Access Modifiers

In this lesson, you will learn about the private, public and protected members.

WE'LL COVER THE FOLLOWING

In C++, we can impose access restrictions on different data members and member functions. The restrictions are specified through **access modifiers**. Access modifiers are tags we can associate with each member to define which parts of the program can access it directly.

There are three types of access modifiers. Let's take a look at them one by one.

Private

A private member cannot be accessed directly from outside the class. The aim is to keep it hidden from the users and other classes. It is a popular practice to **keep the data members private** since we do not want anyone manipulating our data directly. By default, all declared members are private in C++. However, we can also make members private using the `private:` heading.

```
class Class1 {
  int num; // This is, by default, a private data member
  ...
};

class Class2 {
  private: // We have explicitly defined that the variable is private
  int num;
  ...
};
```

Public

Public

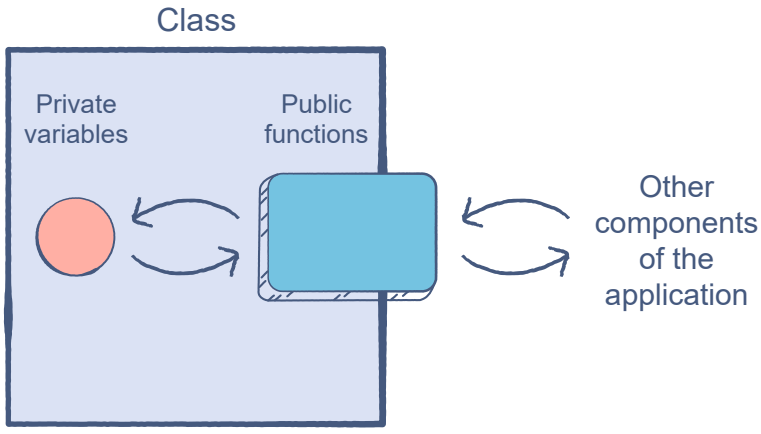
This tag indicates that the members can be directly accessed by anything which is in the same scope as the class object.

Member functions are usually public as they provide the interface through which the application can communicate with our private members.

Public members can be declared using the `public:` heading.

```
class myClass {
  int num; // Private variable

  public: // Attributes in this list are public
  void setNum(){
    // The private variable is directly accessible over here!
  }
};
```



Public members of a class can be accessed by a class object using the `.` operator. For example, if we have an object `c` of type `myClass`, we could access `setNum()` like this:

```
myClass c; // Object created
c.setNum(); // Can manipulate the value of num
c.num = 20; // This would cause an error since num is private
```

Protected

The protected category is unique. The access level to the protected members lies somewhere between private and public. The primary use of the protected tag is to implement **inheritance**, which is the process of creating classes out of classes. Since this is a whole other topic for the future, we'll refrain from

going into details right now.

We've seen a hint of how data members can be created in a class. In the next lesson, we will go into further details on the topic.