Journal of Engineering and Sustainable Development
www.jeasd.org
Vol. 21, No.04, July 2017
ISSN 2520-0917

CONSTRUCTORS AND DESTRUCTUR IN OBJECT ORIENTED PROGRAMMING

Nadia Moqbel Hassan *

Asst. Lecturer, Computer & Software Eng. Department, Al-Mustansiriyah University, Baghdad, Iraq.

Abstract: This paper offers analysis and design of data protection and administration utilizing constructors or special member functions (SMF) and destructor or demolition function (DF) capacities that begin the objects of its class and manipulated the data using encapsulation or concealing data in object-oriented programming (OOP). In (OOP), the accentuation is on data rather than procedures. Class is a technique that ties the data and attributes jointly. SMF work is utilized to begin or initialize the occurrence of a class and DF work crushes the objects when they are no longer needed. Lastly, the paper exhibits Performance Evaluation and Simulation Results for depict how certain security constraints are taken SMF whose errand is to initialize the objects of its class.

Keywords: *object oriented design, software development, information hiding, object oriented programming, encapsulation, constructors and destructor, programming paradigms.*

المنشِآت والمدمر في البرمجة كائنية التوجه

الخلاصة: يقدم هذا البحث تحليل وتصميم إدارة وأمن البيانات باستخدام المنشئ او دوال العضو الخاصة (SMF) والمدمر او دوال الهدم(DF), التي تهيئ الكائنات من فئتها وتعالج البيانات على أساس التغليف أو إخفاء البيانات في البرمجة كائنية التوجه (OOP). في البرمجة كائنية التوجه ، يتم التركيز على البيانات بدلا من الدوال. الفئة هي الطريقة التي تربط البيانات والدوال معاً. دالة العضو الخاصة هي تهيئة للفئة والدالة المدمره تدمر الكائنات عندما لم يعد هناك حاجة إليها. وأخيرا، يقدم البحث تقييم الأداء ومحاكاة النتائج لوصف كيفية التعامل مع القيود الأمنية المعطاة باستخدام دالة عضو خاصة التي تتمثل مهمتها في تهيئة الكائنات من فئتها.

1. Introduction

OOP is a programming model in established on the notion of objects; these are data structures (DS) that include data as fields, frequently known as properties and code. Objects are all over the place, thus it is essential to perceive components, known as objects, from true circumstances and know how they can without much of a stretch be converted into object-oriented code [1]. In the course of recent 15 years, the programming concentrated on the improvement model of (OOP). Most day expansion situations and styles execute OOP.

* nadiamha542@yahoo.com

Artlessly, OOP brews the premise of all that you grow today. OOP is at present getting a large-scale adoption of its methods and languages both in university teaching and in real applications.

As a model OOP has fortified dialect particular books and general acquaintances bound with one dialect (e.g., Eiffel, Smalltalk, and C++)[1,2,3]. To understand OOP ideas, it is vital to see the issue from the programmer point of view and model the arrangement utilizing object demonstrate [4,5,6]. (OOP) utilizes an alternate arrangement of programming languages than old conventional programming using high level languages (COBOL, FORTRAN, C, Pascal, and so on.). Everything in OOP is assembled as self-economical "objects". Subsequently, programmers gain reusability by methods for four major OOP notions [7].

In OOP, the accentuation is on data as opposed to procedures. Class is a method that ties the data and attributes jointly. SMF is an extraordinarily composed class and destructor restores the memory delivers back to the program [8]. OOP displays another and intense approach to cope with intricacy. Rather than review a program as a string of instructions to be run, it seeing it as a gathering of objects that have certain characteristics and ability take certain activities. This may sound darken until the point that you take in more about it, however it brings about projects that are clearer, more dependable, and more easily maintained [10].

2. Literature Survey

There is a common understanding in the writing that code learning is not a simple assignment [12, 13]. Programing in another style on the off chance that one has earlier software knowledge makes pressures between program learning and educating software. Educators' observations and conduct are shaped by their own know-how, both their past experiences and current perspectives. Bergin and Winder [14], among others, trust that OOP is a model, unique in relation to conventional programming, which requires a change in mental model (an outlook change) in the professionals. Subsequent to being utilized to a conventional programming, figuring out how to program in an object oriented manner is by all accounts extremely troublesome.

