# Object Oriented Programming

**Shaobai Kan**

*chapter 9*

# Outline

# Outline

- Classes, Objects, Member Functions and Data Members

# Outline

- Classes, Objects, Member Functions and Data Members

- Defining a class with a Member Function

# Outline

- Classes, Objects, Member Functions and Data Members

- Defining a class with a Member Function

- Data members, *set* functions and *get* functions

# *Classes, Objects, Member Functions and Data Members*

# Object Oriented Programming

**Fact.** Two types of programming

- Procedural - Oriented Programming

- Object - Oriented Programming (OOP)
  — Create new types

# Object Oriented Programming

**Fact.** Two types of programming

- Procedural - Oriented Programming

- Object - Oriented Programming (OOP)
  — Create new types

**Fact.** A C++ programmer can create any type needed, and each of these new types can have all the functionality and power of the built-in types predefined by C++ such as int , long, and double.

# Classes

**Definition.** A class is a collection of variables (data members) - often of different types - combined with a set of related functions (member functions).

# Classes

**Definition.** A class is a collection of variables (data members) - often of different types - combined with a set of related functions (member functions).

Example: A car is

- a collection of *wheels*, *doors*, *seats*, *engine*, *and so forth* (variables).

- A car can *move*, *speed up*, *slow down*, *stop*, *park*, *and so forth* (functions).

# Example: Class

Example:

```
class Cat
{
  public:
        void Meow();
  private:
        unsigned int itsAge;
        unsigned int itsWeight
};
```

# Example: analysis

**Access-specifier.** public *v.s.*private

- public: indicate that the member is "available to the public" — i.e. it can be called by other functions in the program (such as main), and by member functions of other classes (if there are any).

- private: indicate that the variables or functions are accessible only to member functions of the class for which they are declared.

# Example: analysis

**Access-specifier.** public *v.s.* private

- public: indicate that the member is "available to the public" — i.e. it can be called by other functions in the program (such as main), and by member functions of other classes (if there are any).

- private: indicate that the variables or functions are accessible only to member functions of the class for which they are declared.
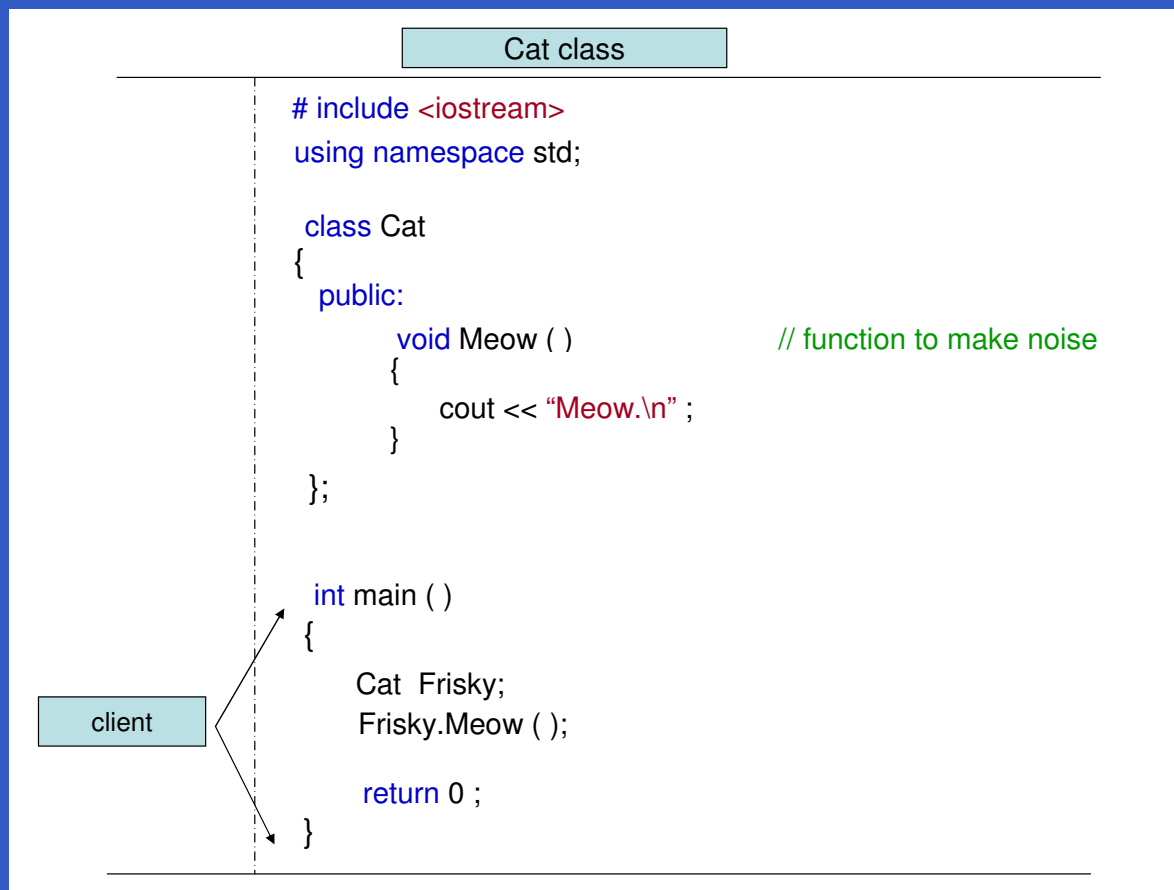
**Fact.** Generally, data member should be declared private and member functions should be declared public.

