Introduction to Classes and Objects

Abstract Data Types

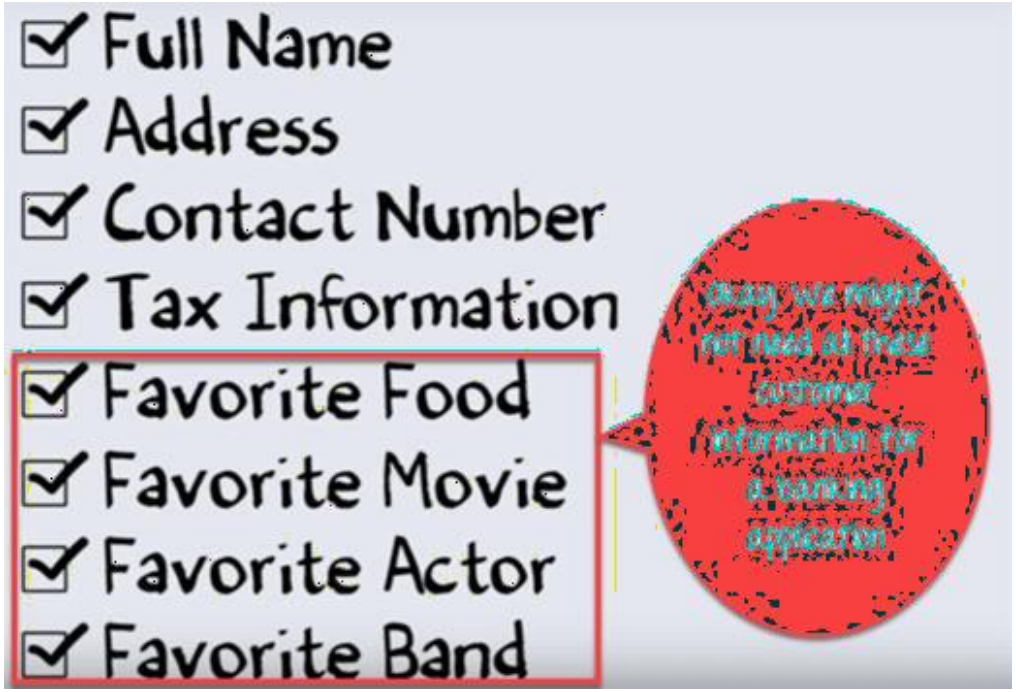
- **Programmer-created data types that specify**
  - **legal values that can be stored**
  - **operations that can be done on the values**

- ***The user of an abstract data type (ADT) does not need to know any implementation details (e.g., how the data is stored or how the operations on it are carried out).***

- **Abstraction allows a programmer to design a solution to a problem and to use data items *without concern for how the data items are implemented*.**

- **Note that, to use the pow function, you need to know what inputs it expects and what kind of results it produces. *You do not need to know how it works*.**

- **Abstraction: a definition that captures general characteristics without details**



Object-Oriented Programming (OOP)

- **OOP is a computer programming model that organizes software design around data, or objects, rather than functions and logic.**
- **An object can be defined as a data field that has unique attributes and behaviour.**
- **OOP focuses on the *objects* that developers want to manipulate rather than the logic required to manipulate them.**

- **Everything in C++ is associated with classes and objects, along with its attributes and methods.**

- ***For example: in real life, a car is an object. The car has attributes, such as weight and colour, and methods, such as drive and brake.***

- **Attributes and methods are basically variables and functions that belongs to the class.**

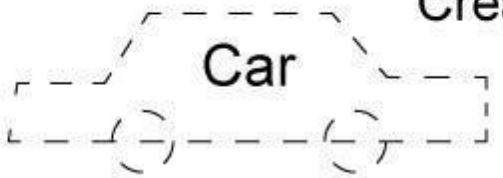
- **These are often referred to as "class members".**

- **A class is a user-defined data type that we can use in our program, and it works as an object constructor, or a "blueprint" for creating objects.**

Class

Object

Create an instance



Properties	**Methods** - behaviors
color	start()
price	backward()
km	forward()
model	stop()

Property values	**Methods**
color: red	start()
price: 23,000	backward()
km: 1,200	forward()
model: Audi	stop()

CLASS

What is the structure of object-oriented programming?

