SOFTWARE ENGINEERING
CS-UH 2012, FALL 2021

Design Patterns - Creational

Designed by: Prof. Mai Oudah

DEFINITION OF A DESIGN PATTERN

- A **design pattern** is a standard reusable solution to a commonly occurring problem in software design

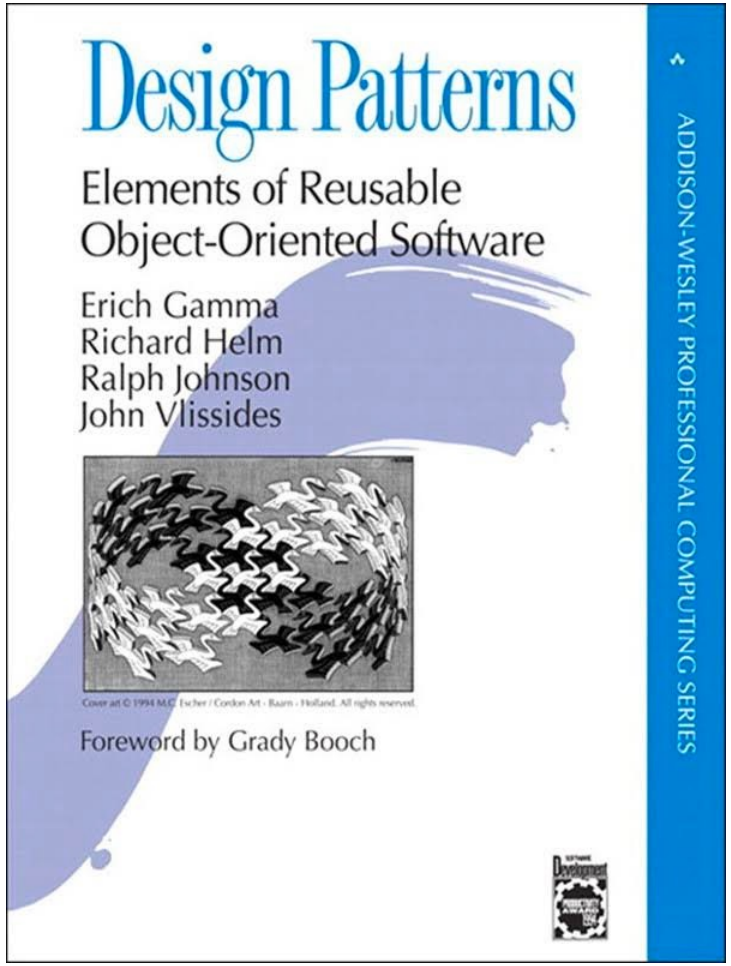
- It is a design or implementation structure that achieves a particular purpose, impacting mainly the class diagram

