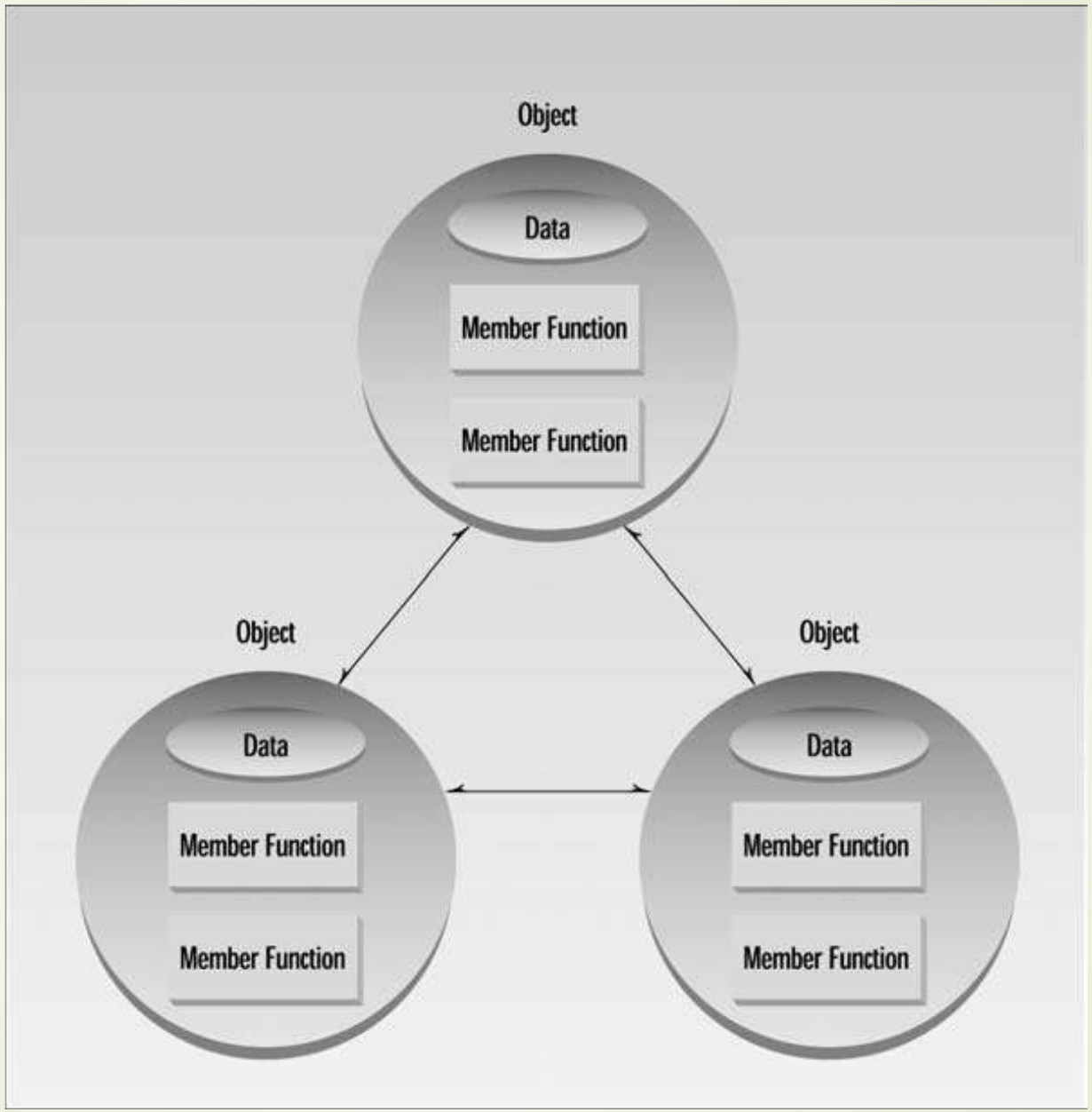
