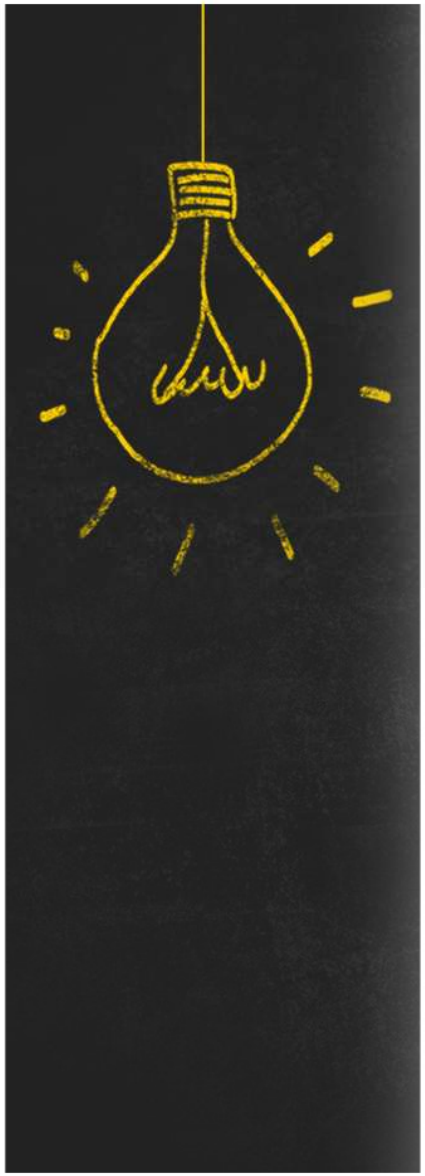
Programación 4
Guía Semana 12 (10/06)

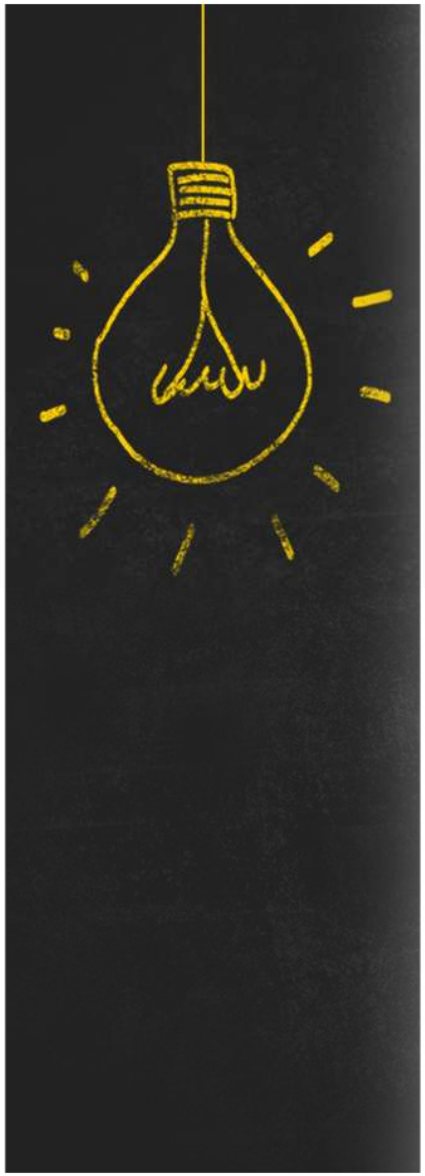
InCo, FING, UdelaR



Objetivo

Los objetivos de esta semana son:

- definir la implementación en C++ de **colecciones de objetos genéricas**;
- ver particularidades en la definición de estas colecciones: **iteradores**, **búsquedas**;
- analizar la **implementación de patrones de diseño** básicos (Singleton, Factory, State y Observer).



