Access Rights for Members

In this lesson, we'll discuss how access rights are defined for class members so that we can access them from outside.

WE'LL COVER THE FOLLOWING

Access rights

The access rights of members determine how the members are accessible from outside of the class. Access rights give the author of the class the ability to decide which class members are accessible to the users of the class, i.e., the interface and which members are for internal use of the class (the implementation.) C++ has three different access rights:

- `public`: No restriction.
- `protected`: Access from inside the class and from all derived classes.
- `private`: Access only from within the class.

The name of every class member has an associated **member access**. When the name of the member is used anywhere in a program, its access is checked, and if it does not satisfy the access rules, then the program does not compile.

Let's have a look at the rules of access rights for class members.

Rules

1. All members of a class are private by default.
2. All members of a `struct` or `union` are public by default.
3. Access rights are determined by the last used access specifier.
4. The access specifier can be used multiple times.

4. The access specifier can be used multiple times.

> The `public` and `protected` members are the interfaces of the class, the `private` members are the implementations of the class.

Embedded declarations

All entities of a class are visible inside the class. The scope operator(`::`) must be used from outside the class to access the elements of the class.

```
struct Account{
   enum Status{ Ok, Error };
   void setStatus(Status s);
...
};

Account * acc = new Account;
acc->setStatus(Account::Status::Ok);
```

Depending on the access specifier, we can use the entities of the class.

In the next lesson, we'll look at an example of access specifiers.