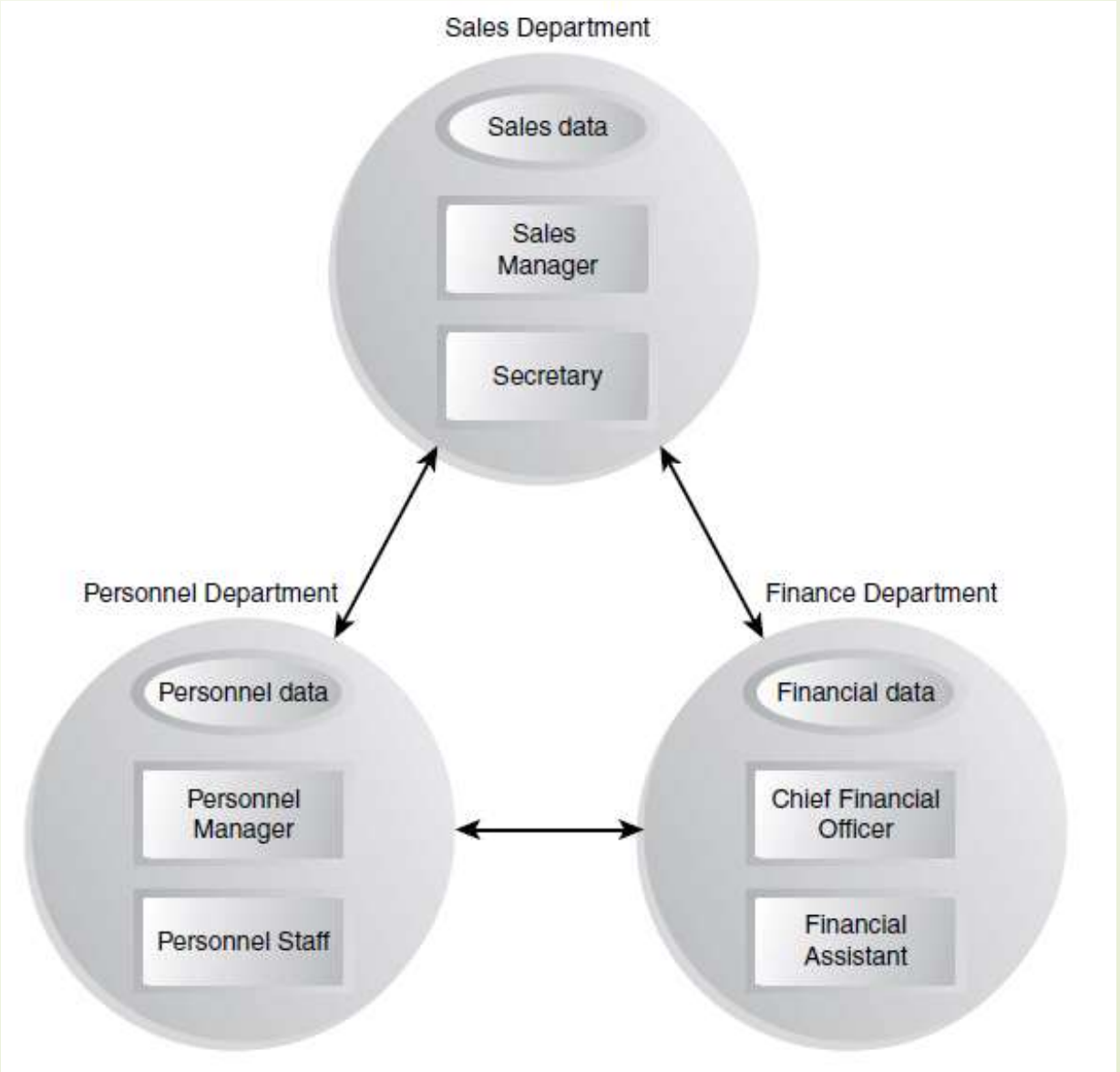
Object Oriented



