Class Members

Data Members
Member Functions (Methods)
Accessing Class Members

Muhammad Hammad Waseem

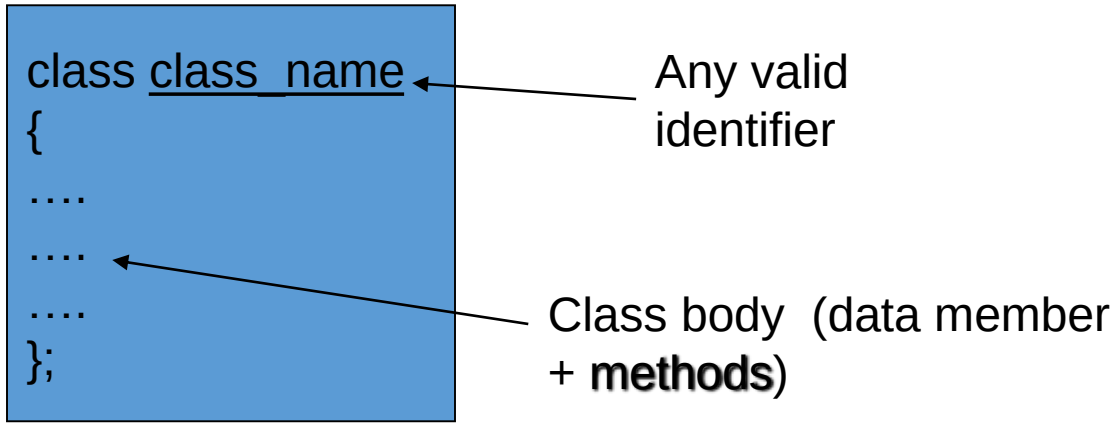
m.hammad.wasim@gmail.com

Object Oriented Programming

- Object-oriented programming (OOP)
  - Encapsulates data (attributes) and functions (behavior) into packages called classes.
- So, Classes are user-defined (programmer-defined) types.
  - Data (data members)
  - Functions (member functions or methods)
- In other words, they are structures + functions

Classes in C++

- A class definition begins with the keyword *class*.
- The body of the class is contained within a set of braces, { } ; (notice the semi-colon).

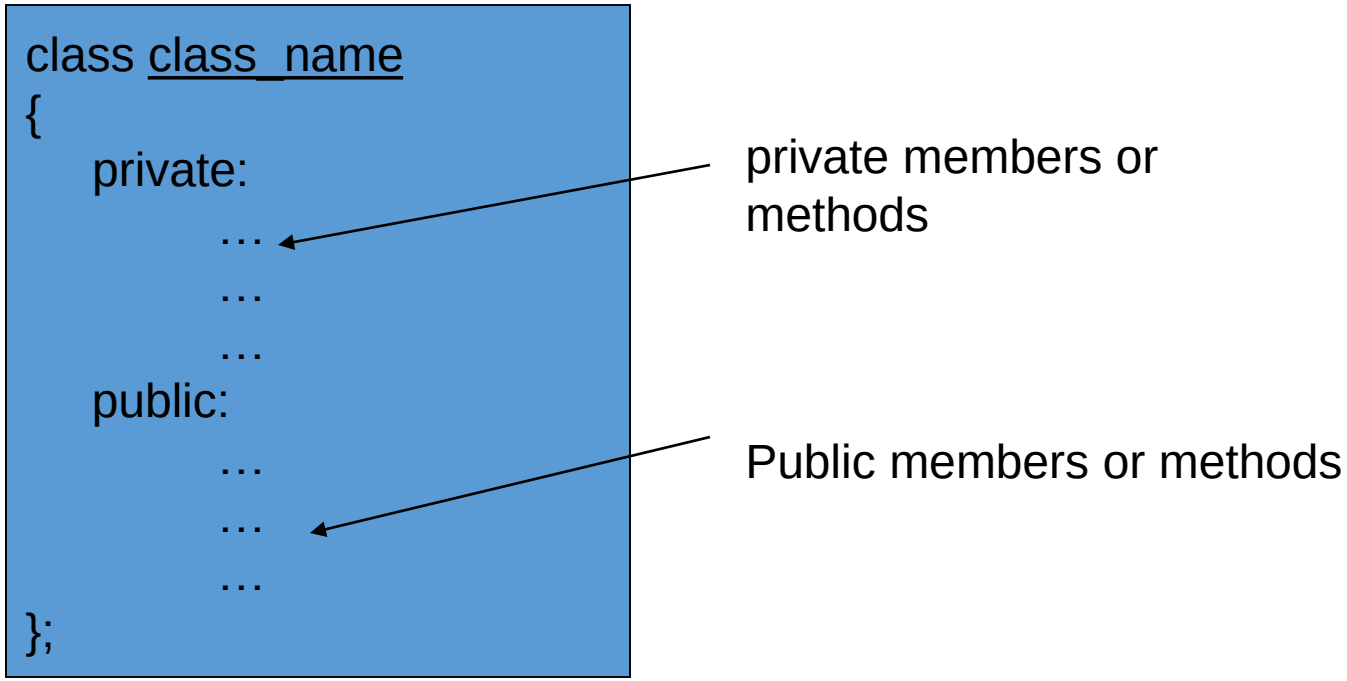


Classes in C++

- 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 *public:* section.

