Features of OOP

(Access Modifiers)

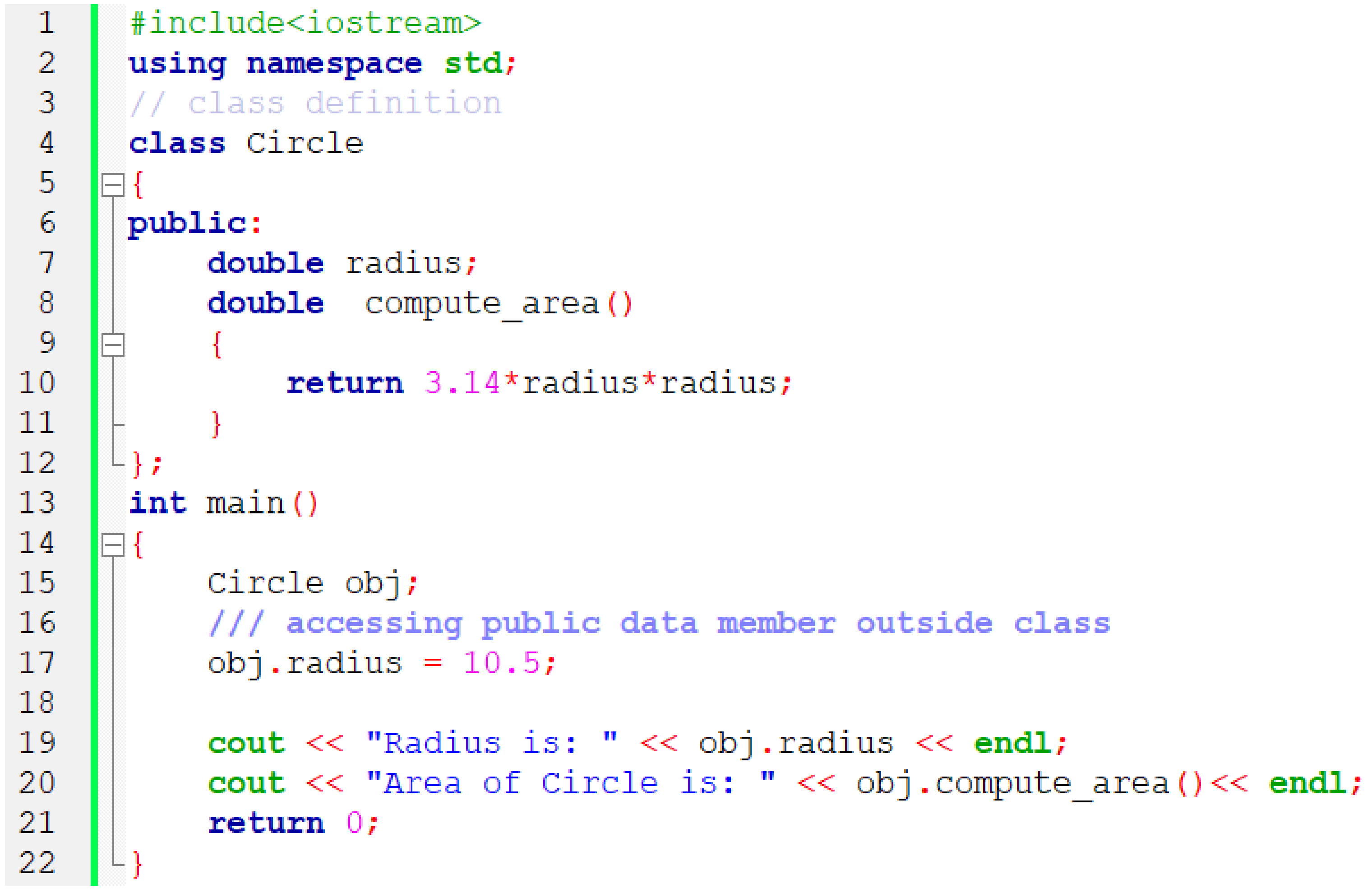
Access Modifiers in C++

- Access Modifiers or Access Specifiers in a class are used to ***set the accessibility*** of the class members.
- It sets some restrictions on the class members not to get directly accessed by the outside functions.
- There are 3 types of access modifiers available in C++:
  - ✓**Public**
  - ✓**Private**
  - ✓**Protected**
- 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**.

Access Modifier :: Public

- All the class members declared under public will be avail.
- The data members and member functions declared public can be accessed able to everyone by other classes too.
- The public members of a class can be accessed from anywhere in the program using the direct member access operator (.) with the object of that class.
- In the below program the data member *radius* is public so we are allowed to access it outside the class.

