Namespaces

Jim Fawcett

CSE687 – Object Oriented Design

Spring 2017

Namespaces

- Namespaces help to avoid naming collisions when software is developed by teams.
- A namespace groups a collection of identifiers in a named scope:

```
namespace CSE687 {
  class tree; class node; void walk(tree*);
}  // note: no semicolon

CSE687::tree myTree;
CSE687::walk(&CSE687::myTree);
```

Extensions

- Unlike classes, namespaces are always open for extension:

```
namespace CSE687 { void anotherFunction(); }
```

- Koenig Lookup:
  You don't have to use the namespace qualifier for functions if one or more of the function's argument types are defined in the namespace of the function. This

```
CSE687::walk(&CSE687::myTree)
```

  could have been written as

```
walk(&CSE687::myTree)
```

Rules of Syntax

- Namespace definitions can only appear at global scope, but namespaces can be nested.
- A namespace name can be aliased, to allow you to shorten a long, unwieldy name, e.g.:

```
namespace shorter = veryLongNamespaceName;
```

Using

- Using declaration:
  With a using declaration you can avoid the need for a qualifier for any type or function defined in a namespace:

```
using CSE687::node;
node myNode;
```

- Using directive:
  A using directive makes all names of a namespace available without qualification:

```
using namespace CSE687;
node myNode;
tree myTree;
```

Caution:

- Never use using declarations or directives in header files or in any scope that others will use. Otherwise you force the change in scope on them, which is always inappropriate.

End of Presentation