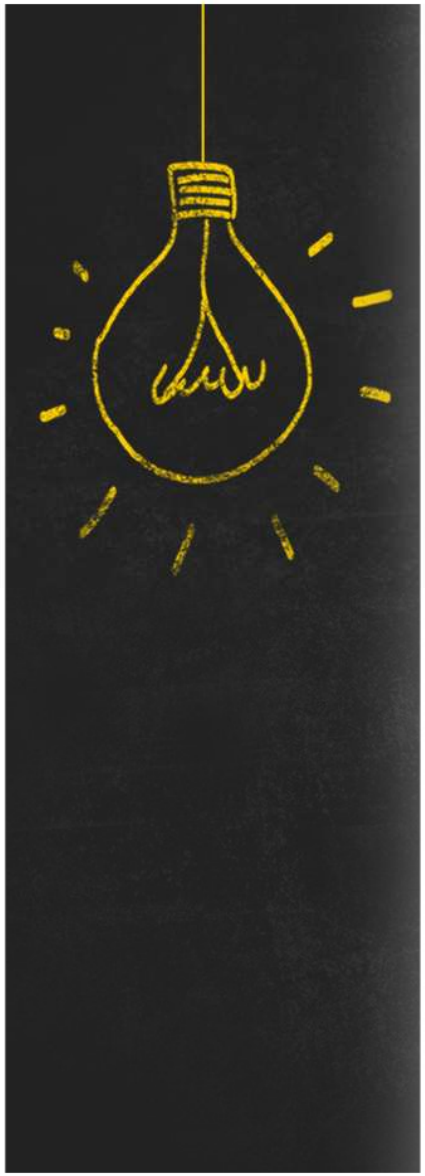
Resumen :: Colecciones

Las **colecciones de objetos** son una herramienta fundamental para la implementación de muchas de las asociaciones presentes en un diseño

Las colecciones deben permitir:

- Realizar **iteraciones** sobre sus elementos
- Realizar **búsquedas** de elementos por clave (en caso de que los elementos tengan una)
- Realizar búsquedas diversas

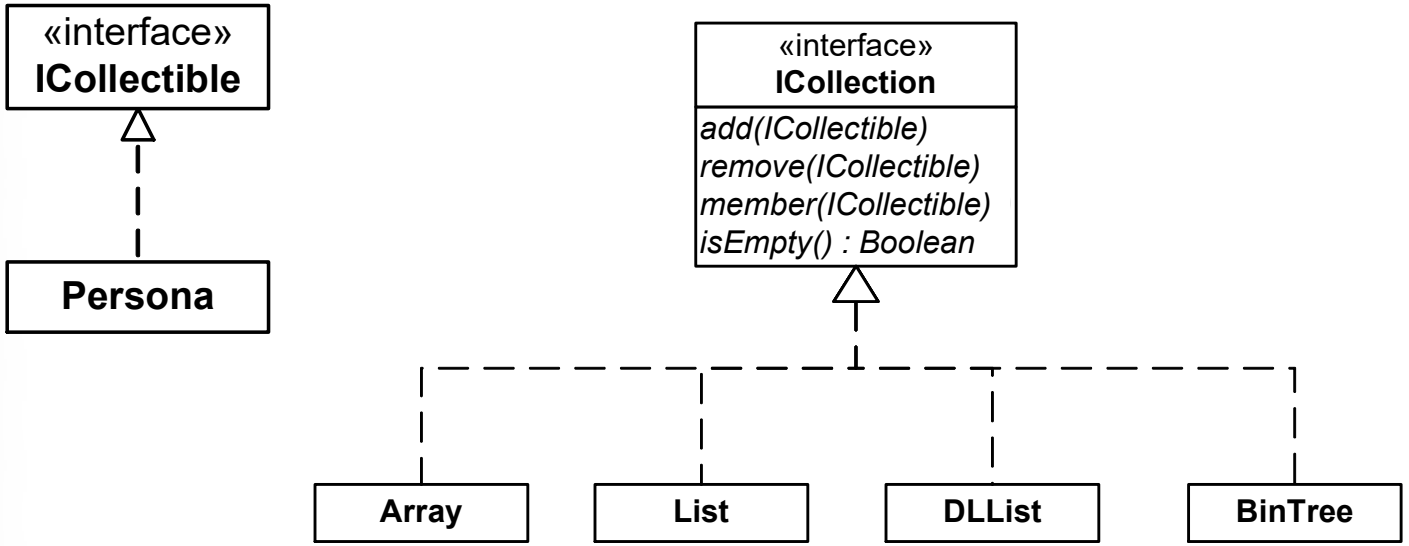
Es posible definir una infraestructura común que sirva de base para todas las colecciones específicas

