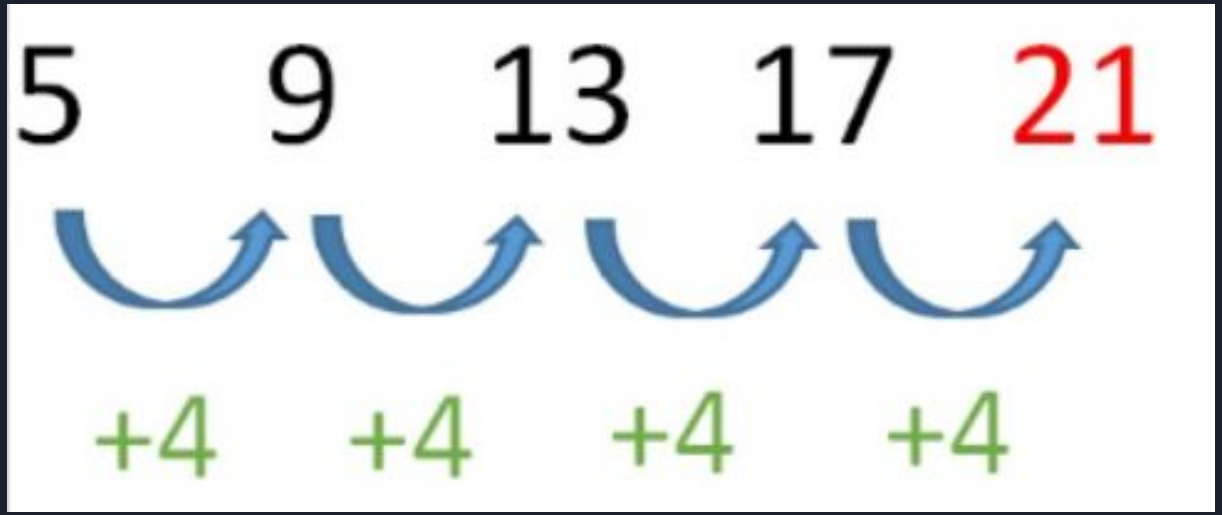
Design patterns

Daniel del Castillo de la Rosa <daniel.del.19@ull.edu.es>
Francisco Jesús Mendes Gomez <francisco.jesus.mendes.gomez.08@ull.edu.es>

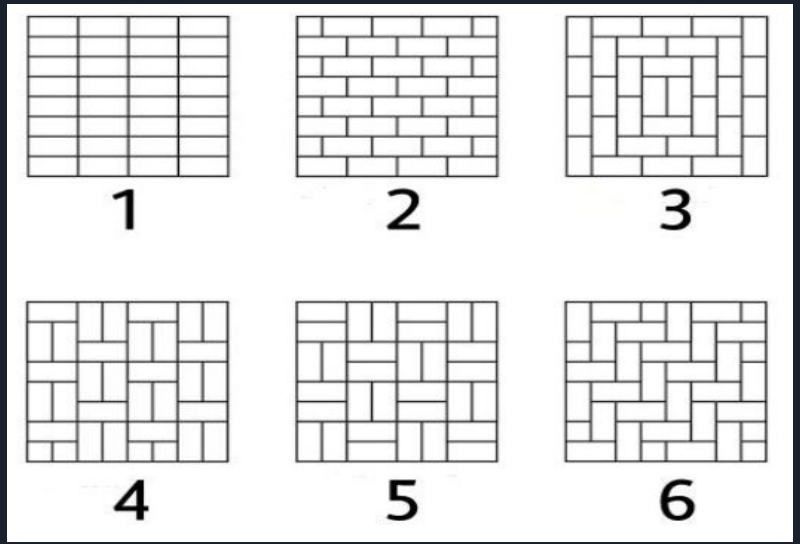
Table of contents

What is a Pattern?