# *Defining a class with a Member Function*

# Example: class Cat

```
                          Cat class

          # include <iostream>
          using namespace std;

           class Cat
          {
            public:
                void Meow ( )                  // function to make noise
                {
                    cout << "Meow.\n" ;
                }
          };


           int main ( )
          {
              Cat  Frisky;
    client    Frisky.Meow ( );

               return 0 ;
          }
```

# Output: class Cat

```
Meow.
```

# Example: class GradeBook



GradeBook class

```cpp
# include <iostream>
using namespace std;

class GradeBook
{
   public:
        // function to display a welcome message to the user
        void DisplayMessage( )
        {
             cout << "Welcome to the Grade Book!" << endl;
        }
} ;

int main ( )
{
     GradeBook  myGradeBook;    //create a GradeBook object
     myGradeBook.DisplayMessage ( );

      return 0 ;
}
```

client
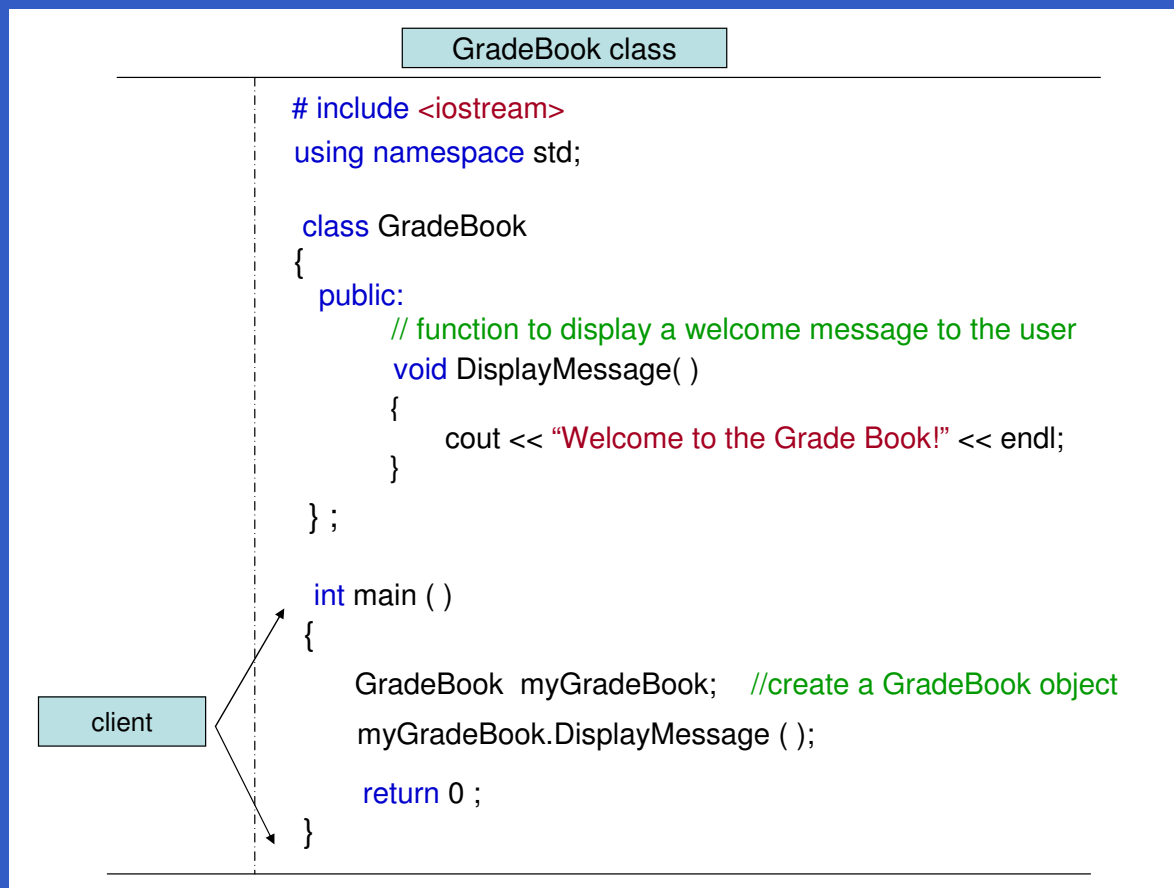
# Output: class GradeBook

```
Welcome to the Grade Book!
```

# Example: class GradeBook II

GradeBook class

```cpp
# include <iostream>
# include <string>
using namespace std;
  class GradeBook
{
 public:
       void displayMessage(string courseName )
       {
             cout << "Welcome to the Grade Book for\n"
                    << courseName << "!" << endl;
       }
} ;

 int main ( )
{
      string nameOfCourse;
      GradeBook  myGradeBook;   //create a GradeBook object

      cout << "Please enter the course name: " << endl;
      getline (cin, nameOfCourse);
      cout << endl;

      myGradeBook.displayMessage (nameOfCourse );
}
```

