2. Object Based Languages: These are those programming languages which don't support all the 4 principles mentioned above but they support some of them as per their requirement . The most popular example in this category is the language called JavaScript which is considered no. 1 programming language for web development. This langauge originally didn't support class based OOP which means that we can't create class in Javascript original version however later on it supported classes but internally converted these classes into functions

Other than JavaScript we have languages like VB Script, Action Script, J Script etc. which also are object based langauges

3. Object Oriented Languages: These are those languages which support all the 4 principles mentioned above but they never force a programmer to always use these principles in his program i.e. it is not compulsory to use class and objects in every program of these languages. Examples in this category are C++, Modern C++, Python, PHP. etc.

4. Pure/Strict/Full Object Oriented Languages: These are those languages which always force a programmer to use class in every program which he writes and following are their 3 impt characterstics.

a) They always make us of class in every program

b) They don't support global declaration as every statement is written inside the class body

c) They don't have variables and only support OBJECTS

In modern era of computer programming languages like Java , C#, SCALA, KOTLIN are considered as almost pure OBJECT ORIENTED LANGUAGES because they voilet the 3rd characteristics mentioned above and allow us to create variables. However languages like Java permit us to convert these variables into OBJECTS using special set of classes called WRAPPER CLASSES. Strictly speaking, there is a language called Small talk which is considered to be 100% pure OBJECT ORIENTED LANGAUGE.

4 pillars of Object Oriented Programming

==========

1. Encapsulation : The word encapsulation is derived from a word called CAPSULE . Just like a capsule has a mixture of multiple medicines wrapped in a single unit, similarly object oriented programming says that every program which we write is also a mixture of just two things
   1. Data (variables)
   2. Functions (Operations/code)

Since data and functions continously intract with each other so OOPs says that they must be kept together also . Thus acoording to OOPs the word encapsulation is defined as **bundelling of data and functions in one single unit.**

For example: when we create a class we can declare data as well as member functions in that class and so we say that **class is an example of encapsulation**

Benefits ?

=========

The most impt benefit offered by encapsulation is data security

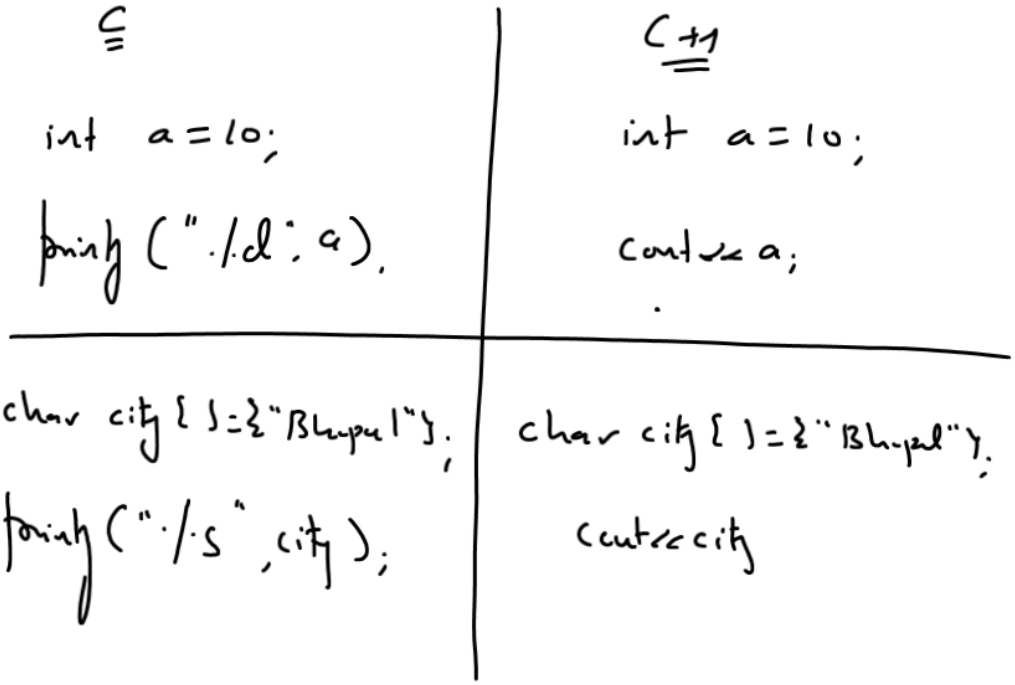
How?

This is because the data members of a class are by default **private** and so they can't be directly accessed from outside the class. Only member functions of the class are eligible for accessing these private data members. Thus we can say that due to encapsulation the data members of the class remain safe and secure from any external tampering

2. Polymorphism : The word polymorphism is composed of two greek words which are

   a) Poly (means multiple)
   b) Morph (means forms)

Thus the word polymorphism means the ability to have multiple forms. In other words we can say that if a single entity can show different behaviours in different situations then we say that it is behaving polymorphically.



For example: In C++ if a programmer wants then he can redefine the builtin operator '+' in such a way that '+' can be used for joining two strings exactly in the same way like it is used to add two integers.

So 10+20 ------>30
and
"Good"+"Morning"---->"GoodMorning"

when we acheive this we say that the operator '+' is behaving polymorphically.

In C++ polymorphism is divided in two categories:

1. Compiletime polymorphism : Examples are **function overloading, contructor overloading and operator overloading**

2. Runtime polymorphism : Examples are **virtual functions, pure virtual functions and abstract classes**

The most impt benefit offered by polymorphism is **SIMPLICITY** i.e. it is much more easier to use polymorphic code as compared to its non-polymorphic alternate

3. Inheritance : To inherits means to acquire features of an existing entity in a newly created entity, just like a child inherits looks of his/her parents. Similarly in C++ if a programmer wishes then he can reuse the features (Data Members and Member Functions) of an existing class in a newly created class. But to do this, programmer will have to apply the principle of INHERITANCE

The class which gets inheritance is called as **PARENT OR BASE CLASS** while the class which inherits is called as **CHILD OR DERIVED CLASS.** Thus via inheritance object of child class can call member functions of parent class.

The major benefits of inheritance is code reuseability i.e. the child class programmer is not required to redesign those functions again which already have been coded by the parent class programmer.
This saves a lot of time and effort on the part of child class programmer.