- **The structure, or building blocks, of object-oriented programming include the following:**

- **Classes are user-defined data types that act as the blueprint for individual objects, attributes and methods.**

- **Objects are instances of a class created with specifically defined data. *Objects* can correspond to real-world objects or an abstract entity.**

- **Methods are functions that are defined inside a class that describe the behaviors of an object.**

- **Each method contained in class definitions starts with a reference to an instance object.**

- **Attributes are defined in the class template and represent the state of an object.**

The other components of oops are:

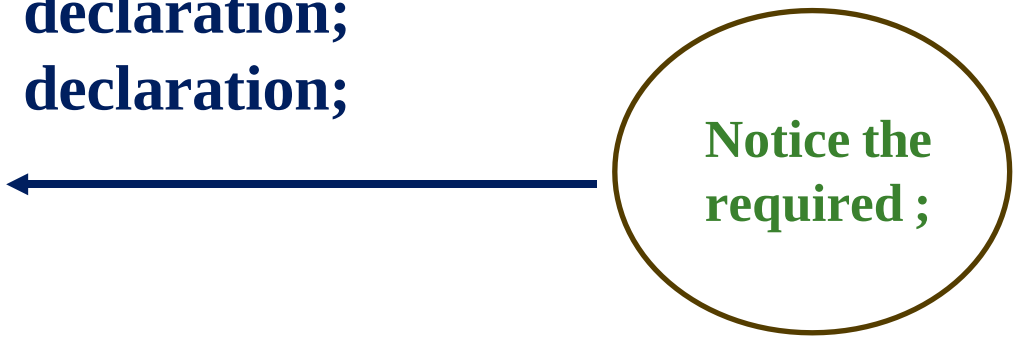
- **Encapsulation**
- **Inheritance**
- **Data Hiding**
- **Polymorphism**