For instance, Stroustrup [15] demonstrates that it takes 6 to year and a half by and large to switch the attitude of a developer from procedural to OOP. What it might be applicable to ask now is, will programming in the new worldview be similarly as, or less, troublesome as it is in the old worldview, or does the move in worldview represent extra challenges. The myth that "question introduction and procedural ideas are fundamentally unrelated" is disproved by Lewis [16].

He contends that an object oriented technique does not toss out the ideas that are appreciated in a procedural approach; rather it enlarges and reinforces them. Various late new studies [17, 9] investigate issues identifying with the OOP worldview. The contention exhibited for grasping the object oriented model is twofold. The first, it is contended that objects are normal features of issue fields and can be illustrated as beings in the software field. The next point, the graph among the scopes is straightforward and

must, along these lines, bolster and encourage OOP outline. Notwithstanding, the looked into writing demonstrates that recognizing objects is not a simple procedure for amateurs, and the among spaces is not straight forward. While the writing on master developers is steady of the expectation and simplicity of object oriented outline, it additionally demonstrates that master object oriented software engineers utilize both object oriented and procedural perspectives of the software field, and switch between them when important [18].

In any case, this paper is especially worried with junior object oriented software engineers. In their survey, Bergin and Reilly [17] found that among the variables that impact programming achievement, self-impression obviously results were the most unequivocally associated to execution. In a different report, Wiedenbeck and Ramalingam [19] found that the primary year tertiary understudies' self-viability of writing computer programs is impacted by their past optional college programming background, which thus impact their execution.

In this paper, in any case, past software knowledge of the members is scout in connection to taking in an alternate programming worldview, OOP. Accordingly, the experience revealed in this paper would enable further to comprehend the learning of OOP and the educating of OOP by beginners, which thus adds to the form of literature [11].

3. Object Oriented Programming Model and Characteristics with Notions

OOP is one of a few programming models. Other programming models incorporate the basic programming models (as exemplified by languages, for example, Pascal or C), the rationale programming model (Prolog), and the procedural oriented programming model (embodied by means of languages, for example, ML, Haskell or Lisp). Logic and procedural languages are told to be declarative languages. Utilized the word model to signify "any illustration or model".

This utilization of the word was promoted by the science history specialist Thomas Kuhn. He utilized the term to depict an arrangement of hypotheses, models and strategies that together speak to a method for planning knowledge a method for survey the world. In this manner a programming model is a

. . . way of conceptualizing what it means to perform computation and how tasks to be carried out on a computer should be structured and organized.

It can recognize two sorts of programming languages: mandatory languages and declaration languages. Mandatory knowledge portrays how-to learning or knowledge while declaration knowledge is what-is knowledge [20]. A program is" declaration" in the event that it depicts what something resembles, instead of how to create it. This is an alternate style from conventional software languages; for example, FORTRAN, and C, which demand the software engineer or programmer to determine an algorithm to be implement.

```
public:
 Student( )
 ~Student( )
{
Code=0;
FirstName=''   '';
LastName=''   '';
GPA=0;
```

Figure 1. Calculate the Student Function

To put it plainly, mandatory systems or programs produce the algorithm express and it remains the objective implicit, while declaration systems or programs produce the objective unequivocal or explicit and leave the algorithm verifiable implicit. Mandatory languages oblige you to record a well ordered formula determining how something is to be finished. For instance, to find the Studentfunction in a mandatory language can would compose something like in the accompanying Figure (1) and Figure (2) outline it:

