Namespaces, friends & smart pointers

Taller de Programación I - FIUBA

Namespaces

```
namespace Graphics {
  namespace Effects {
     class Animation { ... };
  }
}

namespace Graphics {
  class Vertex { ... };
  class Line { ... };
}
```

Permite agrupar funciones, clases, tipos y otros elementos bajo un cierto nombre.

Es una agrupación lógica, independiente de la ubicación física de los elementos al momento de compilación.

A diferencia de una clase, un **namespace** puede ser extendido.

Namespaces

```
Graphics::Vertex v = Graphics::Vertex();
Graphics::Line l = Graphics::Line();

using Graphics::Vertex;
Vertex v = Vertex();
Graphics::Line l = Graphics::Line();

using namespace Graphics:
Vertex v = Vertex();
Line l = Line();
```

Para acceder a los elementos dentro de un **namespace** hay que nombrarlos incluyendo el nombre del **namespace**.

Otra forma es importando algún elemento del **namespace** con la directiva **using**.

O bien importar todos los elementos del **namespace** con **using namespace**.

Friend

```
class List {
    friend class ListIterator;
    void* head;  // private
};

class ListIterator {
     void* get_first_elem(List *l) {
          return l->head;
     }
};
```

Una clase **friend** puede acceder a los atributos y métodos privados.

Es posible hacer **friend** no solo a una clase sino a un solo método.

friend no es transitivo: el amigo de mi amigo no es necesariamente mi amigo.

Smart pointers

```
std::unique_ptr p1 = new Object();
std::unque_ptr q1 = p1; // transfiere ownership

std::shared_ptr p2 = new Object();
std::shared_ptr q2 = p2; // comparte ownership

std::weak_ptr p3 = p2; // accede pero no
                       // comparte ownership
```