National College of Computer Studies

Paknajol, Kathmandu

Term Paper

On

Object Oriented Programming.

Submitted By: Submitted to:

Bishal Phuyal

Class: BIM 1st Semester

Section: A

Roll no. 09

Date of Submission: 2020/02/23

Acknowledgement

I would like to express my sincere gratitude to all of my wonderful teachers and the mentors for providing their invaluable guidance, comments and suggestions throughout the course of the project. I would specially thank Mr. Dhadhi Ram Grimier for giving me a wonderful guidance by letting us know more about the Term Paper writing with his speeches and also like to thank Mr. Sudil Maharjan sir for providing the essentials documents and formats of term paper through mail.

I am able to complete this Term Paper on time with the help of my friend’s encouragements and involvements so they also indirectly have been the part of my writing so thank you all for the great support and gratitude.

Abstract

As an abstract, there are four main principles that stands the modern booming paradigm of writing program called Object Oriented Programming. The OOP basically stands for the style of writing program on which the whole program is divided into different small sub programs called objects. These objects are dynamic in nature means; they can be used anywhere at any time throughout the program. To wrap up the concept, the OOP is different than other programming language like Structure Programming Language it says that structure programming language focuses on procedural or the functional part of programming whereas OOP focuses on the class and its different objects. The four fundamentals where the OOP stands out are Encapsulation, Inheritance, Polymorphism, Data Abstraction.

Beside these in early 20’s a renowned name in tech field - Robert C Martin aka “Uncle Bob” described the five principles of the Object Oriented Programming/Design. And those principles are Single Responsibility Principle(SRP), The Open Closed Principle(OCP), Liskov Substitution Principle(LSP), Interface Segregation Principle(ISP), Dependency Inversion Principle(DIP). There are many advantages of this programming concepts like easy to write, execute and implement, creates real world entities, easily extensible and many more. Then there are some of the limitations which are small but should not be ignored like data duplication, large size, steep learning curve and not suitable for every problem.

Table of Contents

[Object Oriented Programming 1](#_Toc33330882)

[Introduction 1](#_Toc33330883)

[Class 2](#_Toc33330884)

[Object 2](#_Toc33330885)

[History 3](#_Toc33330886)

[Why OOP than other Programming Paradigms? 3](#_Toc33330887)

[Working Mechanism of OOP 4](#_Toc33330888)

[Fundamental Concepts of OOP 5](#_Toc33330889)

[1. Encapsulation 5](#_Toc33330890)

[2. Inheritance 6](#_Toc33330891)

[3. Polymorphism 7](#_Toc33330892)

[4. Data Abstraction 8](#_Toc33330893)

[The five principles by Uncle Bob 9](#_Toc33330894)

[Advantages of OOP 10](#_Toc33330895)

[Limitations of OOP 10](#_Toc33330896)

[Conclusion 11](#_Toc33330897)

# Object Oriented Programming

## Introduction

Object Oriented Programming also known as OOP is the paradigm or a style of writing a program. This programming style mainly focuses on two basic things called objects and classes. Object-Oriented Programming refers to languages that uses objects in programming. Object-oriented programming aims to implement real-world entities like inheritance, hiding, polymorphism etc. in programming. The main aim of OOP is to bind together the data and the functions that operate on them so that no other part of the code can access this data except that function. Programing that can help programmers while writing a program. Computer programming is the process of writing an algorithm and, it is also the encoding of the algorithm into a notation that can produce and provide information to the users. It can be classified into two groups, that is, system programming and application program-mind. System programming is a sub branch of the general programming that is composed of low level instructions which are used to operate and handle computer hardware. Application programming is considered as the improved version of the computer programs which can perform specific tasks for the users. One of the application programming types is the object oriented programming (OOP) which is about how information is represented in human mind.

