CHAPTER

# 1 Principles of Object Oriented Programming

By now you must have used several software packages to do a particular task. You must have observed that you cannot do anything beyond whatever the software is intended for. How about creating a software that will make the computer do what you want it to do? This can be done with the help of a program.

A **program** is a set of instructions given to th computer to perform a particular task.

A program is a set of instructions given to the computer to perform a particular task. Such programs or collection of programs make/makes a software. But there is one problem, the instructions that you give to the computer should be given in a language that the machine understands. Now the machine understands only one language called the binary language. The binary language consists of 1's and 0's which is very difficult for us, humans, to understand and learn. As every instruction that you give to the computer should be a combination of 1's and 0's, it is difficult to learn as we are already accustomed to letters of the alphabet and words.

Therefore, the scientists thought of building computer languages (called High-Level Language) that are more *English-like,* independent of the hardware and is therefore, easier to learn. One such computer language is Java. However, once the program is written using Java, it needs to be translated into Machine Language. This is done with a specialised software called the Translator. A translator comes in two varieties, i.e., Interpreter and Compiler. Whatever may be the variety, its main purpose is to convert the program into machine language. Although the technology behind the Java language is highly complex and you will learn it in the coming years, presently you are going to learn programming using Java at a preliminary level.

A **high-level languag** a programming langu that is closer to hum languages, and throu which programs are written that are mor or less independent of a particular type computer (hardwar

## Difference Between Procedure Oriented and Object Oriented Language

When it comes to programming you need to key in some instructions into an editor abiding by the syntax (or grammar) of that language. A set

A **Procedure Oriented Language** is a type of **programming language** that specifies a series of well-structured steps and **procedures** within its **programming** context to compose a program.

of written ins... The sour...
**code**. You can write and re-write the source code and make the compu...
perform the task until the desired task is attained.

When it comes to writing a program there are various styles a... methodologies of writing. The choice of the methodology depe... upon the task and the language involved. Presently two very popul... methodologies are used to develop a software, namely, Procedu... Oriented Programming and Object Oriented Programming.

Computer Languages are specifically designed according to th... methodologies. Procedure Oriented Languages allow a large progra... to be divided into smaller programs that are also called procedure... or functions. This increases the maintainability and reusability of t... program. However, more emphasis is given to the procedure than... the data upon which the procedure works. Object Oriented Language... on the other hand, emphasises data over a procedure. Programs a... developed in such a way that they represent an object. An object ... anything that has certain characteristics and behaviour. This results ... much better handling of the program with respect to present requirem... of software solutions.

The following table shows some of the basic differences betwe... Procedure Oriented Language and Object Oriented Language.

| Procedure Oriented Programming | Object Oriented Programming |
|---|---|
| A large program is divided into smaller segments or procedures. | A program is represented as an object. |
| More importance is given to the program rather than the data. | More importance is given to the data rather than the program. |
| It follows top down approach. | It follows bottom up approach. |
| Here data can move freely from procedure to procedure in the system. | Here objects can move and communicate with each other through member functions. |
| Adding new data and procedure is difficult in a program. | It provides an easy procedure to add data and procedure to a program. |
| It does not have any proper way for hiding data so it is **less secure**. | It provides *Data Hiding*, thereby providing **more security**. |
| Examples of Procedure Oriented languages are: C, VB, FORTRAN, Pascal. | Example of Object Oriented languages are: C++, JAVA, VB.NET, C#.NET. |

## Concept of Objects

In this world we see several objects around us. The chair on which yo... sit is an example of an object. The computer where you work upon is a...

example of an object. The pen that you use for writing is also an example of an object. Even we ourselves are examples of objects. The objects that you see around yourself may be living or non-living. Whatever may be the situation you will always find two unique features of an object, namely, characteristics and behaviour. For example, a *car*, its characteristics is represented by its *colour*, its *length* and *breadth*, *number of people* that can sit, the *wheel* radius, and its behaviour is the *speed*, how fast it can go from one place to another, the amount of *steering* control it can have, whether the drive is smooth or not, etc.