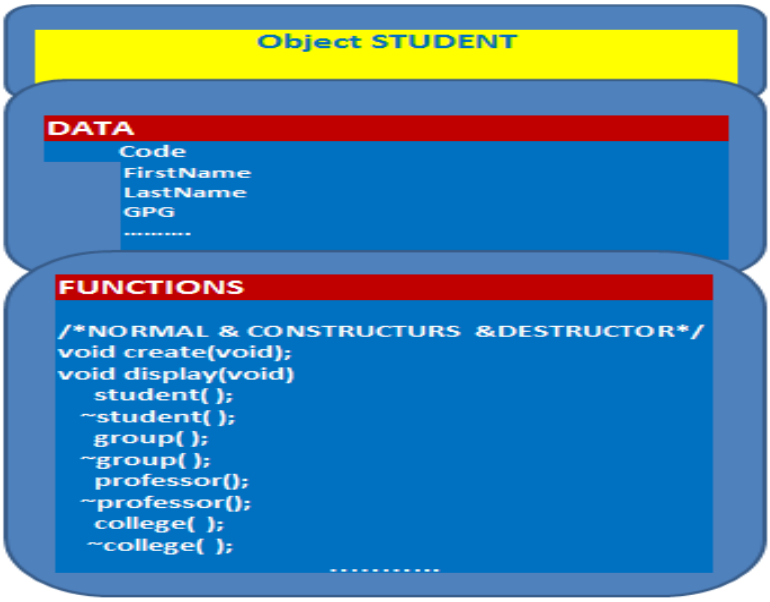

Figure 2. Calculate the StudentFunction of representing an object

In software engineering, useful writing computer programs are a programming worldview that regards calculation as the assessment of mathematical functions and stays away from detail and alterable data. It underscores the use of functions, interestingly with the basic programming style that underlines changes in stat. Object Oriented is a style model for portraying objects and their connections. Articles and relations should be stand near certifiable ideas. This present reality is the world that are execute, that is a level of reflection in the outline or a prerequisites detail and of data security and administration utilizing SMF and DF capacities that initialized the objects of its class and controls the data in view of encapsulation or hide data in OOP. This present reality is a future world in which the program being worked on participates.

This present reality is additionally an abstract world. It is of no worry how something functions, just what it does. This reflection is entering in Object Oriented. In any case, Object Oriented is frequently effectively supplanted with OOP. Be that as it may, a

usage in an OOP language is close to a case of this demonstrating at the least level of abstract of the design [7,10,20,21,25].

4. Basics of Objects and Classes

In the course of the most recent couple of decades programming and informatics have encountered unbelievable development and ideas, which have changed the way for to systems or programs, are assembled. OOP presents such thought. Object oriented can be viewed as a sort of strategy of arranging a program in elements of objects and their relations. It must stand closer to this present reality instead of techniques beginning before Object oriented. Its advantage is the variance between the discernible conduct of objects and the execution of the practices or objects [7,21,25].

Class in the object oriented programming is known as a definition (detail) of a specific sort of objects from this present reality. The class speaks to an example, which depicts the distinctive conduct of the specific objects, which are created from this class.

Object is a duplicate or (copy) created from the definition (details or specification) of a given class, likewise called an instance. When one object is created by the depiction of one class we say the object is from sort "name of the class".

Paradigms of true-world objects are college, student, groups, professor, and so on. Objects are ideas or concepts in an object region, which have been to sample and utilize in a PC program. Cases of abstract objects (AO) are the (DS) Queue, List, Tree, and Stack. In objects from true-world (and additionally in the OA), can recognize the accompanying two gatherings of their qualities:-

Attributes – these are the properties of the object which characterize it in a way and portray it as a rule or in a particular moment.

Behavior – these are the particular unmistakable activities, which should be possible by the object. For instance, an object from true world "Student". The attributes of the Student can be

"name", "color", "ID", "age", "weight" and "length"

Objects in object oriented programming join data and the member functions for their handling in one. They relate to objects in genuine and contain data and activities:

-*Data Members (DM)* – An integral part in objects arguments, which depict their states.

-*Member Functions(MF)* –They are an apparatus for building the objects [2,7,10,21,25].

A basic perspective of an object is a mix of properties and conduct. The member functions name with variables or arguments the interface of an object. The interface or stream is utilized to associate with the outside world. OOP is a packaging technology. Objects encapsulate data and conduct concealing the subtle elements of execution. The idea or concept of execution covering up is otherwise called data hiding. Because it is and since data is vital, the clients can't get to this data straightforwardly. Just the interfaces (member functions) can get to or alter the encapsulated data. In this manner, data hiding is likewise accomplished. The limitation of access to data inside an object just those member functions defined by the object's class are known as encapsulation.

Likewise, usage is freely done enhancing programming reuse idea. Interface encapsulates knowledge about the object [6].

The class particular should be possible in two sections:

(i) Class declaration (DM). It portrays both information individuals and part works.