Example


Sales Department
Sales data
Sales Manager
Secretary
Personnel Department
Finance Department
Personnel data
Personnel Manager
Personnel Staff
Financial data
Chief Financial Officer
Financial Assistant

Information Hiding

- By Information Hiding we mean "*Showing only those details to the outside world which are necessary for the outside world and hiding all other details from the outside world.*"
- Ali's name and other personal information is stored in his brain we can't access this information directly. For getting this information we need to ask Ali about it and it will be up to Ali how much details he would like to share with us.
- *"Hiding the object details (state and behavior) from the users"*
- Object oriented model only had objects and their interactions hiding implementation details
- We can achieve information hiding using **Encapsulation** and **Abstraction**,

Encapsulation

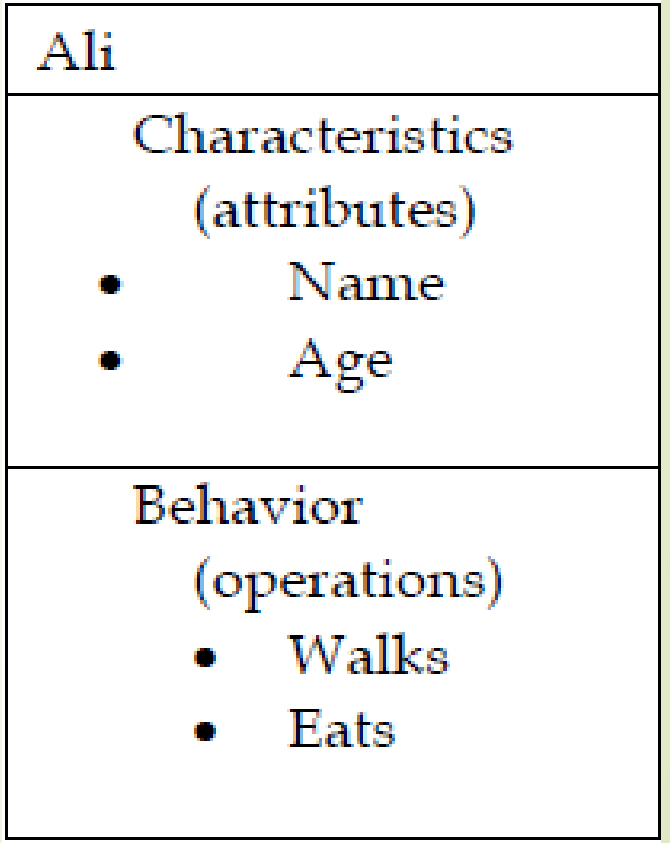
- Encapsulation means *"we have enclosed all the characteristics of an object in the object itself"*
- We have seen in previous lecture that object characteristics include data members and behavior of the object in the form of functions.
- So we can say that Data and Behavior are tightly coupled inside an object
- Consider the same example of object Ali of previous lecture we described it as follows,

Encapsulation

Now it is up to object Ali whether he wants to share that information with outside world or not.

Same thing stands for its behavior if some other object in real life wants to use his behavior of walking it can not use it without the permission of Ali.

Any other object don't know about these things unless Ali share this information with that object through an interface,
Same concept also applies to phone which has some data and behavior of showing that data to user we can only access the information stored in the phone if phone interface allow us to do so.



Interface

- Interface is a set of functions of an object that he wants to expose to other objects.

  Example – Interface of a Car

  - Steer Wheels
  - Accelerate
  - Change Gear
  - Apply Brakes

Interface of an object provides us the list of available functions.

Abstraction

- ***"Capture only those details about an object that are relevant to current perspective"***

Suppose we want to implement abstraction for the following statement,

"Ali is a PhD student and teaches BS students"

*Here object Ali has two **perspectives** one is his **student perspective** and second is his **teacher perspective.***

We can sum up Ali's attributes as follows,

Name

Age

Student Roll No

Year of Study

CGPA

Employee ID

Designation

Salary

Abstraction

- **Similarly we can sum up Ali's behavior as follows,**

  Study

  TakeExam

  PlaySports

  Eat

  DeliverLecture

  Walk

- As was the case with attributes of object Ali, its behavior can also be divided in Ali's student perspective as well as Ali's teacher perspective.

