Introduction To Classes

In this lesson, we will be learning about Classes using object-oriented Methodology.

We'll cover the following

Definition

`Classes` are the building blocks of programs built using the object-oriented methodology. Such programs consist of *independent*, *self-managing modules* and their *interactions*. An object is an *instance* of such module, and a class is its definition.

Body of Class

```
class className
{
  variable var1           // variables names can be of type string, int, float
  variable var2
};              // class body always terminates with ';'
```

A *Keyword* `class` is used with every *declaration* of class followed by the name of the class. You can use any *className* as you want.

Example of Dog

```
class Dog
{
public:
    char name[25];
    string gender;
    int age;
```

```
    int size;
    bool healthy;
};

int main(){
  Dog dogObj;     // creating an object of Dog class called DogObj
  //using the dot operator to access members of a class
  dogObj.gender;  //using object dogObj to access certain accessible variables of the class
}
```

Explanation

A dog has several *member* variables listed such a:

- name,
- the `gender` of the dog,
- the `age` of the dog,
- its `size`,
- whether it is `healthy` or not.

These variables are called *properties* declared inside the class.

An instance of a *dog*, say, a dog named **Lucy**, would be an **object**. So would a dog named **Ruffy**. Hence, you can have *multiple* instances of a *class*, just like you can have *multiple* dogs.

Properties are like **"inner variables"** of each *object* made of type `Dog`. We used the **dot** operator to access members of a class *object*.

Private Members

As you can see above, we have used the word `public` before *declaring* the class members. The reason being:

> C++ restricts the program from directly referencing the member variables.

By default, all *members* declared inside a class are considered `private`. Which means:

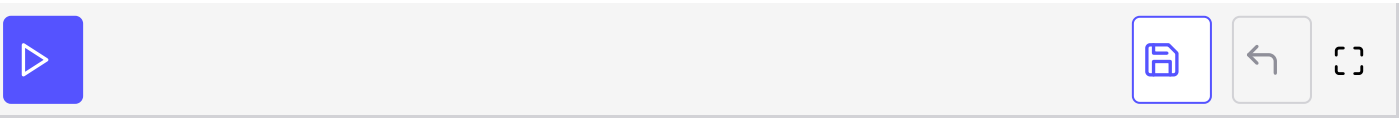
- they can only be referenced within the definitions of member functions
    - If a program tried to access `private` *variables* directly it will get a **compiler error**.

> **Note:** Private members can be *variables* or *functions*.

Try running the code below. It will give an error when you try to compile it as in the code private members of the class are being accessed directly.

```
#include <iostream>
using namespace std;
class Dog
{
private: // these attribute of class are not available in other functions and classes
    char name[25];
    string gender;
    int age;
    int size;
    bool healthy;
};


int main() {
  Dog dogObj;  //making object of Dog class
  dogObj.name;  //this will give an error as data members are private
}
```



Private Variables Access Error

Public Members

The keyword `public` identifies members of a class that can be accessed from outside of the class.

- Members that follow the keyword `public` are public members of the class.

```
#include <iostream>
using namespace std;
class Dog
{
public:  //these attribute of class are available in other functions and classes
    string name = "lucy";
    string gender  = "female";
    int age = 5;
    int size = 5;
    bool healthy = true;
};


int main() {
  Dog dogObj;     //making object of Dog class
  cout << "Dog name is: "<<dogObj.name<<endl;     //by using . operator we can access the member o
  cout << "Dog gender is: "<<dogObj.gender<<endl;    //accessing the public members of class Dog i
  cout << "Dog age is: "<<dogObj.age<<endl;
  cout << "Dog size is: "<<dogObj.size<<endl;
```

```
  cout << "Is dog healthy: " <<dogObj.healthy<<endl;
}
```

Accessing Public Members

In the next lesson, we will look into further details of classes and member functions in a class as well.