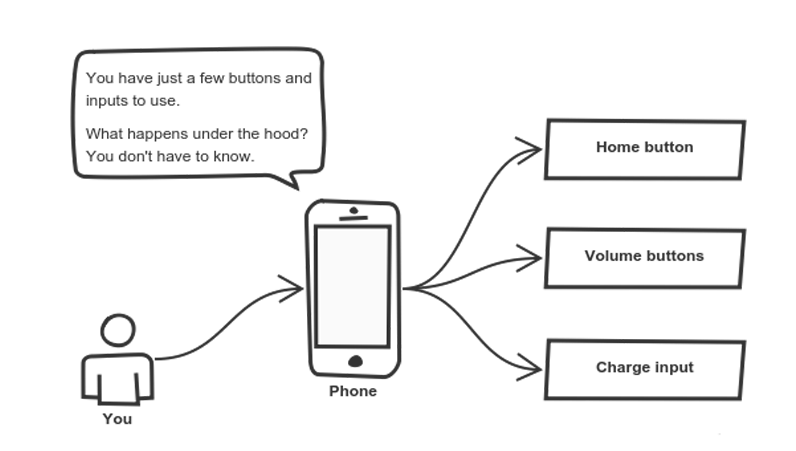


Fig:(Understanding OOP as lots of objects{buttons} in a single class{phone}).

“**Object-oriented programming** (**OOP**) is a programming paradigm based on the concept of objects, which can contain data, in the form of fields (often known as attributes or properties), and code, in the form of procedures (often known as methods)” [1].

### Class

A Class is a user defined data-type which has data members and member functions. Data members are the data variables and member functions are the functions used to manipulate these variables and together these data members and member functions defines the properties and behavior of the objects in a Class. It is a user-defined data type, which holds its own data members and member functions, which can be accessed and used by creating an instance of that class. ... For Example: Consider the Class of Cars.

### Object

 It is an instance of a Class. When a class is defined, no memory is allocated but when it is instantiated (i.e. an object is created) memory is allocated. When a class is defined, only the specification for the object is defined; no memory or storage is allocated. To use the data and access functions defined in the class, you need to create objects.

Some of the popular and mostly used object oriented programming languages are [Java](https://en.wikipedia.org/wiki/Java_(programming_language)),  [C++](https://en.wikipedia.org/wiki/C%2B%2B),  [C#](https://en.wikipedia.org/wiki/C_Sharp_(programming_language)),  [Python](https://en.wikipedia.org/wiki/Python_(programming_language)), [PHP](https://en.wikipedia.org/wiki/PHP), [JavaScript](https://en.wikipedia.org/wiki/JavaScript), [Ruby](https://en.wikipedia.org/wiki/Ruby_(programming_language)), [Perl](https://en.wikipedia.org/wiki/Perl), [Object Pascal](https://en.wikipedia.org/wiki/Object_Pascal), [Objective-C](https://en.wikipedia.org/wiki/Objective-C), [Dart](https://en.wikipedia.org/wiki/Dart_(programming_language)), [Swift](https://en.wikipedia.org/wiki/Swift_(programming_language)), [Scala](https://en.wikipedia.org/wiki/Scala_(programming_language)), [Common Lisp](https://en.wikipedia.org/wiki/Common_Lisp), [MATLAB](https://en.wikipedia.org/wiki/MATLAB), and [Smalltalk](https://en.wikipedia.org/wiki/Smalltalk).

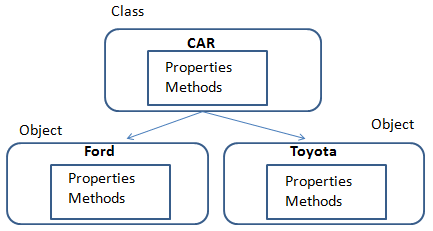


Fig:(Class and Objects examples).

## History

In 1967, the guy name Kristen Nygaard [2] and Ole-Johan Dahl [3] from Norway, developed the programming language called Simula. Which was the first programming language having the features of Object-Oriented Programming. Programming paradigm is a fundamental style of computer programming which classifies programming languages. Different programming paradigms were developed by considering the concepts and abstraction which are used to represent the elements of a program, and steps that compose a computation. Some of the programming languages are designed to support one paradigm and some of them support multiple paradigms. Before OOP languages, there are also some other paradigms such as classic programming [4], modular programming [5] and structural programming [6]. These programming technics were helped the programmers while solving their problems. Depending on improving technology, new structure of OOP has emerged so one of these paradigms is OOP which changed the programming paradigm continued until the day it appeared. Software methodology used before OOP referred to by the name of the procedural programming. This methodology was based on advancing codes in a particular direction and calling the common function that is used to reduce the workload. This methodology, used in the software world for a long time, has some difficulties. First of all, application of the procedural programming developed as a whole cannot be divided. So, each developer working on the application has to know almost every building of application. Due to its building as a whole, it is hard to make changes on the application. The reason of these difficulties is that procedural programming is an abstract that is not able to model the real world. The real world can be simulated by programmers thanks to OOP by using objects and classes.