Attributes:

- **Name** - Employee ID
- **Student Roll No** - Designation
- **Year of Study** - Salary
- **CGPA** - Age

Abstraction

Behaviour:

- **Study** - DevelopExam
- **GiveExam** - TakeExam
- **PlaySports** - Eat
- DeliverLecture - Walk

- **Teacher's Perspective**

Attributes:

- **Name** - **Employee ID**
- Student Roll No - **Designation**
- Year of Study - **Salary**
- CGPA - **Age**

Behaviour:

- Study - **DevelopExam**
- GiveExam - **TakeExam**
- PlaySports - Eat
- **DeliverLecture** - Walk

Abstraction

A cat can be viewed with different perspectives

Ordinary Perspective	Surgeon's Perspective
A pet animal with	A being with
Four Legs	A Skeleton
A Tail	Heart

A car can be viewed with different perspectives

Driver's View

Engineer's View

Class

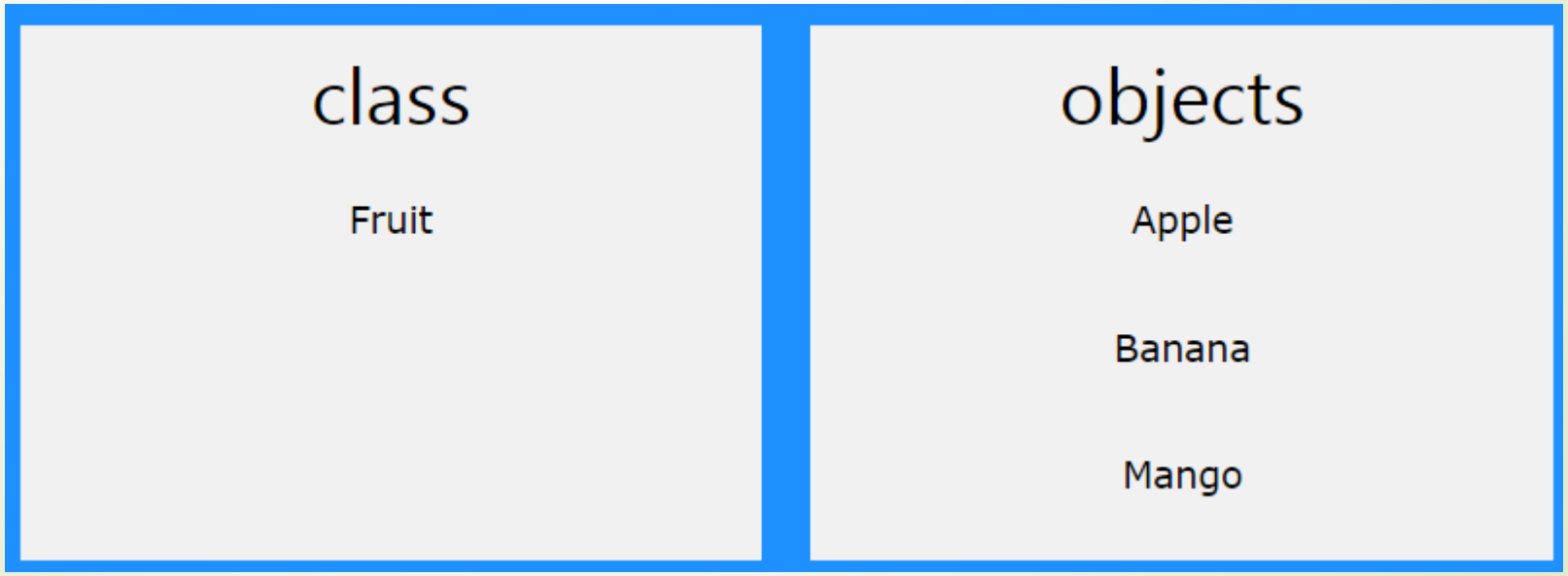
- In OOP we create a general sketch for each kind of objects and then we create different instances using this sketch we call this sketch or prototype or map as “class”.
- All objects of same kind exhibit identical characteristics (information structure and behavior) however they have data of their own.