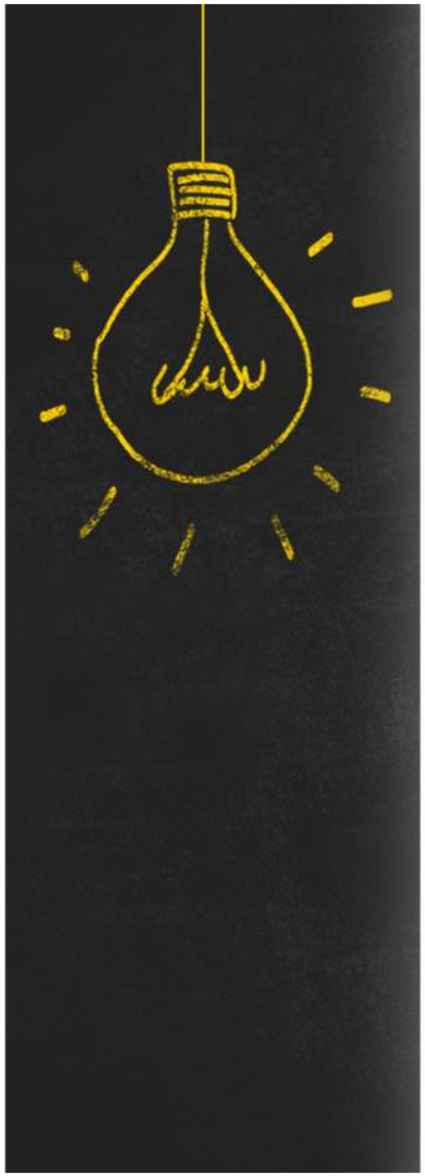


Resumen :: Colecciones

¿Cómo lograr que un elemento de una clase cualquiera pueda ser almacenado en la colección genérica?

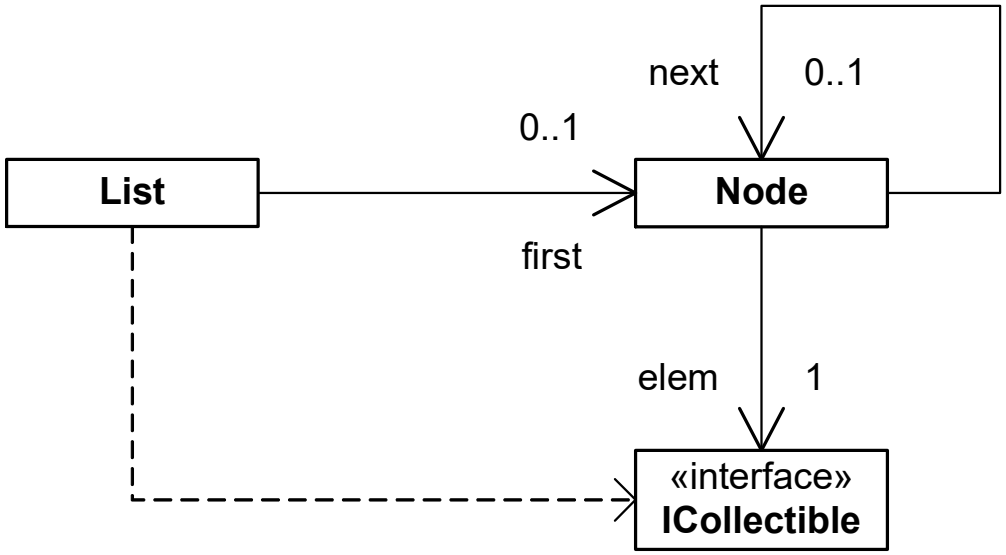
¿Cómo se define una **colección genérica**?

