A using-declaration adds a particular name to the declarative region in which it occurs. For example, a using-declaration of `Jill::fetch` in `main()` adds `fetch` to the declarative region defined by `main()`. After making this declaration, you can use the name `fetch` instead of `Jill::fetch`.

```cpp
namespace Jill {
    double bucket(double n) { ... }  
    double fetch;                  
    struct Hill { ... };            
}
char fetch;
int main()
{
    using Jill::fetch;             // put fetch into local namespace
    double fetch;                  // Error! Already have a local fetch
    cin >> fetch;                  // read a value into Jill::fetch
    cin >> ::fetch;                // read a value into global fetch
    ...
}
```

Because a using-declaration adds the name to the local declarative region, this example precludes creating another local variable by the name of `fetch`. Also, like any other local variable, `fetch` would override a global variable by the same name.

Placing a using-declaration at the external level adds the name to the global namespace:

```cpp
void other();  
namespace Jill {
    double bucket(double n) { ... }  
    double fetch;                  
    struct Hill { ... };            
}
using Jill::fetch;                  // put fetch into global namespace
int main()
{
    cin >> fetch;                  // read a value into Jill::fetch
    other()
```