- Ahsan teaches mathematics
  - Aamir teaches computer science
  - Atif teaches physics

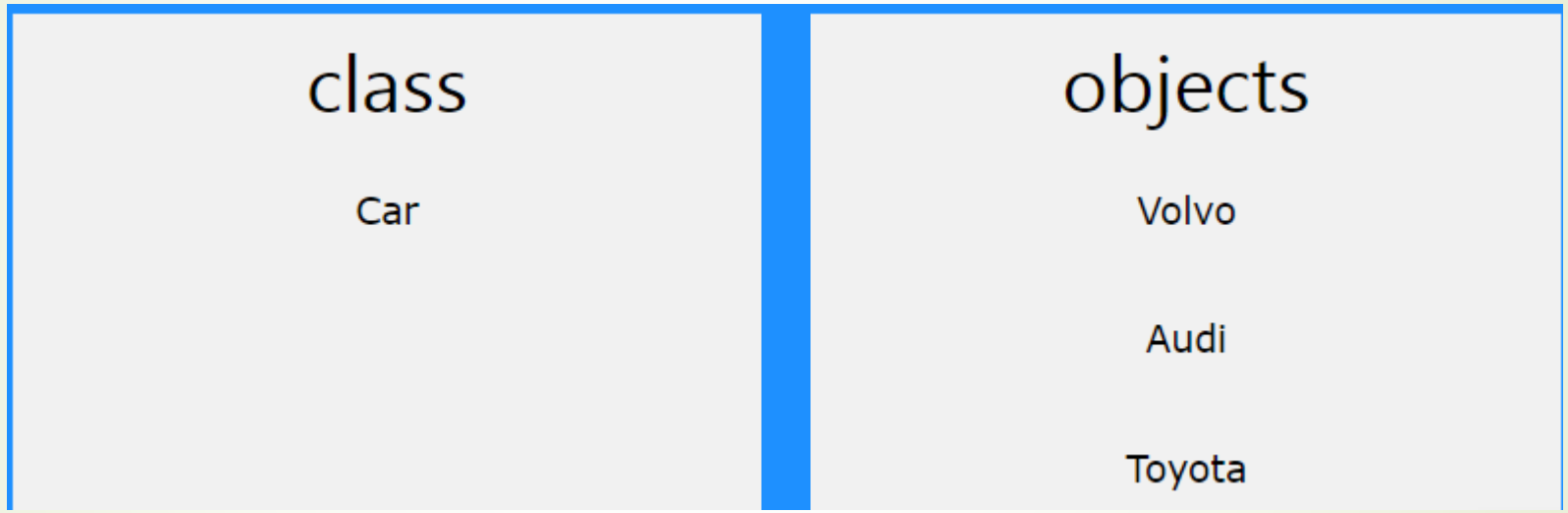
Each one is a teacher so we say these objects are *instances* of the Teacher class

What is a Class

- In OOP we say that objects are members of *classes*.
- you can define many objects of the same class.
- class serves as a plan, or blueprint. It specifies what data and what functions will be included in objects of that class.



What is a Class



So, a class is a template for objects, and an object is an instance of a class. When the individual objects are created, they inherit all the variables and functions from the class