The *book* that you are reading now is also an example of an object. Its characteristics is represented by the information it holds, size, volume and its colour. The behavioural aspect comprises the methods for accessing the information it contains. For example, you can open the *book*, turn a page, read a paragraph, and search the table of contents, and so on. The information contained in the book along with the methods for accessing all the information is what comprises the object known as this *book*.

Trying to read this book will require some sort of light source, which may be the open sunshine or by a electrical bulb or just a lamp. The *lamp* may also be considered to be an object. Apart from its usual characteristics and behaviour, it also contains information about its state. The *state* of an object is the particular condition it is in. For example, a lamp can be *on* or *off*. The lamp's switch (*method*) is used to turn the lamp on and to turn it off to access the state of the lamp.

This *book*, too, has state information. It may be either in open or closed state. While being open, it can be turned to a particular page. The pages by itself are objects in their own right. They contain information and can be accessed through the *Contents* method. The *book* object can be viewed as being composed of *page* objects. The book's methods provide access to pages, and the *page* methods provide access to the information contained on a particular page. Thus, the behaviour aspect of an object contains methods that change the state of an object or provide a usage of the object.

An object is an entity certain characteristi behaviour.

So, we can define an object as an entity with certain characteristics and behaviour. All objects have their own individuality and are distinguishable. For example, you may have two identical pens with the same colour, texture, shape and size, yet they are two distinguishable pens. Similarly, there may be several copies of this book, but each holds a separate identity. For example, the book may be kept open with someone, open to a certain page with someone or open to a different page with someone.

Thus, the book may ... instance. The ...
object is represented through the values/attributes of its characteristics ...
a given point of time.

Programming using objects promotes understanding of the real ... and provides a practical basis for computer implementation.

## Object Oriented Programming

Object Oriented Programming (or OOP) is a technique of implementing programs which are organized as a co-interactive collection of objects, each of which represents an instance of a class.

Programming is a process by which we give instructions to the ... to do a particular task. There are different techniques and terms involved in writing a program. One such technique is Object ... Programming.

Object Oriented Programming (or OOP) is a technique of implementing programs which are organized as a co-interactive collection of ... each of which represents an instance of a *class*. A *class* describes ... of objects with similar properties, common behaviour and ... relationships. Figure 1.1 shows a class diagram (left) and possible ... (right) described by it. Objects *Amit Mitra*, *Anu Mitra*, *Srija Das* ... with their ages are instances of class *Person*.

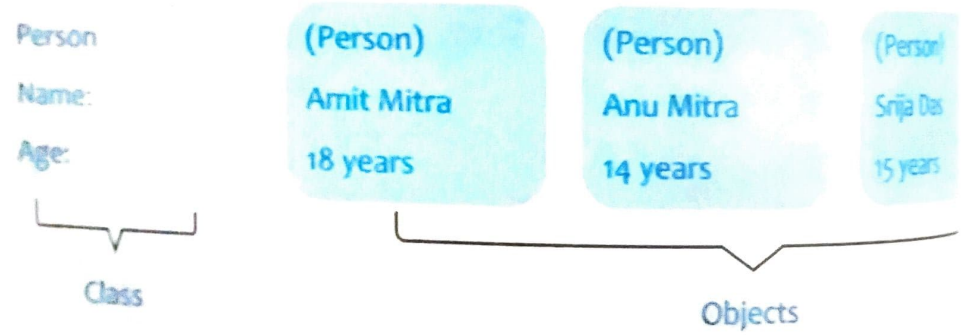


Figure 1.1: The person class and its objects

Another example will help you to understand the relationship ... classes and objects. Let us imagine a class (or group) named ... The **Square** class defines the side length and colour property of ... Every instance (or object) of the **Square** will be represented by ... length and colour property. For example, an instance of the ... may be a square of length 5 units and red in colour, similarly ... instance of the **Square** may be of length 7 units and blue in colour.