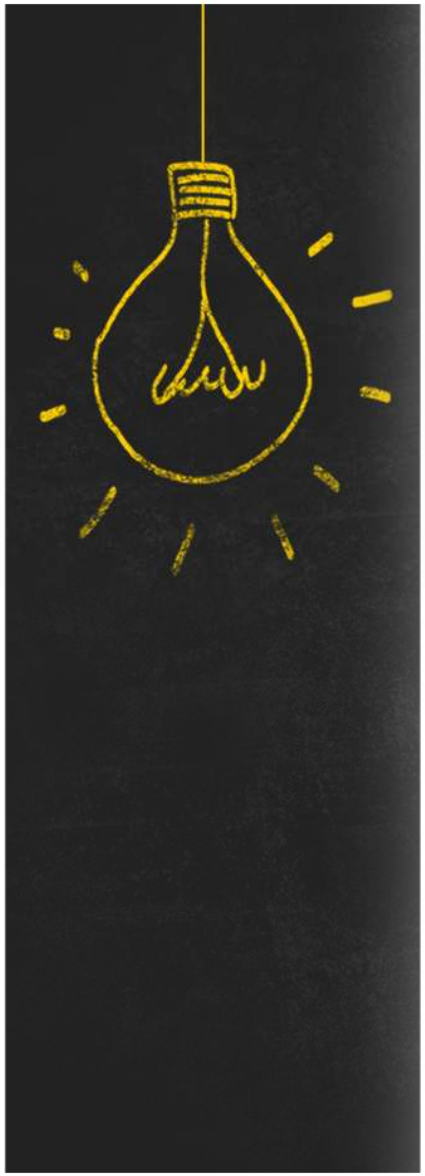




Resumen :: Colecciones

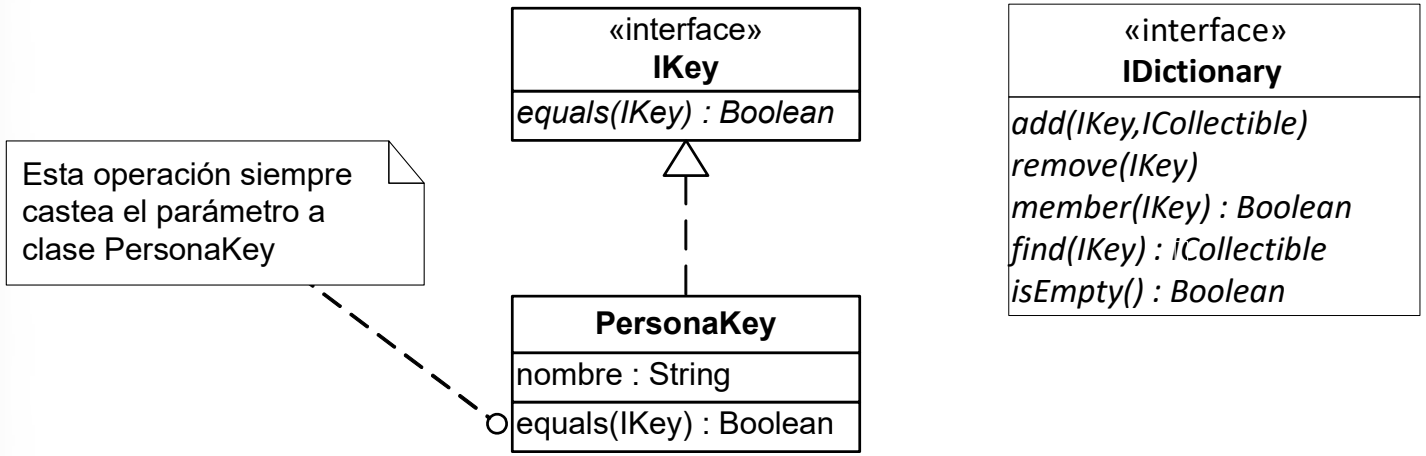
El diseño de una lista común utilizando clases no difiere significativamente del diseño usual

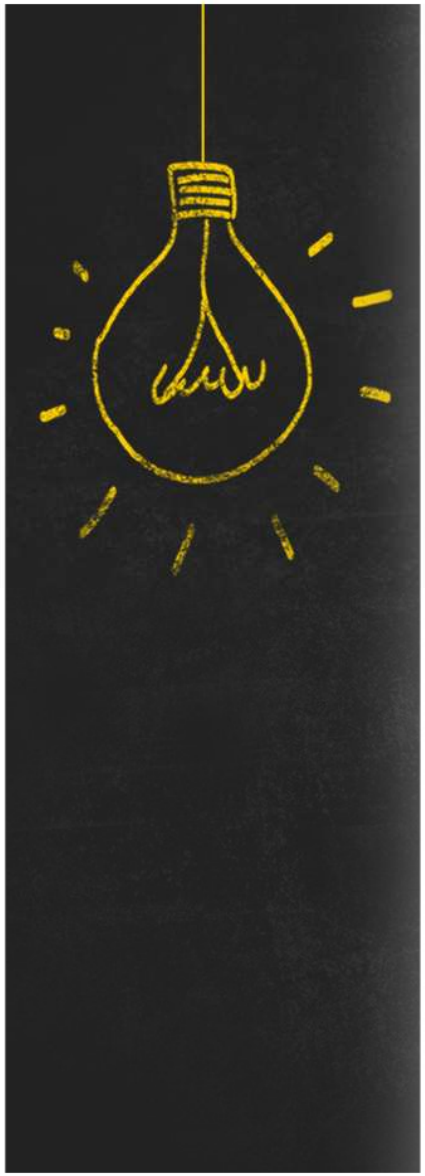




Resumen :: Colecciones

Un **diccionario** es un tipo particular de colección en el cual se almacenan objetos que pueden ser identificados por una **clave**



