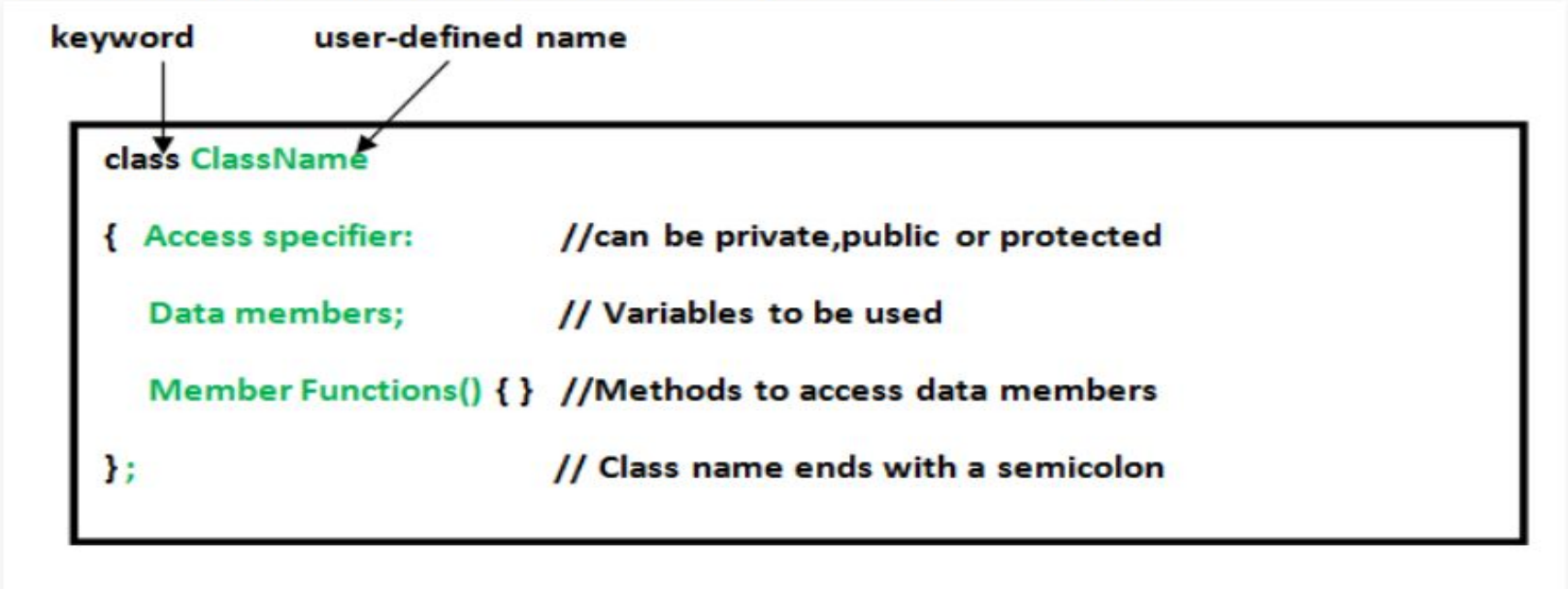
ENCAPSULATION

ENCAPSULATION

- Encapsulation is an Object Oriented Programming concept that binds together the data and functions that manipulate the data, and that keeps both safe from outside interference and misuse. Data encapsulation led to the important OOP concept of **data hiding**.
- **Data encapsulation** is a mechanism of bundling the data, and the functions that use them and **data abstraction** is a mechanism of exposing only the interfaces and hiding the implementation details from the user.
- C++ supports the properties of encapsulation and data hiding through the creation of user-defined types, called **classes**.

Defining Class and Declaring Objects

A class is defined in C++ using keyword class followed by the name of class. The body of class is defined inside the curly brackets and terminated by a semicolon at the end.



Declaring Objects:

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.

Syntax:

ClassName ObjectName;

Accessing data members and member functions: The data members and member functions of class can be accessed using the dot('.') operator with the object.

For example if the name of object is *obj* and you want to access the member function with the name *printName()* then you will have to write :

obj.printName() .

Access Modifiers in C++

Access modifiers are used to implement an important aspect of Object-Oriented Programming known as Data Hiding.

There are 3 types of access modifiers available in C++:

1. Public[everyone can access outside the class]
2. Private[no one can access outside the class]
3. Protected[only child can access outside the class]

Note: If we do not specify any access modifiers for the members inside the class then by default the access modifier for the members will be Private.

Public example:

```
#include<iostream>
using namespace std;


// class definition
class Circle
{
    public:
        double radius;

        double  compute_area()
        {
            return 3.14*radius*radius;
        }

};


// main function
int main()
{
    Circle obj;

    // accessing public datamember outside class
    obj.radius = 5.5;

    cout << "Radius is: " << obj.radius << "\n";
    cout << "Area is: " << obj.compute_area();
    return 0;
}
```

Output:

```
Radius is: 5.5 Area is: 94.985
```

Private example:

```
#include<iostream>
using namespace std;


class Circle
{
    // private data member
    private:
        double radius;

    // public member function
    public:
        double  compute_area()
        {   // member function can access private
            // data member radius
            return 3.14*radius*radius;
        }

};


// main function
int main()
{
    // creating object of the class
    Circle obj;

    // trying to access private data member
    // directly outside the class
    obj.radius = 1.5;

    cout << "Area is:" << obj.compute_area();
    return 0;
}
```

Output:

```
In function 'int main()': 11:16: error: 'double Circle::radius'
is private double radius; ^ 31:9: error: within this context
obj.radius = 1.5; ^
```

```cpp
#include<iostream>
using namespace std;


class Circle
{
    // private data member
    private:
        double radius;

    // public member function
    public:
        void compute_area(double r)
        {   // member function can access private
            // data member radius
            radius = r;

            double area = 3.14*radius*radius;

            cout << "Radius is: " << radius << endl;
            cout << "Area is: " << area;
        }

};


// main function
int main()
{
    // creating object of the class
    Circle obj;

    // trying to access private data member
    // directly outside the class
    obj.compute_area(1.5);


    return 0;
}
```

Output:

```
Radius is: 1.5 Area is: 7.065
```

Member Function inside class and Outside class

A member function of a class is a function that has its definition or its prototype within the class definition like any other variable. It operates on any object of the class of which it is a member, and has access to all the members of a class for that object.

Example: Member function inside the class

```
class Box {
   public:
      double length;      // Length of a box
      double breadth;     // Breadth of a box
      double height;      // Height of a box

      double getVolume(void) {
         return length * breadth * height;
      }
};
```

Member function outside the class

If you like, you can define the same function outside the class using the scope resolution operator (::) as follows −

```
class Box {
   public:
      double length;      // Length of a box
      double breadth;     // Breadth of a box
      double height;      // Height of a box

      double getVolume(void)
};
double Box::getVolume()
{
return length * breadth * height;
}
```

THANK YOU