

# 19CSE201 :Advanced Programming

## Lectures 5

### Classes, Methods and Objects

By  
Ritwik M  
Assistant Professor(SrGr)  
Dept. Of Computer Science & Engg  
Amrita Vishwa Vidyapeetham -  
Coimbatore

# A Quick Recap

- Selection Operations
- Iteration
- Branching
- Functions and Data
- Global Variable
- Local Variable
- Scope
- Formal and Actual Parameters

# Recollect

- We already know how to use the data types that are built into the C++ language.
- We know how to declare variables of these types
  - Eg: `int n;` where `n` is a variable of type `int`.

# Objects & Classes

- A class is a derived complex data type containing members (data items) and functions that operate on these data (that may be of different types.)
  - A class specifies the traits (data) and behavior (operations) that an object can exhibit.
- Attributes – member data of a class.
  - An attribute is the data defined in a class that maintains the current state of an object. The state of an object is determined by the current contents of all the attributes.
- Methods – member functions of a class

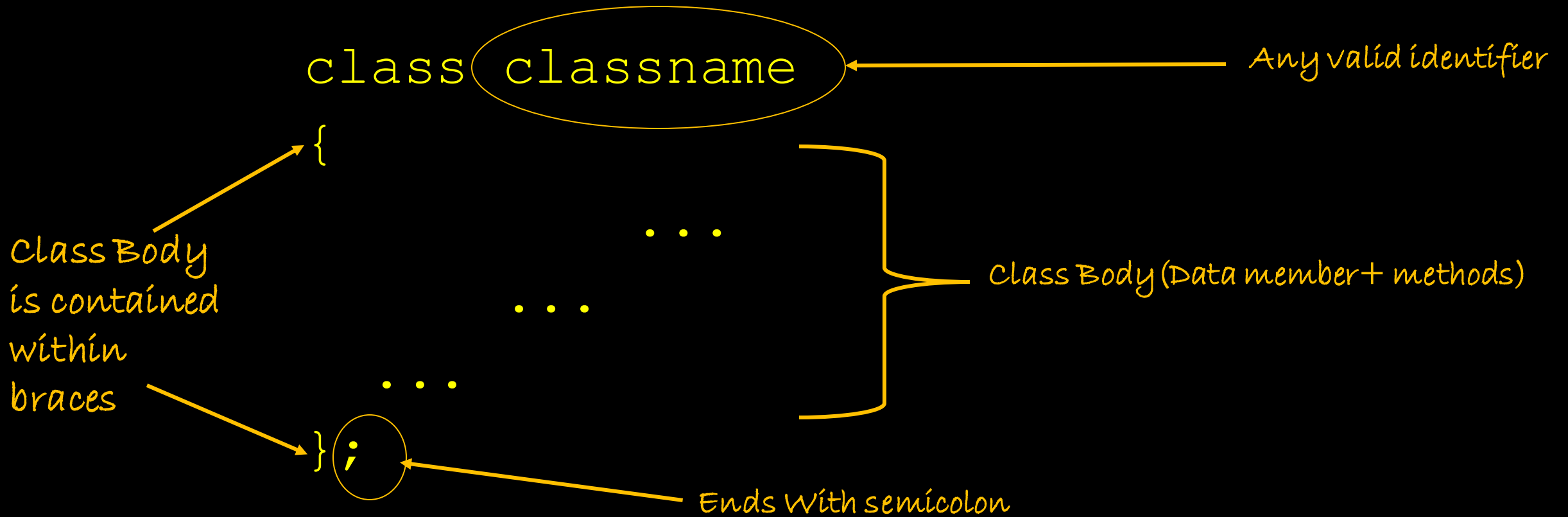
# Objects & Classes

- A class itself does not exist; it is merely a description of an object.
  - A class can be considered a template for the creation of object, e.g.,
  - Blueprint of a building --> class
  - Building --> object
- An object exists and is definable.
  - An object exhibits behavior, maintains state, and has traits.
  - An object can be manipulated.
  - An object is an instance of a class.

# Methods and Messages

- A method is a function that is part of the class definition.
  - The methods of a class specify how its objects will respond to any particular message.
- A message is a request, sent to an object, that activates a method (i.e., a member function).