client
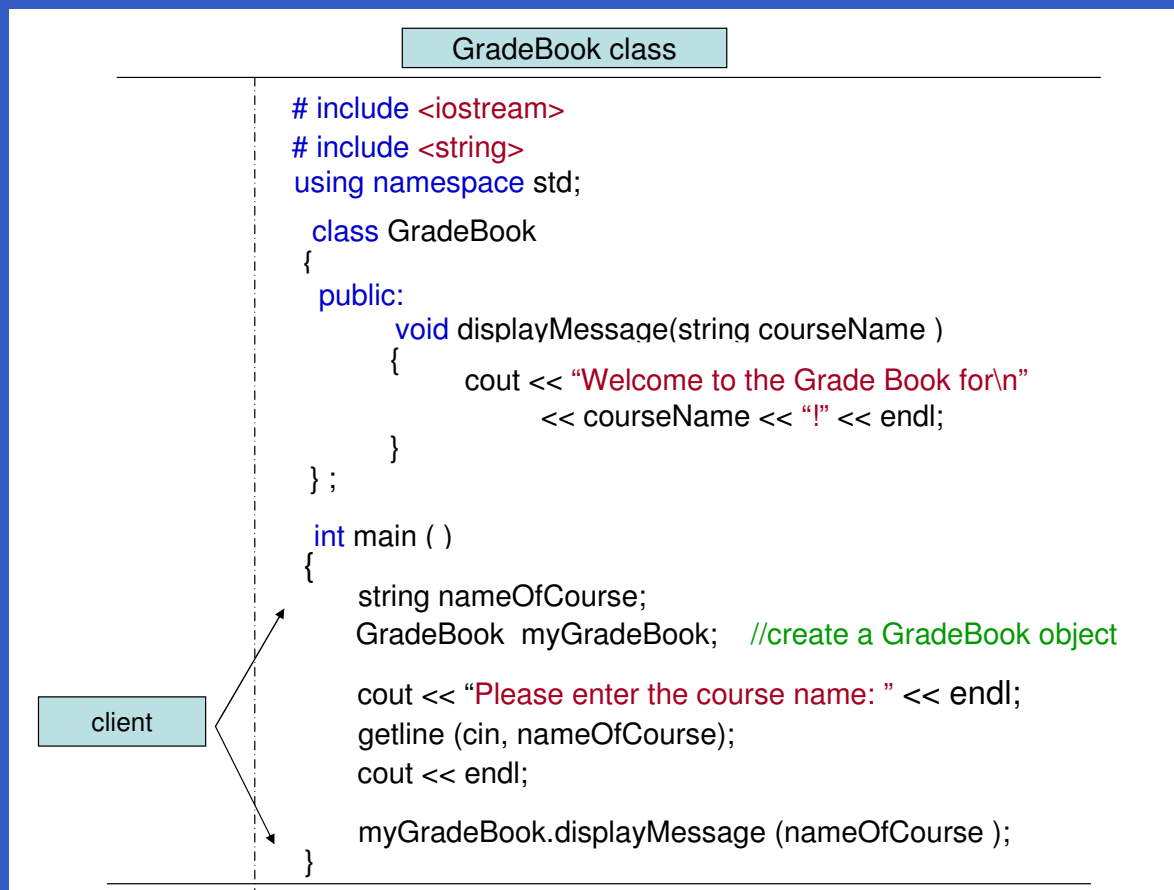
# Example: class GradeBook II

```
Please enter the course name:
CS101 Introduction to C++ Programming ✓

Welcome to the grade book for
CS101 Introduction to C++ Programming
```

# Data members, *set* **functions and** *get* **functions**

# Question: Cat Class

**Recall.** A class's private data members can be
manipulated only by member functions of that class.

```cpp
class Cat
{
  public:
        void Meow();
  private:
        unsigned int itsAge;
        unsigned int itsWeight
};
```

# Set functions *v.s.* get functions

**Fact.** A client of an object — i.e., any class or function that calls the object's member functions from outside the object — calls the class's public member functions to request the class's services for particular objects of the class.

# Set functions *v.s.* get functions

**Fact.** A client of an object — i.e., any class or function that calls the object's member functions from outside the object — calls the class's public member functions to request the class's services for particular objects of the class.

**Example.** In the definition of Cat class, *setItsAge* & *getItsAge*, *setItsWeight* & *getItsWeight* can be defined to manipulate the data members itsAge and itsWeight of the Cat objects from the outside.