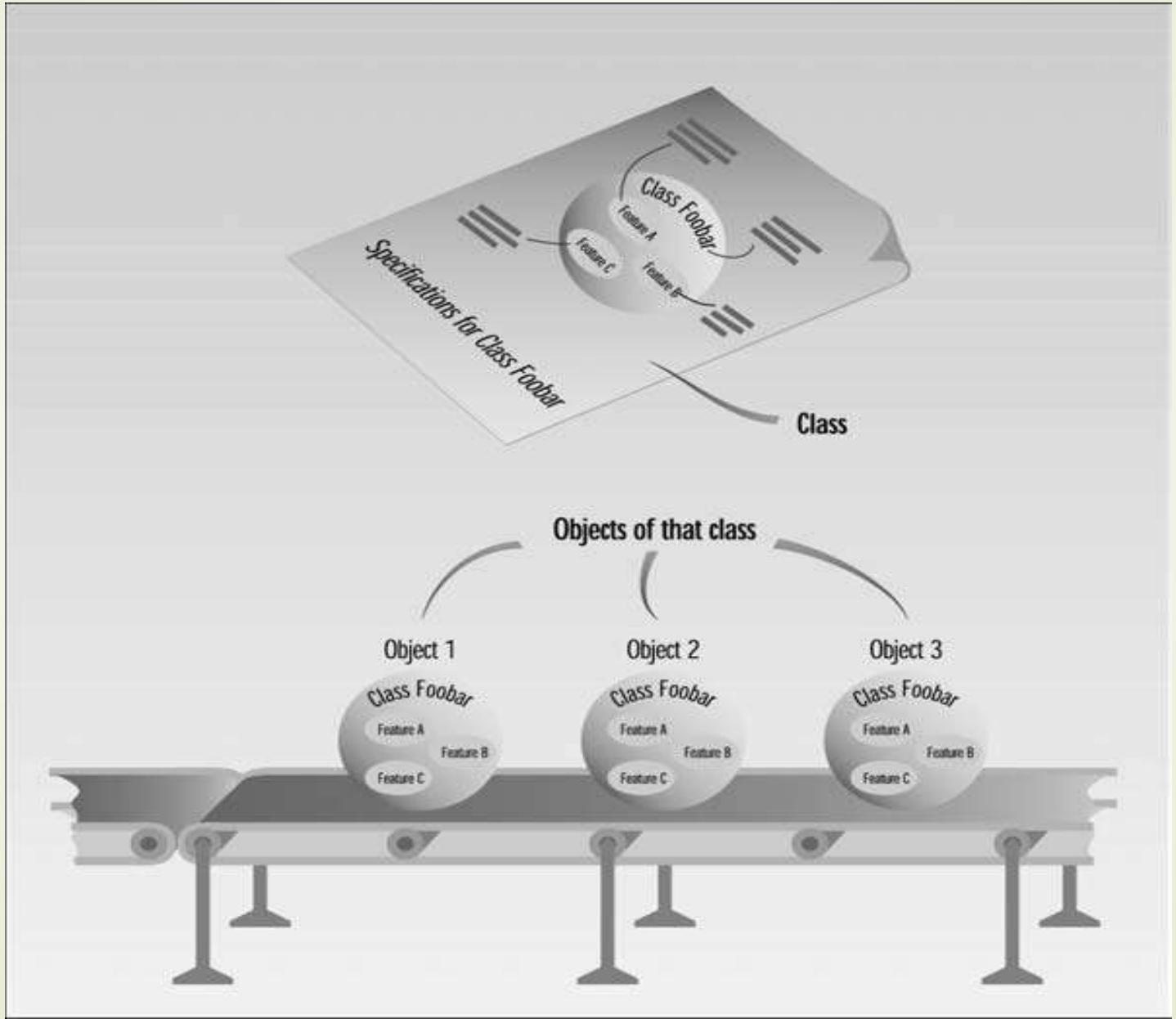
What is a Class

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

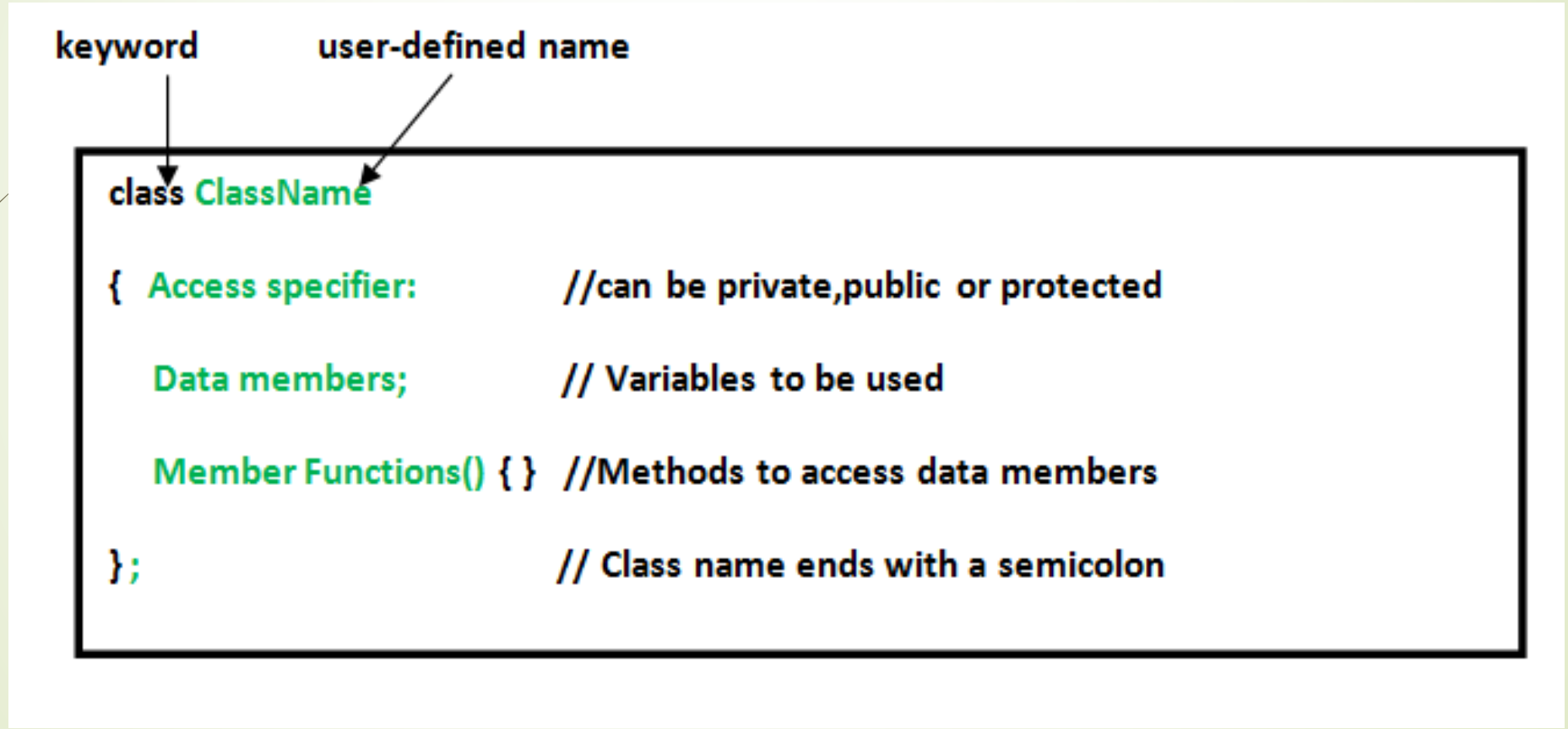
Private/public

What is a Class



What is a Class

An **Object** is an instance of a Class. When a class is defined, no memory is allocated but when it is instantiated (i.e. an object is created) memory is allocated.



Create a class

```
class MyClass {       // The class
  public:             // Access specifier
    int myNum;        // Attribute (int variable)
    string myString;  // Attribute (string variable)
};
```

•The `class` keyword is used to create a class called `MyClass`.
•The `public` keyword is an **access specifier**, which specifies that members (attributes and methods) of the class are accessible from outside the class. You will learn more about access specifiers later.
•Inside the class, there is an integer variable `myNum` and a string variable `myString`. When variables are declared within a class, they are called **attributes**.
•At last, end the class definition with a semicolon `;`.

Create an Object

```
#include <iostream>
#include <string>
using namespace std;
class MyClass {       // The class
  private:            // Access specifier
    int myNum;        // Attribute (int variable)
    string myString;  // Attribute (string variable)
};
int main() {
  MyClass myObj;  // Create an object of MyClass
myObj.myNum = 15;
  myObj.myString = "Some text";
cout << myObj.myNum << "\n";
  cout << myObj.myString;
  return 0;
}
```

state vs behavior

- State simply means data or value. E.g dog breed, color
- Behavior means action or work or task or operation that the object does e.g bark

Methods/functions

- Methods are functions that belongs to the class.
- There are two ways to define functions that belongs to a class:
  - Inside class definition
  - Outside class definition

Methods/functions

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

class MyClass {        // The class
  public:              // Access specifier
    void myMethod() {   // Method/function
      cout << "Hello World!";
    }
};

int main() {
  MyClass myObj;     // Create an object of MyClass
  myObj.myMethod();  // Call the method
  return 0;
}
```