Access Modifier :: Public



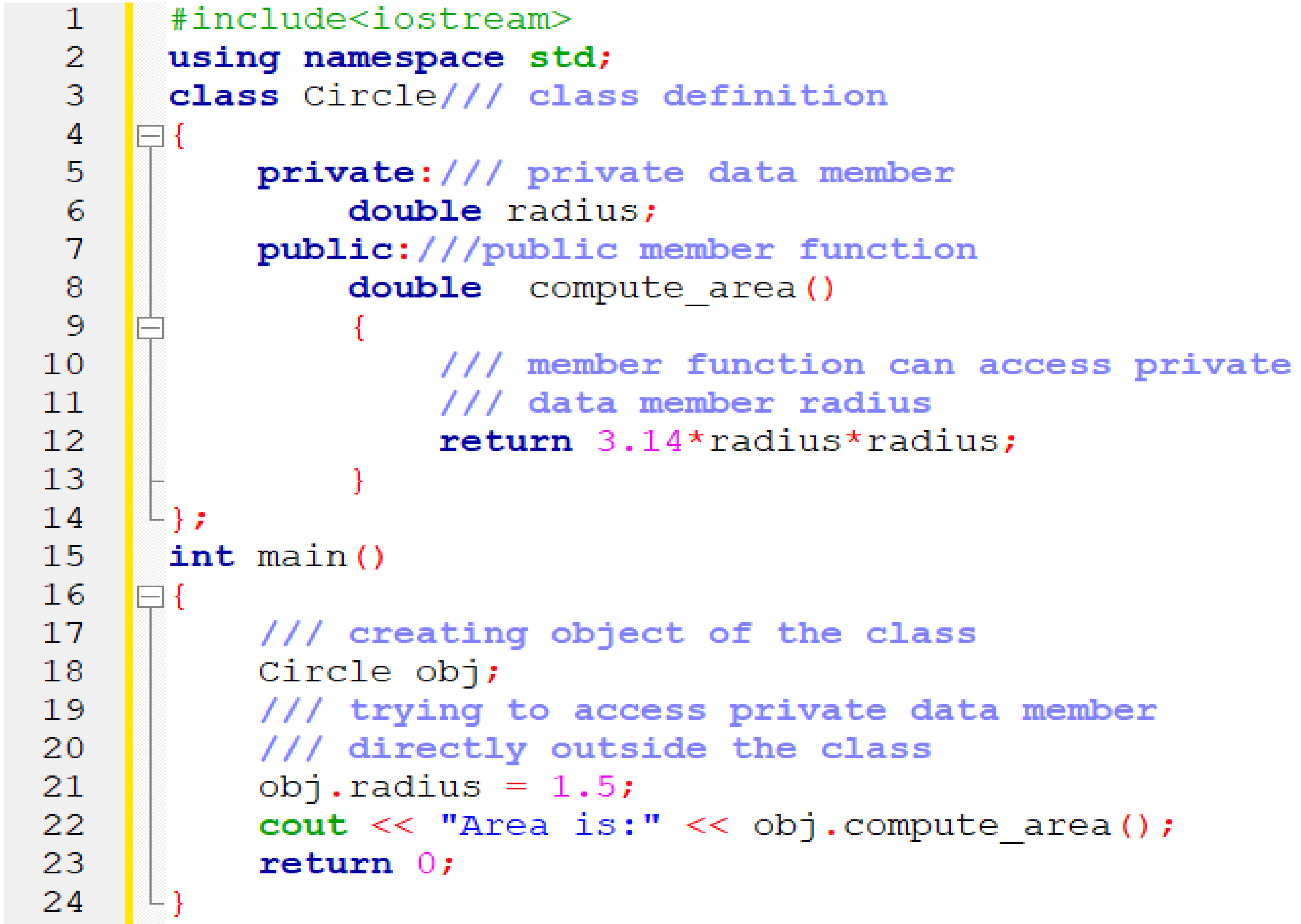
```cpp
#include<iostream>
using namespace std;
// class definition
class Circle
{
public:
    double radius;
    double  compute_area()
    {
        return 3.14*radius*radius;
    }
};
int main()
{
    Circle obj;
    /// accessing public data member outside class
    obj.radius = 10.5;

    cout << "Radius is: " << obj.radius << endl;
    cout << "Area of Circle is: " << obj.compute_area()<< endl;
    return 0;
}
```

Access Modifier :: Private

- The class members declared as ***private*** can be accessed only by the functions inside the class.
- They are not allowed to be accessed directly by any object or function outside the class.
- Only the member functions or the friend functions are allowed to access the private data members of a class.