(ii) Class member functions (MF) definitions. It depicts how certain class methods or member functions are coded [22,23].

5. Proposed Work

Subsequent to working through this module, it will have the capacity to make and utilize objects in C++, fusing encapsulation and data hiding utilizing SMF and DF that initialized the objects of its class and manipulates the data in view of exemplification or hiding data in object oriented programming, and apply it to a genuine paradigm. Begin pondering an issue of the college and exam task. The aim of which is to exam the capacities to utilize object oriented programming(OOP) for demonstrating issues from the genuine living, design classes and connections between them and in addition working with accumulations. To tackle this issue is to utilize object-oriented paradigm the abilities that picked up from "Object-Oriented Programming Principles". Analysis and design of information protection and administration utilizing (SMF) and (DF) capacities that initialized the objects of its class and controls the data in view of encapsulation data in (OOP).

It is let go automatically when an object of the given class is created. It doesn't frank demand to be called. In spite of, it must be put inside the public part area of the class declaration as in the Figure (3).

Module :

```
class student
{
 private:
 int Code, string FirstName, string LastName, double GPA;
 public:
student( ) //Constructor or Special Member Function (SMF)
~ student( ) //Destructor or Demolition Function (DF)
{
Code =0; FirstName="  "; LastName=" "; GPA =0;
} //other public members
};
```

Figure 3. An object of the given class student inside the public part

In this errand, proposing a thought for arrangement is nothing perplexing to invent. It is not algorithmic and there is not something to be reflection. Should be characterize a class for each of the depict in the issue portrayal objects (students, professor, groups, college, and so forth.) and next that it ought to characterize in each class properties to depict its qualities and strategies to execute the activities that the class can do, for example imprinting in intelligible frame.

This is what needs to be done. *Before execution of this issue* separate the issue into sub issues to execute of each of the classes effectively distinguished can be viewed as a sub issue of the given college displaying issue. In this way, there are the accompanying sub issues as shown in figure (4):

1. Class for the students – Student. Students will have Code, first name, last name GPA, and a Member Function (MF) for imprinting in comprehensible shape – print ( ).
2. Class for the groups – Group. Groups will have a name, a professor and a list of students. It will also have a Member Function (MF) for imprinting in humanreadable frame.
3. Class for the professors – Professor. Professors will have first name, subject and publication, and a Member Function(MF) for imprinting in humanreadable shape or.
4. Class for the college – College It will have a name and will hold all students, all professors and all groups.

Figure 4. Accompanying sub issues to execute of each of the classes of the given college

6. Implementation

It's suitable to begin the execution with the class student since it doesn't rely on upon any of last classes.

First step: Class student

The issue of declaration has just four fields illustrate to the code, first name, last name and the GPA of student. May include some another fields, which produce code the student, a string with the full name of the student and GPA of a print () application to print the student in comprehensible frame. Should be might declare the class student like Figure (5):

STUDENT.C++

```
class student
{
private:
        int CODE;
        string FirstName;
        string LastName;
        double GPA;
public:
        student();
        ~student( );
        student(int CODE, string FirstName, string LastName, double GPA);
        void setstudent(int CODE, string FirstName, string LastName, double
        GPA);
        int get CODE ();
        string getName();
        double getGPA();
        void print();
};
```

Figure 5. Issue declaration for Class student

Prior proceeding forward, need to test the class student to make certain it is right as appeared in Figure (6). How about we make a testing class with a main( ) function and create a student in it and get a right outcome as appeared in Figure (7):

Testing the Class student

```
int main()
{
        student n;
        int CODE;
        string FirstName;
        string LastName;
        double GPA;
        cout << "Enter CODE ";
        cin >> CODE;
        cout << "Enter FirstName ";
        cin >> FirstName;
        cout << "Enter LastName ";
        cin >> LastName;
        cout << "Enter GPA ";
        cin >> GPA;
   n.setstudent (CODE, FirstName, LastName, GPA);
   n.print();
   return 0;
}
```

Figure 6. Test the class student

Output the testing program and get a correct result:

```
Enter CODE 1
Enter FirstName Nadia
Enter LastName Moqbel
Enter GPA 95
Code : 1
Name : Nadia Moqbel
GPA : 95
```

Figure 7. True conclusion in Class student

Also, proceed with the execution of another class (group, professor, and college).

