

Conditional operator:

Select between two options based upon a condition.
Unlike if, it can do it inline.

Ex:

```
int x = (1==0) ? 1 : 0;
```

“else” part

“then” part

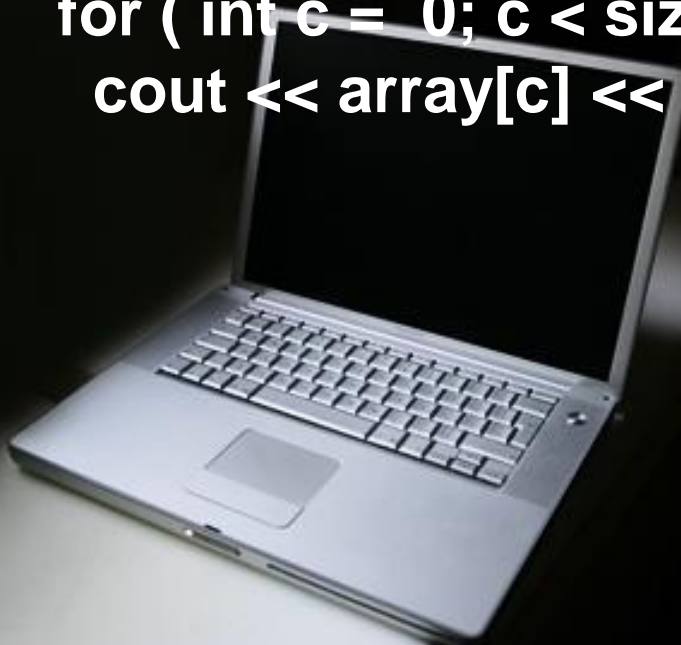
conditional
“if” part



Conditional operator:

```
for ( int c = 0; c < size; c++)  
{  
    cout << array[c] << " ";  
}  
cout << endl;
```

```
for ( int c = 0; c < size; c++)  
    cout << array[c] << ( (c == size - 1) ? "\n": " " );
```



Conditional operator:

Pass in a select value to function

```
bool check = true;
```

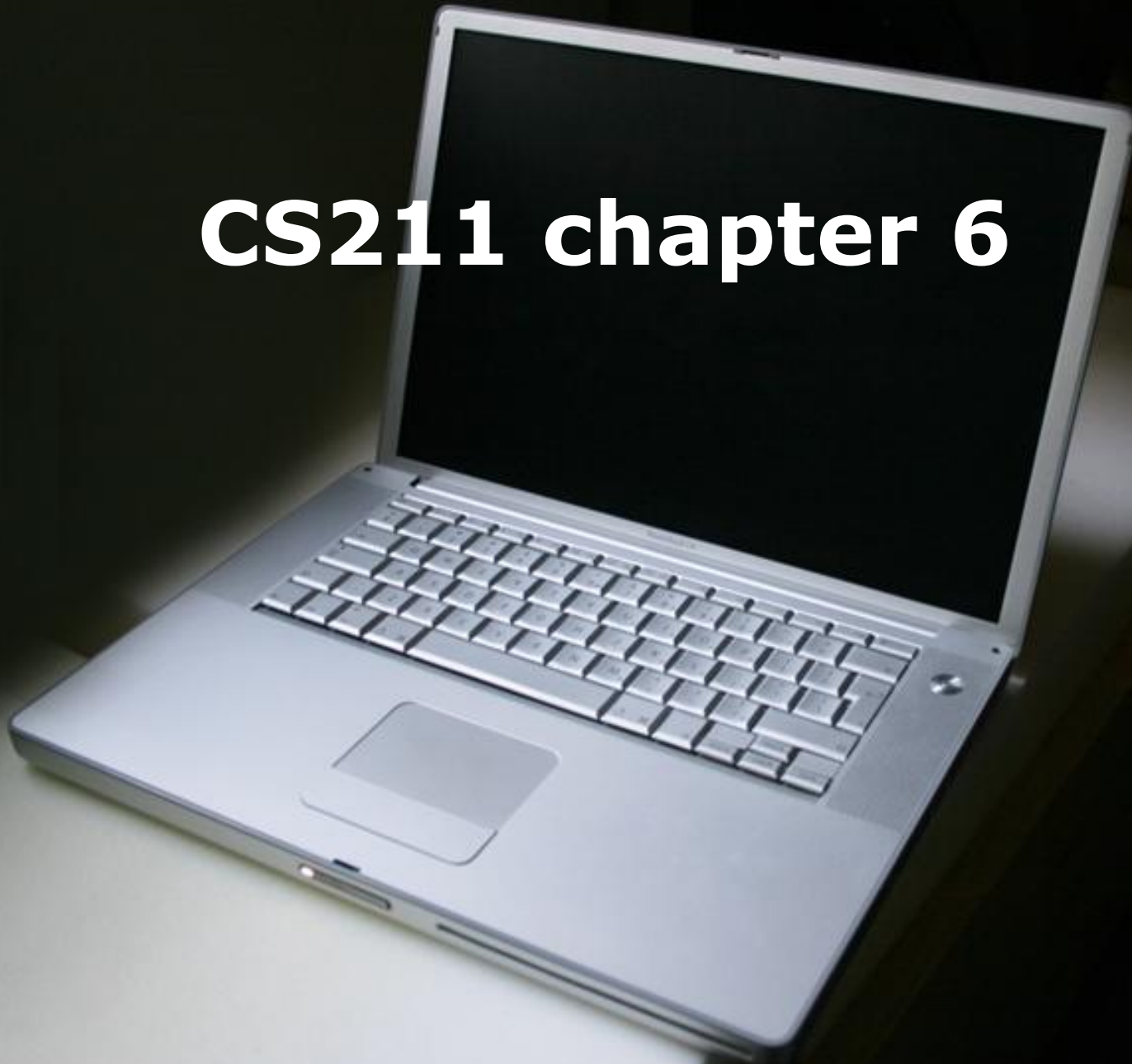
```
func(check?!check:check);
```

```
bool check = 0;
```

```
func(check?check:check+1);
```