Access Modifier :: Private

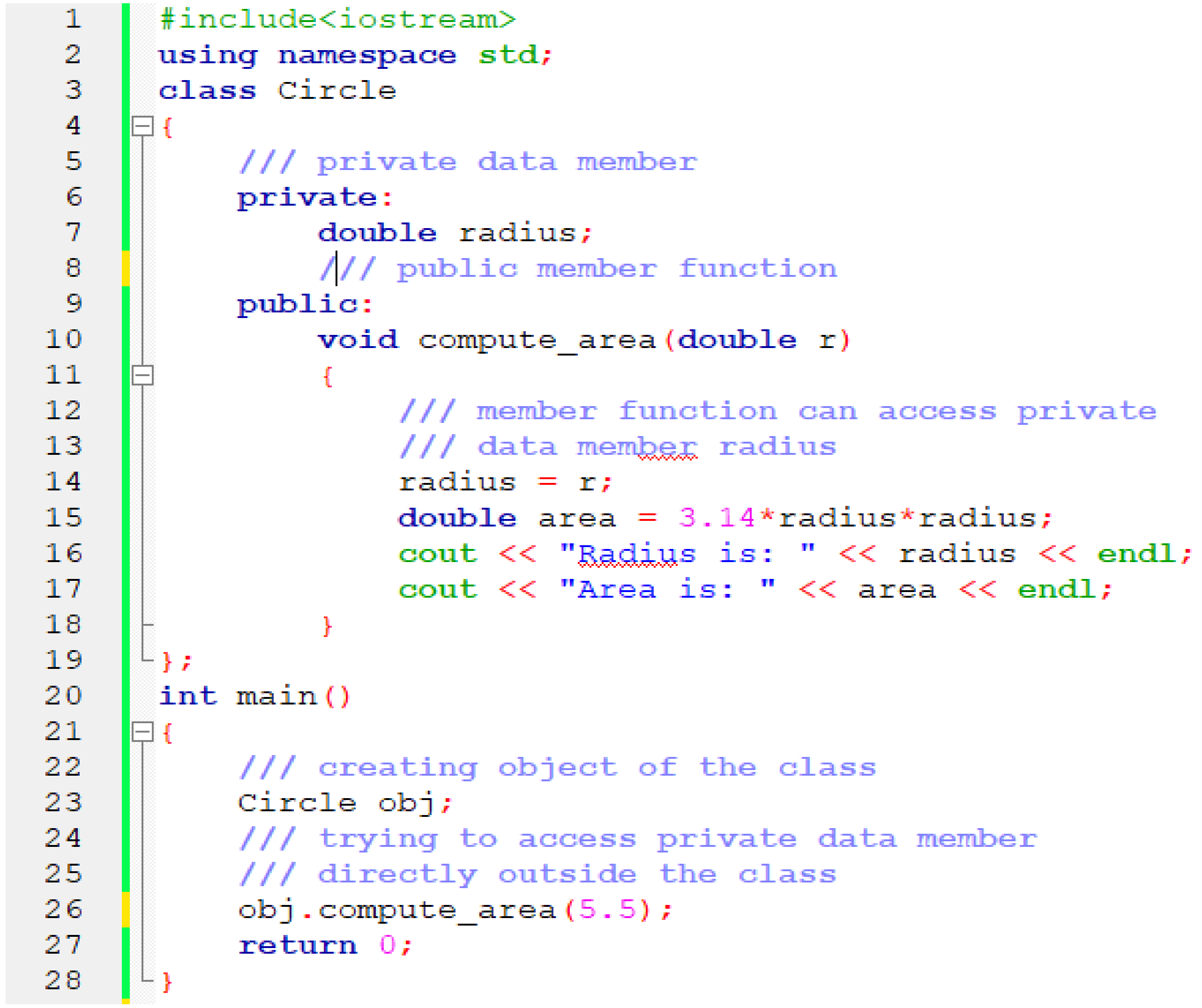


```cpp
#include<iostream>
using namespace std;
class Circle/// class definition
{
    private:/// private data member
        double radius;
    public:///public member function
        double  compute_area()
        {
            /// member function can access private
            /// data member radius
            return 3.14*radius*radius;
        }
};
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;
}
```

Access Modifier :: Private

- The output of the below program will be a compile time error because we are not allowed to access the private data members of a class directly outside the class.
- However, we can access the private data members of a class indirectly using the public member functions of the class.