Second Step: Class group

The following class can be characterize is Group. Also, picked it in light of the fact that the just a single required for its declaration is the class student. The MF, which will characterize, are the name of the group, a rundown of the students, which have a place with the group, and a professor who professors the group. To execute the rundown with of the students using List<student>. Also, add a print () function to empower printing the group in a humanreadable text shape. Should see the execution of the class Group like Figure (8) as follows:

Group.C++

```
class group : public college
{
 private:
        string Nameofprofessor;
        string listofthestudents;
 public:
       group( );
       ~group( );
       void create(void)
         {
                getcollege();
                cout<<"Enter Nameofprofessor :-";
                cin>> NameofProfessor;
                cout<<"Enter listofthestudents :-";
                cin>> listofthestudents;
         }
      void display(void)
         {
                dispcollege();
                cout<<"\nEnter Nameofprofessor    :-"<< Nameofprofessor;
                dispcollege();
                cout<<"\nEnter listofthestudents    :-"<< listofthestudents;
         }
};
```

Figure 8. Issue declaration for Class group

So important when create a group to specify an empty list of students to it. In the event that the rundown of students is unassigned, it will be invalid and when we attempt to include a students, will get a special case. Presently, the group class works effectively. We can proceed with the following class.

Third Step: Class professor

How about characterize the class professor. The professor. must to have first name, Subject and Publication. Characterize it directly iteration the logic in the group class

Professor .C++

```
class professor : public group
{
private:
        char sub[20];
        char pub[20];
 public:
       professor();
       ~professor();
       void create(void)
         {
          get group ();
          cout<<"Enter Subject :-"<<endl;
          cin>>sub;
          cout<<"Enter Publication :-";
          cin>>pub;
         }
       void display(void)
         {
           disp group ();
           cout<<"\nSUBJECT    :-"<<sub;
           cout<<"\nPUBLICATION:-"<<pub;
         }
};
```

Figure 9. Issue declaration for Class professor

Such as the class Group, it is essential to create and empty list of groups instead of outside the group's ownership uninitialized.

Fourth Step: Class college

At long last, College class utilizes all the classes that are already decelerated. It ought to have a name and should hold a list of students, a list of professors and a list of groups such as Figure (10) as follows:

Testing the Class College

```
main( )
{
professor objp[7];
student  objs[7];
group  objg[7];
        .
        .
```

Figure 10. Issue declaration for Class college

Therefore, if the output or printing for college, what ought to be printed? Possibly should print its name, every one of its students (with their inward specifics), every one of its professors (with their inward specifics) and every one of its groups (with their

inward specifics). Presently, need to test the college class to make certain it is correct as appeared in Figure (11):

College.C++

```
class college
{
 protected:
        int code;
        char name[20];
 public:
        college();
         ~college();
        void getcollege(void);
        void dispcollege(void);
};
```

Figure 11. Test the class college

7. Simulation Results

Synchronous processes are managed by scheduler class. This language is most proper to the simulation of programs. It grants classes with attributes and strategies that are public by default. It is possible to proclaim them as private. Heritage and virtual capacities are maintained. Memory is managed automatically with junk accumulation or garbage collection. The above program is designed keeping in mind the end goal to store the information of students, professors, groups, and college. The class's student, professor, group, and college have been declared in the program. This program utilizes So it, after execute issue or problem of the system and getting the accompanying expected. Should be creating a testing class with a main() function and get a right outcome for the implementation ability to simulate the program on constructors and destructor in object oriented programming in C++ program as appeared in Figure (12): Special Member Functions (SMF) and demolition function (DF).