Classes in C++

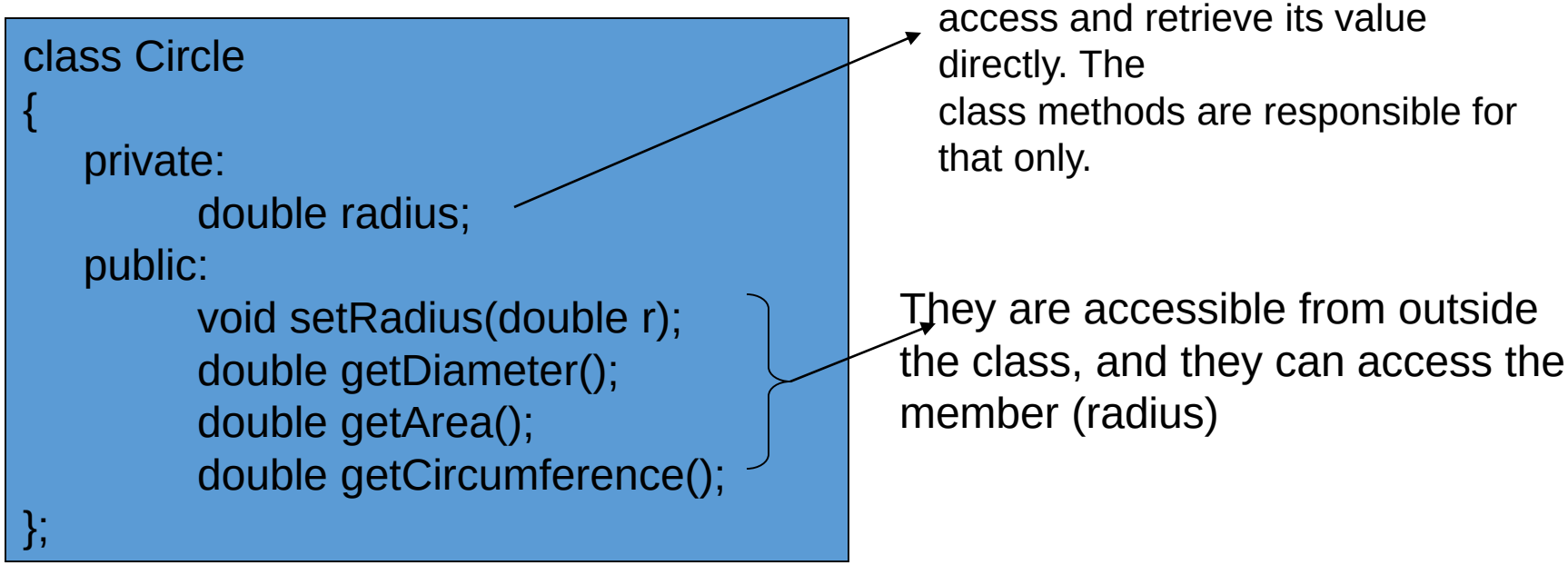

class class_name
{
private:
…
…
…
public:
…
…
…
};
private members or methods
Public members or methods

Classes in C++

- Member access specifiers
  - public:
    - can be accessed outside the class directly.
  - private:
    - Accessible only to member functions of class
    - Private members and methods are for internal use only.

Class Example

- This class example shows how we can encapsulate (gather) a circle information into one package (unit or class)



Special Member Functions

- Constructor:
  - Public function member
  - called when a new object is created (instantiated).
  - Initialize data members.
  - Same name as class
  - No return type
  - Several constructors
    - Function overloading

Special Member Functions

```
class Circle
{
    private:
            double radius;
    public:
            Circle();
            Circle(int r);
             void setRadius(double r);
            double getDiameter();
            double getArea();
            double getCircumference();
};
```

Constructor with no argument

Constructor with one argument

Implementing class methods

- Class implementation: writing the code of class methods.
- There are two ways:

1. Member functions defined inside class

```
MemberFunctionName( )
{
    …
}
```

2. Member functions defined outside class

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

Implementing class methods

1. Member functions defined inside class
   - Do not need scope resolution operator, class name;

```
class Circle
{
    private:
        double radius;
    public:
        Circle() { radius = 0.0;}
        Circle(int r);
        void setRadius(double r){radius = r;}
        double getDiameter(){ return radius *2;}
        double getArea();
        double getCircumference();
};
```

Defined inside class

Defining Member Functions outside Class

- The member function of a class can also be defined outside the class.
- The declaration of member functions is specified within the class and function definition is specified outside the class.
- The scope **resolution operator ::** is used in function declaration if the function is defined outside the class.

