Object Oriented Programming (CS1143)

Week 13

Department of Computer Science

Capital University of Science and Technology (CUST)

Static Members

- Class members (static members)
  - Class Data Members
  - Class Member Functions

Instance Members Example

W13-p1.cpp

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

class Circle
{
    private:
        double radius;
    public:
        Circle (double r)
        {
            radius=r;
        }
        void setRadius (double r)
        {
            radius=r;
        }
        double getRadius () const
        {
            return radius;
        }
};
```

```cpp
int main ( )
{
    Circle circle1 (5.2);
    cout << "Radius: " << circle1.getRadius() << endl;
    return 0;
}
```

Class Data Member

Class Data Member or Static Data Member

- A Class data member is a data member that belongs to the class.
- It is shared by all the instances of a class.
- We use the keyword “static” to declare a class data member

Example

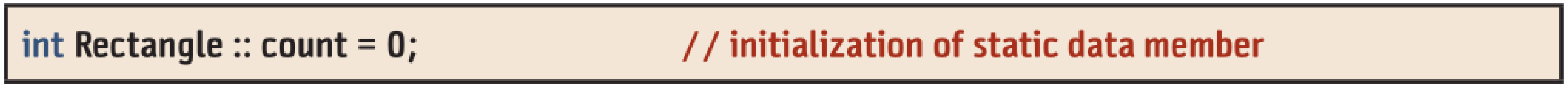
- We want to keep track of the number of current instances in our program.
- We can create a static data member, called count, which is initialized to 0.
- All constructors increment this static member every time an instance is created.
- The destructor decrements it when an object is destroyed (goes out of scope).

Declaring Static Data Members

```
class Rectangle
{
   private:
          ...
          static int count;                          // Static data member
   public:
          ...
}
```

Initializing Static Data Members

- A static data member does not belong to any instance, which means it cannot be initialized in a constructor.
- So it must be initialized after the class definition.



```
int Rectangle :: count = 0;                    // initialization of static data member
```

Class Member Function

Static (or class) Member Functions

- Since a static data member is normally private, we need a public member function to access it.
- Although this can be done by an instance member function, normally we use a static member function for this purpose.
- A static member function is not associated with any instance.

Declaring Static Member Functions

- A static member function, like a static data member, belongs to the class.
- It should be declared inside the class but must be qualified with the keyword static.

```
class Rectangle
{
   private:
         ...
         static int count;              // Static data member
   public:
         static int getCount();         // Static member function
         ...
}
```

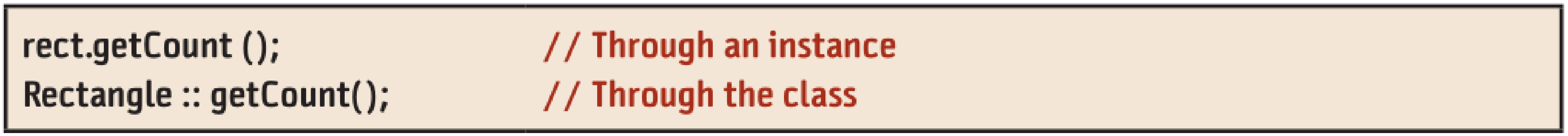
Defining Static Member Functions

- There is no difference between the definition of a static member function and an instance member function.
- We cannot use the const qualifier because there is no host object.

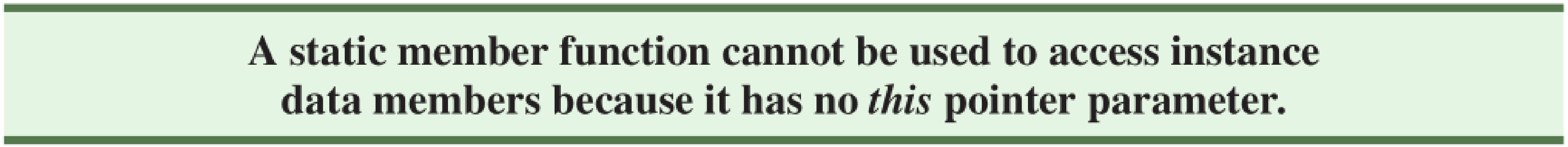
```
int Rectangle :: getCount()
{
   return count;
}
```

Calling Static Member Functions

- A static member function can be called either through an instance or through the class
- To call a static member function through an instance, we use the same syntax we use to call an instance member function
- To call a static member function through the class, we use the name of the class and the class resolution operator (::).