CS211 chapter 6

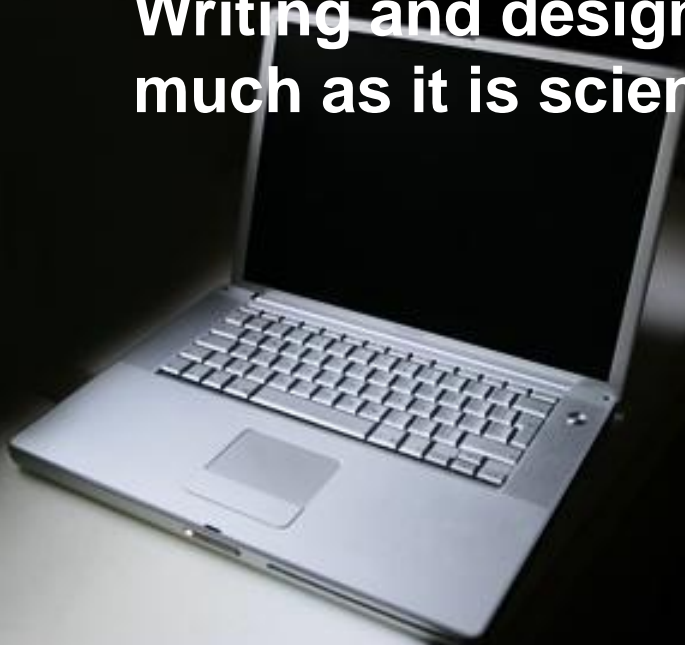


Object Oriented Introduction:

In real world, there are millions of type of things.

With the object oriented design, also known as OO, we can define our own types, to better model the real world problems we want to solve.

Writing and designing software is really an art as much as it is science.



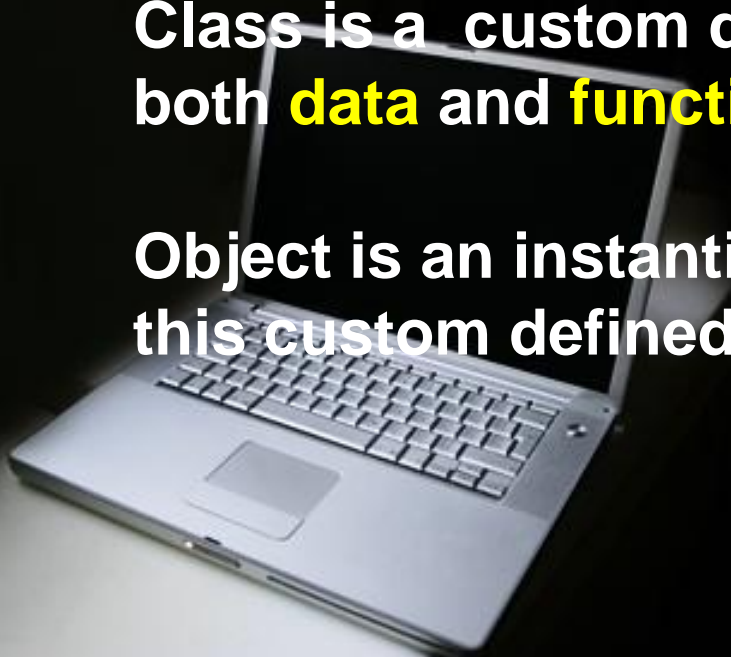
Class:

To define an object, there are two parts:

- What components make up the object?
collection of data
- What actions does the object perform?
function

Class is a custom defined data type which holds both **data** and **functions**.

