

# Class (1A)

---

Copyright (c) 2011-2013 Young W. Lim.

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

Please send corrections (or suggestions) to [youngwlim@hotmail.com](mailto:youngwlim@hotmail.com).

This document was produced by using OpenOffice.

# Class Definition

```
class Ccircle {  
public:  
    int r;  
  
    Ccircle ()      { r = 1; }  
    Ccircle (int x) { r = x; }  
  
    void setR (int x) { r = x; }  
    int  getR () { return r; }  
    double area ();  
};
```

member functions defined  
outside the class definition

```
double Ccircle::area () {  
    return 3.14*r*r;  
}
```

A Member

Constructor  
functions

Member  
functions

Scope Operator ::  
Ccircle::



# Creating Objects

```
class Ccircle {  
public:  
    int r;  
  
    Ccircle ()    { r = 1; }  
    Ccircle (int x) { r = x; }  
  
    void setR (int x) { r = x; }  
    int getR () { return r; }  
    double area ();  
};
```

```
double Ccircle::area () {  
    return 3.14*r*r;  
}
```

```
void main(void) {
```

```
    Ccircle C1;    // default constructor is called
```

```
    Ccircle C2(10);
```

```
}
```

Ccircle C1(0);

object C1

r = 1

setR ()  
getR ()  
area ()

object C2

r = 10

setR ()  
getR ()  
area ()

- Member functions are called from objects
- Objects have their own member data
- So member functions access these distinct member data

# Calling Member Functions

```
class Ccircle {  
public:  
    int r;  
  
    Ccircle ()      { r = 1; }  
    Ccircle (int x) { r = x; }  
  
    void setR (int x) { r = x; }  
    int  getR () { return r; }  
    double area ();  
};
```

```
double Ccircle::area () {  
    return 3.14*r*r;  
}
```

```
void main(void) {
```

```
    Ccircle C1;      default constructor is called  
    Ccircle C2(10);
```

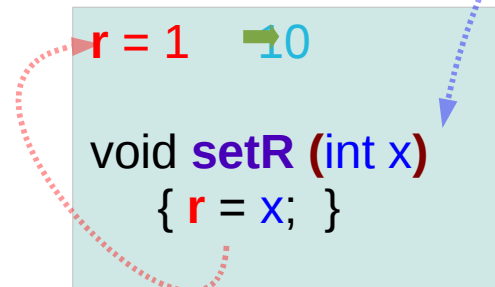
```
    C1.setR (10);  
    C2.setR (20);
```

```
}
```

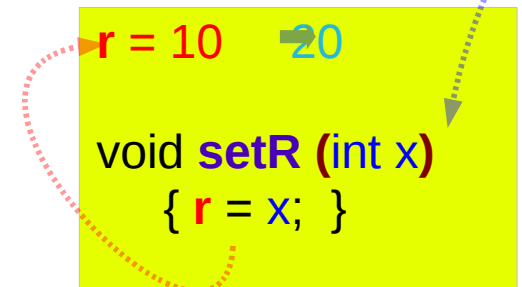
C1.setR (10);

C2.setR (20);