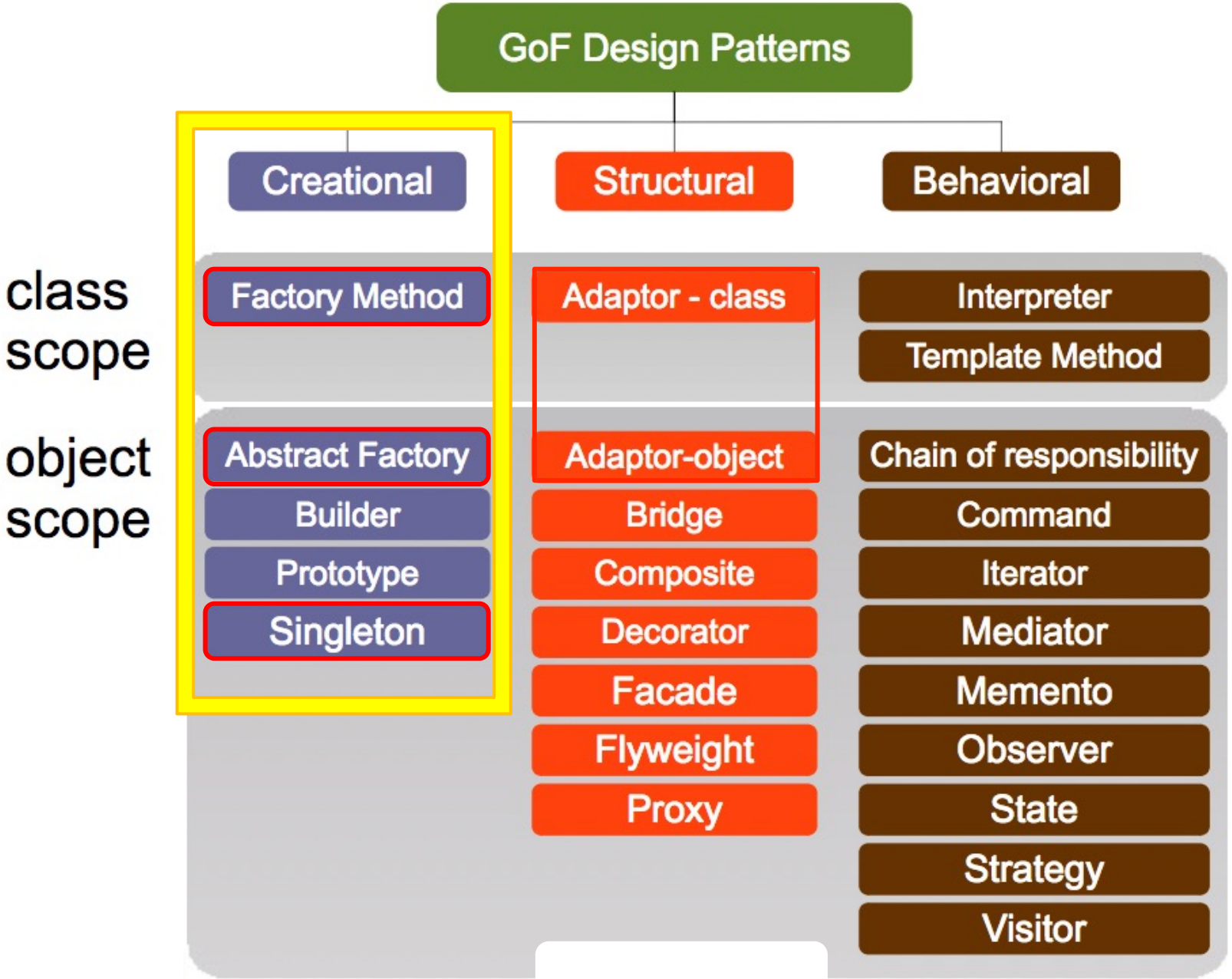
GoF DESIGN PATTERNS

- The **Gang of Four** are the four authors who sat the foundation for all design patterns

- **Classified by purpose:**
  - Creational
  - Structural
  - Behavioral

- **Classified by scope:**
  - Class scope: **(static)**
  - Object scope: **(dynamic)**




GoF Design Patterns
Creational
Structural
Behavioral
class scope
object scope
Factory Method
Adaptor - class
Interpreter
Template Method
Abstract Factory
Builder
Prototype
Singleton
Adaptor-object
Bridge
Composite
Decorator
Facade
Flyweight
Proxy
Chain of responsibility
Command
Iterator
Mediator
Memento
Observer
State
Strategy
Visitor

CREATIONAL PATTERNS

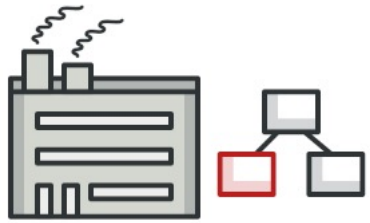
CREATIONAL PATTERNS

- Encapsulate the instantiation process

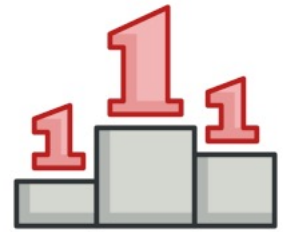
  → make an application independent of how its objects are created

