Universidad Rey Juan Carlos

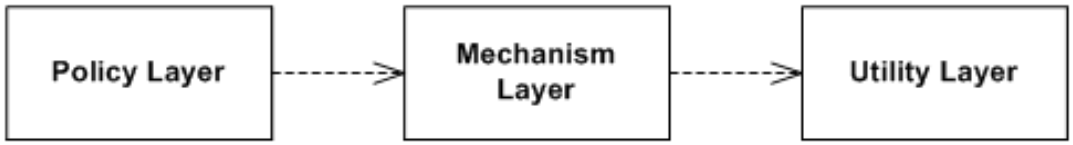
Inversión de Control

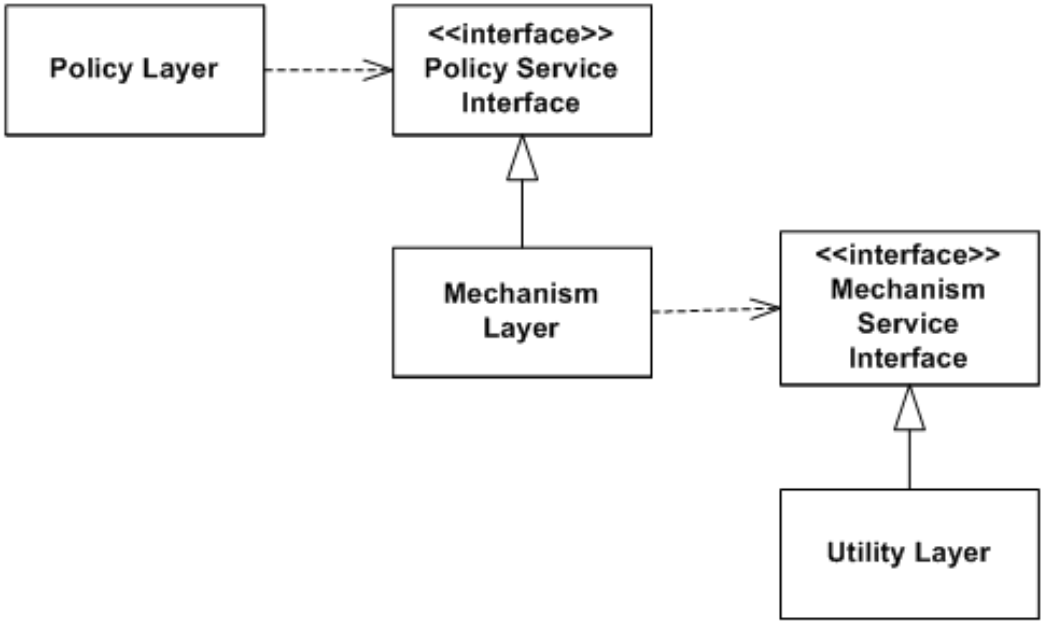
Carlos E. Cuesta Quintero
Depto. Ciencias de la Computación, AC, LSI y EIO
Escuela Técnica Superior de Ingeniería Informática

Inversion of Control

- Inversión de Control (IoC)
  - Principio de Hollywood
    - *Frameworks* (contenedores)
  - Relacionados:
    - Principio de Inversión de Dependencias
    - *Event-Driven Programming* (*Event Loop*)
  - Tipos Específicos:
    - *Callbacks*, *Schedulers*
    - *Event Loop*
    - Inyección de Dependencias
    - Patrón *Template Method*
    - Patrón *Service Locator*
    - Patrón *Strategy*

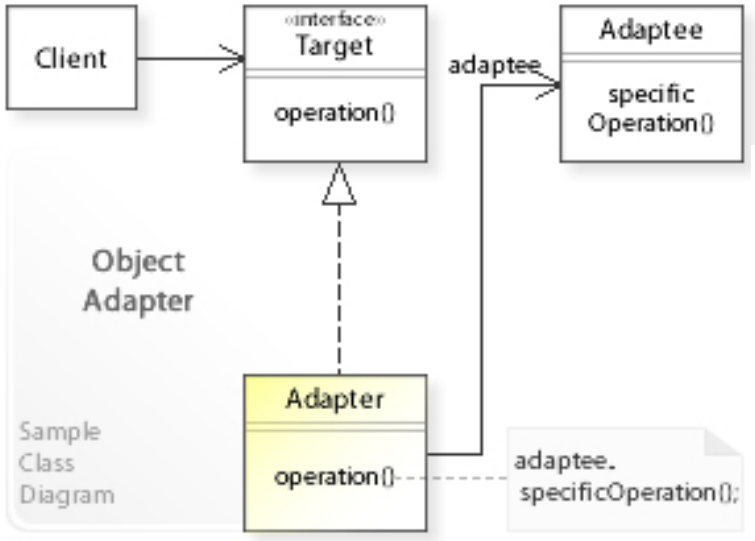
Dependency Inversion (Principle)





