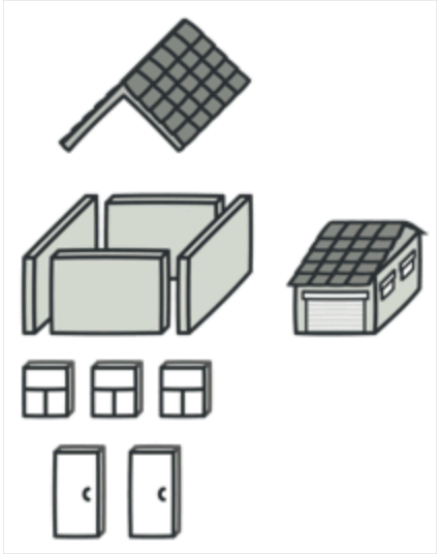
Builder

- Creación de objetos complejos
- atributos pueden ser otros objetos que hay que construir

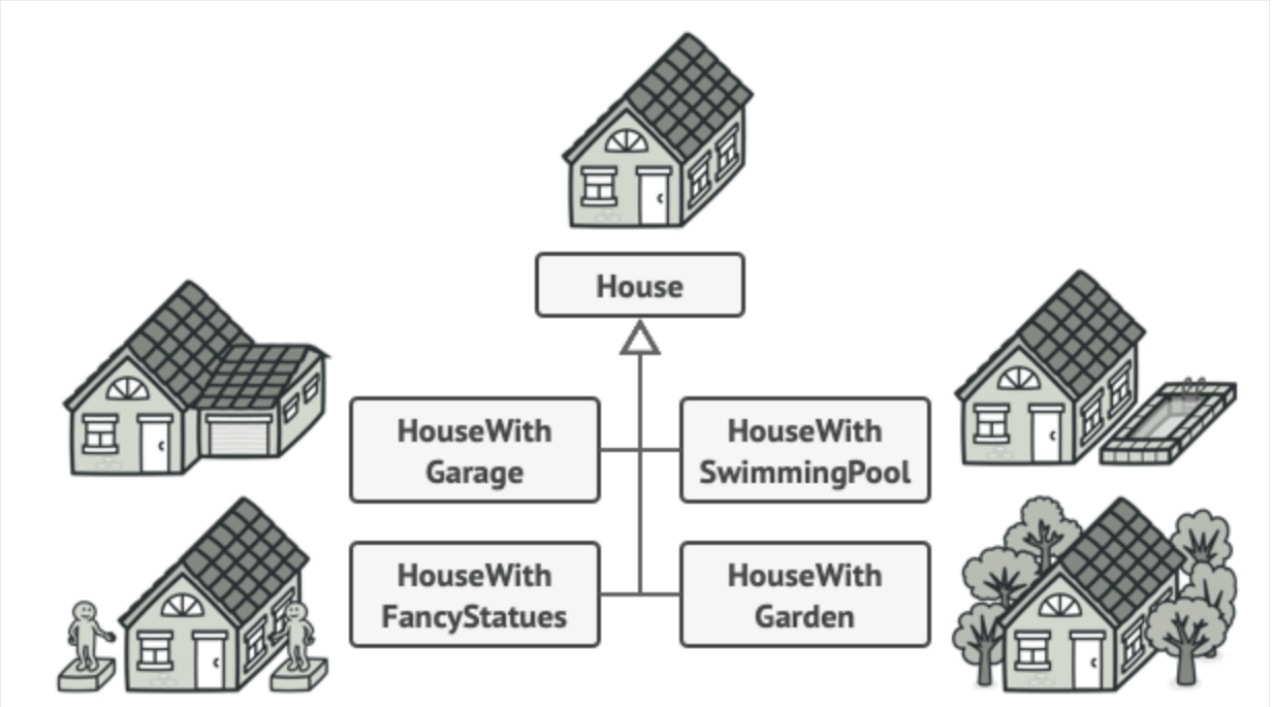
```
class User
  attr_accessor :first_name, :last_name, :birthday,
                :gender, :roles, :status, :email, :password

  def initialize(first_name=nil, last_name=nil, birthday=nil,
                 gender=nil, roles=[], status=nil, email=nil, password=nil)
    @first_name = first_name
    @last_name = last_name
    @birthday = birthday
    @gender = gender
    @roles = roles
    @status = status
    @email = email
    @password = password
  end
end

User.new('John', 'Doe', Time.new('1999-03-02'), 'm', ['admin'],
         'active', 'test@test.com', 'abcdef')
```

Dos Problemas

- el constructor del objeto es demasiado complejo
- cuesta tener variaciones del constructor
- Por ejemplo queremos un constructor para objetos de clase House
  - objeto tiene paredes, puertas, ventanas , techo
  - ¿que pasa si queremos variedades de casas (distintos materiales, con jardín, con piscina, etc) ?