Access Modifier :: Private

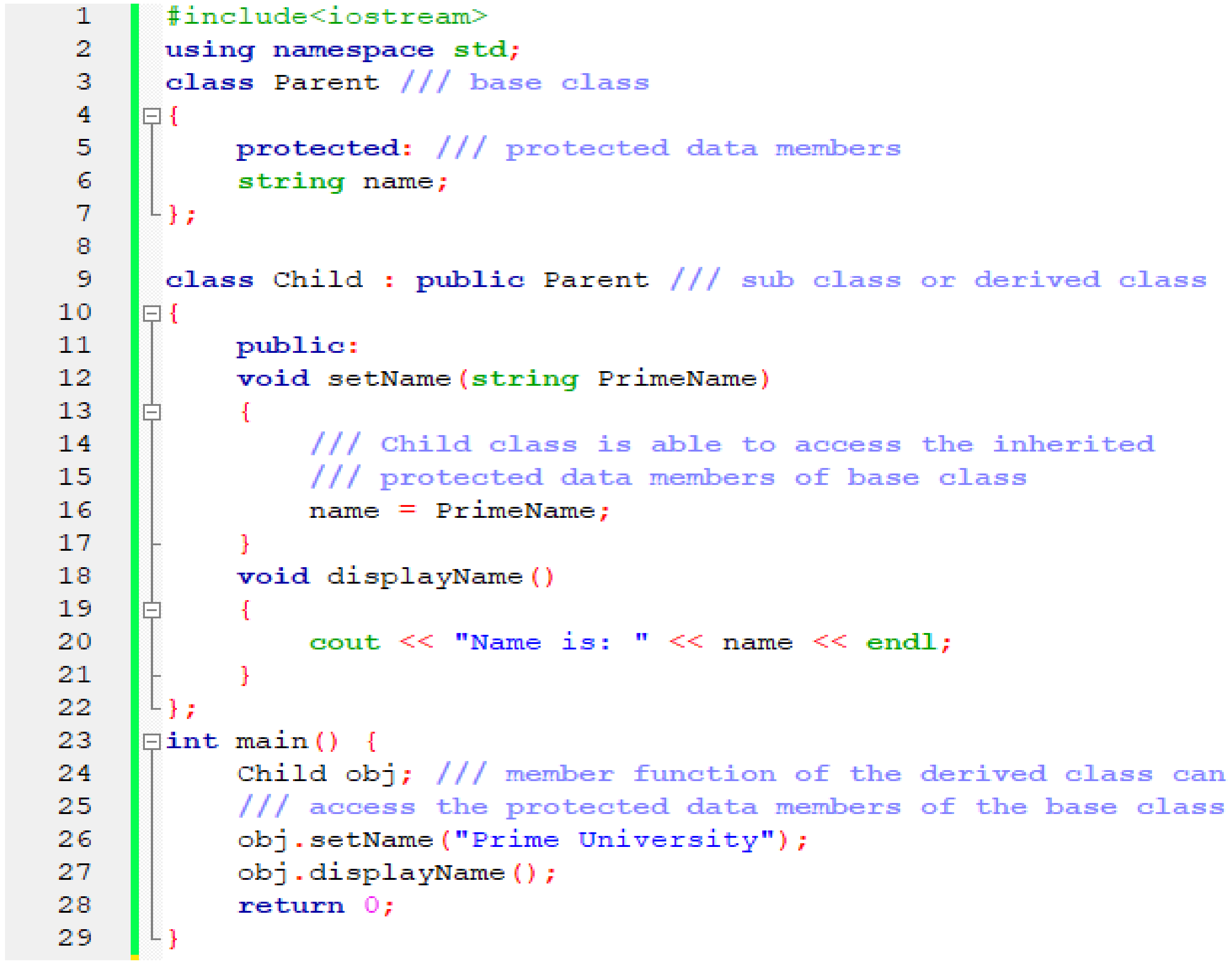


```
#include<iostream>
using namespace std;
class Circle
{
    /// private data member
    private:
        double radius;
        /// public member function
    public:
        void compute_area(double r)
        {
            /// member function can access private
            /// data member radius
            radius = r;
            double area = 3.14*radius*radius;
            cout << "Radius is: " << radius << endl;
            cout << "Area is: " << area << endl;
        }
};
int main()
{
    /// creating object of the class
    Circle obj;
    /// trying to access private data member
    /// directly outside the class
    obj.compute_area(5.5);
    return 0;
}
```

Access Modifier :: Protected

- Protected access modifier is similar to that of private access modifiers.
- The difference is that the class member declared as Protected are inaccessible outside the class but they can be accessed by any subclass(derived class) of that class.