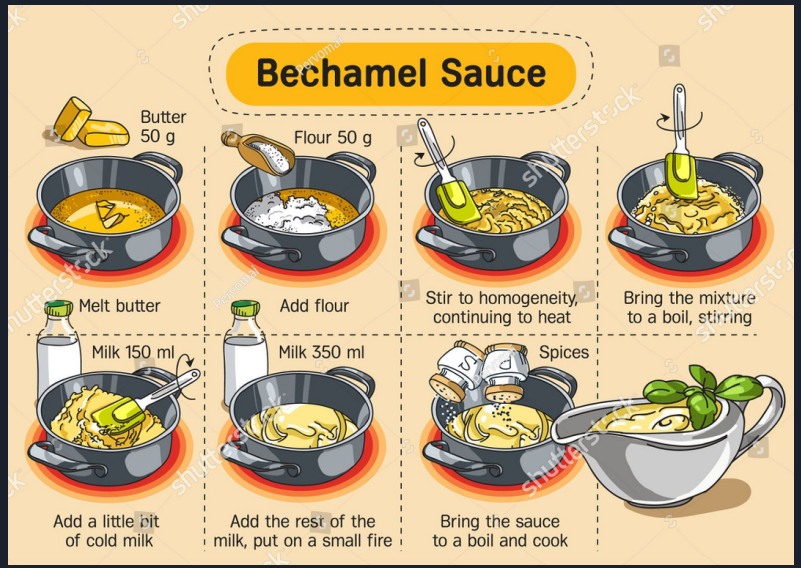
What is a Pattern?

- Another example:



What is a Design Pattern ?

- Definition:
  - Techniques to solve recurring software design problems.



History

- Alexander, 1979 - The Timeless way of Buildings (Book).
- Cunningham and Beck, 1987 - Using Patterns Languages for OOP Programs (Article).
- Gang of Four (GoF), Gamma, Helm, Johnson and Vlissides - Design Patterns (Book).

Anti-patterns.

- Common response
- Usually ineffective
- Can be counterproductive.

Types of Design Patterns

- We will talk about 3 types of Design Patterns:
  - Creational
  - Structural
  - Behavioural

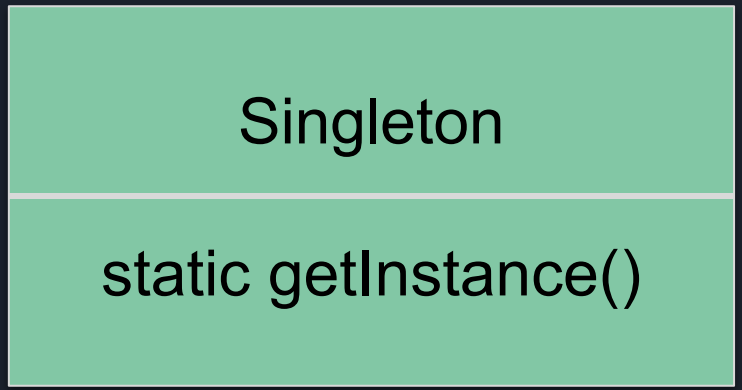
Creational patterns

- Singleton
- RAII
- Builder pattern

Singleton

- Ensure a class only has one instance
- Frequently criticized

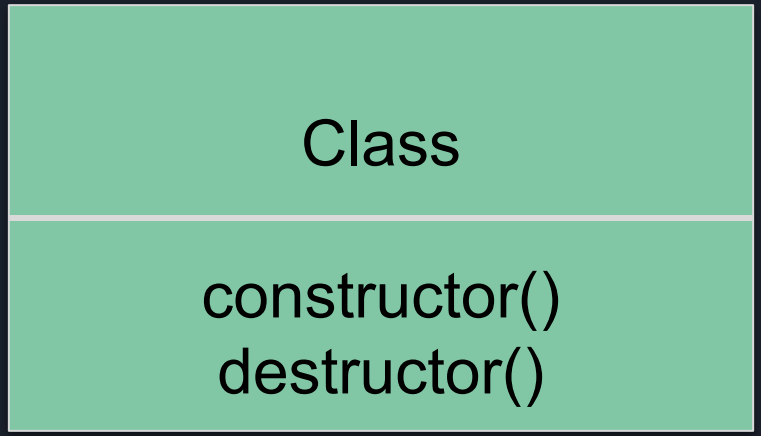
Singleton



RAII

- Resource Acquisition Is Initialization
- Based on use of constructors and destructors
- Encapsulate resource management