```
rect.getCount ();            // Through an instance
Rectangle :: getCount();     // Through the class
```

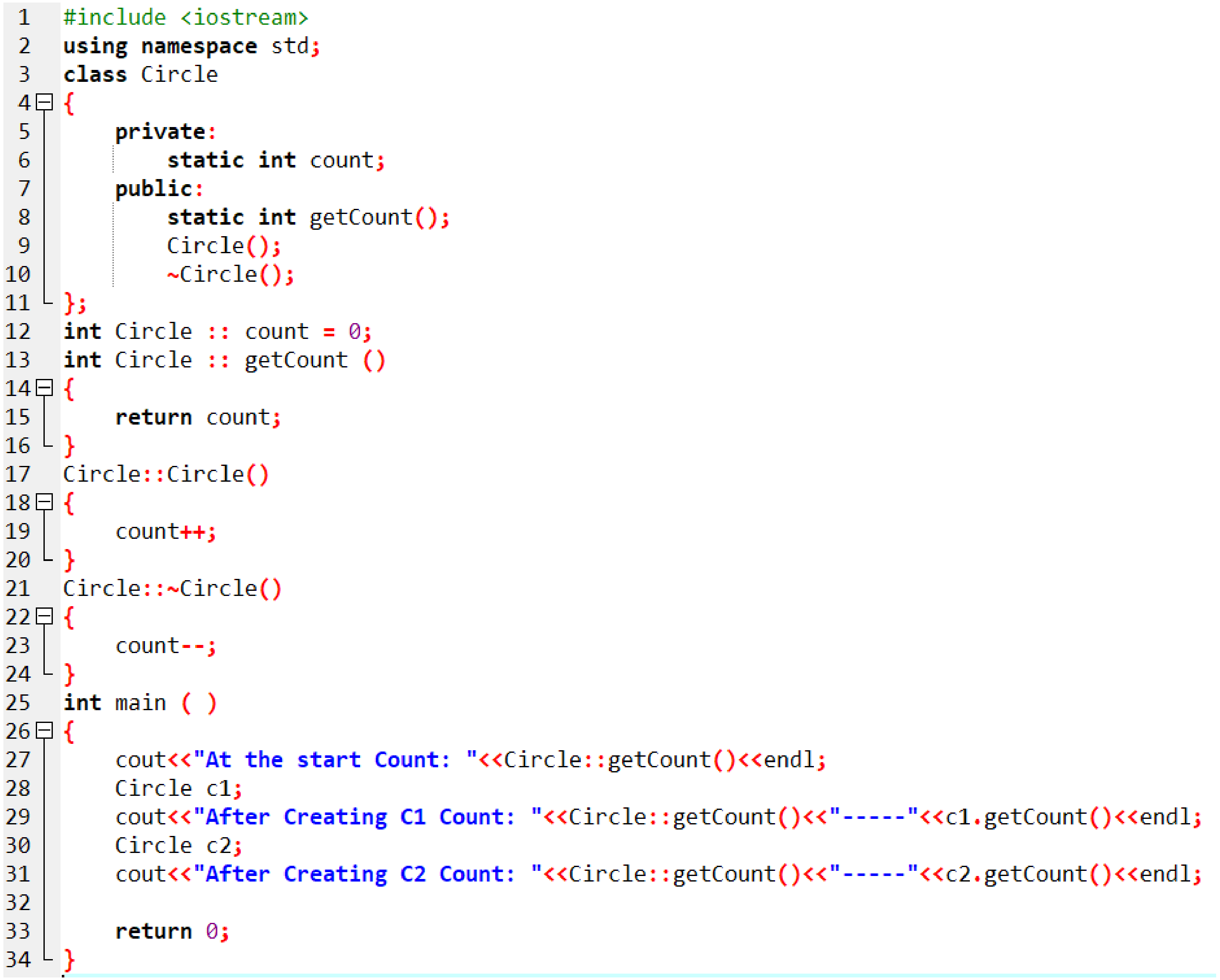


A static member function cannot be used to access instance data members because it has no *this* pointer parameter.

Calling Static Member Functions Contd..

- We cannot use a static member function to access an instance data member because a static member function does not have the hidden this pointer, which defines the instance that needs to be referenced.
- On the other hand, an instance member function can be used to access static data members (the this pointer is not used), but we usually avoid this.
- A good practice is to use instance member functions to access instance data members and static member functions to access static data members.

Example



