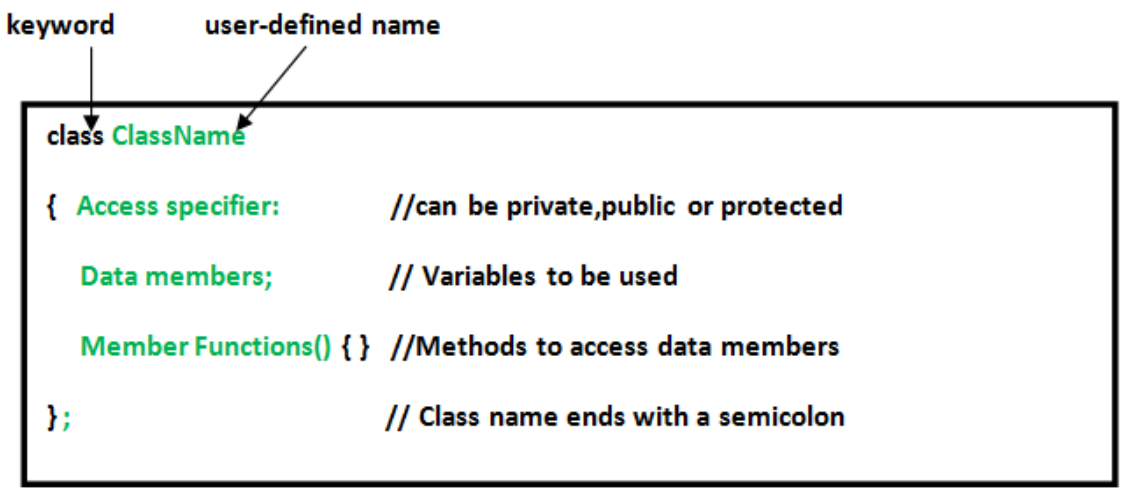
2.OBJECTS AND CLASSES

CLASSES & OBJECTS

- It is a user-defined data type, which holds its own data members and member functions, which can be accessed and used by creating an instance of that class.
- A C++ class is like a blueprint for an object.
- 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.

Defining Class

- A class is defined in C++ using keyword class followed by the name of class. The body of class is defined inside the curly brackets and terminated by a semicolon at the end.



Declaring Objects

- **Declaring Objects:** When a class is defined, only the specification for the object is defined; no memory or storage is allocated. To use the data and access functions defined in the class, you need to create objects.
- **Syntax:** ClassName ObjectName;

- **Accessing data members and member functions**: The data members and member functions of class can be accessed using the dot('.') operator with the object. For example if the name of object is *obj* and you want to access the member function with the name *printName()* then you will have to write *obj.printName()* .

Member Functions in Classes

- There are 2 ways to define a member function:

1. Inside class definition:

   If we define the function inside class then we don't not need to declare it first, we can directly define the function.

2. Outside class definition

- To define a member function outside the class definition we have to use the scope resolution :: operator along with class name and function name.

Member Functions in Classes

- Outside class definition:
- Syntax:

```
return_type class_name::function_name (argument declaration)
{
   Function body
}
```

- The membership label class_name:: tells the compiler that function *function_name* belongs to the class *class_name*. That is, scope of the function is restricted to the class_name specified in the header line.

Arrays within a class

- Arrays can be declared as the members of a class.
- The arrays can be declared as private, public or protected members of the class.

Access Specifiers

- Access specifiers in C++ class defines the access control rules.
- C++ has 3 access specifiers,

  public

  private

  protected
- These access specifiers are used to set boundaries for availability of members of class be it data members or member function.

Access Specifier

- Public: Public, means all the class members declared under public will be available to everyone.
- The data members and member functions declared public can be accessed by other classes too.
- class PublicAccess

```
{
 public:    //public access specifier
 int Y;    //Data member declaration
 void display(); //Member function declaration
}
```

Access Specifier

- Private: Private keyword, means that no one can access the class members declared private outside that class.
- If someone tries to access the private member, they will get a compile time error.
- By default class variables and member function are private.
- class PrivateAccess

```
{
 private:      //private access specifier
 int Y;    //Data member Declaration
 void display(); //Member function declaration
```