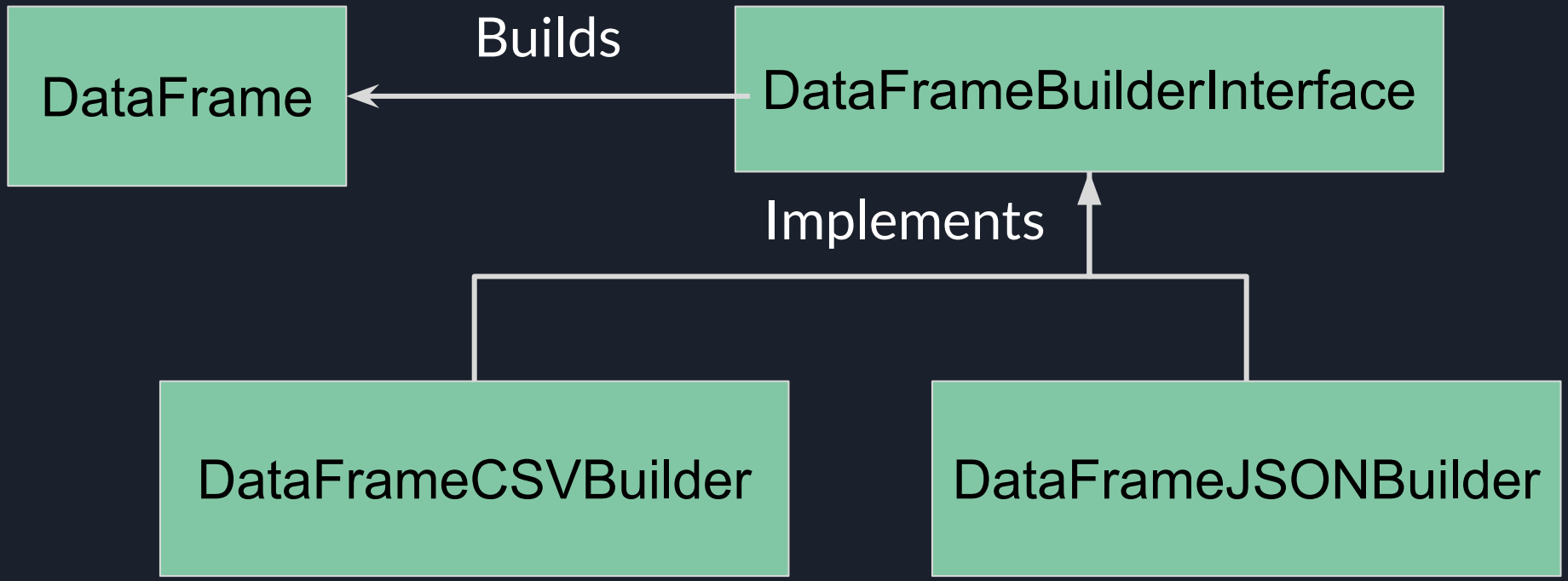
RAII



Builder pattern

- Simplify classes
- Separate construction of a complex object from its representation

