**What is C++?**

* C++ is a cross-platform language that can be used to create high-performance applications.
* C++ was developed by Bjarne Stroustrup, as an extension to the C language.
* C++ gives programmers a high level of control over system resources and memory.

**Why Use C++?**

* C++ is one of the world's most popular programming languages.
* C++ can be found in today's operating systems, Graphical User Interfaces, and embedded systems.
* C++ is an object-oriented programming language which gives a clear structure to programs and allows code to be reused, lowering development costs.
* C++ is portable and can be used to develop applications that can be adapted to multiple platforms.
* C++ is fun and easy to learn!

C++ Program:

#include <iostream>  
using namespace std;  
  
int main() {  
  cout << "Hello World!";  
  return 0;  
}

**Line 1:** #include <iostream> is a **header file library** that lets us work with input and output objects, such as cout (used in line 5). Header files add functionality to C++ programs.

**Line 2:** using namespace std means that we can use names for objects and variables from the standard library.

**Line 4:** Another thing that always appear in a C++ program, is int main(). This is called a **function**. Any code inside its curly brackets {} will be executed.

**Line 5:** cout (pronounced "see-out") is an **object** used together with the *insertion operator* (<<) to output/print text. In our example it will output "Hello World".

**Note:** Every C++ statement ends with a semicolon ;.

**Note:** The body of int main() could also been written as:  
int main () { cout << "Hello World! "; return 0;}

**Remember:** The compiler ignores white spaces. However, multiple lines makes the code more readable.

**Line 6:** return 0 ends the main function.

**Line 7:** Do not forget to add the closing curly bracket} to actually end the main function.

***Enhancement in c towards C++***

* **cin and cout**
* cin and cout are objects of stream class.
* Cin is used with extraction (>>) operator.
* Cout is used with insertion (<<) operator.
* **Scope and Scope resolution operator (: :)**
* Scope: It Means Until where the variable can be accessed.
* Scope resolution operator is used to resolve scope of variable.
* **New- Delete** 
  + In c Dynamic memory is allocated using:
    - malloc() : initial value is garbage.
    - calloc() : initial value is zero.
    - Free() : used to deallocate memory.
* In C++
  + New: used to allocate memory.
    - EG: Ptr = new num;
    - New calls constructor while malloc does not call constructor.
  + Delete: used to free memory.
    - Delete may call destructor while free() will not call destructor.
* Function Overloading:
  + When task to be performed is same, we can write more the one functions with same name but with different arguments (type, order, number).
  + Compiler do name mangling or name decoration process and calls the function.

***Features of object-oriented language (C++):***

* Object:
  + Basic Runtime Entity.
  + Instance of Class.
* Class:
  + Collection of Objects of similar type.
  + Collection of member functions and member variables wrapped together in a single unit.
* Abstraction:
  + Act of representing essential features without including its background details.
* Encapsulation:
  + Wrapping of data and functions into single unit that unit is known as class and this process is known as Encapsulation.
* Inheritance:
  + It is process in which object of one class can acquire properties of another class
  + Types:
    - Single Level
    - Multilevel
    - Hierarchical inheritance
    - Multiple inheritance
    - Hybrid inheritance
* Polymorphism:
  + The Word polymorphism means having many forms.
  + One Interface many forms.
  + Types:
    - Compile time polymorphism. (Static)
      * Function overloading
      * Operator overloading
    - Run time polymorphism. (Dynamic)
      * Function overriding.
* Message Passing:
  + When two or more objects communicate with each other that means that those objects are sending and receiving messages.
* Dynamic:
  + In dynamic binding, the code to be expected in response to function call is decided at runtime.
* Constructor:
  + It is Special Member Function used to initialise member variable.
  + Class name and constructor name is same.
  + Constructor don’t have return type.
  + It can be overloaded.
  + It Invokes as soon as object is created.
  + Types:
    - Default constructor:
      * It does not have arguments.
      * When we don’t write constructor compiler by default provides this constructor.
      * Syntax:
        + classname(){ }
* Parameterised Constructor:
  + It has minimum 1 parameter.
  + Syntax:
* classname (datatype variable1, datatype variable2) { }
* Copy Constructor:
  + It is called whenever object is initialized using another object, object is passed as argument.
  + Syntax:
    - classname (classname &ref)

{

var = ref.var;

var2 = ref.var2;