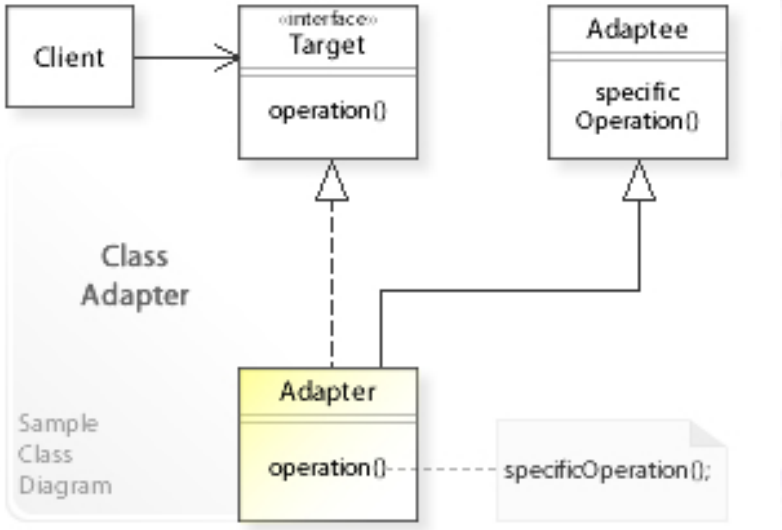
Inversión de Dependencias

- Relacionado con el patrón Adaptador
  - También con los patrones Plugin, Decorador