Object is an instantiation of a class. The variable of this custom defined type.

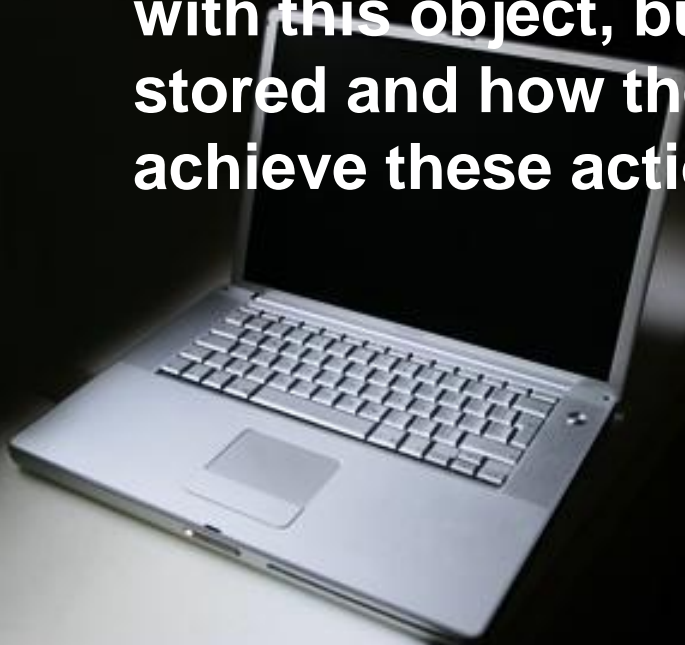


Class:

Example: String

- Create a string object
- Manipulate (concatenate, insert, ...)
- Information about this object (length, substring)

We only know there are functions that associate with this object, but we do not care how the data is stored and how the function is implemented to achieve these actions.



Class vs. Struct:

In user defined type - structure, we used . (dot) to get to it's member variable. Thus, we have to be aware exactly what made up the structure, what are the member name.

One principle of OOP is encapsulation. All the internal information is hiding from the rest of the world.



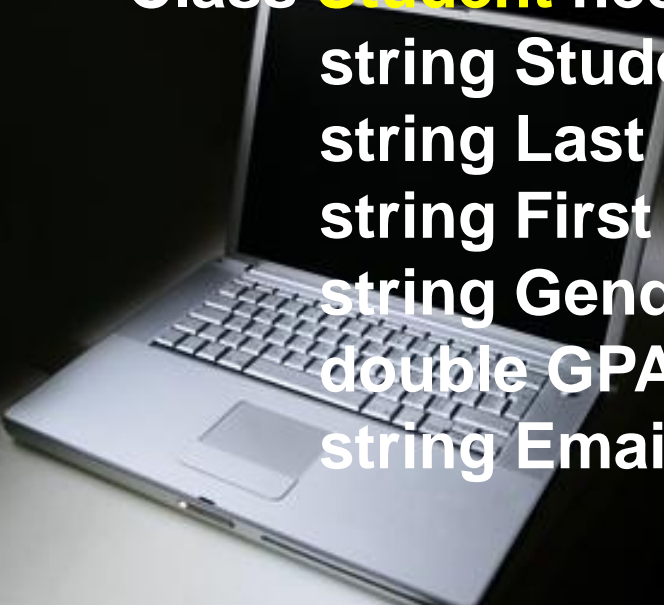
Example:

Let's say we want to define a real world type – **Student**

What component makes up the student?

Student ID, Last Name, First Name, Gender, GPA,
Email address

Class **Student** need variables to hold these data.



```
string Student ID,  
string Last Name,  
string First Name,  
string Gender,  
double GPA,  
string Email address
```

Example:

After defining variables, it also need to act on the variables.

Ex:

Student ID

– to setup the student ID, ask for the student ID

One function perform one action.