## Why OOP than other Programming Paradigms?

* In **Structure Programming** style, the program is broken in different functions whereas in OOP paradigm the program is broken in different objects so OOP has advantage over structure programming language that OOP is more flexible and easy to written and debug. Structure programming approach is something like top-to-bottom approach where the program won’t get executed without the completion of the program but OOP is the concept where the program is divided in the sub programs where objects are created and those objects are used wherever the programmer wants. Which makes OOP more versatile and easy because it solves the problem breaking them down into other small problems.
* Again, of-course the OOP is much better than those **Classic Programming languages** which were some hard assembly languages using different mnemonics codes. In OOP the natural languages are used to write the code.
* An object orientated language creates closures with multiple entry points (object) and uses message passing to communicate between them. This might be compared to imperative languages which are lists of instructions, or functional languages that create closures with a single entry point and which communicate by calling each other **Scripting languages** tend to be interpreted at runtime, not compile ahead of time.

## Working Mechanism of OOP

In Object Oriented Programming the two methods called class and objects works where many objects form the class and it can be used in many times in different scenarios. Each object is said to be an [instance](https://en.wikipedia.org/wiki/Instance_(computer_science)) of a particular class (for example, an object with its name field set to "Tesla" might be an instance of class Car). Procedures in object-oriented programming are known as [methods](https://en.wikipedia.org/wiki/Method_(computer_science)) variables are also known as [fields](https://en.wikipedia.org/wiki/Field_(computer_science)), members, attributes, or properties. Objects are accessed somewhat like variables with complex internal structure, and in many languages are effectively pointers, serving as actual references to a single instance of said object in memory within a heap or stack. They provide a layer of abstraction which can be used to separate internal from external code. External code can use an object by calling a specific instance method with a certain set of input parameters, read an instance variable, or write to an instance variable. Objects are created by calling a special type of method in the class known as a [constructor](https://en.wikipedia.org/wiki/Constructor_(object-oriented_programming)). . A program may create many instances of the same class as it runs, which operate independently. This is an easy way for the same procedures to be used on different sets of data.

Variables terms in OOP:

* **Class Variable** is any [variable](https://en.wikipedia.org/wiki/Variable_(programming)) declared with the static modifier of which a single copy exists, regardless of how many [instances](https://en.wikipedia.org/wiki/Instance_(computer_science)) of the class exist. Note that in Java, the terms "field" and "variable" are used interchangeably for [member variable](https://en.wikipedia.org/wiki/Member_variable).[7]
* **Instance Variable** is a [variable](https://en.wikipedia.org/wiki/Variable_(programming)) defined in a class (i.e. a [member variable](https://en.wikipedia.org/wiki/Member_variable)), for which each instantiated [object](https://en.wikipedia.org/wiki/Object_(computer_science)) of the class has a separate copy, or instance. An instance variable is similar to a [class variable](https://en.wikipedia.org/wiki/Class_variable).[8]
* **Member Variable** is a [variable](https://en.wikipedia.org/wiki/Variable_(programming)) that is associated with a specific object, and accessible for all its [methods](https://en.wikipedia.org/wiki/Method_(computer_science)).[9]

Object-oriented programming that uses classes is sometimes called [class-based programming](https://en.wikipedia.org/wiki/Class-based_programming), while [prototype-based programming](https://en.wikipedia.org/wiki/Prototype-based_programming) does not typically use classes. As a result, a significantly different yet analogous terminology is used to define the concepts of object and instance.