  - Hide how objects are created and put together from other components within the system

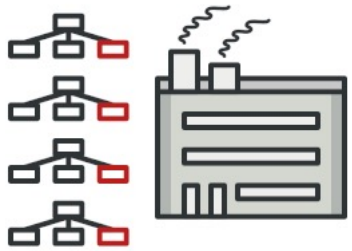
CREATIONAL PATTERNS



Factory Method



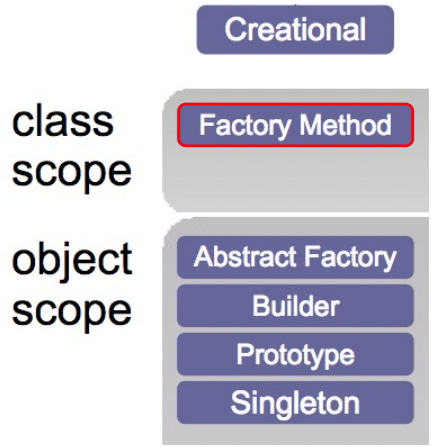
Singleton



Abstract Factory

CREATIONAL PATTERNS

FACTORY METHOD



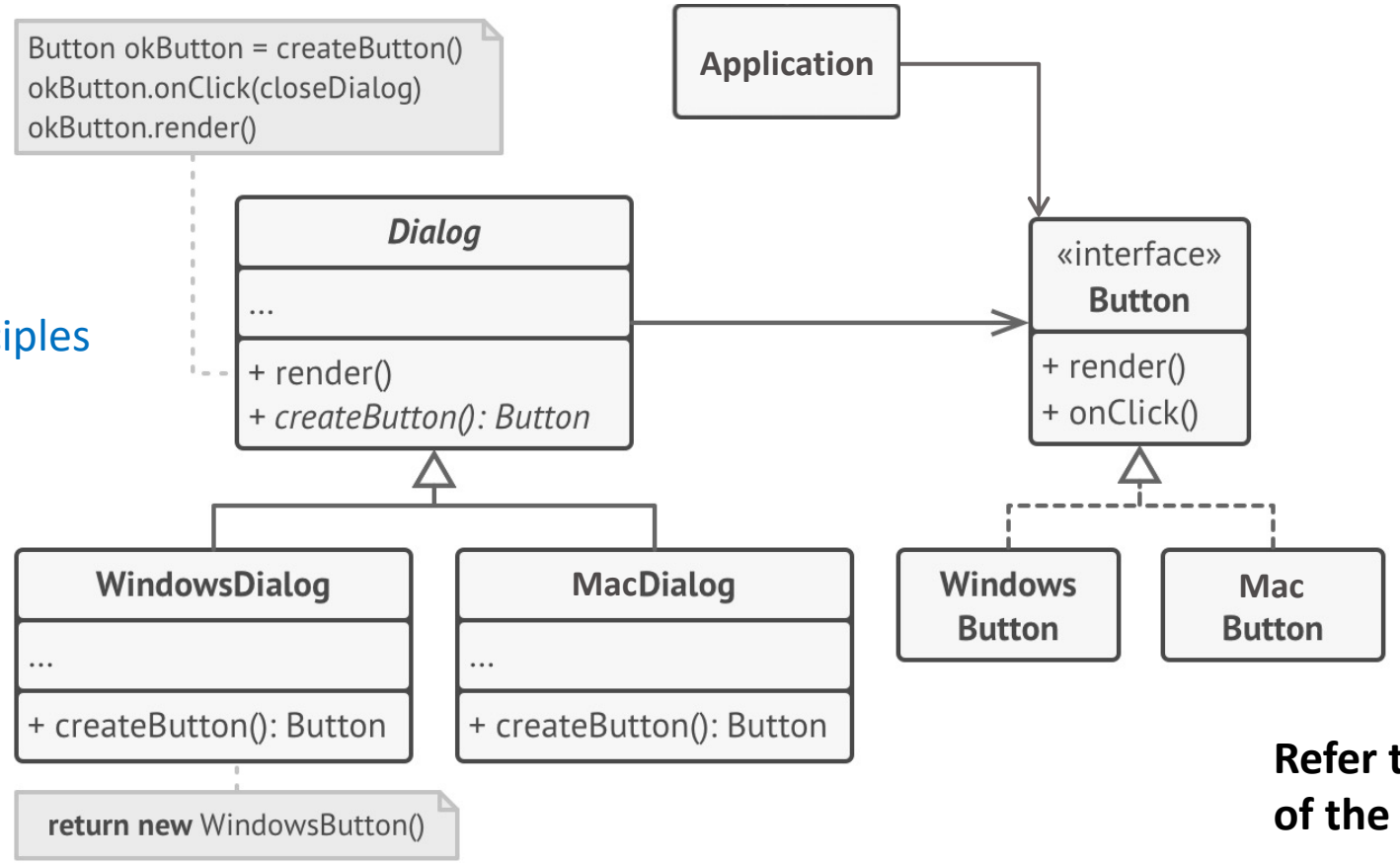
- It provides an interface for creating a set of objects based on predetermined factor(s) or dependencies

