CS 247: Software Engineering Principles

Namespaces

Reading: Eckel, Vol. 1
Ch. 10 Name Control: Namespaces

Namespaces

Namespaces are used to package together related classes, functions, and types in a common named scope.

```
namespace myLib {

   void f();
   ...
}
```

```
namespace myLib2 {

   void f();
   void g() { f(); }          // local name
   void h() { myLib::f(); } //nonlocal name
}
```

```
namespace myLib {
   void g();
   ...
}
```

Rational Example

Best Practice: put related types, classes, functions together in the same namespace (e.g., ADT + nonmember functions)

```
namespace RatADT {

   class Rational {
   public:
     Rational (int numer = 0, int denom = 1);
      ...
   };

   Rational operator+ (const Rational&, const Rational&);
   Rational operator* (const Rational&, const Rational&);

   bool operator== (const Rational&, const Rational&);
   bool operator!= (const Rational&, const Rational&);

   std::ostream& operator<< (std::ostream &sout, const Rational &r);
   std::istream& operator>> (std::istream &sin,  Rational &s);
   ...
}
```

Global Namespace

global namespace is implicitly declared in every program and includes all names defined at the global scope.

Each file that defines entities at the global scope adds those names to the global namespace

```
:: member_name  // refers to member of global namespace
```

Unnamed Namespace

An unnamed namespace is used to declare a local namespace for file-scope names.

```
namespace
{
    void f();
}
```

Referencing Namespace Members

There are constructs for allowing nonlocal namespace members to be referenced as if they were local

1) Using declaration: makes one name on par with local names

```
using std::cout;
```

- Conflicts with local declarations of the same name

2) Using directive: makes all of the names visible

```
using namespace std;
```

- Allows multiple occurrences of a name (can be disambiguated with scope qualification)
- Superseded by local declarations of the same name

Namespace Etiquette

Never place a using directive (`using namespace X`)...

Inside of a header file

- header could be included in any number of other files
- affects the set of names visible in those files

Before an `#include` directive in an implementation file

- affects the set of names visible in the subsequent header