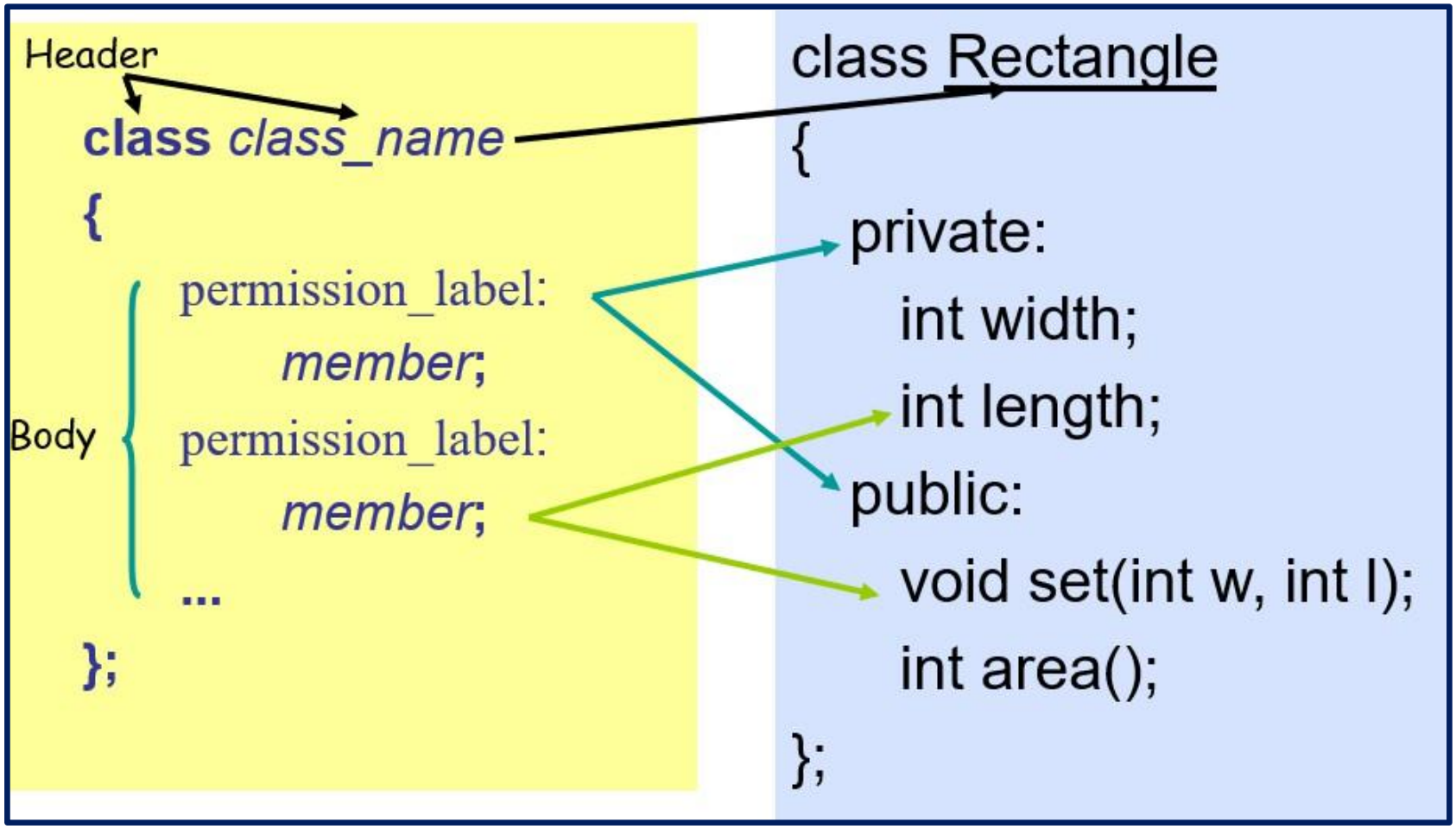
A Simple Class

- **Class: a programmer-defined data type used to define *objects*.**

- **Class declaration format:**

```
class className
{
    declaration;
    declaration;
};
```




Header
class class_name
{
permission_label:
member;
permission_label:
member;
...
};
Body
class Rectangle
{
private:
int width;
int length;
public:
void set(int w, int l);
int area();
};

- **Consider the following program that contains a class and two objects of that class.**

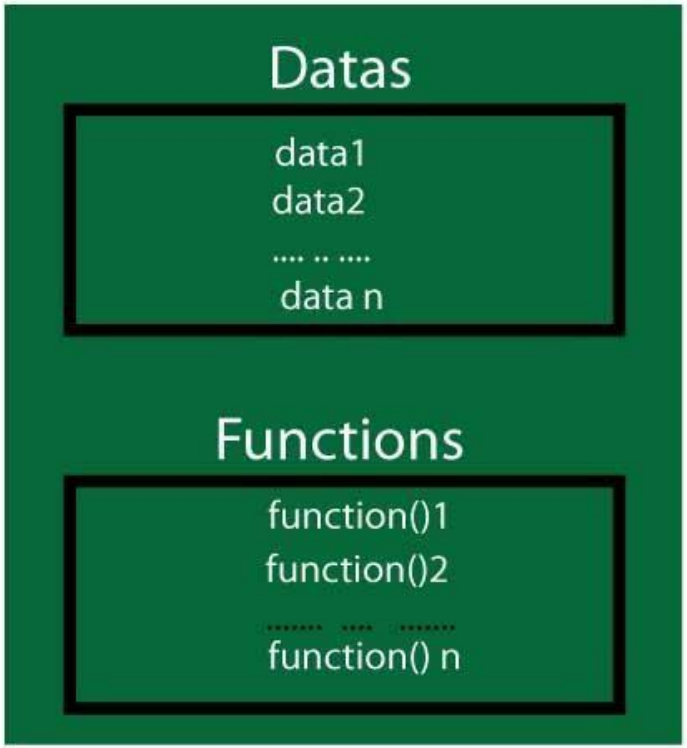
```
#include <iostream>
using namespace std;
class smallobj          //define a class
{
    private:
    int somedata;       //class data
    public:
    void setdata(int d)     //member function to set data
    {
          somedata = d;
    }
    void showdata() //member function to display data
    {
          cout << "Data is " << somedata << endl; }
};
```

