && (current bal>+amt))
Current bal-+amt;
}
Float query bal(float amt =100.0){return currentbal;}
};
```

III. INHERITANCE

In object-oriented programming, inheritance be an object or class is based some other object or class, using the same implementation (inheriting from an object or class) specifying implementation to maintain the same character (realizing an interface; inheriting behaviour). It is a mechanism for code reuse and to permit independent extensions of the genuine software via public classes and interfaces. The relationships in objects and classes through inheritance develop in a hierarchy. The invention of Inheritance was in 1967 for Simula. The term "inheritance" is generally used for the class-based and prototype-based programming, but in narrow use is fixed for class-based programming (one class inherits from another), with the same technique in prototype-based programming being instead called delegation (one object delegates to another). [3]

Inheritance is the reuse of an existing code. Suppose that we have a class Maruti that define features and functionality of Maruti cars now Swift is also a Maruti car but it has some its own extra functionality. Now what Swift will do? It will inherit the existing properties of Maruti instead of writing that same code again and will also add on its own functionality in the code . So here Maruti is a parent class and Swift is a child class.

Example**:**

```
#include <iostream>
using namespace std;
class A{
   public:
     void display()
     {
 cout<<"This is base class";      }
};
class B : public A
{};
class C : public B
{};
int main()
{    C obj;
   obj.display();
   return 0;
}
```

IV. ENCAPSULATION

In Object Oriented Programming, encapsulation is in regard of object design. It means that all of the object's data is stored and hidden in the object and access to it is not allowed to members of that class.

When information is wider and more open, it becomes very risky as it leads to surges and various changes elsewhere. It is safe to restrict the direct way in of one piece of data. Encapsulation is one of the basic concepts in object-oriented programming (OOP) that states the notion of bundling data and methods that work on that data without one unit. When it restricts entry to the member of a class, it protects access to the member of a class from manipulating objects in ways that the designer does not want. [4]

Its use is to hide the interior representation of an object, from the inside. It sums up elements to generate a new entity. If there is an attribute that is not in view from the outside of an object and tie-up with methods that provide read or write access then it is possible to hide the information and handle access to the internal state of the subject. It works as a mechanism decreasing the accessibility of attributes to the current class and uses public getter and setter methods to control and limit external access. It also enables to prove the new value before modifying the attribute. Encapsulations preven data from any accidental corruption and lower coupling between objects and keeps code maintainability.

Generally, there is uncertainty that, what is different b/w encapsulation and abstraction
Programmatically, when we can have a way in the hidden data somehow and know something.

Abstraction and when we know zero about the interior its Encapsulation.

Example*:*

```
#include <iostream>
using namespace std;
class add {   public:
          add(int i = 0) {
        total = i;      }
          void addNum(int number) {
        total += number;      }
          int getTot() {
        return tot;      };
  private:
     int tot;
};
int main() {   add a;
  a.add(10);
  a.add(20);
  a.add(30);
  cout <<"Total "<< a.getTot() <<endl;
  return 0;}
```

V. POLYMORPHISM

Polymorphism is derived from 2 greek words: poly and morphs. The word "poly" means many and "morphs" means forms. The meaning of Polymorphism is form. Polymorphism is not a programming concept, but it is one of the basic of OOPs.

Following are the advantages of polymorphism

1. Simplicity

- If you need to write code that gets with family of types, the code can neglect type-specific details and just contacted with the base type of the family
- This thing makes code easy to write and to understand also.

2. *Extensibility*

Other subclasses could be added after to the family of types, and objects of those new subclasses would also work with the existing code.[5]

Examples:

```
class Shape {
  protected:
    int width, height;

  public:
    Shape( int a = 0, int b = 0) {
      width = a;
      height = b;
    }
    virtual int area() {
      cout <<"This is the area of parent class :"<<endl;
      return 0;
    }
```

VI. CONCLUSION

This review paper shows us the benefits of OOPs to the programming. Here, we had analysed the business world and word with encapsulation of related strings and numbers into classes. Achieve reuse without instruction lists from the programmer who wrote the code you are reusing.

ACKNOWLEDGEMENT

We would like to express our special thanks of gratefulness to Dr. D.S Bankar Head of Department of Electrical Engineering for their able guidance and support for completing my research paper. I would also like to thank the faculty members of the Department of Electrical Engineering would helped us with extended support.

REFERENCES

[1] Balguruswami, “Object oriented programming with C++”, (Mc Grawhill Education).
[2] Sumita Arora, “Concept of C++”,(Dhanpat Rai &co.)
[3] Prof. Dharminder Kumar, “Introduction of OOP”.2011
[4] Shivam, “A Study on Inheritance Using OOP with C++”, Issue 2 July, 2013.
[5] Ashwin Urdhwareshe, “Object-Oriented Programming and its Concepts”, Issue 1 Aug, 2016.
[6] Saïda Benlarbi, “Polymorphism Measures for Early Risk Prediction”,IEEE.