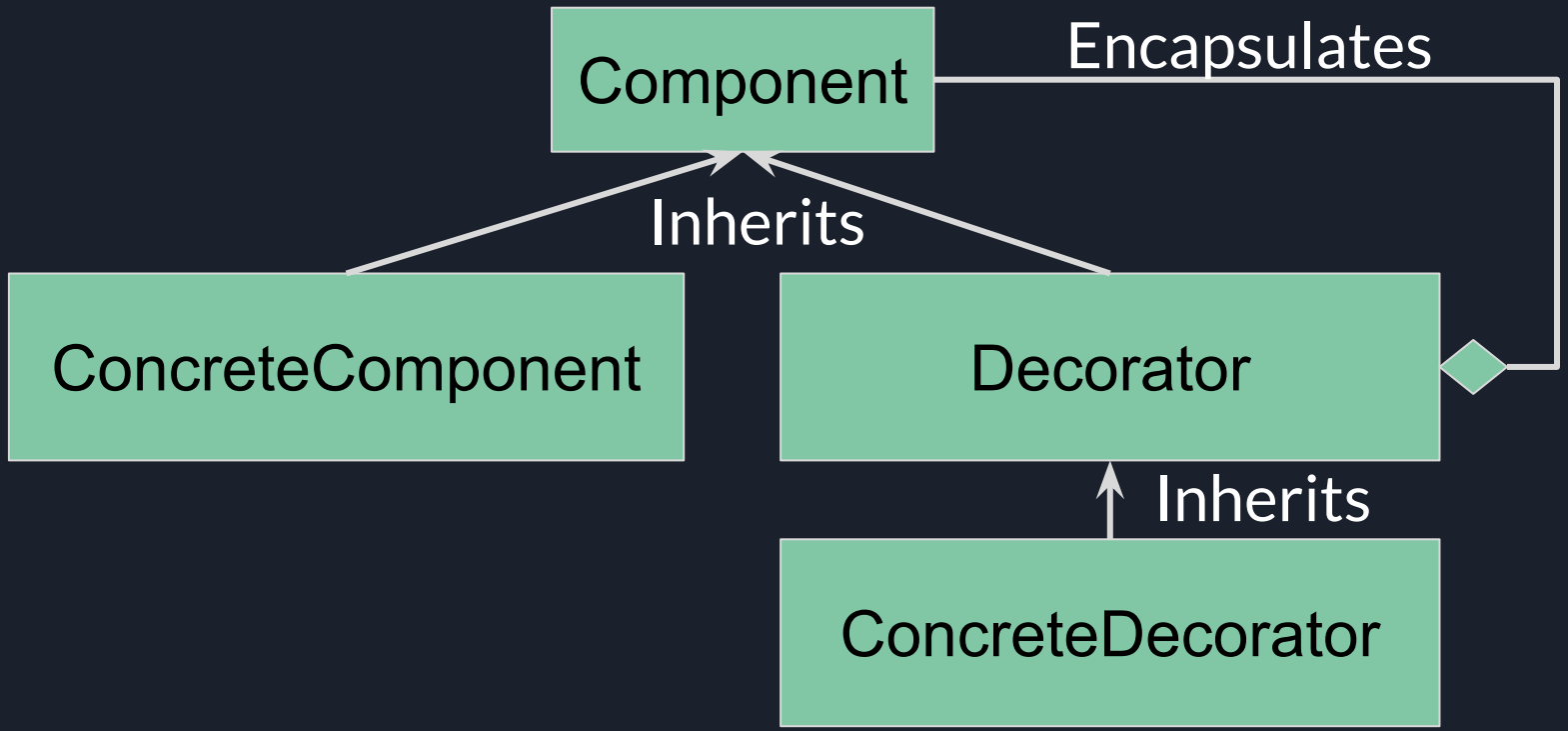
Builder pattern



Structural Patterns

- Decorator
  - Adds functionality to existing classes dynamically

Decorator Pattern



Decorator Pattern.

Advantages:

- Extends the functionality of an object without creating new subclasses.
- Allow us add or remove functionalities in execution time.