Access Specifier

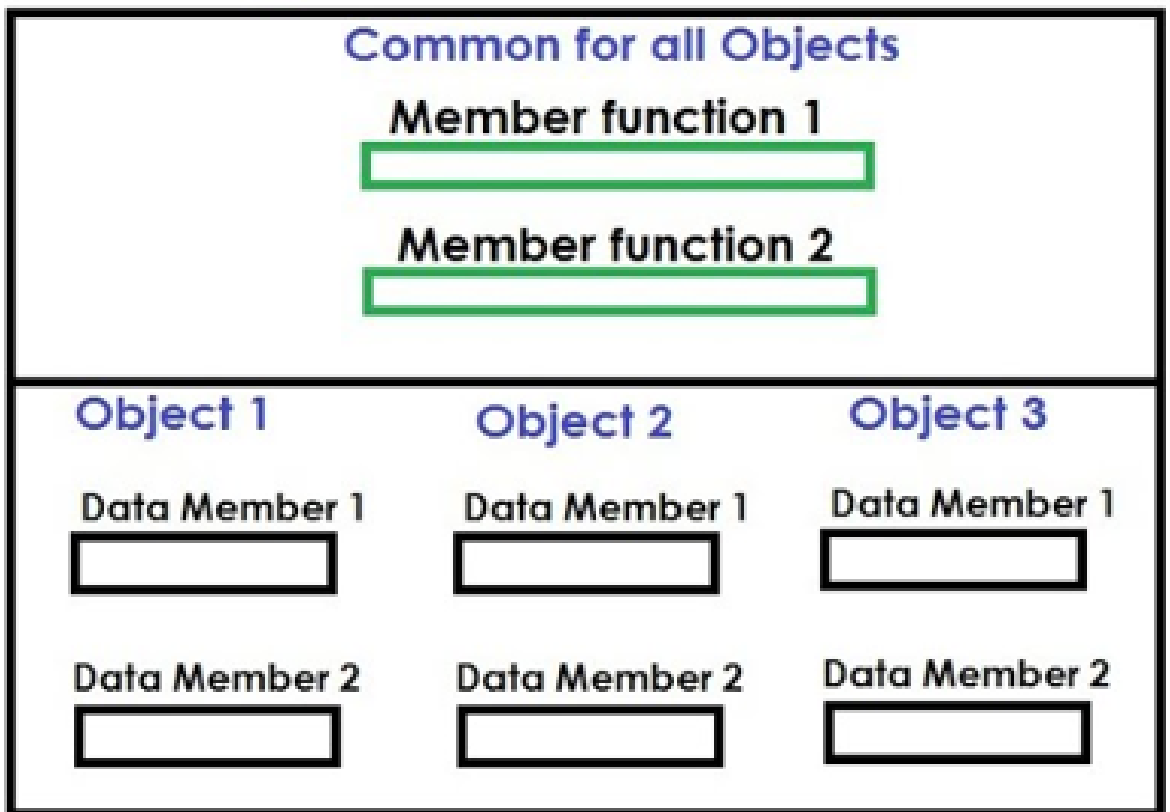
- Protected: It is similar to private, it makes class member inaccessible outside the class. But they can be accessed by any subclass of that class. (If class A is inherited by class B, then class B is subclass of class A.)
- class ProtectedAccess

```
{
 protected:    //protected access specifier
 int Y;    //Data member declaration
 void display(); //Member Function Declaration
}
```

Memory Allocation for Objects

- The memory space for objects is allocated when they are declared and not when the class is specified.
- Actually, the member functions are created and placed in the memory space only once when they are defined as a part of a class specification.
- All the objects belonging to that class use the same member functions, no separate space is allocated for member functions when the objects are created.
- Only space for member variables is allocated separately for each object.
- Seperate memory locations for the objects are essential, because the member variables will hold different data values for different objects.

Memory Allocation for Objects



Static Data Members

- A data member of a class can be made as static.
- A static member variable has certain special characteristics.

1. **It is initialized to zero** when the first object of its class is created. No other initialization is permitted.
2. **Only one copy of that member is created for the entire class and is shared by all the objects** of that class, no matter how many objects are created.
3. It is visible only within the class, but its lifetime is the **entire program.**

Static Data Members

- Declaration Syntax:

  static type variable_name;
- Definition Syntax:

  type class_name :: variable_name;

- Each static variable must be defined outside the class definition.

Static member functions

- A member function that is declared static has the following properties:

1. A static function can have access to only other static members (functions or variables) declared in the same class.
2. A static member function can be called using the classname instead of its objects as follow:

class_name :: function_name;

Friend Function

- C++ allows the common function to be made friendly with the classes, by allowing the function to have access to private data of that class.
- Such function need not be a member of any of these classes.
- The function that are declared with the keyword friend are known as friend functions.
- Syntax:

  friend data_type function_name(class_name object_name);
- Example:

```
class ABC{
public:
friend void add(sample s);
};
```

Friend Function

- The function declaration should be preceded by the keyword friend.
- The function definition does not use either the keyword friend or the scope resolution operator ::
- The friend function has full access rights to the private members of the class.
- Friend function can be declared as private or public.

Characteristics of Friend Function

- It is not in the scope of the class to which it has been declared as friend. Since it is not in the scope of the class.
- It can not be called using the objects of that class.
- It can be invoked like a normal function without the help of any object.
- Unlike member functions, it cannot access the member names directly and has to use an object name and dot membership operator with each member name.
- It can be declared either in the public or private. Part of a class without affecting its meaning.
- Usually, it has the objects as arguments.

Arrays of Objects

- Array of variables that are of type class are called array of objects.
- Syntax:

  classname objectname[size];