

Classes

- Defining Classes
 - Data Members
 - Member Functions
- Static vs. Non-static members
- Constant functions
- Access Control Mechanisms
- Constructors & Destructors



Classes & Objects

- Four pillars of OOP are achieved from classes
 - Encapsulation
 - Data hiding
 - Inheritance
 - Polymorphism
- Class
 - Consists of both data and functions/methods
 - Defines properties and behavior of set of entities
 - Defines a new type (like *int*, *float* etc.)
- Object
 - An instance of class
 - Has *identity*, *state*, and *behavior*



Classes & Objects

```
class Rectangle
```

```
{  
  private:  
    int width;  
    int length;  
  public:  
    void set(int w, int l);  
    int area();  
}
```

```
Rectangle r1;  
Rectangle r2;  
Rectangle r3;
```

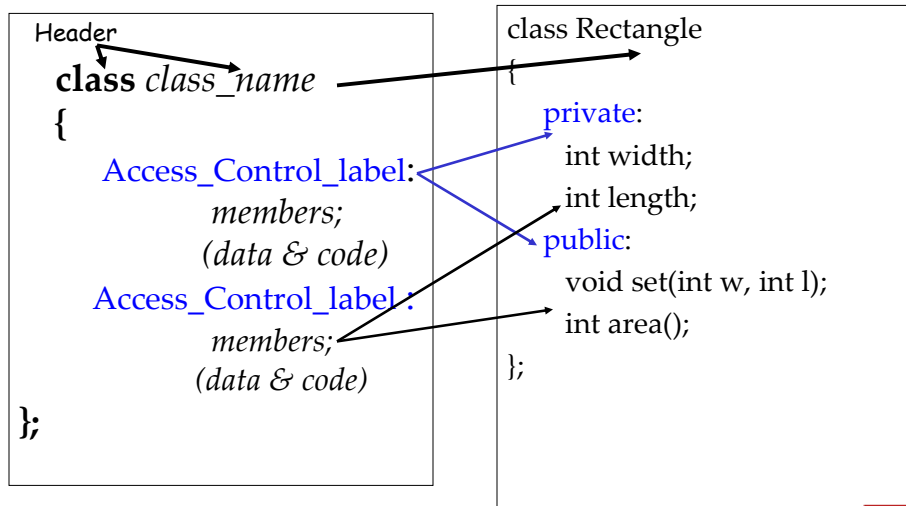
```
int a;
```

Copyright 2006, The Ohio State University

3



Defining a class



Copyright 2006, The Ohio State University

4



Class Definition – Data Members

- Abstracts the general attributes of the entity defined by class
- Type can be anything: *built-in* or *user-defined*

	Non-Static Member	Static Member
Existence	<i>Local</i> to object – each object has its <i>own copy</i>	<i>Global</i> to class – all objects <i>share the same copy</i>
Initialization	object level in member functions or constructor	Class level
Access	Qualified by object <object name>.<var>	Qualified by class <class name>.<var>

- Static data members and functions **MUST** be defined somewhere

Copyright 2006, The Ohio State University

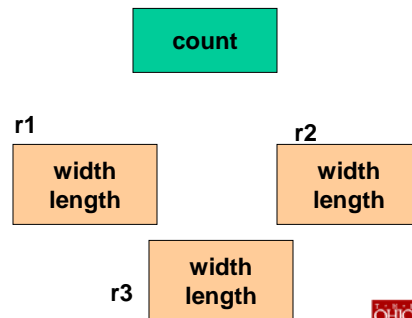
5



Static Data Member

```
class Rectangle
{
    private:
        int width;
        int length;
        static int count;
    public:
        void set(int w, int l);
        int area();
}
```

```
Rectangle r1;
Rectangle r2;
Rectangle r3;
```



Copyright 2006, The Ohio State University

6



Class Definition - Member Functions

- Non-static Member Function
 - Invoked from objects [*object.function()*]
 - Can access both *static* and *non-static* data members
- Static Member Function
 - Invoked using class [*classname::function()*]
 - Can only access *static* data members (but not *non-static*)
- Constant Member Function
 - Does not modify the state of the object !
- Const function can be invoked from both const & non-const objects
- Non-const function can be invoked only from non-const object