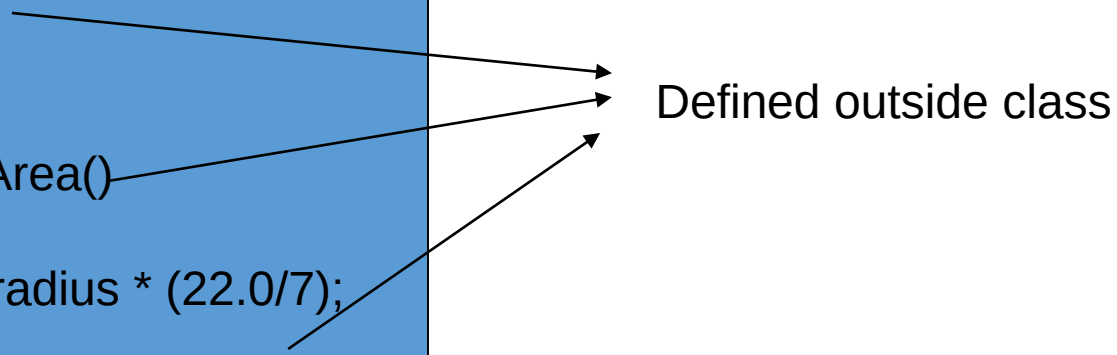
Syntax

- The syntax of defining member function outside the class is as follows:

```
Return_type Class_name:: function_name(params)
{
          Function body
}
```

- Return_type
  - It indicates the type of value to be returned by the function.
- Class_name
  - It indicates the name of class to which the function belongs.
- ::
  - It is the scope resolution operator to define member function outside the class.
- function_name
  - It is the name of the member function to be defined.
- Function body
  - It is the body of the function.

```
class Circle
{
    private:
            double radius;
    public:
            Circle() { radius = 0.0;}
            Circle(int r);
            void setRadius(double
r){radius = r;}
            double getDiameter(){ return
radius *2;}
            double getArea();
            double getCircumference();
};
Circle::Circle(int r)
{
    radius = r;
}
double Circle::getArea()
{
    return radius * radius * (22.0/7);
}
double Circle:: getCircumference()
{
    return 2 * radius  * (22.0/7);
```



Accessing Class Members

- Operators to access class members
  - Identical to those for **`struct`**s
  - Dot member selection operator (`.`)
  - Arrow member selection operator (`->`)
    - Pointers

```
class Circle
{
    private:
            double radius;
    public:
            Circle() { radius = 0.0;}
            Circle(int r);
            void setRadius(double r){radius = r;}
            double getDiameter(){ return radius *2;}
            double getArea();
            double getCircumference();
};
Circle::Circle(int r)
{
    radius = r;
}
double Circle::getArea()
{
    return radius * radius * (22.0/7);
}
double Circle:: getCircumference()
{
    return 2 * radius  * (22.0/7);
}
```

```
void main()
{
    Circle c1,c2(7);

    cout<<"The area of c1
        <<c1.getArea()<<"\n";

    //c1.raduis = 5;//syntax error
    c1.setRadius(5);

    cout<<"The circumference of c1:"
       << c1.getCircumference()<<"\n";

    cout<<"The Diameter of c2:"
       <<c2.getDiameter()<<"\n";
}
```

The second constructor is called

Since radius is a private class data member

```
class Circle
{
    private:
            double radius;
    public:
            Circle() { radius = 0.0;}
            Circle(int r);
            void setRadius(double r){radius = r;}
            double getDiameter(){ return radius *2;}
            double getArea();
            double getCircumference();
};
Circle::Circle(int r)
{
    radius = r;
}
double Circle::getArea()
{
    return radius * radius * (22.0/7);
}
double Circle:: getCircumference()
{
    return 2 * radius  * (22.0/7);
}
```

```
void main()
{
    Circle c(7);
    Circle *cp1 = &c;
    Circle *cp2 = new Circle(7);

    cout<<"The are of cp2:"
            <<cp2->getArea();

}
```