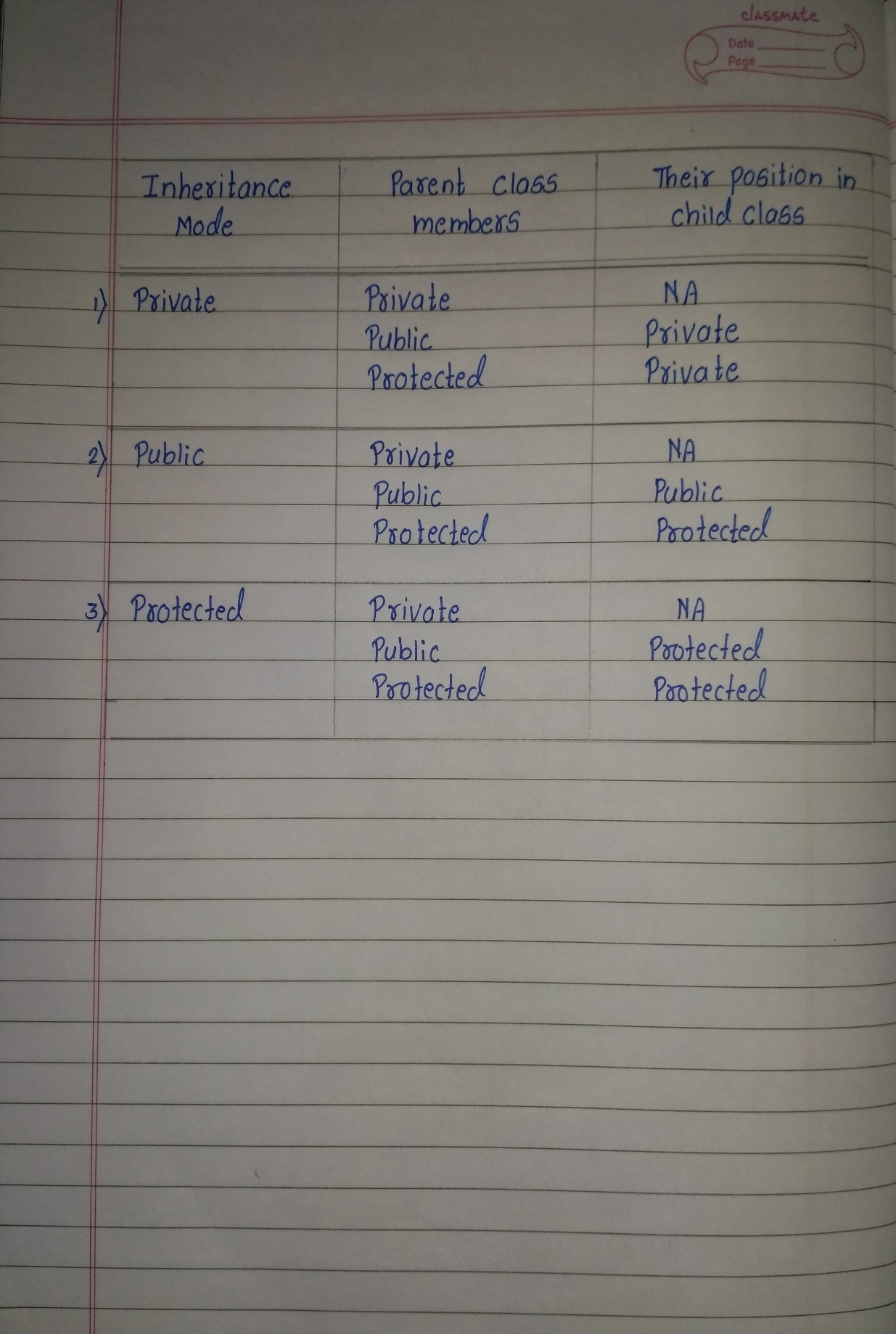
}

* Destructor:
  + It is Special member function, used to destroy member variable.
  + Same name as class name.
  + Cannot be overloaded.
  + It doesn’t have argument and return type.
  + Syntax:

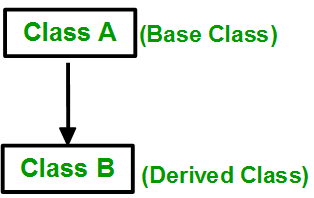
~classname()

***Inheritance***

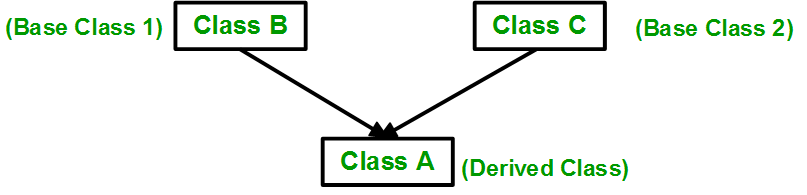
* Inheritance is one of the most important features of Object-Oriented Programming.
* The below table summarizes the above three modes and shows the access specifier of the members of base class in the sub class when derived in public, protected and private modes:



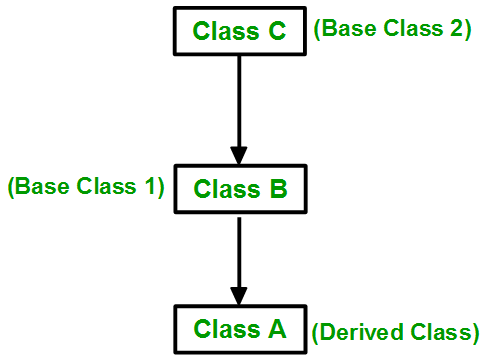
* **Single Inheritance**: In single inheritance, a class is allowed to inherit from only one class. i.e. one sub class is inherited by one base class only.



* **Multiple Inheritance:** Multiple Inheritance is a feature of C++ where a class can inherit from more than one classes. i.e one **sub class** is inherited from more than one **base classes**.



* **Multilevel Inheritance**: In this type of inheritance, a derived class is created from another derived class.



* **Hierarchical Inheritance**: In this type of inheritance, more than one sub class is inherited from a single base class. i.e. more than one derived class is created from a single base class.



* **Hybrid (Virtual) Inheritance**: Hybrid Inheritance is implemented by combining more than one type of inheritance. For example: Combining Hierarchical inheritance and Multiple Inheritance.