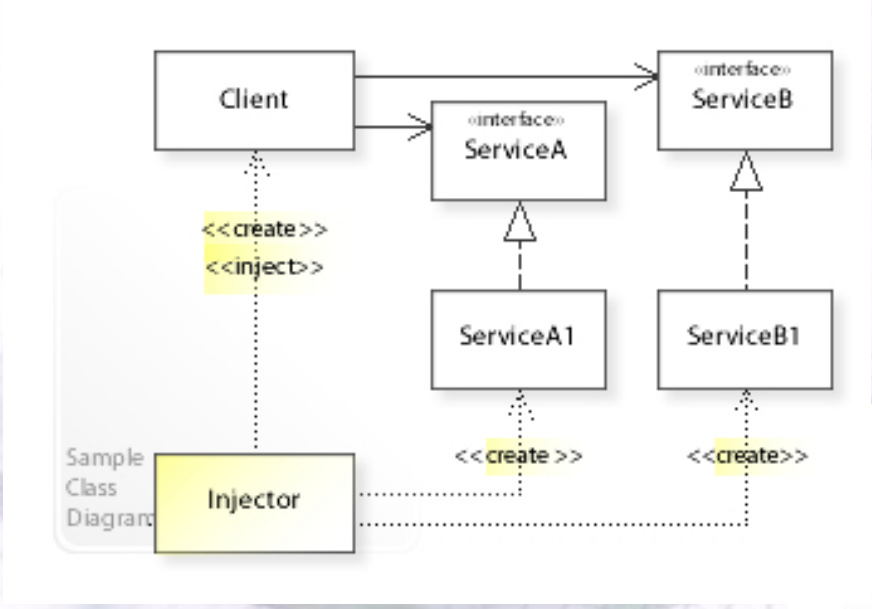


Por Delegación

Por Inversión de Dependencias

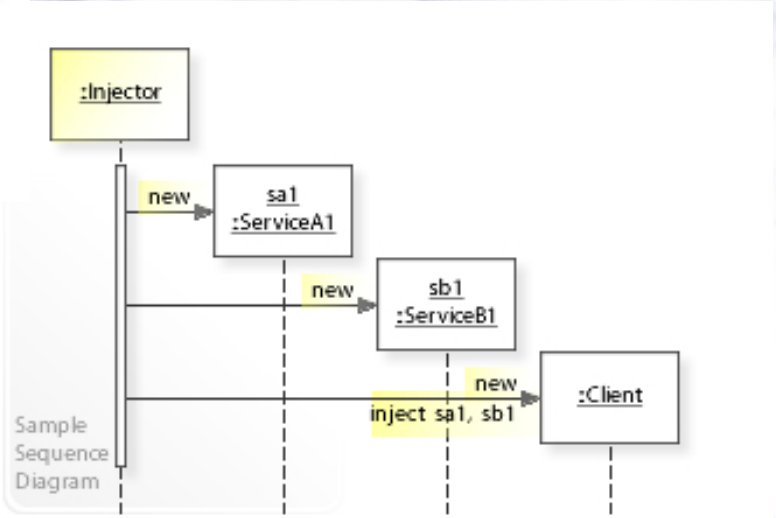


Inyección de Dependencias

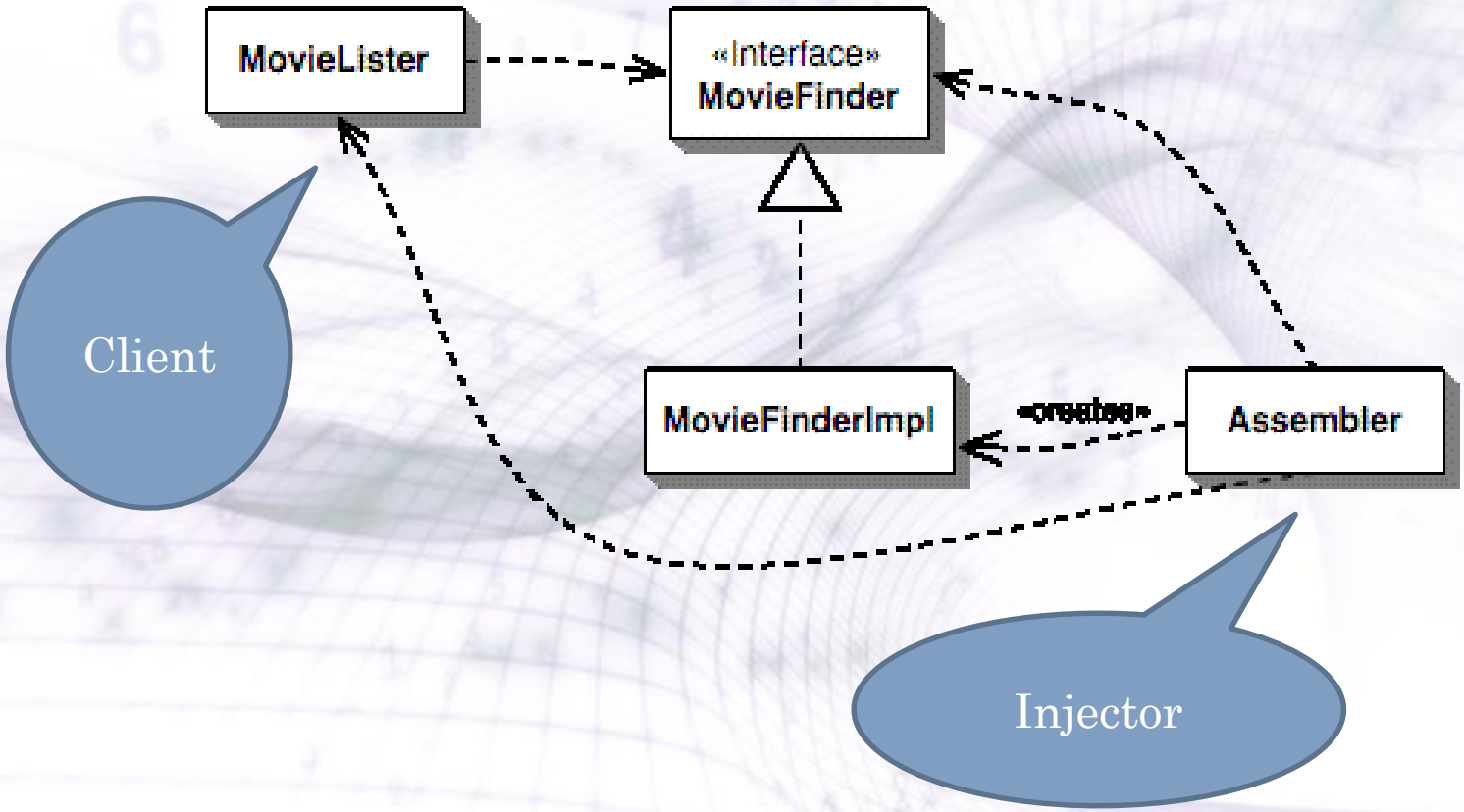


Orden de las operaciones (distinto, p.e. de la Factoría Abstracta)

Rol del Inyector (con Inversión de Dependencias)



EJEMPLO: MOVIEFINDER



Tipos de Inyección de Dependencias

- Setter Injection (Type-2 IoC)

```
class MovieLister {...
  private MovieFinder finder;
  public void setFinder (MovieFinder finder) {
    this.finder = finder;}
```