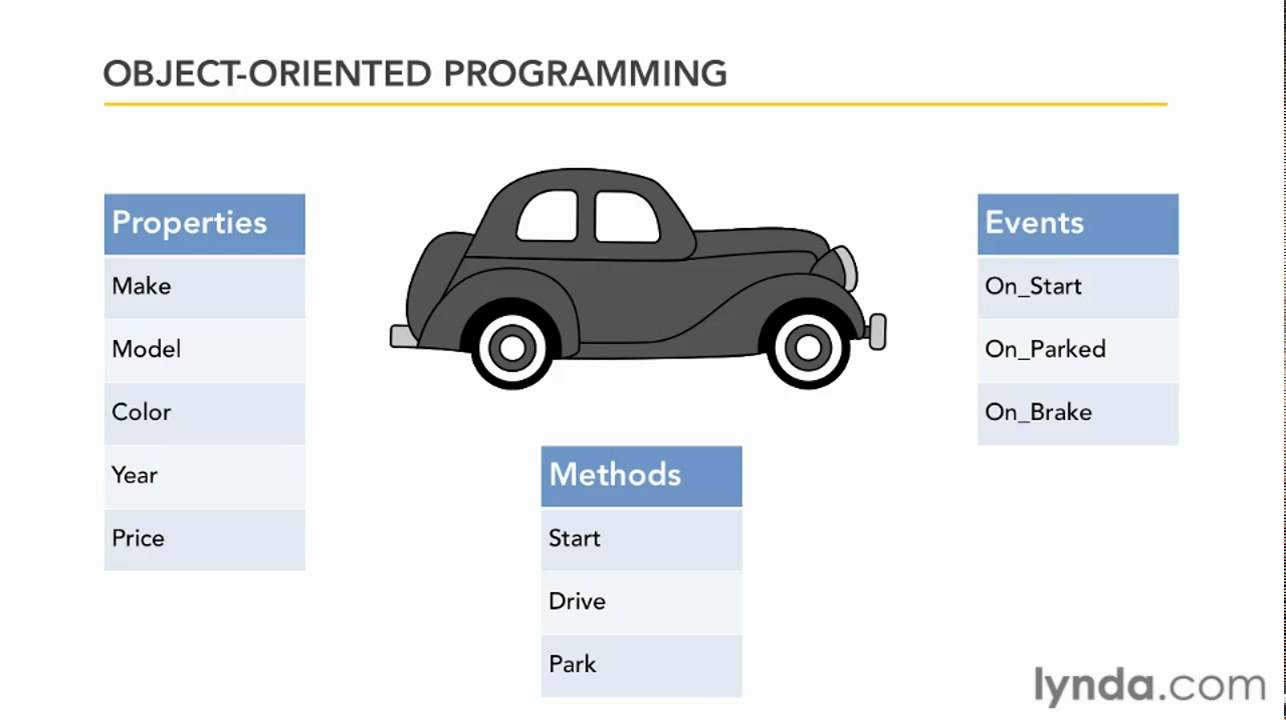


FIG:(Object Orient Programming basic model [10])

## Fundamental Concepts of OOP

Object oriented Programming works on the basic four concepts which are also their features; named as Encapsulation, Inheritance, Polymorphism and Data Abstraction. These four are the pillars of the object oriented programming paradigm. They are described briefly down below:

### Encapsulation

 It is the mechanism that binds together code and the data it manipulates. other way to think about encapsulation is, it is a protective shield that prevents the data from being accessed by the code outside this shield. Technically in encapsulation, the variables or data of a class is hidden from any other class and can be accessed only through any member function of own class in which they are declared. As in encapsulation, the data in a class is hidden from other classes using the data hiding concept which is achieved by making the members or methods of class as private and the class is exposed to the end user or the world without providing any details behind implementation using the abstraction concept, so it is also known as combination of data-hiding and abstraction. Encapsulation can be achieved by declaring all the variables in the class as private and writing public methods in the class to set and get the values of variables.

In Java the get methods like getAge() , getName() , getRoll() are set as public, these methods are used to access these variables. The setter methods like setName(), setAge(), setRoll() are also declared as public and are used to set the values of the variables. These are some examples of the encapsulation beside these there are lot more in many other programming languages.

