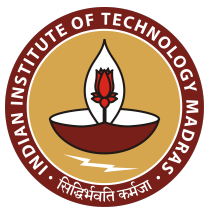
Objects and Classes

Saurav Samantaray

Department of Mathematics

Indian Institute of Technology Madras

August 18, 2024



A simple Class

now, the topics you've all been waiting for: objects and classes

```
//////////////////////////////////////////////////////////////
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; }
  };
//////////////////////////////////////////////////////////////
int main()
```

A Simple Class

- 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 `setdata` sets the data item to a value,
- the second member function `showdata` displays the value

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

  s1.showdata( );     //call member function to display data
```

Classes

Placing data and functions together into a single entity is a central idea in object-oriented programming.

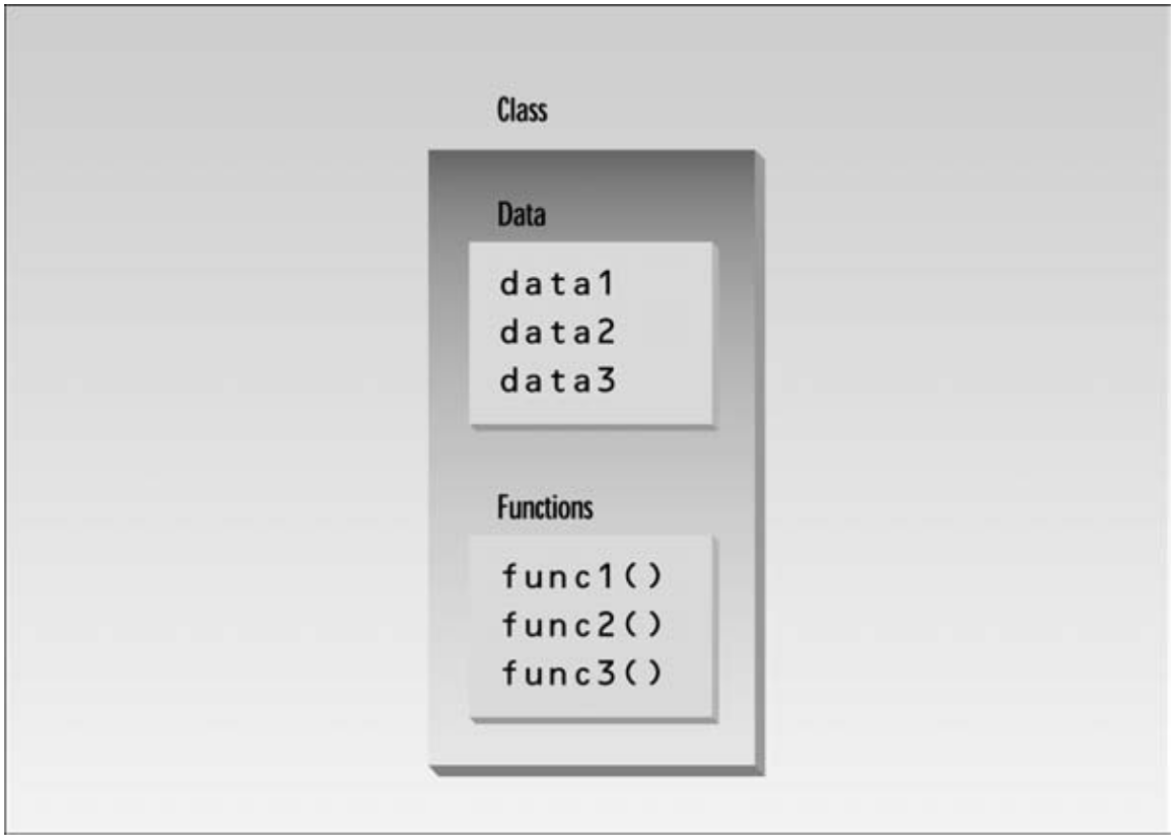


Figure 1: Classes contain data and functions.

Classes and Objects

- an object has the same relationship to a class that a variable has to a data type
- an object is said to be an instance of a class,