# Example: class Cat II

```
                         Cat class II

            # include <iostream>
            using namespace std;

             class Cat
             {
               public:
                    void setItsAge ( int age );          //accesor function
                    int getItsAge ( );                   //accesor function
                    void  Meow ( );                      //general function
                 private:
                    int itsAge;                          //data member

             } ;


            void Cat::setItsAge ( int age )
            {
                  itsAge = age;
            }

            int Cat::getItsAge ( )
            {
                  return itsAge;
            }
```
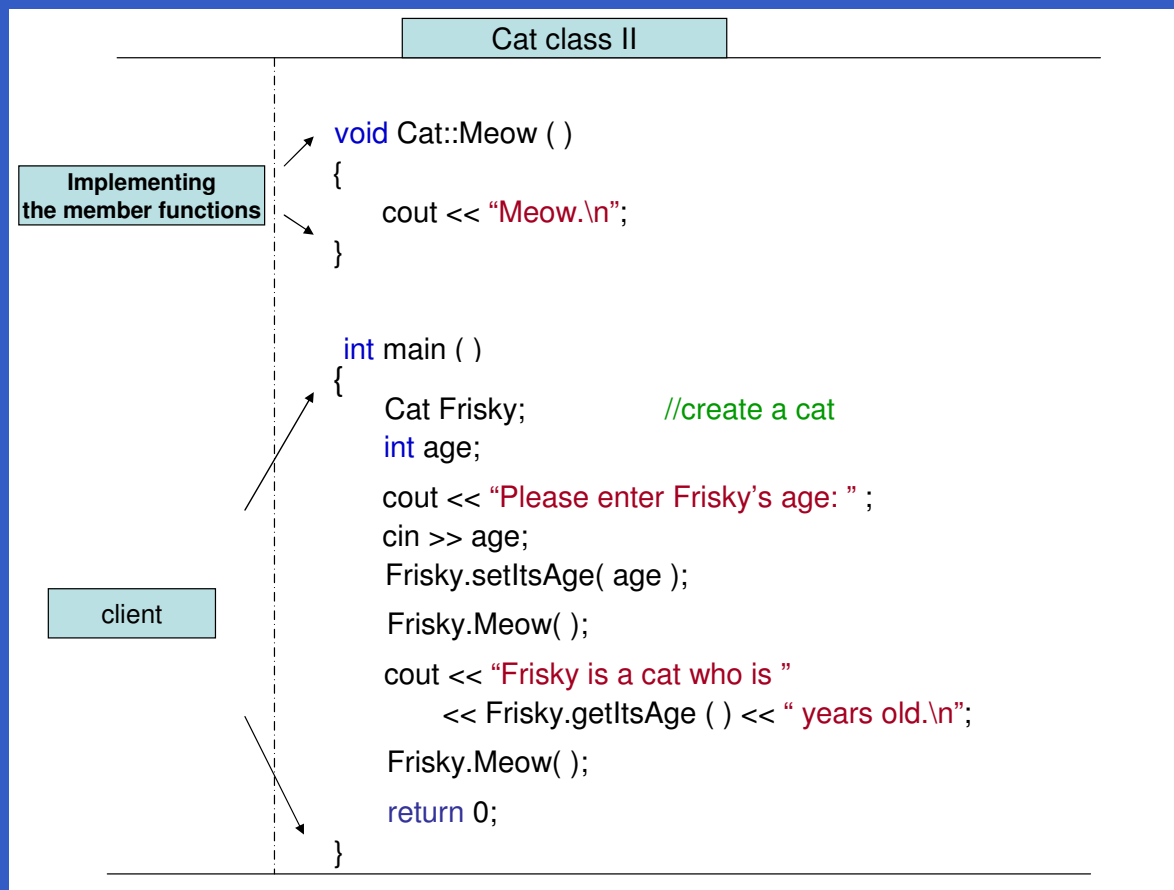
**Implementing the member functions**

# Example: class Cat II



Cat class II

void Cat::Meow ( )
{
    cout << "Meow.\n";
}

**Implementing the member functions**

int main ( )
{
    Cat Frisky;               //create a cat
    int age;

    cout << "Please enter Frisky's age: " ;
    cin >> age;
    Frisky.setItsAge( age );

    Frisky.Meow( );

    cout << "Frisky is a cat who is "
         << Frisky.getItsAge ( ) << " years old.\n";

    Frisky.Meow( );

    return 0;
}

**client**

# *Homework:*

- Read Sec. 9.1 - 9.4