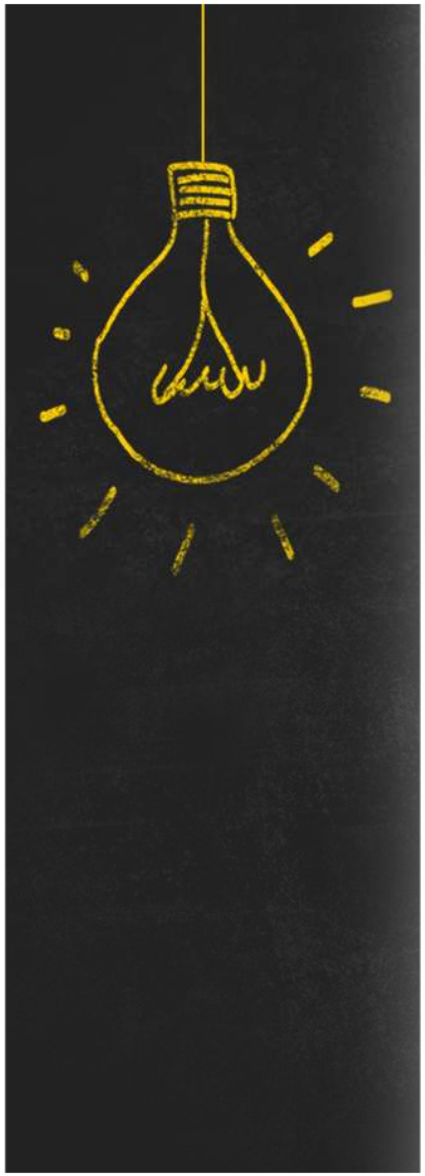


Resumen :: STL

C++ ofrece la **Standard Template Library**, un conjunto de contenedores (clases paramétricas), iteradores y otras utilidades para manejar conjuntos de objetos.

Es una alternativa a implementar colecciones genéricas (no se reinventa la rueda)

- **set<T>**: Una colección de objetos. Permite opcionalmente ordenamiento
- **map<K,V>**:Un Diccionario de claves de tipo K y valores de tipo V
- Otros: **vector**, **queue**, **list**, **stack**

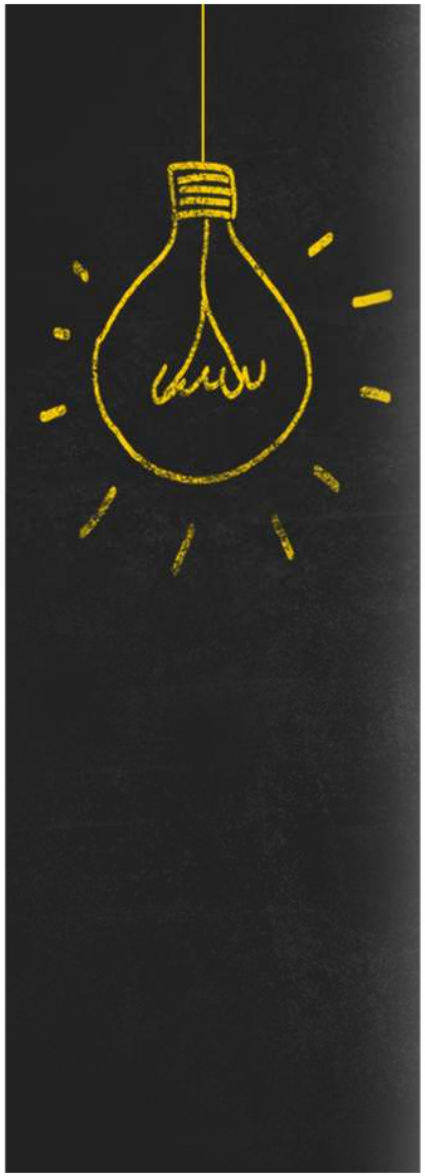


Resumen :: Singleton

¿Cómo restringir que una clase tenga una sola instancia y que se tenga visibilidad global hacia ella?

