Tic-Tac-Toe Project

- Discussion of project implementation…

Random related comments:

- It is perfectly OK to delete a null-pointer
- It is NOT OK to call delete twice on the same address!
- There is NO built-in implementation of a 2-dimensional dynamically allocated array

Inline functions

For efficiency, sometimes declare a function as “inline”:

```
inline void foo()
{
  …
}
```

- Request for compiler to paste individual copies of this code in each place it is called.
- Compiler will decide whether to honor the request
- Can appear in declaration or definition

Static Functions

Principly the same as Java:

- Part of a class
- Called without an invoking “current object”
- Using “this” makes no sense
- No access to data members

Slightly different calling syntax:

Java: Class.staticMethod();

C++: Class::staticMethod();

C++ can differ from Java on “where things go”:

- You can declare and define them in a class (like Java)
- More typically, the declaration goes in the class declaration in the .h file and the implementation goes in the .cpp file

Static Data Members

In principal, same as Java:

- Part of a class
- One copy shared by everyone

Declared/defined very differently:

Java: Usually declaration and initialization go together inside a class, or could be initialized in a static initialization block.

C++:

- Declared in the class declaration (in a .h file)
- Defined and Initialized outside class declaration (in a .cpp file)

Why? We have to **declare** the variable in the .h file so that other modules can see the declaration. If actually **defined** in the header file, it would be defined repeatedly for each .cpp file includes it. (The ifndef trick only prevents duplicate inclusions when compiling a single .cpp file.)

Example: Tickets

Concept: Each Ticket object must have a unique ID number.

Illustrates:

- Static functions
- Static data members
  - Declaration
  - Definition

Nested Classes

We are **not** talking about nested objects. We mean:

```
Class A
{
   Class B
   {
   }
}
```

- B is **not** an inner class like Java
  - An instance of B does **not** have access to an instance of A's members
  - A and B **cannot** see each other's private members
- B is like a **static** nested class in Java
- Why do this? Normally, the inner class is made private, so that it can only be used inside the outer class
- Why didn't I do this on the project with the Node class? I should have!

Class Weirdness

Java allows “anonymous” classes, which are weird. C++ does not.

C++ allows a class definition inside a function, which is weird. Java does not.

struct vs. class

Surprise!

In C++, **`struct`** and **`class`** are the same thing.

The only difference between `class` and `struct` is:

- Members in a `struct` default to **public**
- Members in a `class` default to **private**

(Note, a `struct` in C is not the same as a `struct` in C++, but C++ is backwards compatible in this regard.)

Age-Old Problem

File Chicken.h:

```
#include "Egg.h"
Class Chicken
{
   Egg *x;
}
```

File Egg.h:

```
#include "Chicken.h"
Class Egg
{
   Chicken *y;
}
```

You can't compile Chicken without seeing declaration for Egg.
You can't compile Egg without seeing declaration for Chicken.

How can we resolve this?

Incomplete Class Declaration

File Chicken.h:

```
Class Egg;        // "Incomplete" class declaration
Class Chicken
{
  Egg *x;
}
```

File Egg.h:

```
#include "Chicken.h"
Class Egg
{
  Chicken *y;
}
```

Won't help if the Chicken class needs to access **members** of the Egg!