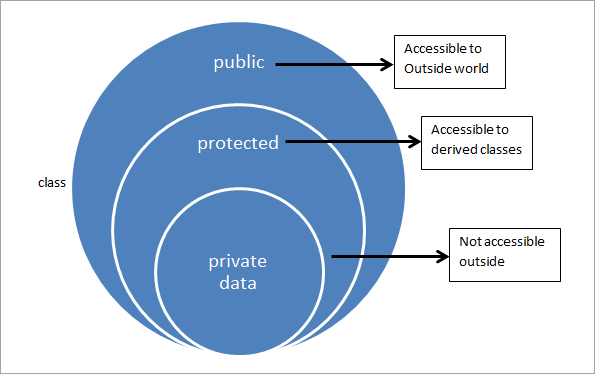


Fig: (Encapsulation: hiding the private data from the world.)

### Inheritance

Inheritance the process where one class acquires the properties (methods and fields) of another. With the use of inheritance, the information is made manageable in a hierarchical order. The class which inherits the properties of other is known as subclass and the class whose properties are inherited is known as superclass.

The idea behind inheritance in Java is that you can create new [classes](https://www.javatpoint.com/object-and-class-in-java) that are built upon existing classes. When you inherit from an existing class, you can reuse methods and fields of the parent class. Moreover, you can add new methods and fields in your current class also.

Terms in Inheritance:

**Class:** A class is a group of objects which have common properties. It is a template or blueprint from which objects are created.

**Sub Class**: Subclass is a class which inherits the other class. It is also called a derived class, extended class, or child class.

**Super Class**: Superclass is the class from where a subclass inherits the features. It is also called a base class or a parent class.

**Reusability:** As the name specifies, reusability is a mechanism which facilitates you to reuse the fields and methods of the existing class when you create a new class. You can use the same fields and methods already defined in the previous class.

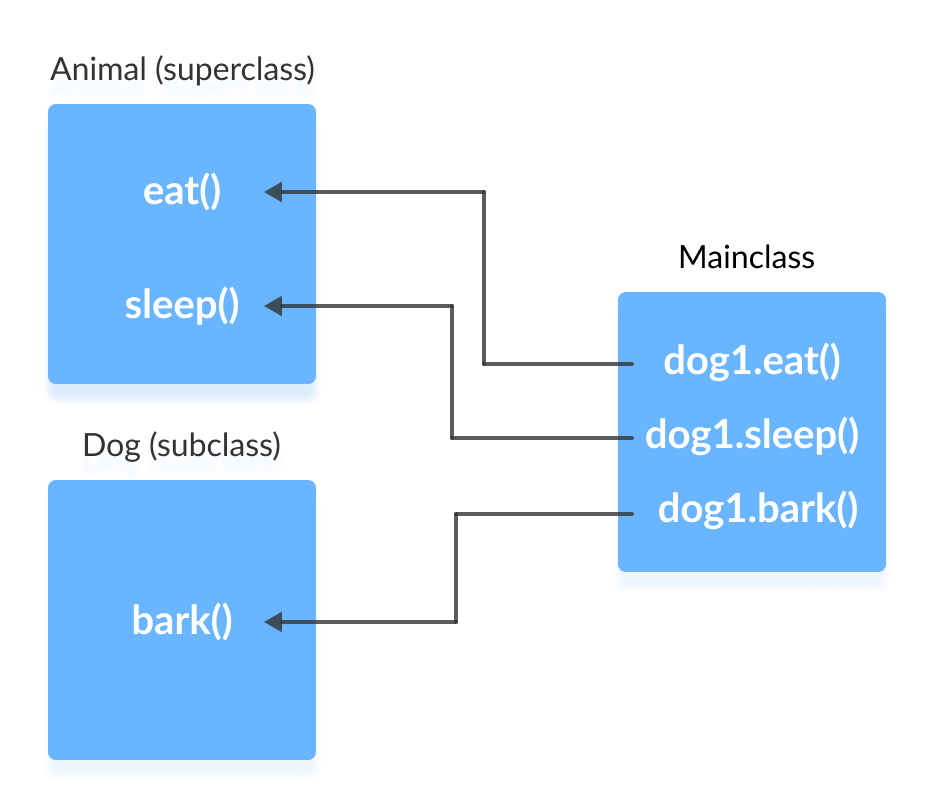


Fig:(Inheritance, showing that object of subclass used in main class).