- Interface Injection (Type-1 IoC)

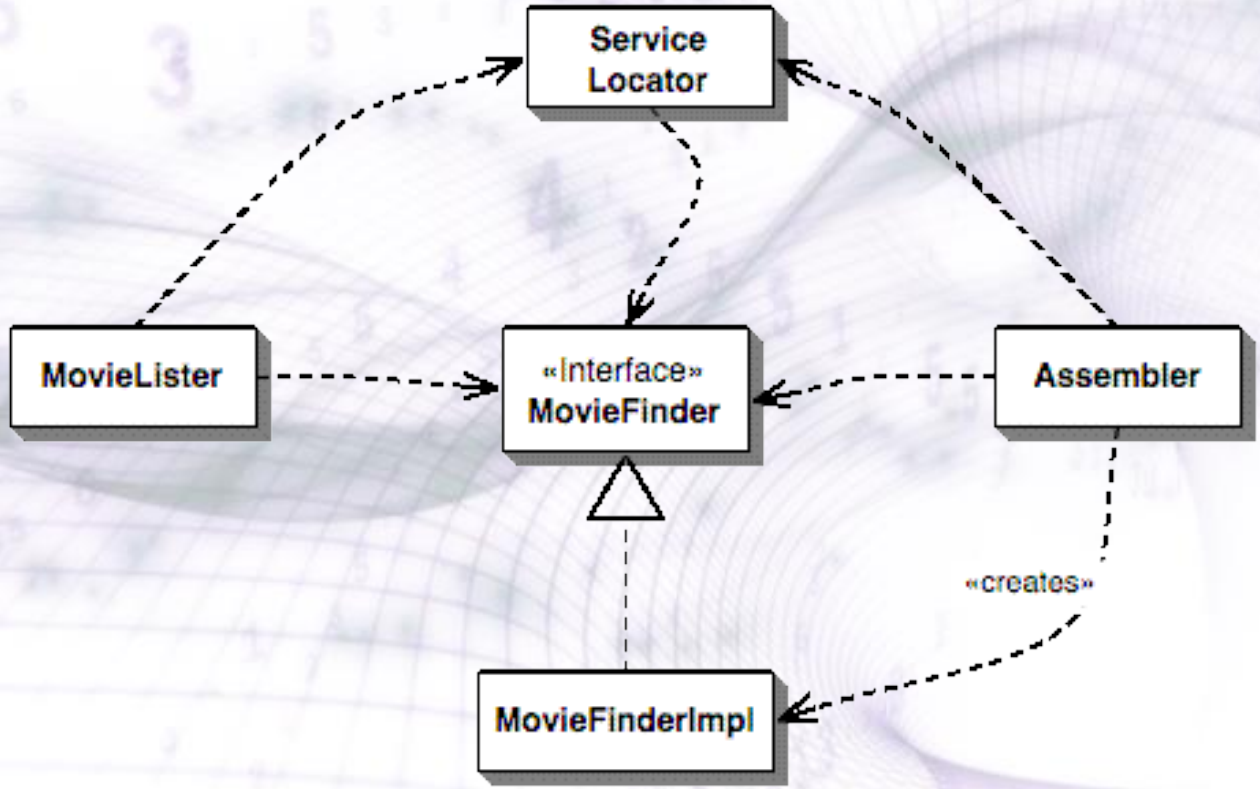
```
public interface InjectFinder {
  void injectFinder (MovieFinder finder);}
class MovieLister implements InjectFinder {...}
```

- Constructor Injection (Type-3 IoC)

```
 public MovieLister (MovieFinder finder) {
  this.finder = finder; }
```

Service Locator (Pattern)

- Alternativa: utilizar un localizador



Service Locator

- Sin inyección de dependencias:

