Static Members

By Avani M. Sakhapara
Asst. Professor,IT Dept
KJSCE

Non-static (ordinary) member variables

- Regular member variables of a class exist in every object.
- That is, when you declare a class and list the member variables, you are saying that every object in the class should have its own space in memory for each member variable.

Example

```
class Thing {
    public:
        int x;
        int y;
                };
int main()
{
Thing t1, t2;
……}
```

Static member variables

- Sometimes it is handy to have a member variable that is associated with the whole class instead of individual objects.
- Such a variable occupies its own piece of memory, by itself, instead of being part of an individual object.
- keyword "static" is used before the member variables declaration, so we have "static member variables”.

Characteristics of a static member variable

- It is initialized to zero when the first object of its class is created.
- **Only one copy** of that member is created for the entire class and is **shared by all the objects** of that class.
- It is visible only within the class, but its lifetime is the entire program.

Example

```
class Thing {
        public:
                int x;
                int y;
                static int count;
                Thing ()
                {count++;} // there is now one more Thing
                ~Thing ()
                {count--;} // there is now one fewer Thing
};
```

Definition and Initialization of static member variable

- You have to define and initialize a static member variable somewhere in your code, at the top level (outside of any function)
- In this definition and initialization line, you provide the full name of the static member variable using the scope resolution operator.
- **Eg:** int Thing::count = 0; // define the static member variable

Accessing a static member variable

- At any point in your code, you can refer to a static member variable either in the context of an object, or just with the scope resolution operator
- **Eg:** Thing t;...

```
int x;
x = t.count; // get the count using object
x = Thing::count; // get the count using
                  class name
```

Static member functions

- A static member function is like an ordinary function preceded by the keyword **static.**
- A static member function can access only other static members(functions or variables) declared in the same class.

Accessing a static member function

- At any point in your code, you can refer to a static member function either in the context of an object, or just with the scope resolution operator.

Static member function Example

```
class Thing {
private:
static int count;

public:
        Thing ()
        {count++;}
        ~Thing ()
        {count--;}
        static int get_count()
{return count;}
};
```

```
//defining static member variable
int Thing::count;
int main(){
Thing t;
...

// get the count
x = t.get_count();

// get the count
x = Thing::get_count();
return 0;
}
```