- **Applicability:**
  - When you don't know beforehand the exact types you should request to work with
  - When you want to allow for future extensions
  - When you want to allow for reusability of existing objects

CREATIONAL PATTERNS

FACTORY METHOD: *EXAMPLE*

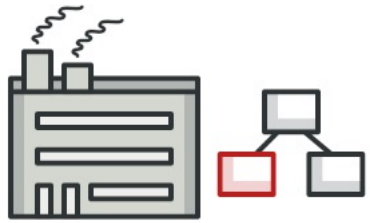
What GRASP design principles are applied here?



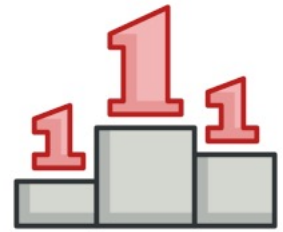
The cross-platform dialog example.

Refer to the Java implementation of the example in the handout

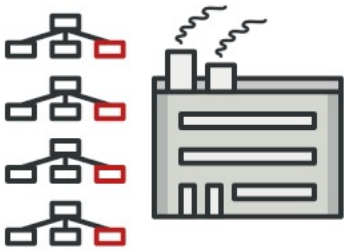
CREATIONAL PATTERNS



Factory Method



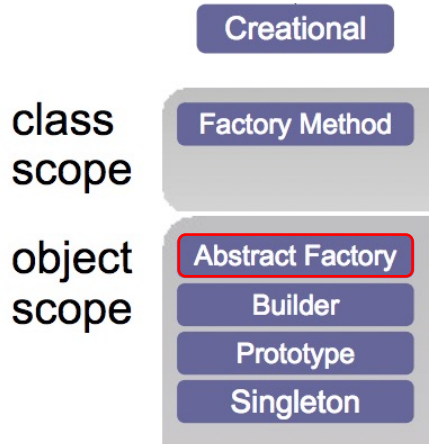
Singleton



Abstract Factory

CREATIONAL PATTERNS

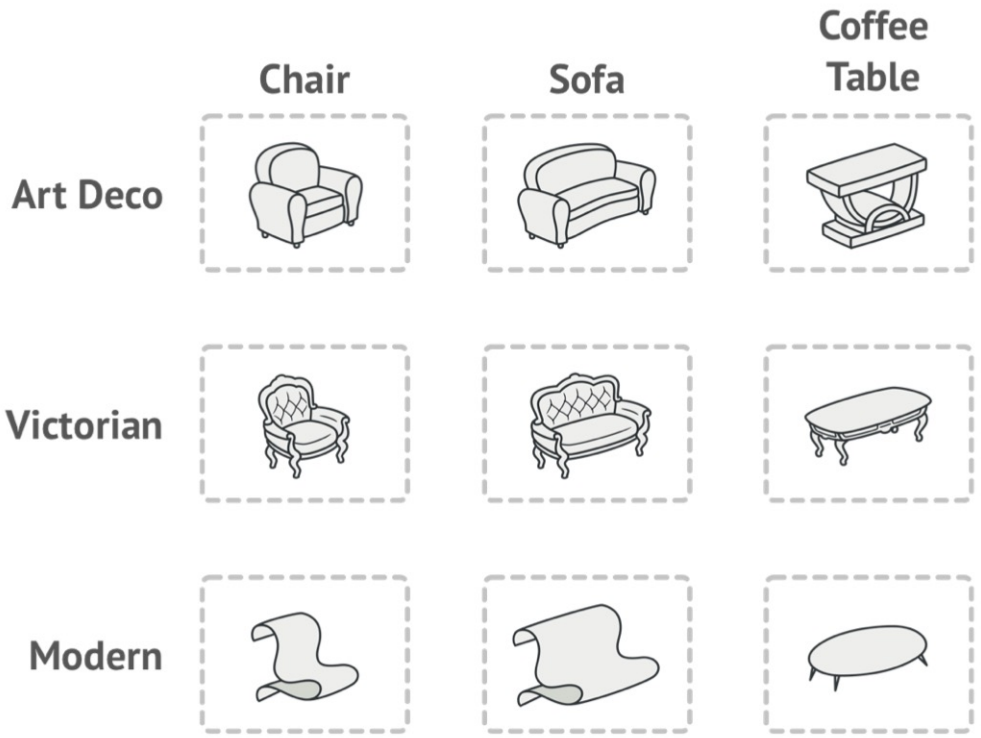
ABSTRACT FACTORY



- Allows you to produce families of related objects without directly specifying their classes

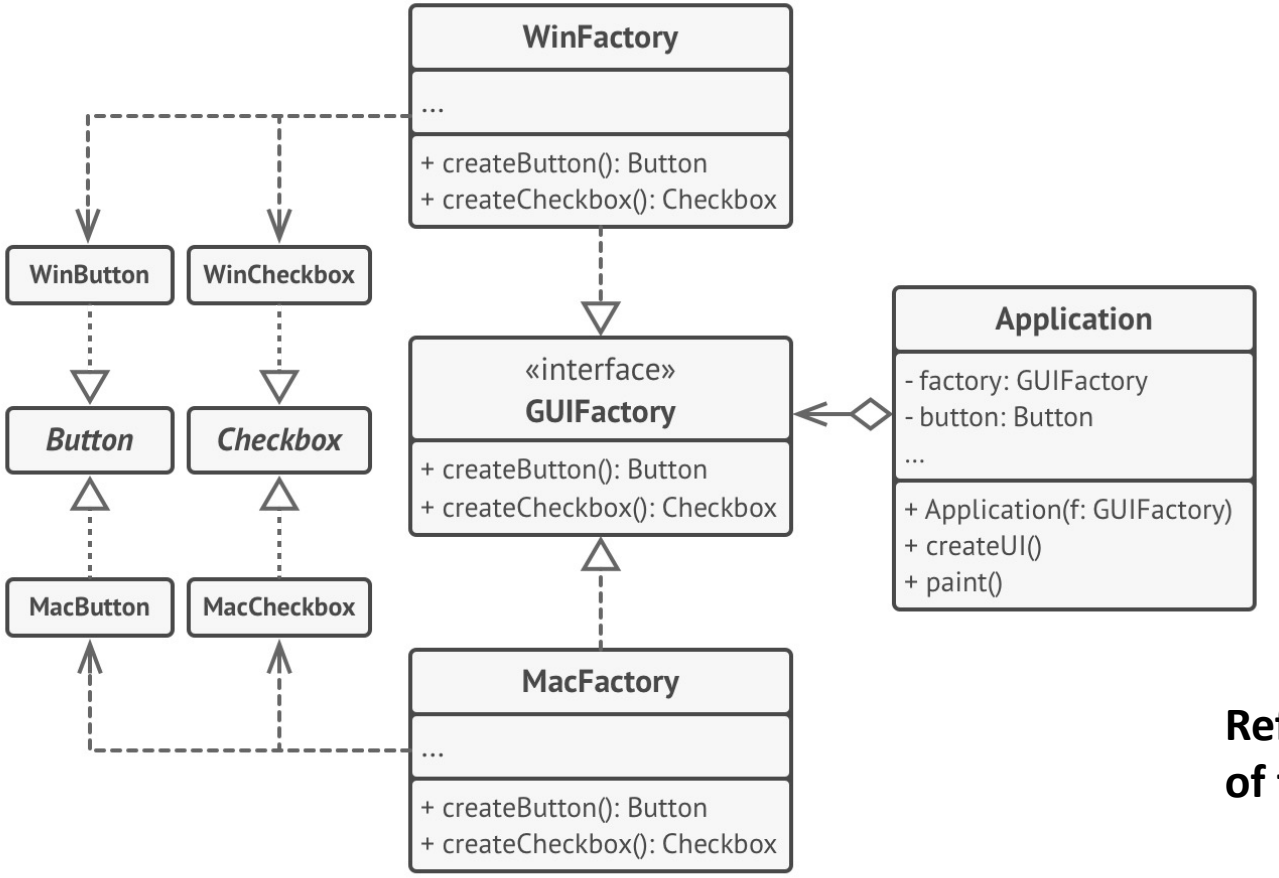
- **Applicability:**
  - When the object classes may be unknown beforehand
  - When you don't want to user to worry about the details of variant objects across families
  - When you simply want to allow for future extensions