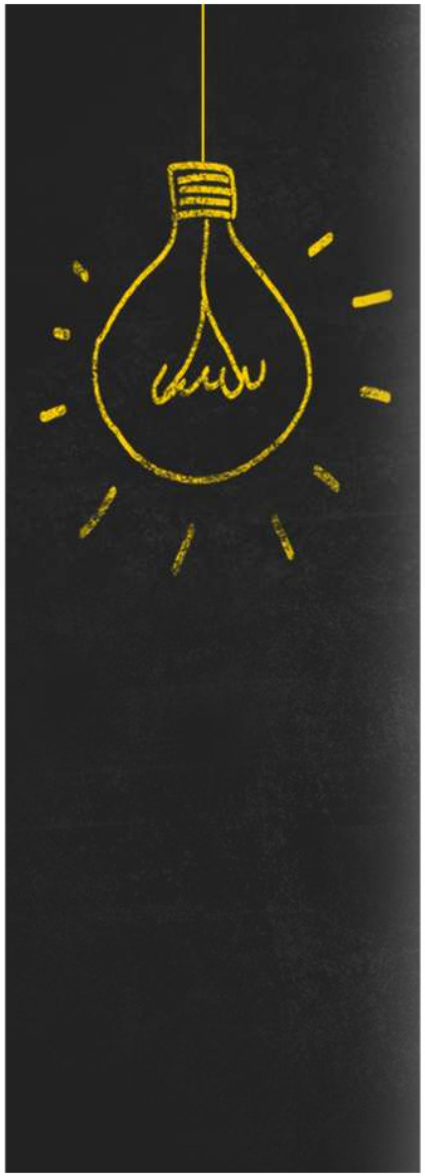
```
// Singleton.h

class Singleton {
private:
        static Singleton * instancia;
        Singleton();
public:
        static Singleton * getInstancia();
        void operacion();
};
```



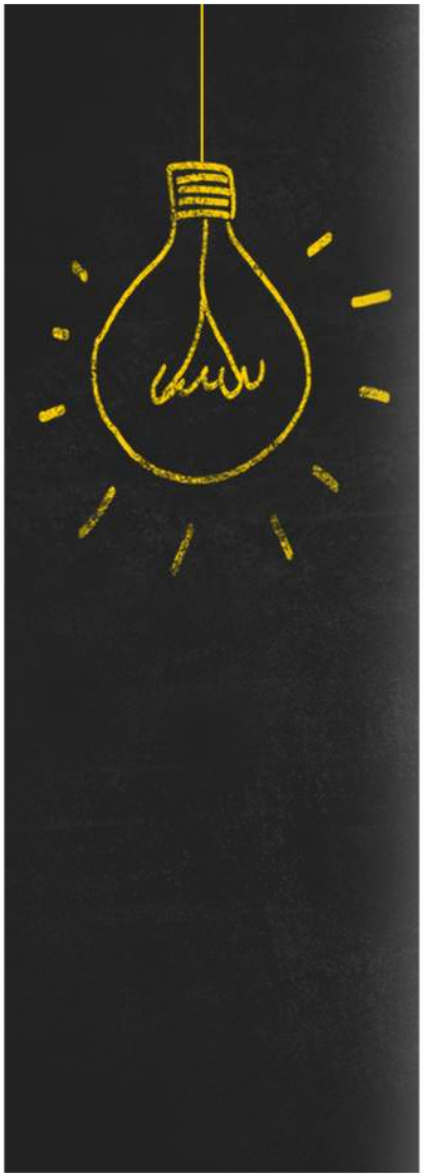
Resumen :: Singleton

```
// Singleton.cpp
#include "Singleton.h"
Singleton * Singleton::instancia = NULL;
Singleton::Singleton() {...}
Singleton * Singleton::getInstancia() {
   if (instancia == NULL)
      instancia = new Singleton();
   return instancia;
}
void Singleton::operacion() {...}
```