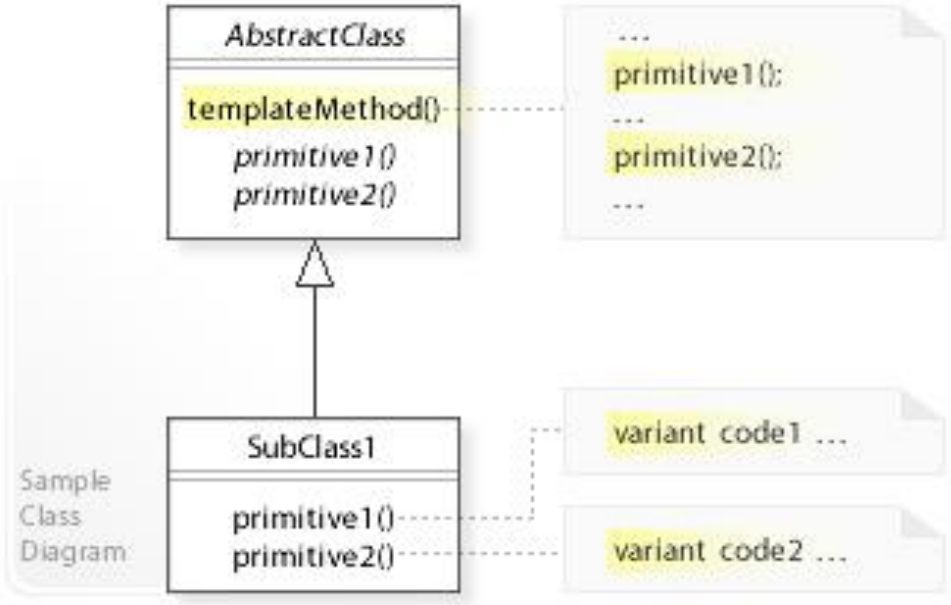
```
class MovieLister { ...
  MovieFinder finder = ServiceLocator.movieFinder(); }

class ServiceLocator {...
  public static MovieFinder movieFinder() {
     return singleton.movieFinder; }
  private static ServiceLocator singleton;
  private MovieFinder movieFinder;
...}
```

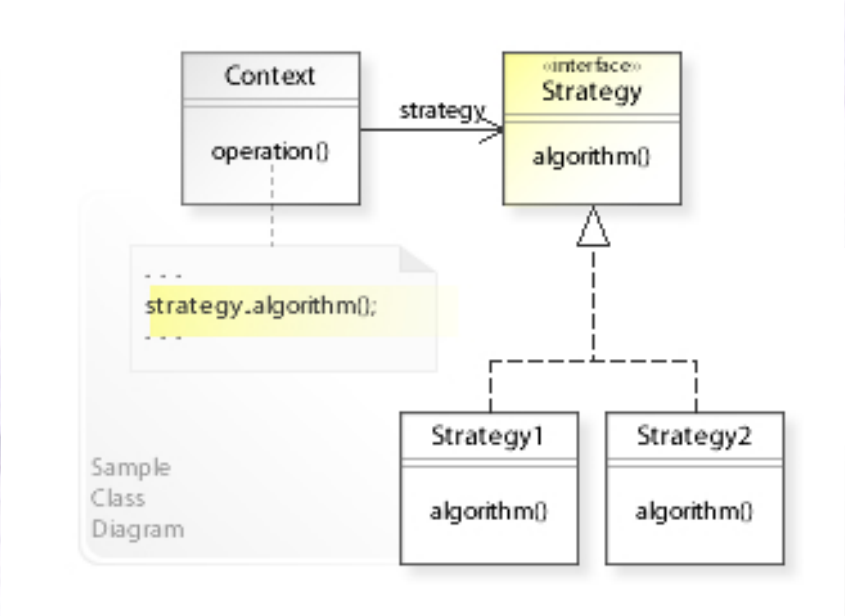
- Se puede utilizar un patrón Singleton (éste es el caso)
- Algunos consideran este enfoque un antipatrón

Template Method (Pattern)

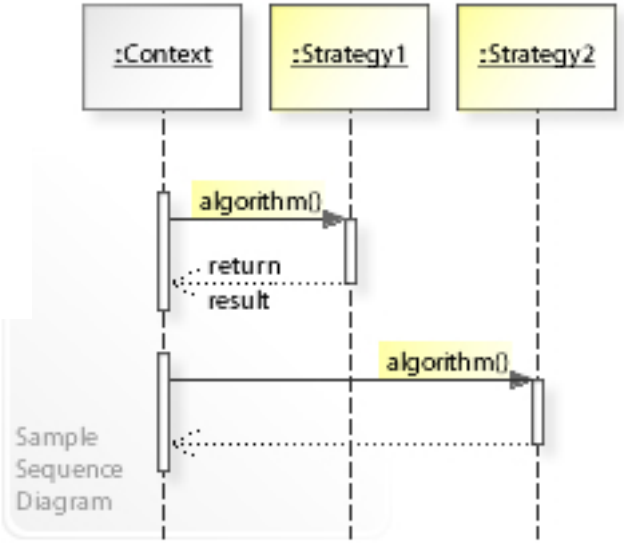
- El método plantilla invoca a los métodos concretos
  - Sin conocerlos en tiempo de diseño (abstracción)



Strategy (Pattern)



También similar en el patron State



Por inversión de dependencias