Copyright 2006, The Ohio State University

7



Define a Member Function

```
class Rectangle
{
  private:
    int width, length;
  public:
    void set (int w, int l);
    int area() {return width*length; }
}
```

class name

member function name

inline

```
r1.set(5,8);
rp->set(8,10);
```

```
void Rectangle :: set (int w, int l)
{
  width = w;
  length = l;
}
```

scope operator

Copyright 2006, The Ohio State University

8



Constant Member Functions

- **const** member function
 - declaration
 - *return_type func_name (para_list) const;*
 - definition
 - *return_type func_name (para_list) const { ... }*
 - *return_type class_name :: func_name (para_list) const { ... }*
 - Makes no modification about the data members (safe function)
 - It is illegal for a const member function to modify a class data member

Copyright 2006, The Ohio State University

9



Example of Const Member Function

```
class Time
{
private:
    int  hrs, mins, secs;

public:
    void  Write () const;
};
```

function declaration

function definition

```
void Time :: Write() const
{
    cout << hrs << ":" << mins << ":" << secs << endl;
}
```

Copyright 2006, The Ohio State University

10



Access Control Specifiers

- Mechanism to achieve *Information Hiding*
- To prevent the internal representation from direct access from outside the class
- **Public**
 - Can be accessible from anywhere in the program
- **Private (default)**
 - Can be accessible only from member functions and friends
- **Protected**
 - Acts as *public* for objects of its own class and derived classes
 - Acts as *private* to rest of the program
- Public functions – Class Interface
- Private functions – helper functions (can be accessed by class objects and friends)



class Time Specification

```
class Time
{
    public :

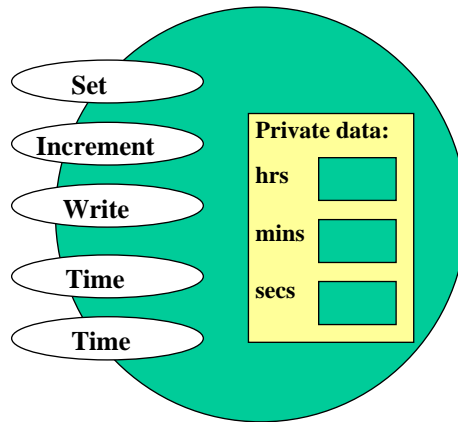
        void Set ( int hours , int minutes , int seconds );
        void Increment ( );
        void Write ( ) const ;
        Time ( int initHrs , int initMins , int initSecs ); // constructor
        Time ( ); // default constructor

    private :

        int hrs ;
        int mins ;
        int secs ;
};
```

Class Interface Diagram

Time class

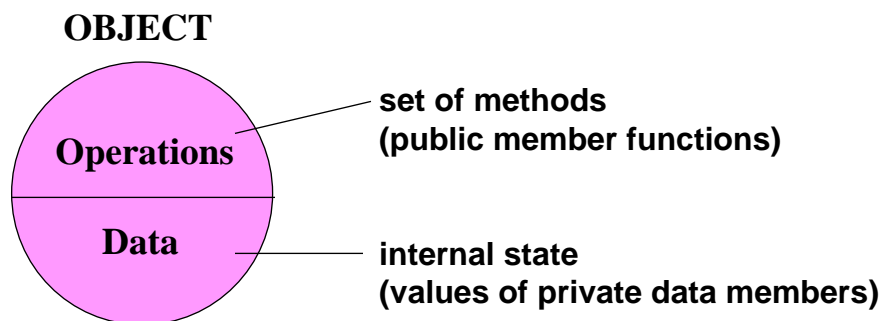


Copyright 2006, The Ohio State University

13



What is an object?



Copyright 2006, The Ohio State University

14



Declaration of an Object

```
class Rectangle
{
    private:
        int width;
        int length;
    public:
        void set(int w, int l);
        int area();
}
```

```
main()
{
    Rectangle r1;
    Rectangle r2;

    r1.set(5, 8);
    cout<<r1.area()<<endl;

    r2.set(8,10);
    cout<<r2.area()<<endl;
}
```



Take Home Message

- Class can be considered as a user-defined data type, while an object is just a variable of certain class.
- There are three parts in the definition of a class: data members, member functions, and access control.

