Class and Class Members

Classes are the building blocks of Object-Oriented programming. It is a user-defined data-type which contains data members carrying values that describe the characteristics of the entity the class represents. It contains member methods/functions which describe the behavior of the class-entity. Note that the **class is just a blueprint** that has no physical memory associated with it. **When a class is instantiated, an object is created** which is provided a memory/storage location.

Class Definition and Object Instantiation

The syntax for class declaration is as:

```
class < class-name > {
< access-modifier >:
    //data-members
    //constructors
    //member-functions
    //destructors
};   //don't forget the semi-colon here !!
```

Each of the above terms (constructors, destructors, access-modifiers etc.) will be explained in detail afterwards.

```
#include <bits/stdc++.h>
using namespace std;

class Employee {  //Class Declaration
    public:
        string id, name;
        int years;  //experience (in years)

        Employee(string id, string name, int years) {
            this->id = id;
            this->name = name;
            this->years = years;
        }
        void work() {
            cout << "Employee: " << this->id << " is working\
        }
};

int main()
{
    //Class Instantiation (Direct)
    Employee emp("GFG123", "John", 3);

    //Class Instantiation (Indirect)
    Employee *emp_ptr = new Employee("GFG456", "James", 4);

    cout << "Employee ID: " << emp.id << endl;
    cout << "Name: " << emp.name << endl;
    cout << "Experience (in years): " << emp.years << endl;

    emp.work();
    cout << endl;

    cout << "Employee ID: " << emp_ptr->id << endl;
    cout << "Name: " << emp_ptr->name << endl;
    cout << "Experience (in years): " << emp_ptr->years << en

    emp_ptr->work();
    return 0;
}
```

Output:

```
 Employee ID: GFG123
Name: John
Experience (in years): 3
Employee: GFG123 is working

Employee ID: GFG456
Name: James
Experience (in years): 4
Employee: GFG456 is working
```

NOTE:

Don't bother about **this** and other keywords which might as of now seem to be unexplained, as they require an additional whole article to explain. We shall cover each of them in detail afterwards. The statement *Employee emp("GFG123", "John", 3)* inside the *main()* is the object instantiation statement. It is a direct-instantiation of the class. i.e. emp itself is the object stored in RAM, (much like declaring an int, float, struct, etc.). Upon execution of the statement, storage is allocated according to the size of the class and thereafter the constructor - Employee(...) inside the class is run to initialize the data members. We shall study in depth more object initialization formats (there are multiple ways to do so), and more about constructors in later sections.

The statement *Employee *emp_ptr = new Employee("GFG456", "James", 4)* is the indirect way of instantiating class-objects. Here, we use a pointer to point to the object created. We use the **new** keyword and refer to the member entities using → operator.

Data Members

Data Members are the identifiers that describe the characteristics and hold important data related to the class. In our Employee class, we have *id*, *name*, *years* as the data members. Data members can be anything from primitive data-types (int, char, double, etc.) to collections (arrays, strings, etc.) to user-defined data-types (structs, unions, even other class objects, etc.). Anything that can hold data value can be a data-member.

Member Functions

Member Functions/Methods are class-functions/methods which describe the behavior of the class. i.e. what kind of operation we can perform with objects of this class. e.g. In our Employee example, *work()* is a member function. Member functions in C++ can be defined inside/outside the class, however, it is mandatory to have prototype declaration inside the class if go for an outside-the-class definition. e.g.

```
#include <bits/stdc++.h>
using namespace std;

class Employee {  //Class Declaration
    public:
        string id, name;
        int years;  //experience (in years)

        Employee(string id, string name, int years) {
            this->id = id;
            this->name = name;
            this->years = years;
        }

        //Prototype Declaration
        void work();
};

//Outside-class definition
void Employee::work() {
    cout << "Employee: " << this->id << " is working\n";
}

int main()
{
    //Class Instantiation (Direct)
    Employee emp("GFG123","John",3);

    emp.work();
    return 0;
}
```

Output:

```
Employee: GFG123 is working
```

As we observe, while providing the definition for a class member-function we use the **scope-resolution operator ::** with the **class-name::function-name** format to differentiate it from normal global-scoped functions.