### Polymorphism

The word polymorphism means having many forms. In simple words, we can define polymorphism as the ability of a message to be displayed in more than one form. Real life example of polymorphism: A person at the same time can have different characteristic. Like a man at the same time is a father, a husband, an employee. So the same person possesses different behavior in different situations. This is called polymorphism. Polymorphism is considered as one of the important features of Object Oriented Programming. Polymorphism allows us to perform a single action in different ways. In other words, polymorphism allows you to define one interface and have multiple implementations. The word “poly” means many and “morphs” means forms, So, it means many forms. It has basically two types:

* Static polymorphism also known as Compile time polymorphism is the type of polymorphism is achieved by function overloading or operator overloading.
* [Runtime polymorphism](https://www.geeksforgeeks.org/dynamic-method-dispatch-runtime-polymorphism-java/) also known as Dynamic Method Dispatch. It is a process in which a function call to the overridden method is resolved at Runtime. This type of polymorphism is achieved by Method Overriding.

For example, think of a superclass called Animal that has a method called animalSound(). Subclasses of Animals could be Pigs, Cats, Dogs, Birds - And they also have their own implementation of an animal sound (the pig oinks, and the cat meows, etc.)[11]

### Data Abstraction

Abstraction is selecting data from a larger pool to show only the relevant details to the object. It helps to reduce programming complexity and effort. In Java, abstraction is accomplished using Abstract classes and interfaces. It is one of the most important concepts of OOP.

Data abstraction is the programming process of creating a data type, usually a class, that hides the details of the data representation in order to make the data type easier to work with. Data abstraction involves creating a representation for data that separates the interface from the implementation so a programmer or user only has to understand the interface, the commands to use, and not how the internal structure of the data is represented and/or implemented.

Levels in Data Abstraction are:

* Physical(internal) level: The lowest level connects to the database and performs the actual operations required by the users.
* Conceptual level: It consolidates external concepts and instructions into an intermediate data structure that can be devolved into physical instructions.
* External level: It is exposed to users and developers and supplies a consistent pattern for performing database operations.

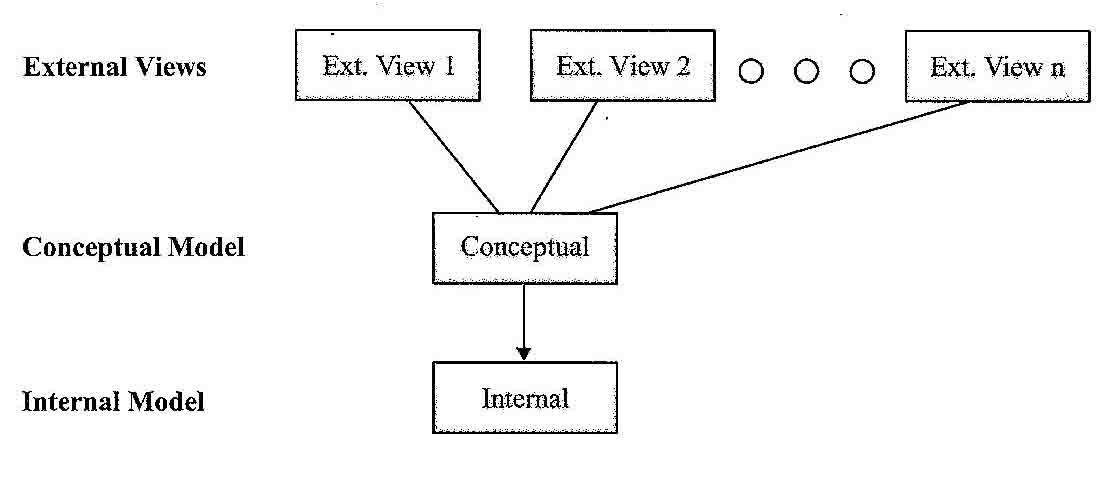


Fig:(Three layers of the Data Abstraction)

