**Static Data Fields/Static Methods**

**Overview**

Consider the following class declaration:

#ifndef EMPLOYEE\_H

#define EMPLOYEE\_H

class Employee

{

private:

int id;

int salary;

public:

Employee(int salary);

~Employee();

int getId();

int getSalary();

};

#endif

We want to **automatically** increment the *id* field each time an object is created:

Employee emp1(23000); // we want id = 0

Employee emp2(33000); //we want id = 1

Employee emp3(43000); //we want id = 2;

. . .

Employee empn(19000); //We want id = n-1

How can we do this?

//Could we implement the constructor like this?

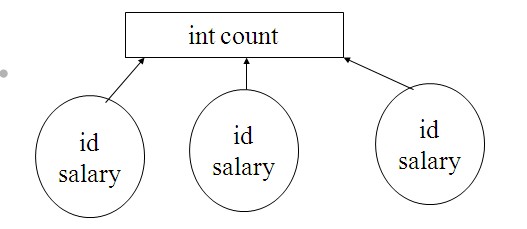
Employee::Employee(int salary)

{

id++; //How do we initialize id? - **Won’t work**

**…..**

}

Would it be useful in this case that there be a **global** data field that **all** objects could access? The global data field (e.g., *count*) would store how many objects have been created.

**Static Data Fields**

Static Data Fields are like the global data fields mentioned above. They are declared as follows:

---------------------------

//**Step 1** - Declaring Static Data Fields

class Employee

{

private:

int id;

int salary;

**static int count; //Only one storage location for every obj**

………..

Static Data Fields require an initialization step (usually done in .cpp file). **Conventional data fields do not require this step.**

**-----------------------------------**

//**Step 2**- Initialization of Static Data Fields

int Employee::count = 0; //Notice Scope oper! - Top of Employee.cpp **before** methods

Static Data fields are created and initialized **only once** for all objects.

Now consider the following Employee Constructor:

Employee::Employee(int salary)

{

//Set salary

this->salary = salary;

id = count++; //count gets incremented each time an

//Employee object is created

}

**Static Methods**

I can also create static methods for a class.

Suppose I wanted to declare a method called **getCount()** as static:

//**Step 1** - Declaring Static Method

class Employee

{

private:

int id;

int salary;

static int count;

public:

Employee(int salary);

~Employee();

int getId();

int getSalary();

**static int getCount(); //Declaration of static method**

};

------------------------

//**Step 2** - Implementing Static Method

**int Employee::getCount()**

{

**//Cannot access non-static private data fields**

**//return (id) - this won't work**

return count; //Can only access static data fields

}

//Notice that static qualifier not used in method header!

Cannot access non-static pdf because static methods can exist **before** any objects have been declared!

There is no implicit *this* pointer associated with static methods

Syntax for invoking a static method

ClassName::methodName(); //methodName must be declared as static

Employee::getCount();