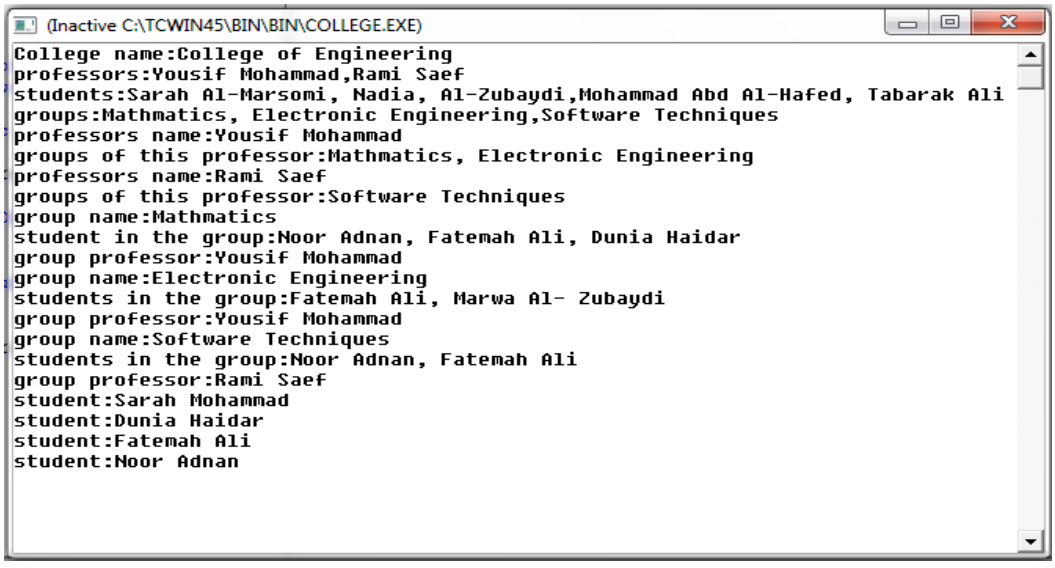

Figure 12. Correct result in Class college

Administration of information for the most part concentrates on the specify for the information components of classes (student, group, professor, college), how it is organized, stored and moved. Administration of data is more worried about the security, exactness, culmination and convenience of different parts of information.

The multifaceted nature of programming or the complexity of software required an adjustment in the style of programming. It was intended to:

1. Create trustworthy programming
2. Diminish production cost
3. Create and develop reusable programming functions
4. Diminish upkeep cost
5. Speed the finishing time of programming development.

The object oriented paradigm was developed for tackling complex issues. It resulted OOP paradigms. OOP appear viable in taking care of the perplexing issues confronted by programming enterprises. The end-programmers and in addition the product experts or software professionals by object oriented software. OOP gives a steady method for correspondence between analysts, designers, programmers, and end-programmers. The result enables the move from the begin to the end state as appeared in Figure (13):