Access Modifier :: Protected



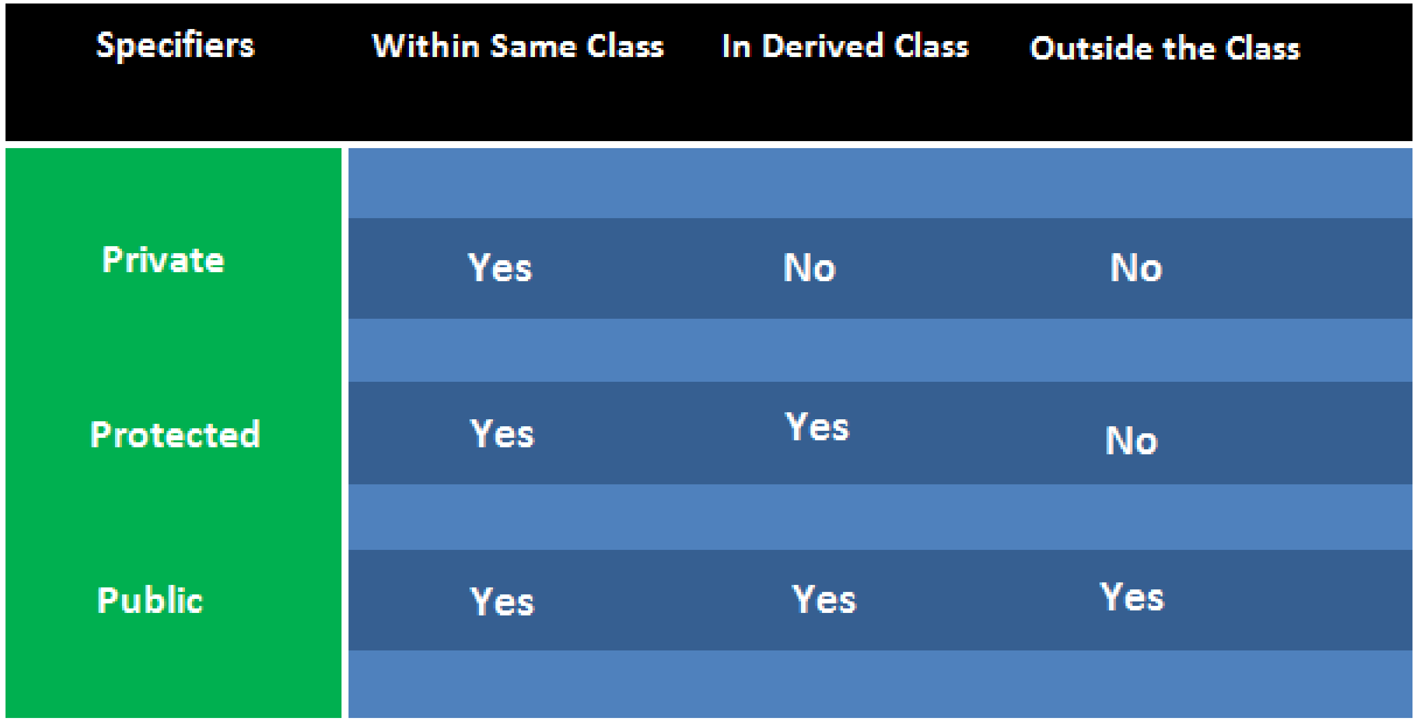
```cpp
#include<iostream>
using namespace std;
class Parent /// base class
{
    protected: /// protected data members
    string name;
};

class Child : public Parent /// sub class or derived class
{
    public:
    void setName(string PrimeName)
    {
        /// Child class is able to access the inherited
        /// protected data members of base class
        name = PrimeName;
    }
    void displayName()
    {
        cout << "Name is: " << name << endl;
    }
};
int main()  {
    Child obj; /// member function of the derived class can
    /// access the protected data members of the base class
    obj.setName("Prime University");
    obj.displayName();
    return 0;
}
```

Access Modifier

- **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 Modifier in a View



Specifiers	Within Same Class	In Derived Class	Outside the Class
Private	Yes	No	No
Protected	Yes	Yes	No
Public	Yes	Yes	Yes

Thank You