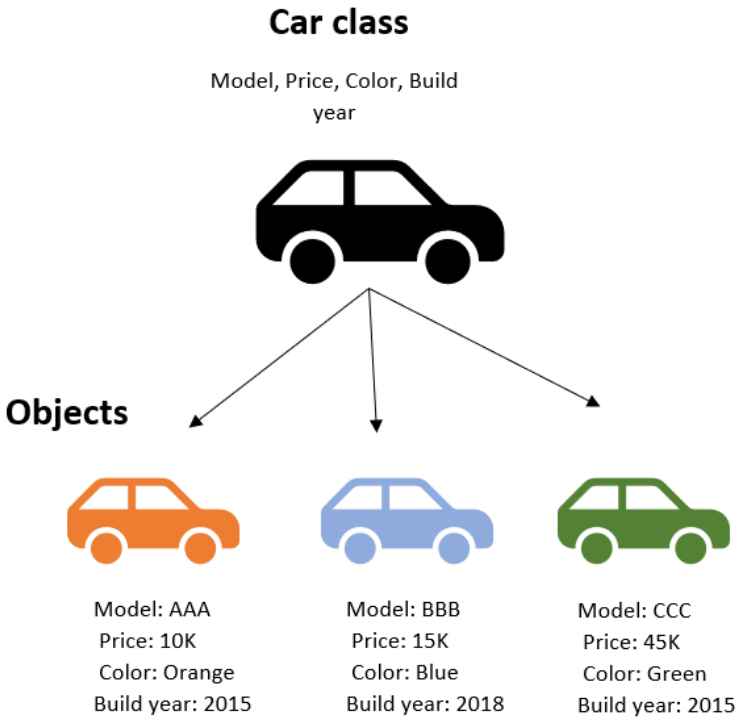
Continued

```
int main()
{
   smallobj s1, s2;      //define two objects of class smallobj
   s1.setdata(1066);    //call member function to set data
   s2.setdata(1776);
   s1.showdata();     //call member function to display data
   s2.showdata();
   return 0;
}
```

- **The class *smallobj* defined in this program contains *one data item* and *two member functions*.**
- ***The two member functions provide the only access to the data item from outside the class*.**
- **The first member function sets the data item to a value, and the second displays the value.**
- **Placing data and functions together into a single entity is a central idea in OOP as illustrated in the following Figure**

Class


Datas
data1
data2
....
data n
Functions
function()1
function()2
....
function() n


Car class
Model, Price, Color, Build year
Objects
Model: AAA
Price: 10K
Color: Orange
Build year: 2015
Model: BBB
Price: 15K
Color: Blue
Build year: 2018
Model: CCC
Price: 45K
Color: Green
Build year: 2015

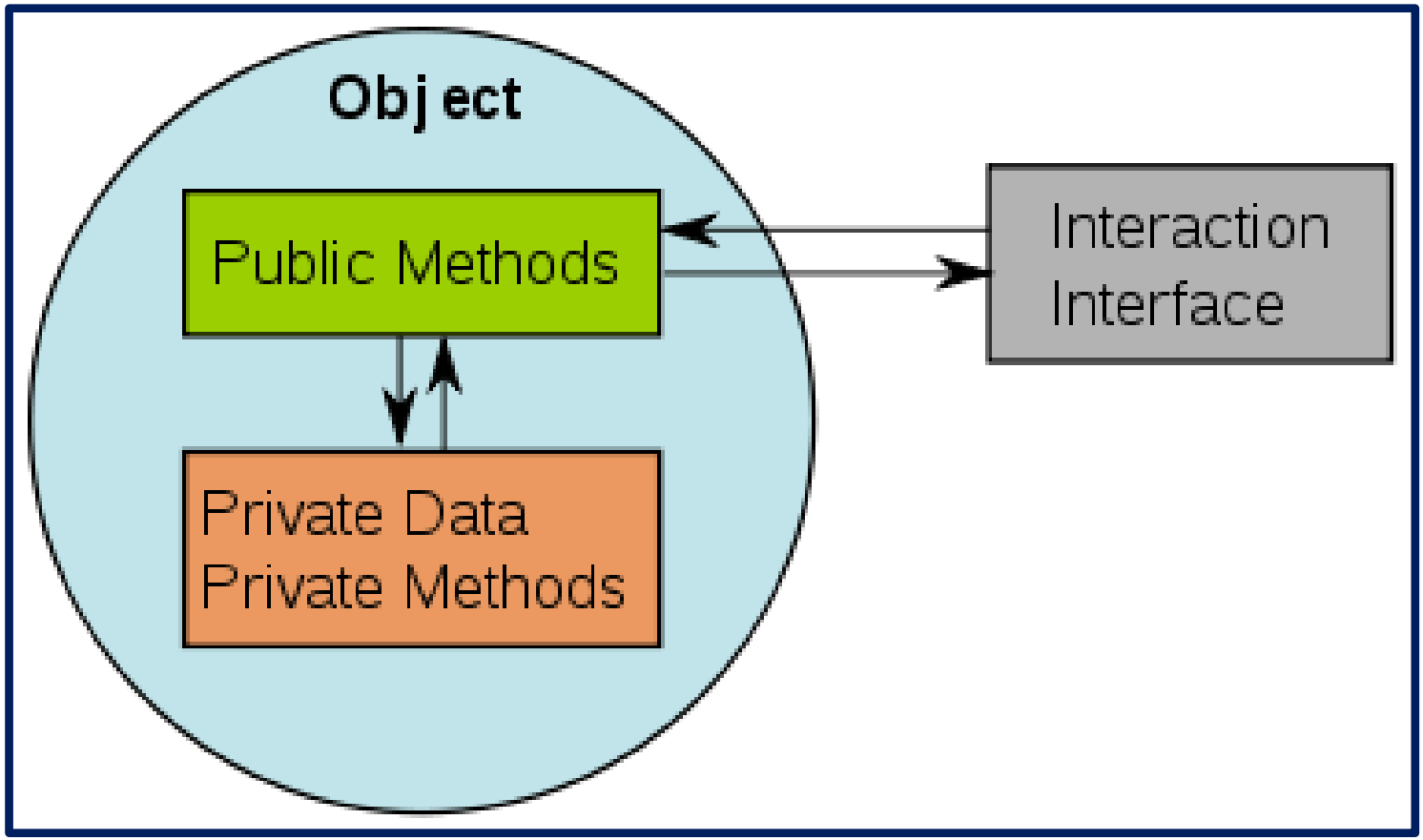
- **An object is said to be an instance of a class.**
- **In SMALLOBJ, the class—whose name is smallobj—is defined in the first part of the program.**
- **Later, in main(), two objects—s1 and s2— are defined, that are instances of that class.**
- **Each of the two objects is given a value, and each displays its value. *Here's the output of the program*:**
- **Data is 1066 ← object s1 displayed this**
- **Data is 1776 ← object s2 displayed this**