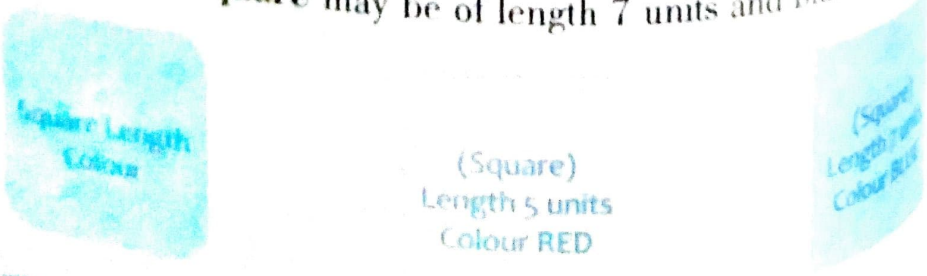


Figure 1.2: The ...

Thus, through the OOP technique it is possible to represent objects having characteristics and behaviour. It forms a powerful technique to represent objects as in the real world. The characteristics of an object are represented through *data members* and behaviour is represented through *member functions in programming world*.

## Characteristics of OOP

Object Oriented Programming (OOP) is based upon certain principles. This section describes the principles of OOP:

- Abstraction
- Encapsulation
- Polymorphism
- Inheritance

### Abstraction

*Abstraction refers to the art of hiding the complexities and giving a simple interface.* For example in a car, for a layman, it is good enough to know driving. Hardly will he know the intricacies of the movement of the engine, or the working of the electrical and electronic components. What he will know or will be interested with, is the *steering* for *turning*, the *accelerator* for *speed*, gear for power and the *switches*. This is because he has been given a simple interface and the complexity of the car engine movement has been kept completely hidden from him. Thus he doesn't bother upon the intricacies of the car's engine movement, but concentrates on driving. This is what is required for programming; all the complexities should be *encapsulated* in such a way, that a simpler interface is given and henceforth the programmer can carry on with the development process at a higher level.

**Abstraction** is a principle of Object Oriented Programming (OOP) that hides certain details and only shows the essential features of the object.

However, abstraction is the selective examination of certain aspects of a problem. The goal of abstraction is to isolate those aspects that are important for some purpose and suppress or hide those aspects that are unimportant. Abstraction must always be for some purpose, because the purpose determines what is and is not important. Many different abstractions of the same thing are possible, depending on the purpose for which they are made.

For example, a simple switchboard that is used in your room, is an abstraction. Imagine if the concept of switchboard did not exist. You would find a great number of wires hanging around your room. To start an appliance, you would need to join two wires. But essentially you need to know which group of wires should be joined to start the appliance.

Data abstraction is the reduction of a particular body of data to a simplified representation of the whole.

Moreover, more the number of wires, more it will be confusing t identify the wires to start an appliance. Thus, joining of wires is not on confusing but may also prove fatal as far as safety is concerned. T the electrician installs a switchboard that connects each of the wire t switch. So, it is just enough to know for the user to use that switch to s the appliance.

Similarly, when a rider drives a bike, the internal complexity of ignit acceleration, mileage, etc. hardly concerns him. He is only interested driving. He knows the components that accelerate the bike. He is aw that pushing the break helps to stop the bike, and is able to use switches that put on the lights, etc. Thus the abstraction of the bike mak the rider only concentrate on riding the bike rather than understand internal complexities.

However, abstraction is relative to the context of the requirem It changes in accordance to the purpose or requirement. For exampl car produces several abstractions or levels of abstractions. For a dr of the car, some of the abstractions are:

- Steering position, type (Power Steering or not), shape and size.
- Accelerator position, its smoothness and the speed of acceler that the car can gain.
- The brake system, which can stop the car and how fast.
- The head lights, their position and power that can give a clear vis at night.