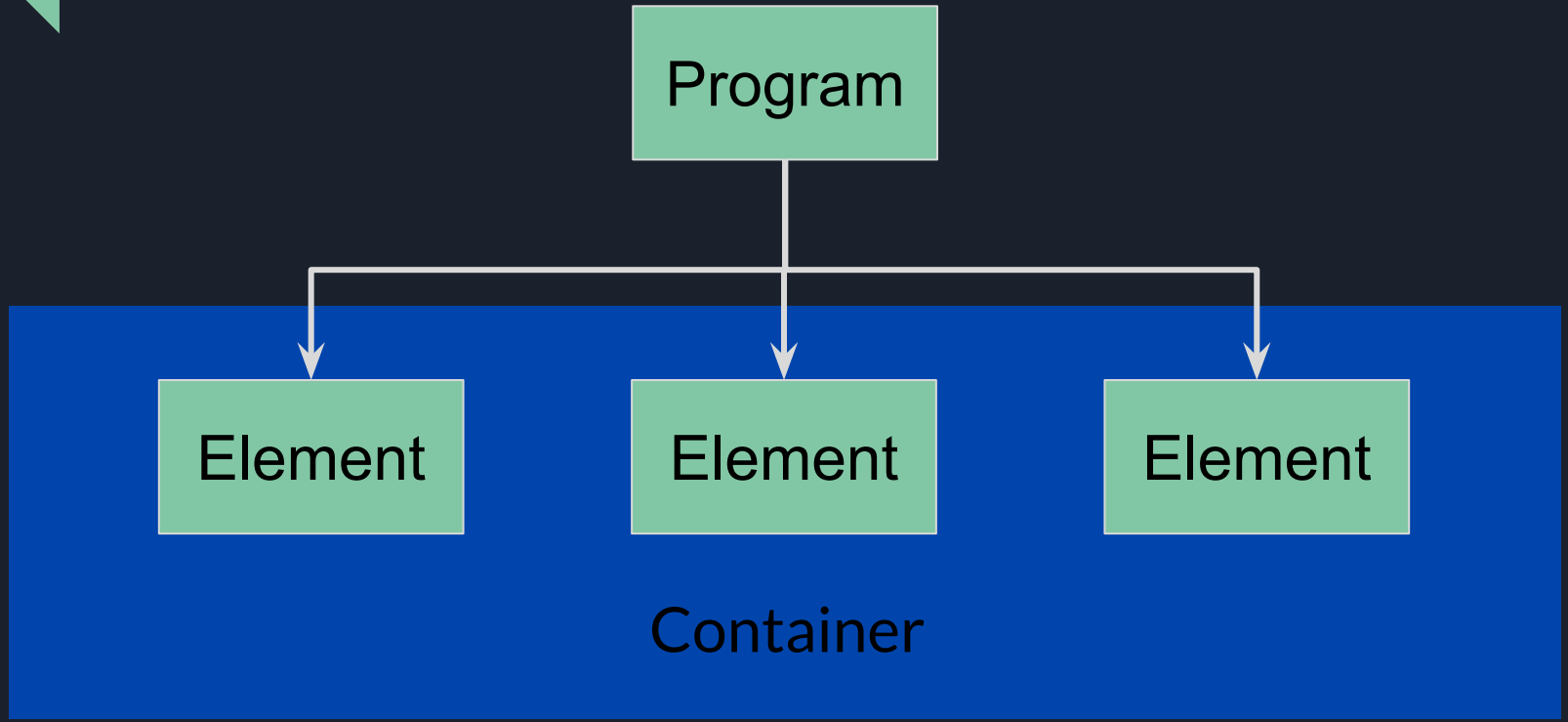
Behavioural patterns

- Iterator pattern
- Observer pattern
- Chain of responsibility pattern

Iterator pattern

- Traverse a container
- Independent from container type

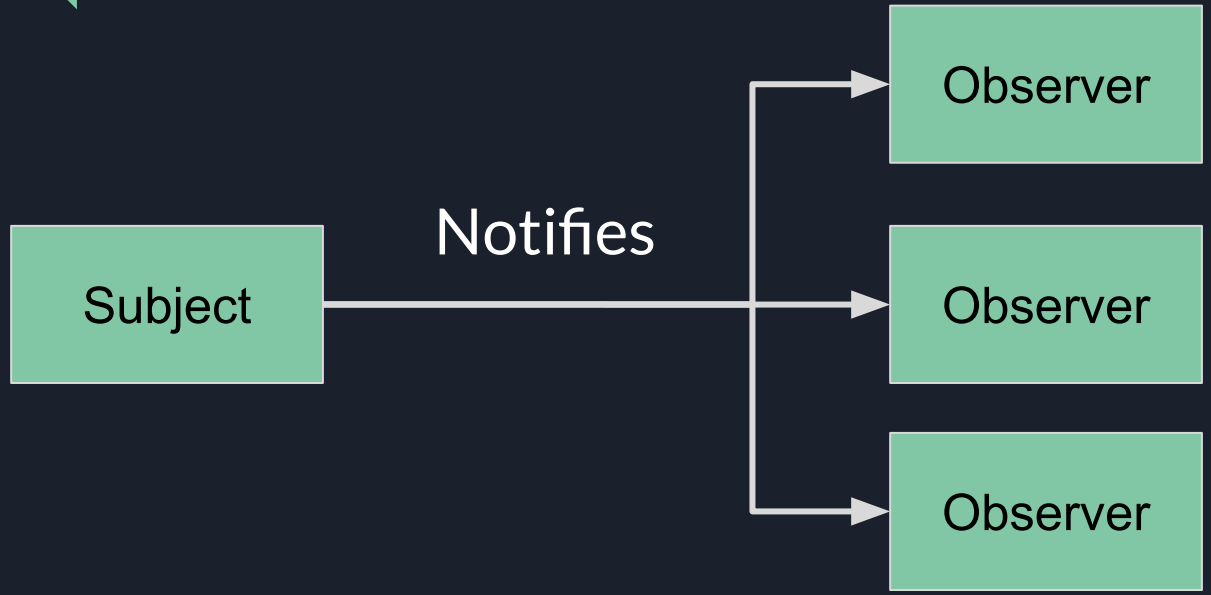
Iterator pattern



Observer pattern

- Observers wait for an event
- Another class notifies events to observers
- The observers perform an action when notified
- One-to-many dependency without tight coupling

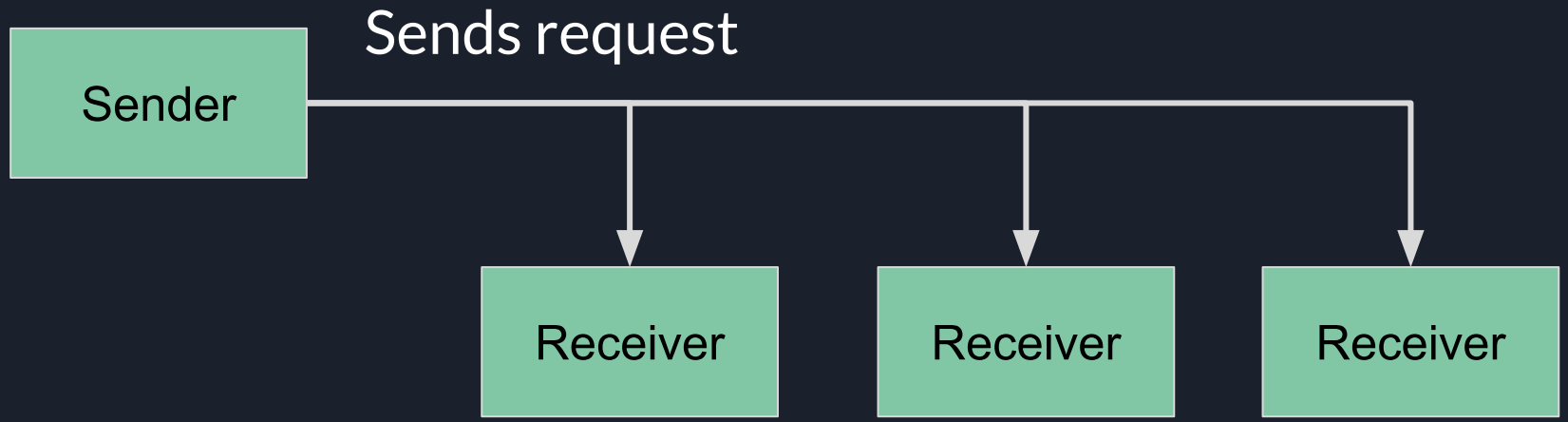
Observer pattern



Chain of responsibility pattern

- Commands and processing objects
- Each command gets passed until it finds an object that can process it
- Loose coupling between senders and receivers

Chain of responsibility pattern



Importance of Design Patterns

- Main benefits:
  - Expressive.
  - Proven solutions.
  - Easily reused.
  - Recognizable.
- Best practice.

Summary

- Our goal
- Design patterns are really cool, but …

Bibliography

- Wikipedia - Design Patterns
- Design Patterns Book - GoF
- Learning JavaScript Design Patterns by Addy Osmani
- JavaScript Design Patterns
- Rust unofficial
- JavaScript - Patrones de diseño en JS

Thanks.

Daniel del Castillo de la Rosa <daniel.del.19@ull.edu.es>
Francisco Jesús Mendes Gómez <francisco.jesus.mendes.gomez.08@ull.edu.es>