## The five principles by Uncle Bob

Uncle Bob “Robert C. Martin” listed 11 principles of good Object Oriented Design in 2000’s. The first five principles are principles of what makes good class design. These five principles are what have become known by the acronym “SOLID” which Michael Feathers helped coin. These principles when learned and implemented correctly help developers create and maintain a codebase that is strong but flexible to grow and change with small difficulty. When you look at a quick description for each of this principles, you might find yourself saying, “Well that’s obvious” but keep in mind that for many young developers these ideas aren’t always so clear as to why they are important. The five principles are:

1. Single Responsibility Principle: states that “A class should have one, and only one, reason to change.” Think of this as a warning not to create god objects that try to do too many different things.
2. Open Closed Principle: states that “You should be able to extend a classes behavior, without modifying.” This is all about using polymorphism and abstraction to build new functionality.
3. Liskov Substitution Principle: states that “Derived classes must be substitutable for their base classes.” The idea behind this principle is that code should be written such that if you create a new derived class from a base class you should not have to modify code to work with this derived class.
4. Interface Segregation Principle: states “Make fine grained interfaces that are client specific.”
5. Dependency Inversion Principle: states “Depend on abstraction, not on concretions.” Dependency Inversion, often confused with Dependency Injection, is all about building classes that are created to use abstractions of 3rd party objects so that the concrete objects can be changed at will without needing to rewrite your code.