Similarly for the maintenance incharge of the car, the following are th abstractions:

- The engine capacity and life.
- The wheel alignment type, status, etc.
- The coolants, engine oil status, etc.
- The mileage aspect, which is essential for oil conservation.

## Encapsulation

**Encapsulation** is a principle of Object Oriented Programming (OOP) that binds together the characteristics and behaviour of an object.

The term encapsulation refers to the act of enclosing one or m items within a (physical or logical) container. In software e encapsulation binds together characteristics and behaviour of object. The characteristics is represented by *data* and the behavi is represented by *functions* or *methods*. Thus encapsulation i technique that *binds together function and data into a single unit*. Y can imagine it to be as a protective wrapper that prevents the co and data from being accessed by some other code defined outside wrapper. To relate this to the real world, consider any switch base

It encapsulates several wires for different electrical or electronic appliances. You, as the user, have only one method of affecting this complex encapsulation: by switching on a particular switch, which in turn will start a particular appliance connected to it and not any other appliance. This same idea can be applied to programming. The power of encapsulated code is that everyone knows how to access it and thus can use it regardless of its implementation details –and without fear of unexpected side effects.

Encapsulation is also frequently confused with abstraction, since the two concepts are closely related. Abstraction is a process of hiding the complexity and giving a simple interface. Encapsulation on the other hand is the mechanism by which the abstraction is implemented.

Consider this real world example. A company wants to setup a huge building. The details regarding the materials that would be used for the construction (*i.e.*, bricks, cement, iron rods, wood, etc.), type of work, manager of the company, number of floors, design of the building, cost of the building, etc., can be classified as Abstraction. Whereas, the type of bricks used, who all work for which departments and how they will work, cost of each and every element in the building, etc., comes under Encapsulation. Thus the term Abstraction is used to define the outer layout used in terms of design and the term Encapsulation is used to define the inner layout used in terms of implementation.

## Polymorphism

**Polymorphism** is an Object Oriented Principle that enables an object to take on many forms. Thus we can perform a *single action by different ways.*

Before defining the term polymorphism let us begin with an example. Say in a room stand a mouse, a cat and a dog. Now from the far corner of the room there comes a screeching sound. The mouse, the cat and the dog all immediately turn towards the direction of the sound. The dog barks, the cat twitches its tail and the mouse sniffs the air. Thus, you can see, for the same screeching sound, the animals behave differently. So, the animals behaved differently for the same effect. This is exactly what polymorphism is.

Here is another example. Shobha is a married woman and the mother of 2 children. She has a teaching job. She is a women first, teacher in a school when she is at school, someone's wife at home, mother of her children and obviously someone's daughter. Thus, you can see that Shobha plays different roles at different times, and that is what polymorphism is.

The term Polymorphism is actually a combination of two terms *Poly* and *Morphism*. Here the term *Poly*-means many and *Morphism* means forms.

It is the ability of a method or a function to assume differen
In Object-Oriented Programming, this refers to the ability of o
have many methods of the same name, but each one responds to di
type of specific behaviour as it has different forms.

## Inheritance

Inheritance is a mechanism in Object Oriented Programming that allows a new class to be created from an existing class.

In real world you must have seen inheritance. As the name inhe
suggests, an object is able to inherit characteristics and behavio
another object. In more concrete terms, an object is able to pass
state and behaviour to its children. For inheritance to work the
need to have characteristics in common with each other.

It is basically a term that is used to represent hierarchical relation
*generalization.*

For example, let's say we make a class called "Human" that repr
our physical characteristics. It's a generic class that could represe
me or anyone in the world. Its state keeps track of things like num
legs, number of arms, and blood type. It has behaviours like eat,
and walk. "Human" is good for getting an overall sense of what ma
all the same but it can't, for instance, tell me about gender differ
For that we'd need to make two new class types called "Man" and "W
(fig. 1.3). The state and behaviours of these two classes will diffe
each other in a lot of ways except for the ones that they inheri
"Human".