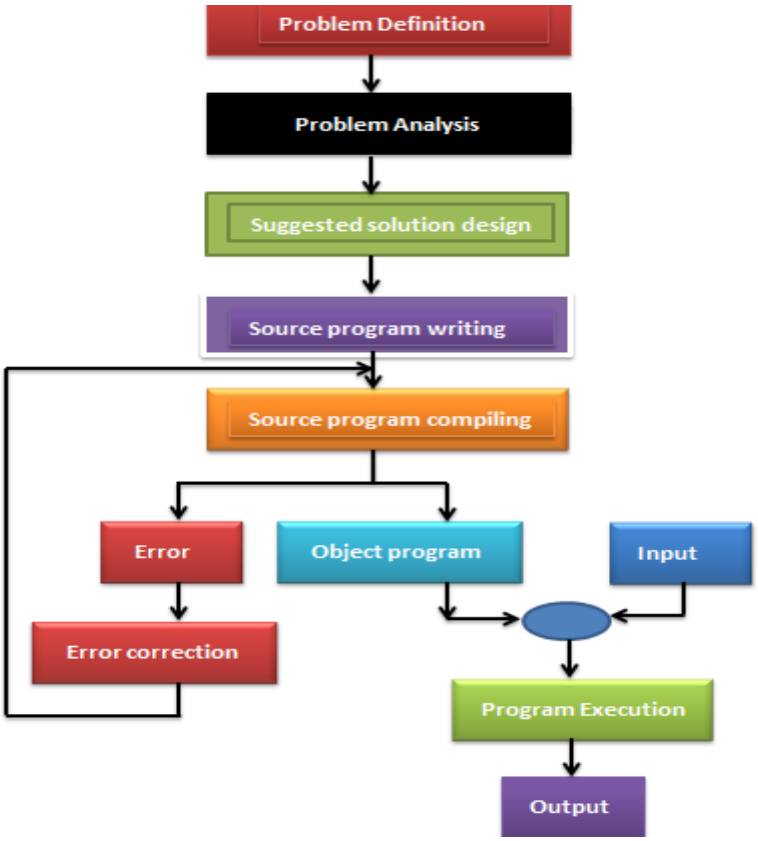

Figure 13. Solution to a problem

8. Conclusions

The essential objective of this paper is to advance object oriented design utilizing C++ and to represent the utilization of the developing object oriented design models. Experienced OOP engineers find that specific methods for doing things work best and that these routes happen again and again. The paper demonstrates how these paradigms are utilized to create great programming designs utilizing special member functions called constructors and demolition function called destructor. Information administration is the improvement, execution and supervision of arrangements, strategies, projects and practices that control, secure, deliver, the value of data and information assets. Information security administration, arranging usage and control exercises to guarantee protection and secrecy and to anticipate unapproved and wrong information get to, creation or change. The idiom, object orientation (OO) has been used to many subjects, for example, analysis, design implementation, data modeling in databases, and distribution. In this paper the idiom object orientation programming (OOP) is intended to cover every one of these subjects, since one of the OO is that it gives a brought together way to deal with these subjects.

9. References

1. Gaston C. H. (2016)." Object Oriented Programming with Swift 2".
2. Gianluca A. Maria C. M. Francesca S.(2016)" Exploiting Linearity in Sharing Analysis of Object-oriented Programs".
3. (2016)"Object Oriented Languages", 1st Edition.
4. Rajkumar B. Thamarai S. S. Xingchen C. "Object Oriented Programming with Java: Essentials and Applications".
5. Chayan V. G. MAYAPYTHON PI, "Concepts of object-oriented programming".
6. (2015)."Introduction to Object Oriented Programming Concepts (OOP) and More".
7. E Balaguruswamy.(2008)."Object Oriented Programming With C++" Fourth Edition.
8. "13 Classes & Objects with Constructors/Destructors".
9. (2013). "Constructors & Destructors in C++ (OOPs).
10. Robert L. (2002). "Object-Oriented Programming in C++", Fourth Edition.
11. Irene G. (2010). "From Procedural to Object-Oriented Programming (OOP) –An exploratory study of teachers' performance".
12. Gal-E. J. Vilner, T. & Zur, E. (2009). "Has the paradigm shift in CS1 a harmful effect on data structures courses: a case study". SIGCSE Bulletin 41(1): 126-130.
13. Kölling, M. (1999). "The Problem of teaching Object- Oriented Programming", Part 1: Languages. Journal of Object-Oriented Programming, 11(8):8-15.
14. Bergin, J.,& Winder,R.(2000)."Understanding Object-Oriented Programming". Retrieved: May, 2010. http://csis.pace.edu/~bergin/patterns/ppoop.html.
15. Stroustrup, B (1994). "The Design and Evolution of C++". Addison- Wesley, Reading MA.
16. Lewis, J. (2000). "Myths about Object-Orientation and its Pedagogy". SIGCS 2000 3/00 Austin TX, USA.

17. Bergin, S., & Reilly, R. (2005). "Programming: Factors that Influence Success". SIGCSE ’05 February 23-27. St.Louis, Missouri, USA.
18. Détienne, F. (1990). "Expert programming knowledge: A schema based approach". In J.M. Hoc, T.R.G. Green, R. Samurcay, & D.J. Gillmore (Eds.), Psychology of programming (pp. 205-222). London: Academic Press.
19. Wiedenbeck, S. & Ramalingam, V. (1999). "Novice comprehension of small programs written in the procedural and object-oriented styles". International journal Human-Computer Studies 51:71-87.
20. (2007)"Object-Oriented Programming ", School of Computer Science University of KwaZulu-Natal February 5.
21. Bob D. "On Object-Orientation", section Theory of Computer Science, Faculty of Science, University of Amsterdam.
22. Dharminder K. "Object-Oriented Programming with C++ ".
23. "COMPUTER SCIENCE C++(083)"ASSIGNMENT BOOKLET, Academic Session: 2016-17.
24. Joyce F. (2009)."Object-Oriented Programming Using C++",Fourth Edition.
25. Herbert S. (1998)."C++: The Complete Reference", Third Edition.