- my Macbook pro is an instance of a laptop
- in `SMALLOBJ`, the class—whose name is `smallobj`—is defined in the first part of the program
- later, in `main()`, we define two objects—s1 and s2—that are instances of that class
- the definition starts with the keyword `class`
- followed by the class name-`smallobj` in this example.

private and public

- The body of the class contains two unfamiliar keywords: `private` and `public`.
- What is their purpose?
- A key feature of object-oriented programming is data hiding.
- This term does not refer to the activities of particularly paranoid programmers;
- rather it means that data is concealed within a class so that it cannot be accessed mistakenly by functions outside the class.
- The primary mechanism for hiding data is to put it in a class and make it private.
- Private data or functions can only be accessed from within the class.
- Public data or functions, on the other hand, are accessible from outside the class.
- Programmers who really want to can figure out a way to access private data, but they will find it hard to do so by accident.

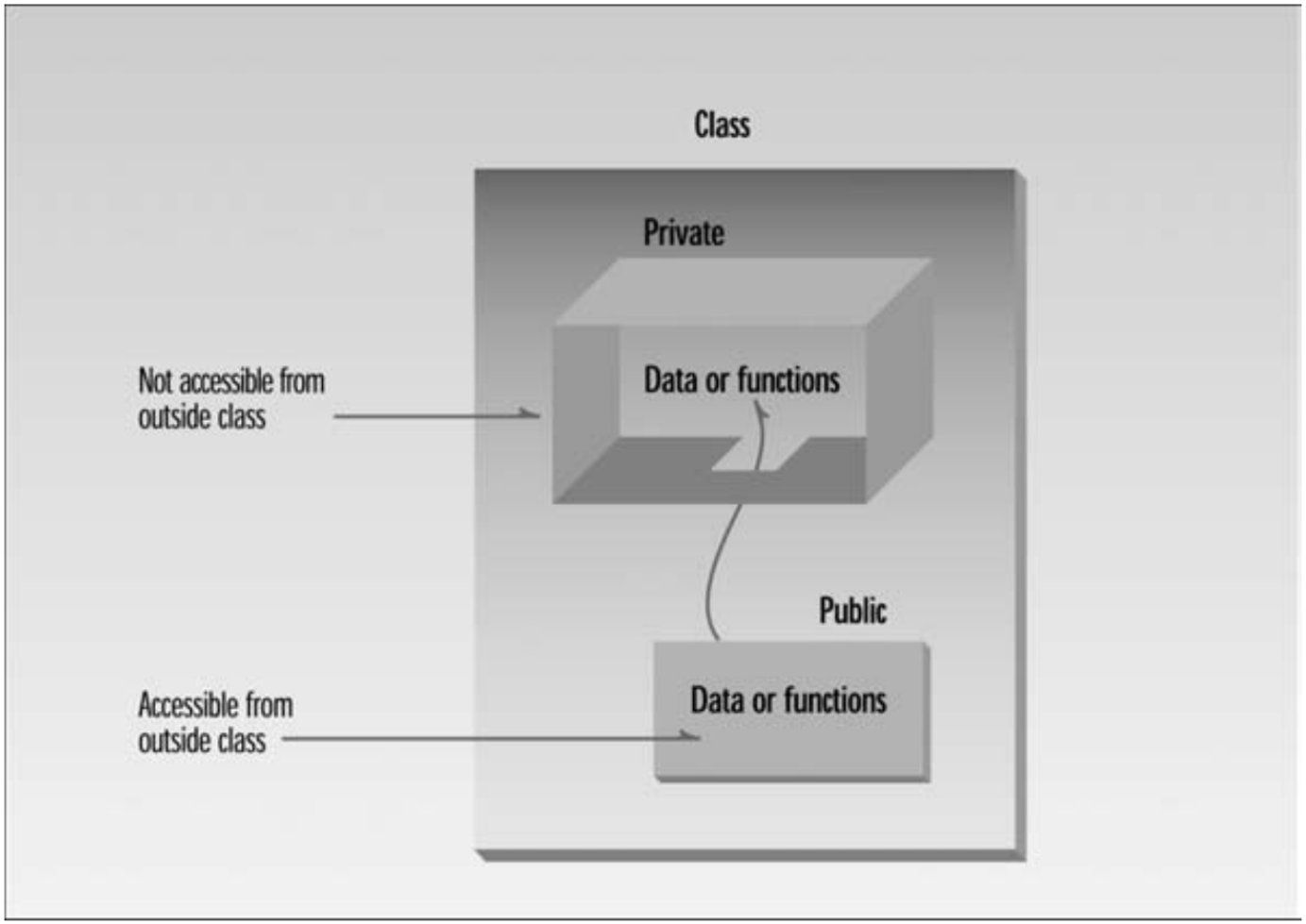


Figure 2: private and public.

Class Data

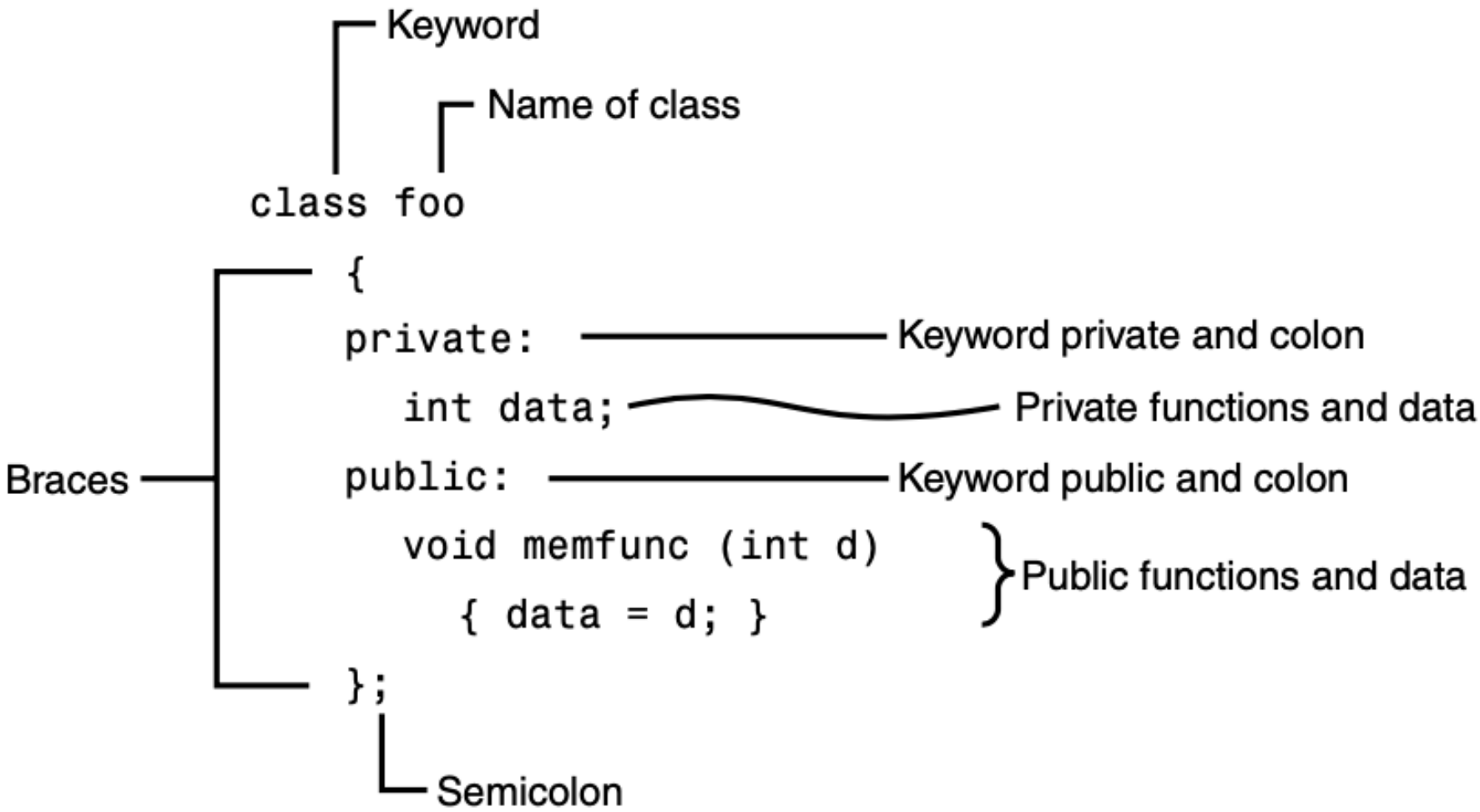
- The `smallobj` class contains one data item: `somedata`, which is of type `int`.
- The data items within a class are called data members (or sometimes member data).
- there can be any number of data members in a class,
- just as there can be any number of data items in a structure
- the data member `somedata` follows the keyword private,
- so it can be accessed from within the class
- but not from outside

Member Functions

- *Member functions* are functions that are included within a class.
- There are two member functions in `smallobj`: `setdata()` and `showdata()`.
- The function bodies of these functions have been written on the same line as the braces that delimit them.
- one could also use the more traditional format for these function definitions
- However, when member functions are small, it is common to compress their definitions this way to save space.
- Because `setdata()` and `showdata()` follow the keyword public, they can be accessed from outside the class.