Inheritance allows all members of one class to be the members of another class.

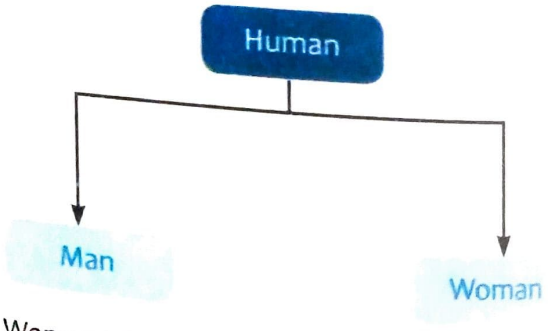


Figure 1.3: Man and Woman inherit traits from Human class

Therefore, inheritance allows us to encompass the parent class' sta
behaviours into its child. The child class can then extend the sta
behaviours to reflect the differences it represents. The most imp
aspect of this concept to remember is that the child class is a
specialized version of the parent.

Here is another example, generalization of the term "mammal", w
may have several levels of *sub-generalizations* as depicted in fig
1.4. Inheritance is a term that is used to establish relationship
hierarchical classification.

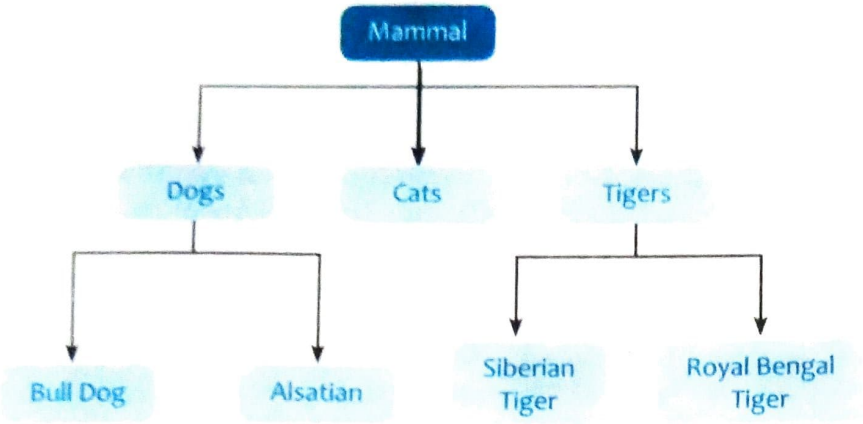


Figure 1.4: The Mammal hierarchy

Inheritance enforces generalization, thus helping in categorically identifying the relationship between each levels of inheritance. For example, the Bull Dog and Alsatian are some popular breeds of Dogs (generalization), similarly the Siberian Tiger and Royal Bengal Tiger are some breeds of Tiger (generalization). The Dog, Cats and Tigers are again a generalization of a generic Mammal.

In OOP the technique of Inheritance is generally used for facilitating reusability. For example, a developer after modeling a system, tries to group similar classes together and reuses common code. Often code is available from past work, which the developer can reuse and modify, where necessary, to get the precise desired behaviour. Thus inheritance also reduces the hazard of rewriting a code, simplifying the task of a developer.

## Recapitulation

- A program is a set of instructions given to the computer to perform a particular task.
- A program written in a text editor or any editor is also called a source code.
- A Procedure Oriented Language specifies a series of well-structured steps and procedures within its programming context to compose a program.
- Object Oriented Programming (or OOP) is a technique of implementing programs which are organized as a co-interactive collection of objects, each of which represents an instance of a class.
- An object is an entity with certain characteristics and behaviour.
- The characteristics of an object in programming is represented by data members and the behaviour of an object is represented by member functions.
- The four principles of Object Oriented Programming are: Abstraction, Encapsulation, Polymorphism and Inheritance.
- Hiding the complexity and giving a simple interface is called Abstraction.