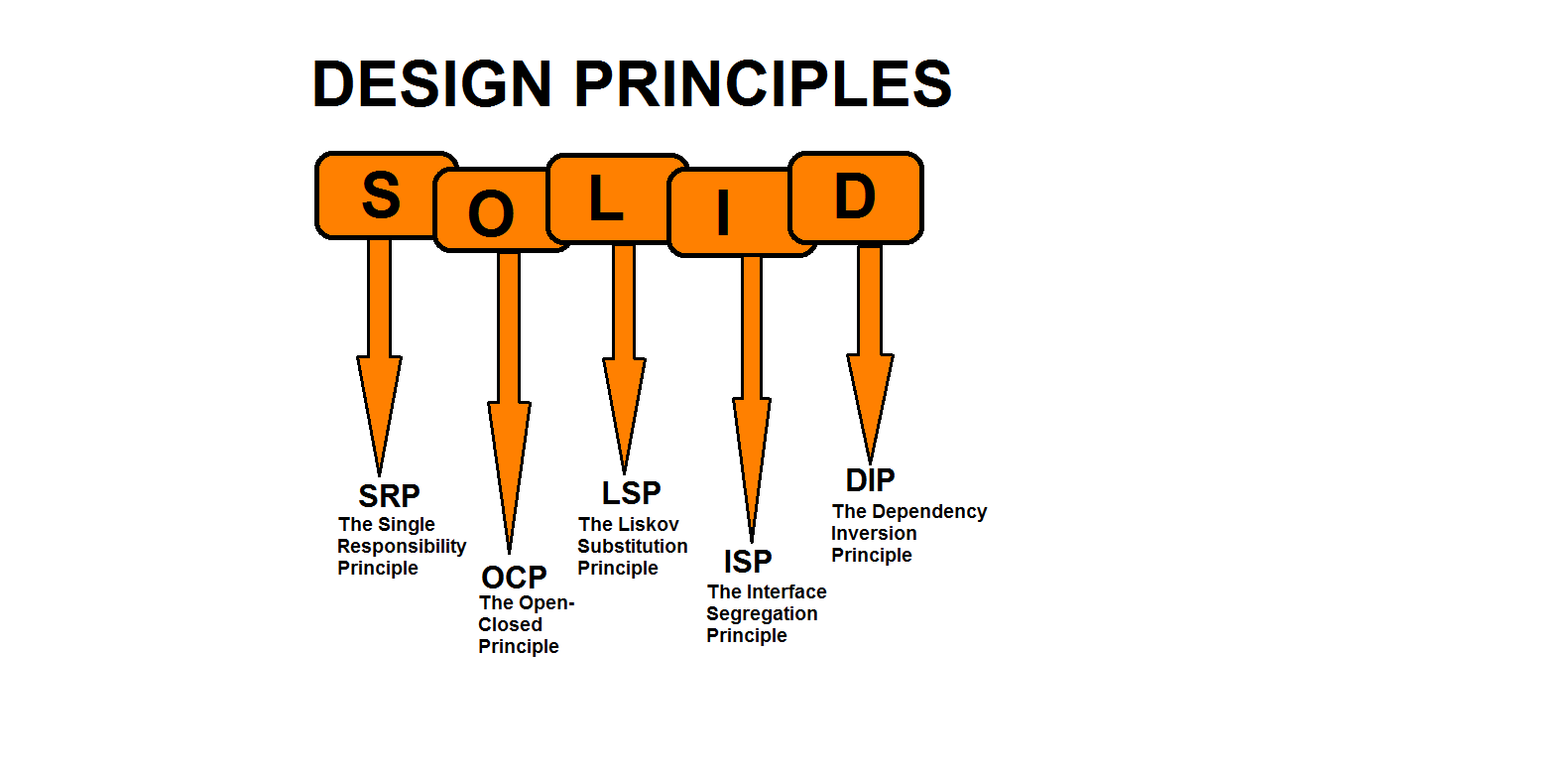


Fig :(Solid Principles by Robert C Martin)

## Advantages of OOP

Due to the flexible and dynamic property of the OOP it has enormous amount of benefits in writing program. Here are some of the benefits of the object oriented programming languages:

1. Object oriented programming can be easily reused in other programs due to its re-usability feature.
2. Software maintenance of the object oriented programming is easier than as compared with structured oriented programming as much time has been invested while doing the planning of the code so minimum flaws are found.
3. Through object oriented programming language, complex systems of real world can be converted into software solutions.
4. Objected oriented programming is easy to implement.
5. Object oriented programming can easily extensible. New features or changes in operating environment can be easily done.
6. Large programs are difficult to develop. Objected oriented programs focuses the developer to do extensive planning which will reduces programming flaws and better design.
7. Objects are created on real world entities.

## Limitations of OOP

Though there are lot more benefits of this programming paradigm it also have some of its own limitations which cannot be ignored. They are:

1. Larger program size: Object-oriented programs typically involve more lines of code than procedural programs.
2. Steep learning curve: The thought process involved in object-oriented programming may not be natural for some people, and it can take time to get used to it. It is complex to create programs based on interaction of objects. Some of the key programming techniques, such as inheritance and polymorphism, can be challenging to comprehend initially.
3. Not suitable for all types of problems-for smaller problems it is in general not suitable.
4. Data Duplication might occur.

## Conclusion

To conclude, OOP is the paradigm with many principles and functionalities. It is a vast subject to roll up since I have tried to cover up as many possible things I could and can in a term paper. It is a Programming concepts that separates the huge disgusting procedural programming and builds a complete set of programming in the form of independent objects, it never complete set of program because it does not have beginning and ending point it just starts at nowhere and ends at nowhere. It is a paradigm that gives us many options how we can implement our solution. With many options come many places where it can go wrong. And so people of FP say OOP adds unnecessary complexity. And that is true, but only for the cases when OOP is done wrong. Proper OOP makes code easier to read, understand and maintain.

Lastly, OOP is a very useful and popular method or a style to write a program which increases the effectiveness and the productivity in programming since it works on the small parts dividing the big chunks of programming.

References

[1] A definition of the OOP from <https://en.wikipedia.org/wiki/Object-oriented_programming>.

[2][3] Krysten Nygard and Ole-Johan Dahl the inventor of the first OOP language called Stimula <https://en.wikipedia.org/wiki/Simula>

[4] Classical Programming (Assembly level languages and mnemonics) <https://www.johndcook.com/blog/2014/07/01/classical-programming/>

[5] Modular Programming (design that emphasizes separating the functionalities) <https://en.wikipedia.org/wiki/Modular_programming>

[6] Structure Programming (full structure or procedural basis) <https://en.wikipedia.org/wiki/Structured_programming>

[7][8][9] Different variables in OOP <https://www.sciencebuddies.org/science-fair-projects/science-fair/variables>

[10] A picture of Basic OOP model from Lynda.com <https://i.ytimg.com/vi/NUl8lcbeN2Y/maxresdefault.jpg>

[11] An example of Polymorphism from w3schools.com

Beside this; youtube.com, OOP in Javascript(book), some pdfs and other websites are the sources of the research.