- **As can be noted in the program given above, the definition starts with the keyword class, followed by the class name—smallobj.**

- **Like a *structure*, the body of the *class* is delimited by *braces* and terminated by a *semicolon*.**

- **The body of the class contains two keywords: *private* and *public*.**

- **Simply, *private* data or functions can only be accessed from within the class and *public data* or functions are accessible from outside the class.**

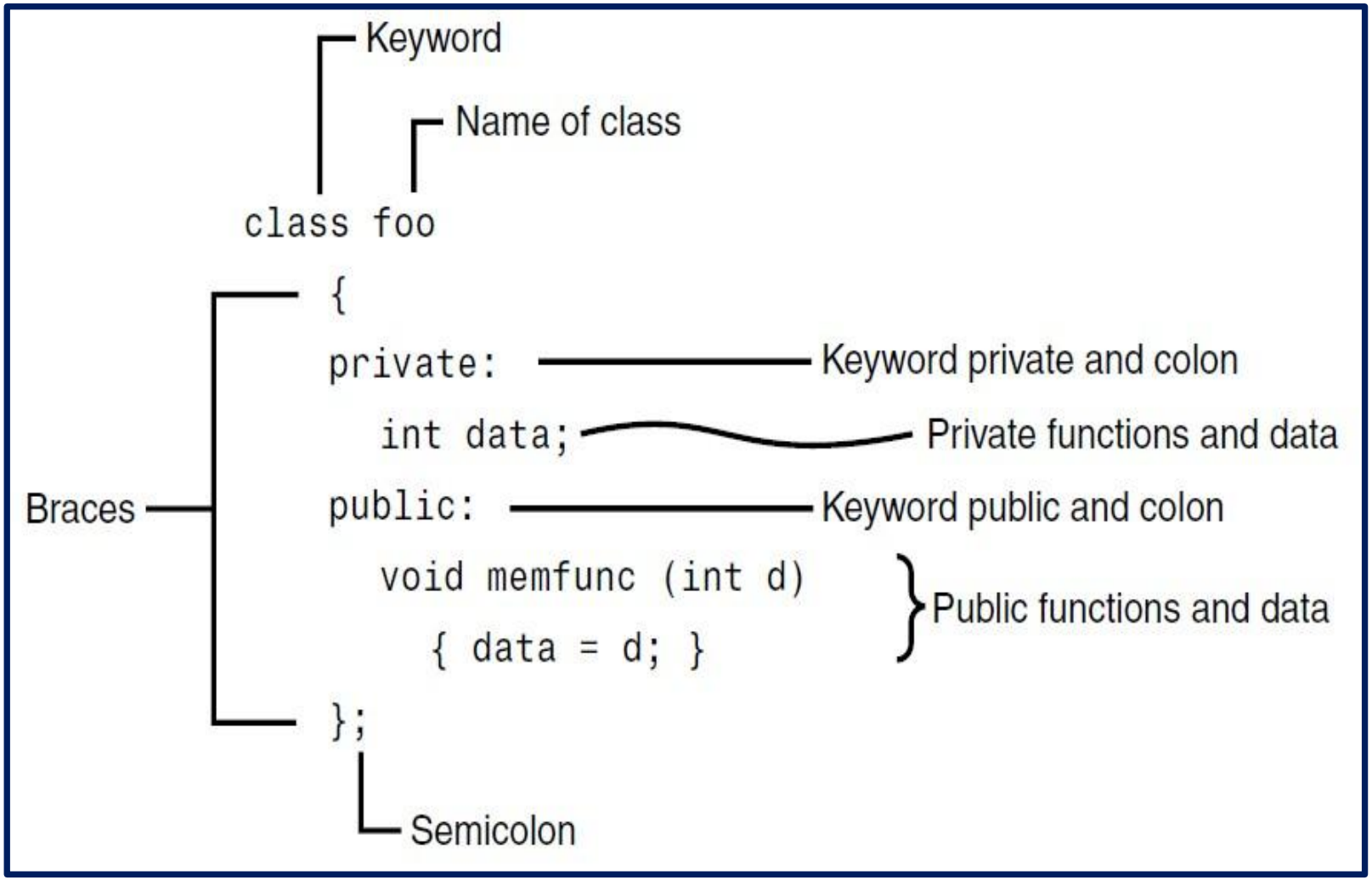

Object
Public Methods
Private Data
Private Methods
Interaction
Interface

- **Using the keyword *private* is a key feature of OOP called data hiding.**

- **The goal of data hiding is to protect data within a class from unwanted access and to prevent unneeded intrusion from outside the class. *The intent is to allow only member functions to directly access and modify the object's data.***

- **To provide a security measure you might, for example, require a user to supply a password before granting access to a database.**

- **The password is meant to keep unauthorized users from altering (or often even reading) the data.**


Keyword
Name of class
class foo
{
private:
Keyword private and colon
int data;
Private functions and data
Braces
public:
Keyword public and colon
void memfunc (int d)
{ data = d; }
Public functions and data
};
Semicolon

- **Some object-oriented languages refer to calls to member functions as messages.**

- **Thus, the call s1.showdata(); can be thought of as sending a message to s1 telling it to show its data.**

Ex 1. The following program implements a class called Circle that has private member variables for radius and includes member functions to calculate the circle's area and circumference.

```
#include <iostream>
using namespace std;
class Circle
{
    private:
        float r;
    public:
        float area(float rad)