# A Class Definition begins with the keyword class



# The Class Body

- Within the body, the keywords `private:` and `public:` specify the access level of the members of the class.
  - the default is `private`
- Usually, the data members of a class are declared in the `private:` section of the class and the member functions are in the `public:` section.

```
class classname  
{
```

```
    private:
```

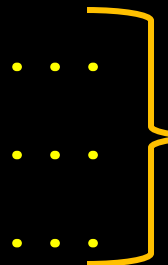
```
    ...  
    ...  
    ...
```



private members  
or methods

```
    public:
```

```
    ...  
    ...  
    ...
```



public members  
or methods

```
};
```



# Access Specifiers

- Data members or member functions may be public, private or protected
  - Public
    - Members can be accessed outside the class directly
    - Acts as an *interface*
  - Private
    - Accessible only to member functions of the class
    - Private members and methods are only for internal use within the class and cannot be accessed outside the class
  - Protected
    - Data members and member functions can be used in the same class and its derived class (for one level)
    - Cannot be used in main function

# Implementing Class Methods

- Inside the class -- straightforward
- Outside the class
  - Using binary scope resolution operator (::)
    - Connects member name to class name
    - Uniquely identifies functions of a particular class
  - Different classes can have member functions with same name
  - Format:

```
ReturnType    ClassName :: MemberFunctionName() {  
    ...  
}
```

# Example - defining inside class

```
#include<iostream>
using namespace std;
class student{
    int rollNo;
    char name[20];
public:
    void getData() {
        cin>>rollNo;
        cin>>name;
    }
    void putData() {
        cout<<rollNo<<" "<<name;
    };
};
```

# Example - defining outside class

```
#include<iostream>
using namespace std;
class student{
    public:
        int rollNo;
        char name[20];
        void getData();
        void putData();
};
```

```
void student :: getData() {
    cin>>rollNo;
    cin>>name;
}
```

Scope Resolution  
Operator

Return Type

Class Name

Member Function  
Name

```
void student :: putData() {
    cout<<rollNo<<" "<<name;
}
```

# Member Functions

- Different classes can have the same function name
- Using a "membership label" will resolve the scope
- Only Member functions and Friend functions can access private data of the class
- A member function can call another member function without using the dot operator

# Accessing Class Member Functions

- Operators – similar to using structures in C
  - Dot member selection operator (.)
    - Object
    - Reference to object
  - Arrow member selection operator (->)
    - Pointers
    - Discussed at that "point" :)

```
void putData() {  
    cout<<rollNo<<"    "<<name;  
}  
};
```

```
int main() {  
    student s;  
    s.getData();  
    s.putData();  
    return 0;  
}
```

Notice the "dot"  
operator  
AKA  
Member Selection  
Operator

Object Creation/  
instantiation

Accessing class  
members

# Example - Accessing Class Attributes

```
#include<iostream>
using namespace std;
class student{
    public:
        int rollNo;
        char name[20];
};
```

```
int main() {
    student s;
    cin>>s.rollNo;
    cin>> s.name;
    cout<<s.rollNo<<"    "<<s.name;
    return 0;
}
```

Any difference from  
accessing member  
functions?

# Terminology!!

- **Accessor**

- An accessor is a class method used to read data members

- **Mutator**

- a class method used to change data members.

- **Auxiliary Function**

- A function written for a specific purpose called when required and not in a specific library
- Eg. `max()`, `min()`



# Note on Static Data Members

- The data and functions of the class may be declared "static" in the class declaration
- The static members are initialized to zero when the first object of the class is created.
  - No other initialization is permitted
- Only a single copy of that member is created for the entire class and is shared by all the other members of the class
- It is visible only within the class, but its lifetime is the entire program.

# Note on Static Member Functions

- Declared like static data members
- Static member functions can have access to only other static members declared (functions or variables) declared in the same class
- Can be called using the class
  - Does not need an object
  - Example
    - `ClassName::functionName`

# Example: Static Members

```
class test{
    int code;
    static int count;
public:
    void setcode()
    {
        code = count++;
    }
    void showcode()
    {
        cout<<"object number: "<<code<<endl;
    }
    static void showcount(){
        cout<<"count: "<<count<<endl;
    }
};
```

Declaring Static Member

Accessing static member within the class

Accessing static member function

```
int test :: count=10;
int main(){
    test t1,t2;
    t1.setcode();
    t2.setcode();
    test :: showcount();
    test t3;
    t3.setcode();
    test :: showcount();
    t1.showcode();
    t2.showcode();
    t3.showcode();
    return 0;
}
```

Initializing static member

# Quick Summary

- Classes
- Methods
- Messages
- Access Specifiers
- Class Attributes
- Static data members
- Accessors, Mutators and Auxiliary Functions

Up Next

More on Objects, Methods and  
Classes