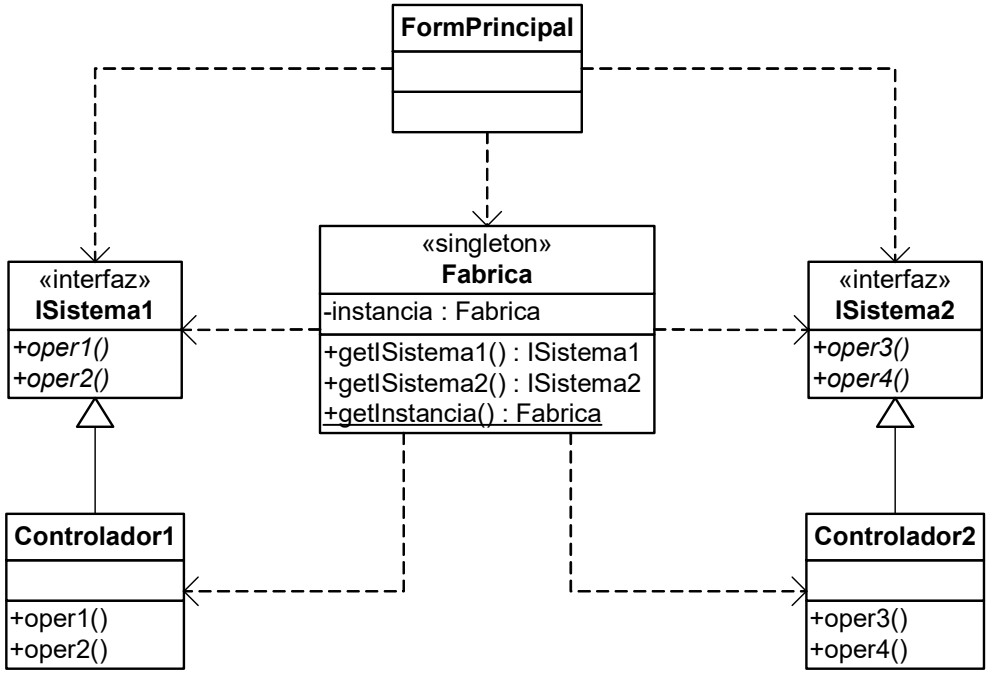
Resumen :: Singleton

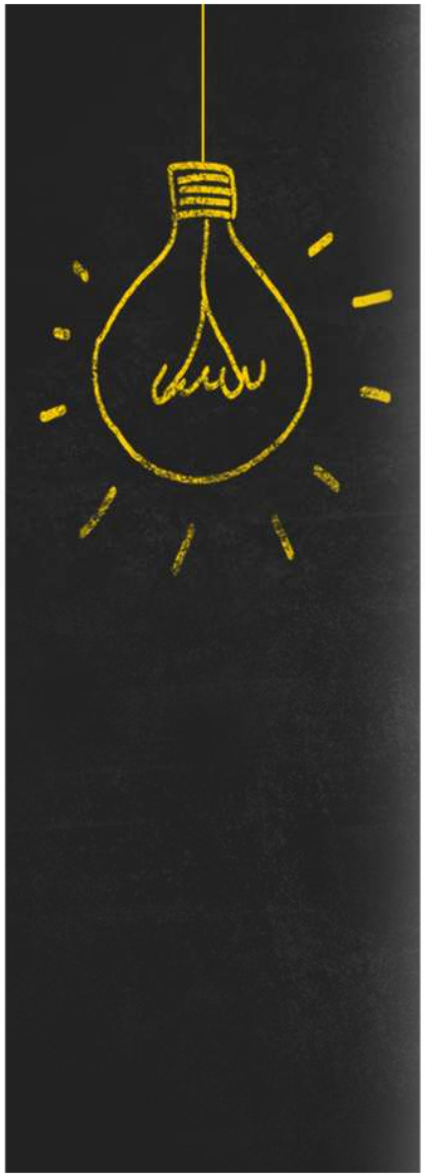
```
// Ejemplo de uso
#include "Singleton.h"
int main() {
        Singleton * ms;
        ms = Singleton::getInstancia();
        ms -> operacion();
        return 0;
}
```



Resumen :: Factory

¿Cómo acceder a objetos (ej.: Controladores), sin acoplarse directamente con ellos?