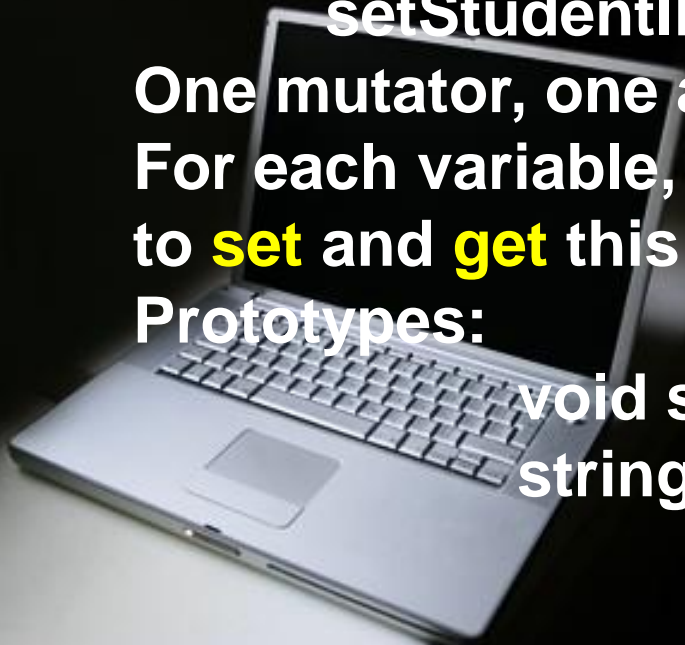
setStudentID, getStudentID

One mutator, one accessor

For each variable, at least need two member functions to **set** and **get** this variable.

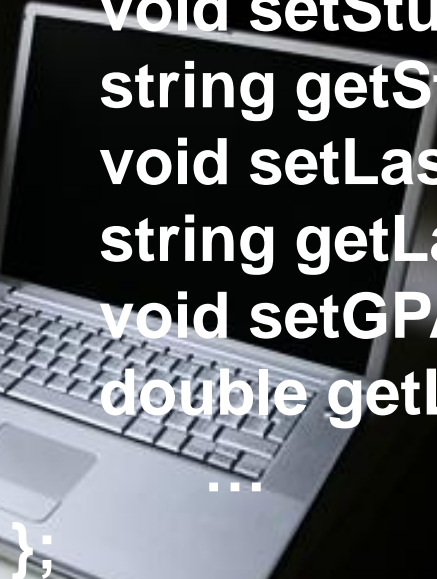
Prototypes:

```
void setStudentID(string id);  
string getStudentID();
```



Sample Class:

```
class Student
{
    private:
        string student_ID, last_name, first_name;
        double gpa;
        Date dob;
    public:
        void setStudentID(string id);
        string getStudentID();
        void setLastName(string lname);
        string getLastName();
        void setGPA(double gpa);
        double getGPA();
        ...
};
```



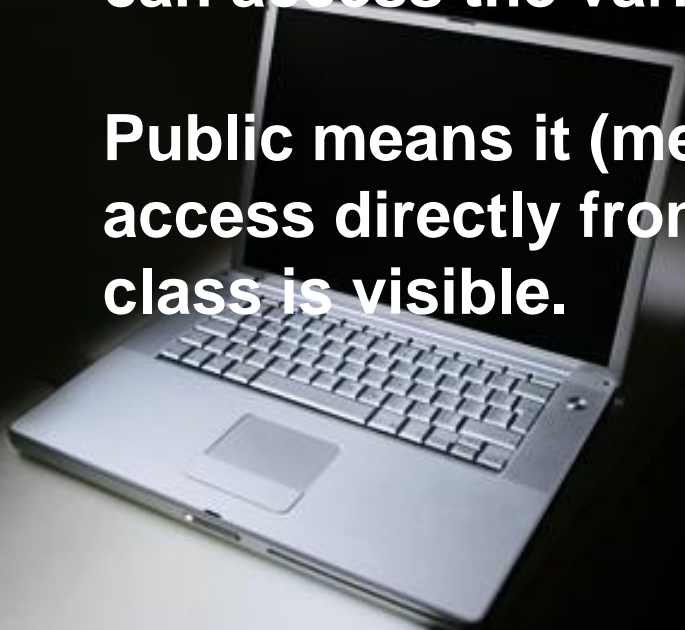
Accessibility:

For each variables and functions, we can define its accessibility.

Access modifier decides who can access the variables or the functions from outside of the class.

Private means only members of the class (or friends) can access the variables and modify it.

Public means it (member function or variable) can be access directly from outside of the class where this class is visible.



Accessibility:

Note: for **Student** Class as example.

Let's say we created Student s1 in a main function outside of class Student. We can not modify it's gpa variable directly by `s1.gpa = 2.0` from main function, because this variable is **private**.

I can call it's member function `s1.setGPA(2.0)` to set the gpa value, because this member function is **public**.

