Jaydeep Shah
radhey04ec@gmail.com
<Self learning notes>

TUTORIAL 7: CONSTRUCTOR & Access Specifier

Function auto call when Object created known as Constructor (Method name = Class Name)

A constructor in C++ is a special method that is automatically called when an object of a class is created.

To create a constructor**, use the same name as the class, followed by parentheses ():**

Note: The constructor has the same name as the class, it is always public, and it does not have any return value.

No return type,
Always Public,
Basic use initialize process

Note: The constructor has the same name as the class, it is always public, and it does not have any return value.

Example:

Note: Not possible to call constructor method using dot(.) operator with object

```
#include<iostream>

using namespace std;

   class myClass{

      public:

      myClass()

      {

      cout<<"constructor call";

      }

   };

int main()

{

   myClass myObj;

   return 0;

}
```

OUTPUT: constructor call

Constructor mostly used to set attribute with some default value when Object will create.

Lets take on example

Class >> **CAR** , Attribute>>**Company name, Year**

EXAMPLE:

```
#include<iostream>
using namespace std;
//Lets create one class (Class of CAR)
class CAR
{
   public:
   //Attribute
   string brand;
   int year;
   string type;

   //Constructor (same as car NAME)  (With Default Argument)
   CAR(string a = "No Data ",int b=0,string c=" No Data")
   {
      //Set attribute value when create object of this class
      brand = a;
      year = b;
      type = c;
   }
};

//Create Object
CAR mycar1("TATA",2020,"PUNCH");
int main()
{
```

```
    CAR mycar2;
    cout<<"\n Your Car name = "<<mycar1.brand<<" Year = "<<mycar1.year<< " Model = "<<mycar1.type;
    cout<<"\n Your Car name = "<<mycar2.brand<<" Year = "<<mycar2.year<< " Model = "<<mycar2.type;

    return 0;
}
```

OUTPUT:

Your Car name = TATA Year = 2020 Model = PUNCH

Your Car name = No Data Year = 0 Model = No Data

ACCESS SPECIFIER:

Now we are familiar with "**Public"** keyword, this is nothing but access specifier.

OOP supports mostly three type of access specifier.

public - members are accessible from outside the class

private - members cannot be accessed (**or viewed**) from outside the class

protected - members cannot be accessed from outside the class, however, they can be accessed in inherited classes

Access specifiers define how the members (attributes and methods) of a class can be accessed.

Example:

```
class MyClass {
  public:    // Public access specifier
    int x;   // Public attribute
  private:   // Private access specifier
    int y;   // Private attribute
```

```
};

int main() {
  MyClass myObj;
  myObj.x = 25;  // Allowed (public)
  myObj.y = 50;  // Not allowed (private)
  return 0;
}
```

If you try to run you will get **Error "error: y is private" ,try to use attribute y directly from outside of class and y declared as private member.**

NOTE: Note: By default, all members of a class are private if you don't specify an access specifier:

Example

```
class MyClass {
  int x;   // Private attribute
  int y;   // Private attribute
};
```

Note: It is possible to access private members of a class using a **public method inside the same class**. Encapsulation concept is here.

IMPORTANT NOTES

Tip: It is considered good practice to declare your class attributes as private (as often as you can). This will reduce the possibility of yourself (or others) to mess up the code. This is also the main ingredient of the Encapsulation concept.