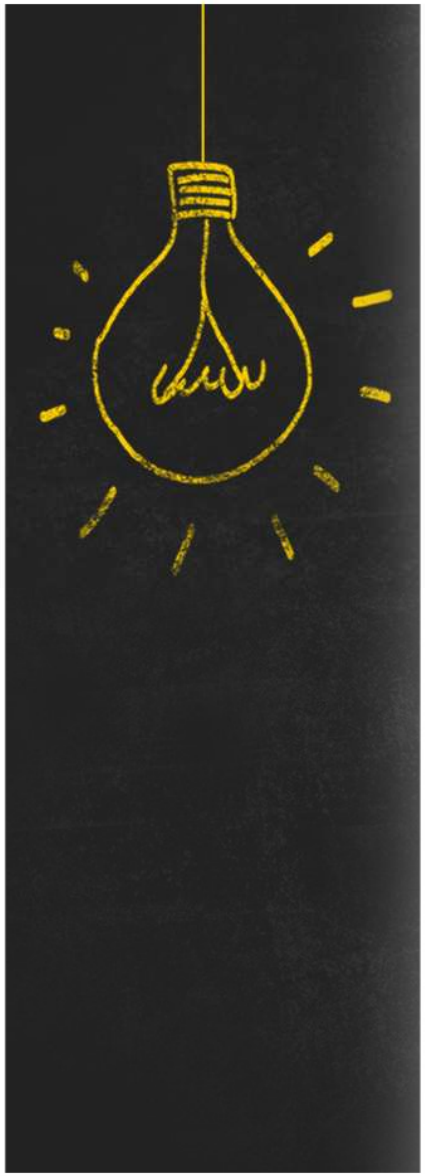


Resumen :: Factory

Alternativa 1 :: Que los controladores sean Singleton y simplemente sean accedidos mediante `getInstance()` desde la Fábrica.

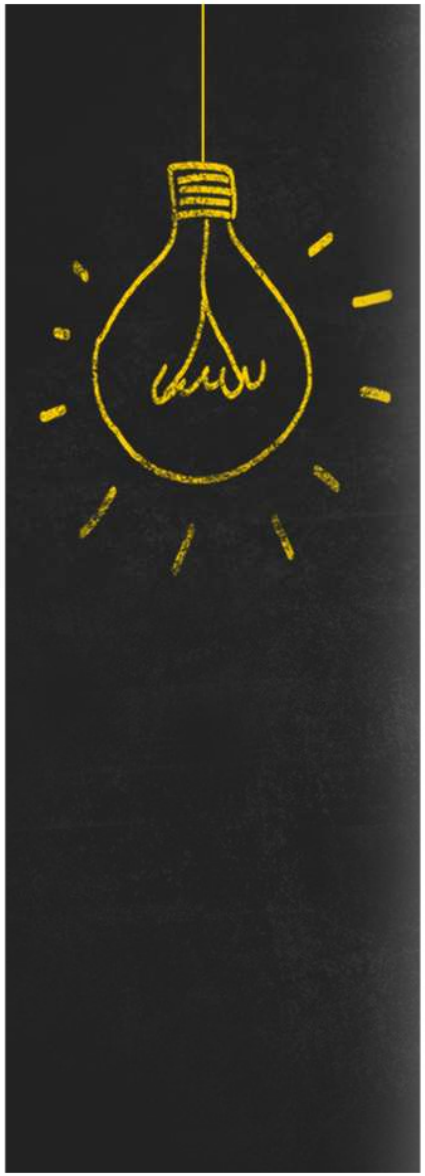
Alternativa 2 :: Que los controladores no sean Singleton y que la Fábrica mantenga una referencia a cada uno de ellos controlando su unicidad.

Alternativa 3 :: Que los controladores no sean Singleton y que la Fábrica no mantenga referencias, sino que devuelva una nueva instancia del controlador cada vez.



Resumen :: Factory

```
// ------------ Alternativa 1 ------------
ISistema1 Fabrica::getISistema1() {
   return Controlador1.getInstancia();
}
// ------------ Alternativa 2 ------------
ISistema1 Fabrica::getISistema1() {
   if (this->sistema1 == NULL)
      this->sistema1 = new Controlador1();
   return this->sistema1;
}
// ------------ Alternativa 3 ------------
ISistema1 Fabrica::getISistema1() {
   return new Controlador1();
}
```



¿Qué hago esta semana?

1. Estudio los materiales de Teórico y las lecturas recomendadas. Las clases correspondientes se encuentran en OpenFing.
    - 18 - Implementación: Colecciones
    - 19 - Implementación: Patrones de Diseño
2. Finalizo el Práctico 6 “Implementación”.
3. Continúo el Laboratorio 4 “Implementación”. Plazo de entrega: lunes 24/06, 15hs