```
void setdata(int d)
  {
  somedata = d;
  }
void showdata()
  {
  cout << " \nData is " << somedata;
  }
```

Syntax of a class definition



Functions Are Public, Data Is Private

- usually the data within a class is private and the functions are public
- the data is hidden so it will be safe from accidental manipulation
- while the functions that operate on the data are public so they can be accessed from outside the class
- however, there is no rule that says data must be private and functions public;
- in some circumstances one may find the need to use private functions and public data.

Member Functions Within Class Definition

- The member functions in the `smallobj` class perform operations that are quite common in classes:
- setting and retrieving the data stored in the class
- Note that the member functions `setdata()` and `showdata()` are definitions in that the actual code for the function is contained within the class definition.
- The functions are not definitions in the sense that memory is set aside for the function code;
- this doesn't happen until an object of the class is created
- it is also possible to declare a function within a class but to define it elsewhere

Using the Class

- Now that the class is defined, let's see how `main()` makes use of it.
- We'll see how objects are defined, and,
- once defined, how their member functions are accessed.

Defining Objects

- The first statement in `main()`
- `smallobj s1, s2;`
- defines two objects, `s1` and `s2`, of class `smallobj`
- It is objects that participate in program operations.
- Defining an object is similar to defining a variable of any data type: Space is set aside for it in memory.

- Defining objects in this way means creating them (instantiating)
- term instantiating arises, as an instance of the class is created.
- An object is an instance of a class.

Calling Member Functions

- The next two statements in `main()` call the member function `setdata()`:
- ```
  s1.setdata(1066);
  s2.setdata(1776);
  ```

- These statements don't look like normal function calls.
- Why are the object names `s1` and `s2` connected to the function names with a period?
- This strange syntax is used to call a member function that is associated with a specific object.
- Because `setdata()` is a member function of the `smallobj` class,
- it must always be called in connection with an object of this class.
  ```

Using the Class

It doesn't make sense to say `setdata(1066);`

- because a member function is always called to act on a specific object,
- not on the class in general.
- Attempting to access the class this way would be like trying to drive the blueprint of a car.
- Not only does this statement not make sense, but the compiler will issue an error message if you attempt it.

Member functions of a class can be accessed only by an object of that class.

Using the Class

- To use a member function, the dot operator (the period) connects the object name and the member function.
- The syntax is similar to the way we refer to structure members
- but the parentheses signal that we're executing a member function rather than referring to a data item
- The dot operator is also called the class member access operator

The first call to `setdata()`

```
s1.setdata(1066);
```

- executes the `setdata()` member function of the `s1` object.
- This function sets the variable `somedata` in object `s1` to the value 1066.
- similarly the second call does the same for `s2`.

Similarly, the following two calls to the `showdata()` function will cause the two objects to display their values:

```
s1.showdata();
s2.showdata();
```