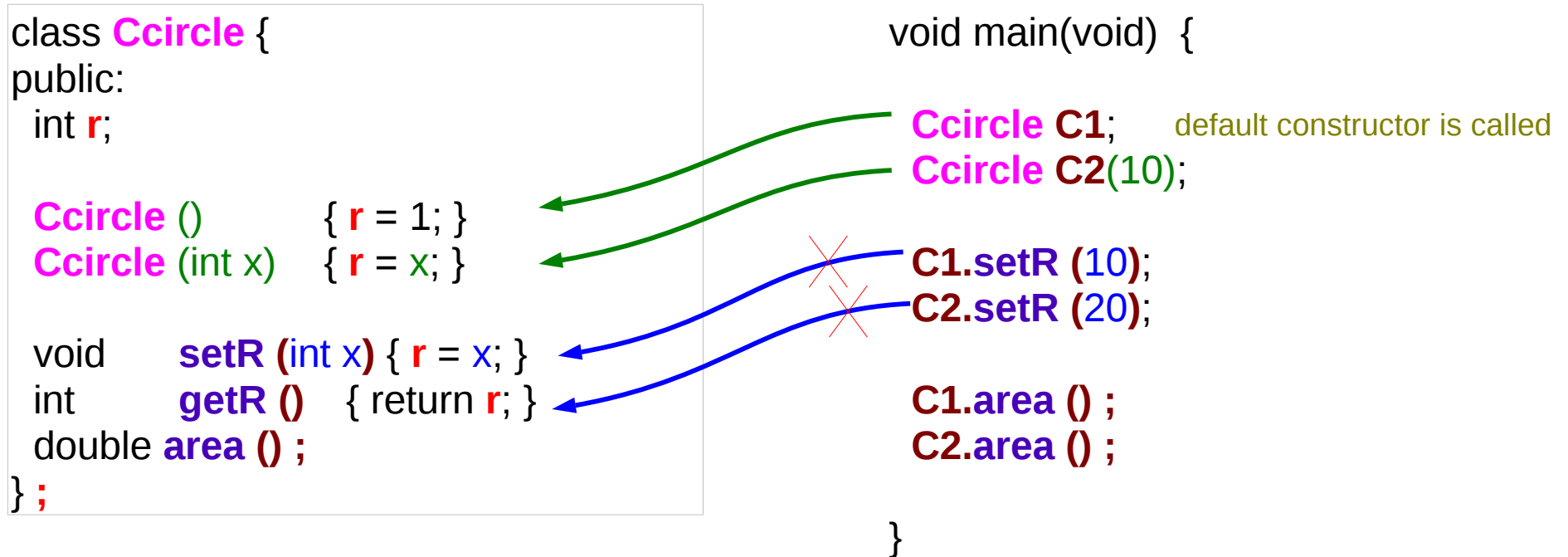
object C1



object C2



# Member Functions : called from objects



```
double Ccircle::area () {  
    return 3.14*r*r;  
}
```

- Member functions are called from objects
- Objects have their own member data
- So member functions access these distinct member data

# Objects and Member Function Calls

```
Ccircle C1;  
C1.setR (10);
```



object C1

```
r = 10  
  
setR ()  
getR ()  
area ()
```

```
Ccircle C2(10);  
C2.setR (20);
```



object C2

```
r = 20  
  
setR ()  
getR ()  
area ()
```

```
C1.area ();
```

➔  $3.14 \cdot 10^2$

object C1

```
r = 10  
  
double area ()  
{ return 3.14*r*r; }
```

```
C2.area ();
```

➔  $3.14 \cdot 20^2$

object C2

```
r = 20  
  
double area ()  
{ return 3.14*r*r; }
```

# Member Function Call

```
class Ccircle {  
public:  
    int r;  
  
    Ccircle ()      { r = 1; }  
    Ccircle (int x) { r = x; }  
  
    void setR (int x) { r = x; }  
    int  getR () { return r; }  
    double area () ;  
};
```

```
void main(void) {  
  
    Ccircle C1;  
    Ccircle C2(10);  
  
    C1.setR (10);  
    C2.setR (20);  
  
    C1.area () ;  
    C2.area () ;  
  
}
```

possible implementation :

```
void Ccircle::setR (Ccircle *this, int x)  
{  
    this->r = x;  
}
```



r = x;

passing a pointer hidden to a programmer

```
Ccircle::setR (&C1, 10);
```



# Access Specifier

```
class Ccircle {  
public:  
    int r;  
  
    Ccircle ()      { r = 1; }  
    Ccircle (int x) { r = x; }  
  
    void setR (int x) { r = x; }  
    int  getR () { return r; }  
    double area ();  
};
```

```
class Ccircle {  
    int r;          default private:  
  
public:  
    Ccircle ()      { r = 1; }  
    Ccircle (int x) { r = x; }  
  
    void setR (int x) { r = x; }  
    int  getR () { return r; }  
    double area ();  
};
```

```
void main(void) {
```

```
    Ccircle C1;  
    Ccircle C2(10);
```

```
    C1.setR (10);  
    C2.setR (20);
```

```
C1.r = 13;  
C2.r = 24;
```

```
}
```

# Member Function Definition within a class

```
int func1() {  
    mem2 = 10;  
    func2 ();  
    mem3 = 10;  
    func3 ();  
}
```

*member functions of  
the same class*

```
int func2() {  
    mem1 = 10;  
    func1 ();  
    mem3 = 10;  
    func3 ();  
}
```

*member functions of  
the same class*

```
int func3() {  
    mem1 = 10;  
    func1 ();  
    mem2 = 10;  
    func2 ();  
}
```

*member functions of  
the same class*

```
class CC {  
private:  
    int mem1;  
    int func1();  
protected:  
    int mem2;  
    int func2();  
public:  
    int mem3;  
    int func3();  
};
```

The diagram illustrates the class `CC` with its members grouped by access level: `private:` (orange box), `protected:` (green box), and `public:` (blue box). Dashed arrows indicate that each member can access all other members within the same class, regardless of their access level.

Each members can be accessed by other members of the same class

# Member Function Definition within a derived class

The members of a derived class can access public and protected members of the base class

```
class CC {  
  private:  
    int mem1;  
    int func1();  
  
  protected:  
    int mem2;  
    int func2();  
  
  public:  
    int mem3;  
    int func3();  
};
```

```
class EE : public CC {  
  int func4() {  
    mem2;  
    func2 ();  
    mem3;  
    func3 ();  
  }  
};
```

# Member Function Call from objects

```
void main(void) { the main function  
  CC C1;  
  
  C1.mem3;  
  C1.func3 ();  
}
```

```
int foo(CC *X) { C-style functions  
  
  X->mem3;  
  X->func3 ();  
}
```

```
class DD { member functions of  
other classes  
  int faa(CC *Y) {  
    Y->mem3;  
    Y->func3 ();  
  }  
};
```

```
class CC {  
  private:  
    int mem1;  
    int func1();  
  
  protected:  
    int mem2;  
    int func2();  
  
  public:  
    int mem3;  
    int func3();  
};
```

Only public members can be accessed

# Class Structure

---

# Class Structure

---

## References

- [1] W Savitch, "Absolute C++"
- [2] P.S. Wang, "Standard C++ with objected-oriented programming"
- [3] <http://www.cplusplus.com>