CREATIONAL PATTERNS
ABSTRACT FACTORY



Product families and their variants.

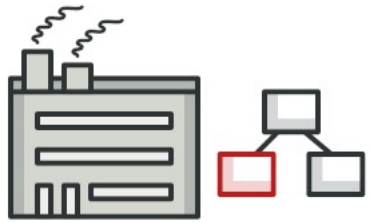
CREATIONAL PATTERNS
ABSTRACT FACTORY: *EXAMPLE*



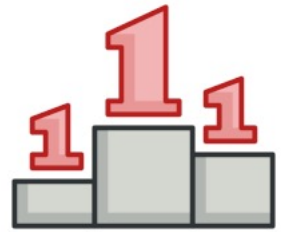
The cross-platform UI classes example.

Refer to the Java implementation of the example in the handout

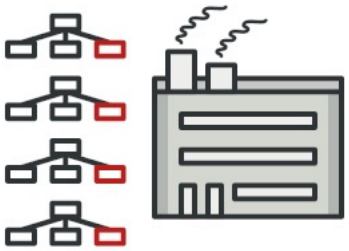
CREATIONAL PATTERNS



Factory Method



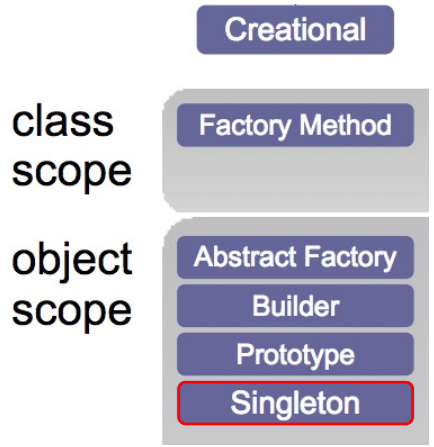
Singleton



Abstract Factory

CREATIONAL PATTERNS

SINGLETON



- Ensures that a class has only one object, while providing a global access point to that object.

- **Applicability:**
  - When a class in your program should have just a **single** instance available **to all clients to use**
  - When you need **stricter control** over global variables
    - The Singleton will guarantee that there is just one object of a class

CREATIONAL PATTERNS
SINGLETON : *EXAMPLE*