        {
            r = rad;
            return 3.14 * r * r;

        }
        float cir(float rad)
        {
            r = rad;
            return 2 * 3.14 * r;
        }
};
```

```
int main()
{
    Circle s1;
    cout << "The area is  " << s1.area(5.0) << endl;
    cout << "The circumference of the circle is  "
    << s1.cir(5.0);
}
```

Sample output

```
The area is  78.5
The circumference of the circle is  31.4
```

Ex. 2 A program that uses class named Convert used to convert from Fahrenheit to Celsius. The class contains private member variable called r and public member function called Con.

```
#include <iostream>
using namespace std;
class Convert
{
    private:
        int temp_grade;
    public:
        float Con(int F)
        {
            temp_grade = F;
            return (5.0/9.0)*(temp_grade - 32.0);

        }
};
int main()
{
    Convert FC;
    cout << "Fahrenheit to Celsius is  " << FC.Con(100) << endl;
    return 0;
}       //Sample Run
```

```
Fahrenheit to Celsius is   37.7778
```

Ex.3 Write a program to create a class called Rectangle that has private member variables for length and width. Implement member functions to calculate the rectangle's area and perimeter.

```
#include <iostream>
using namespace std;
class Rectangle
{
    private:
        float length, width;
    public:
        float recArea(float l, float w)
        {
            length = l; width = w;
            return length * width;

        }
        float recfler(float l, float w)
        {
            length = l; width = w;
            return 2 * (length + width);

        }

};
```

Continued

```
int main()
{
   Rectangle b;
   cout << "The Rectangle Area is    " << b.recArea(3.0,4.0)
   cout<<endl;
   cout << “The Rectangle perimeter is "<<b.recfler(3.0,4.0);
   return 0;
}
```

Sample Run

```
The Rectangle Area is   12
The Rectangle perimeter is   14
```