```cpp
#include <iostream>
using namespace std;
class Circle
{
    private:
        static int count;
    public:
        static int getCount();
        Circle();
        ~Circle();
};
int Circle :: count = 0;
int Circle :: getCount ()
{
    return count;
}
Circle::Circle()
{
    count++;
}
Circle::~Circle()
{
    count--;
}
int main ( )
{
    cout<<"At the start Count: "<<Circle::getCount()<<endl;
    Circle c1;
    cout<<"After Creating C1 Count: "<<Circle::getCount()<<"-----"<<c1.getCount()<<endl;
    Circle c2;
    cout<<"After Creating C2 Count: "<<Circle::getCount()<<"-----"<<c2.getCount()<<endl;

    return 0;
}
```

Some more slides

`static` Class Members

- **`static` class members**
  - Shared by all objects of a class
  - Efficient, when a single copy of data is enough
    - Only the `static` variable has to be updated
  - May seem like global variables, but have class scope (only accessible to objects of same class)
  - Exist even if no instances (objects) of the class exist
  - Both variables and functions can be `static`
  - Can be `public`, `private`

`static` Class Variables

- **Two-Step Procedure:**
  1. Declare (Inside Class): `static int count;`
  2. Define (Outside Class): `int Circle::radius=100;`

- `static` variables initialization:
  - Default initialization: 0
  - Or initialize to user defined value
  - Initialization is made ***just once, at compile time.***

Public static Class Variables

- Can be accessed using class name:

```
cout<< Employee::count;
```

- Can be accessed via any object of the class:

```
cout<< e1.count;
```

- Can be accessed via non-static member functions:

```
cout<< e1.getCount();
```

- Can be accessed via static member functions:

```
cout<<Employee::Stat_getCount();
cout<<e1.Stat_getCount(); //public
static
```

Private static Class Variables

- Cannot be accessed using class name:

  // ERROR → `cout<<Employee::count;`

- Cannot be accessed via class object:

  // ERROR → `cout<<e1.count;`

- Can be accessed via non-static member functions:

```
    cout<<e1.getCount();
```

- Can be accessed via static member functions:

```
    cout<<Employee::Stat_getCount();
     cout<<e1.Stat_getCount(); //public
static
```

`static` Class Functions

- Non-static function:
  - Can access: static/non-static data members and static/non-static methods

- Static functions:
  - Can access: static data and static functions
  - Cannot access: non-static data, non-static functions, and this pointer

`Public static` Class Functions

- Can be invoked using any object of the class:

```
cout<<e1.getCount();
```

- Can be invoked using class name:

```
cout<<Employee::getCount();
```

`Private static` Class Functions

- Cannot be invoked using class’s object

  `//ERROR → cout<<e1.getCount();`

- Cannot be invoked using Class name

  `//ERROR →`
  `cout<<Employee::getCount();`

- **Can be invoked within class:**
  - Static member functions
  - Non-Static member functions

This is all for Week 13