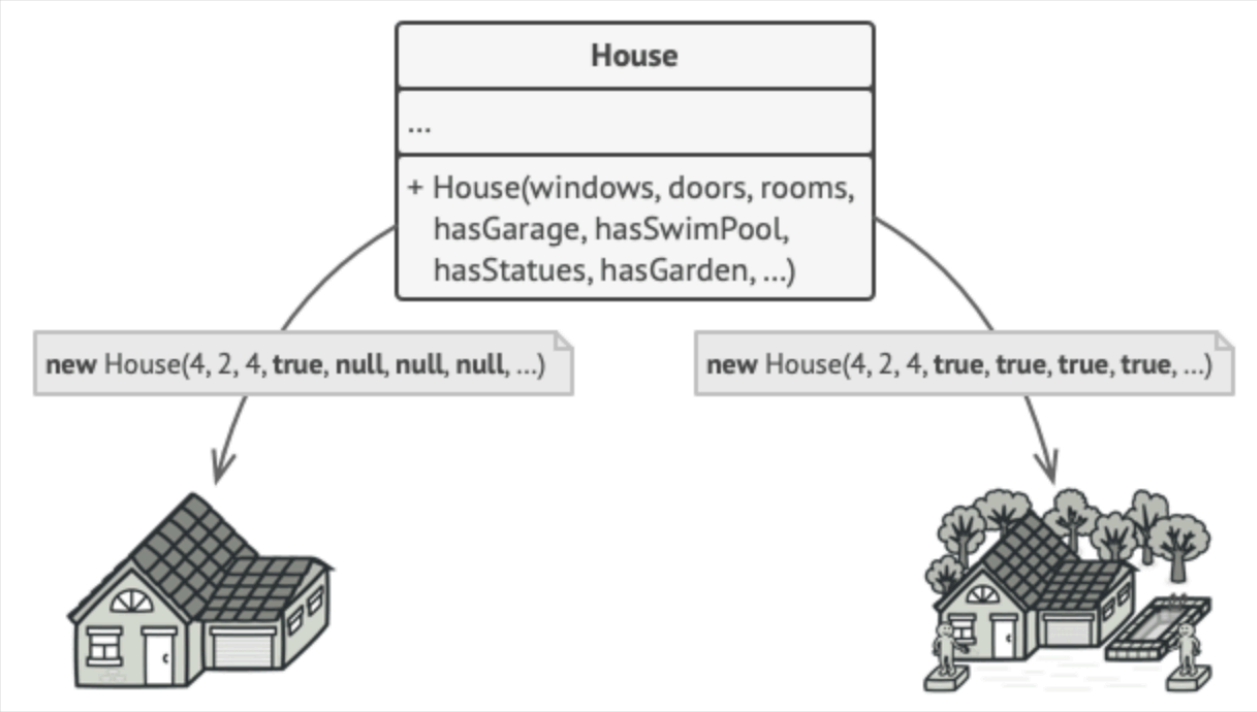


La herencia siempre salva ...



¿O no?

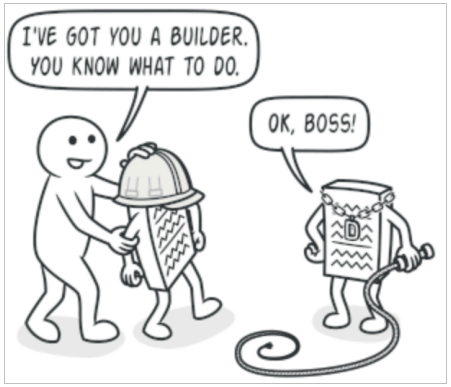
Constructor queda con elementos sin uso



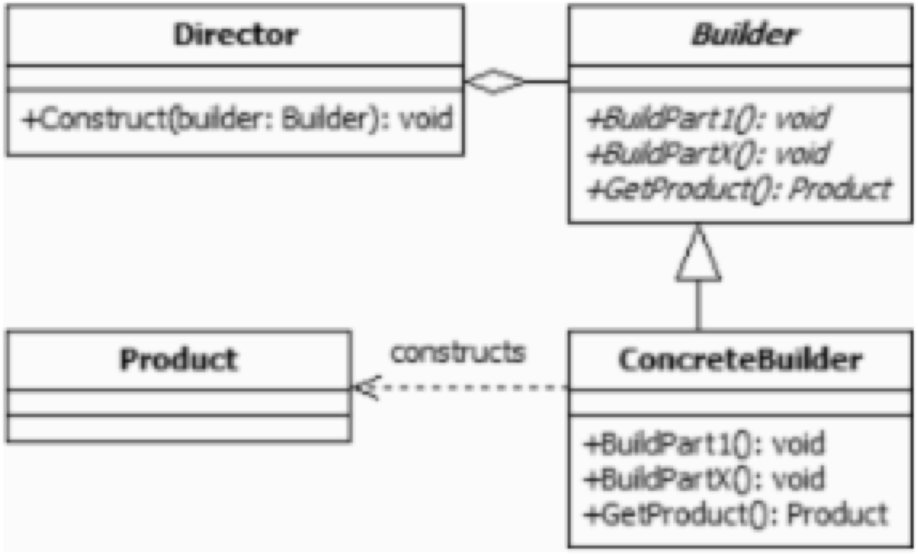
La esencia del patrón builder

- Pasar a otro objeto la responsabilidad de construir las partes del objeto a crear: un builder

- La clase builder provee de métodos para crear una casa y para agregarle cada uno de los elementos necesarios
- Si se requieren variaciones de la casa se usan varios builders
- Un objeto director dirige la construcción del builder (no siempre aparece)



El patrón formalmente



- El director sabe construir los objetos gracias a la referencia a un builder
- El builder genera la instancia del objeto a construir y proporciona métodos para que el director lo construya

Un Ejemplo: UserBuilder

```
class UserBuilder
  def self.build
    builder = new
    yield(builder)
    builder.user
  end

  def initialize
    @user = User.new
  end

  def set_name(first_name, last_name)
    @user.first_name = first_name
    @user.last_name = last_name
  end

  def set_birthday(birthday)
    @user.birthday = Time.new(birthday)
  end

  def set_as_active
    @user.status = 'active'
  end

  def set_as_on_hold
      @user.status = 'on_hold'
  end

  def set_as_men
    @user.gender = 'm'
  end

  def set_as_women
    @user.gender = 'f'
  end

  def set_as_admin
    @user.roles = ['admin']
  end

  def set_login_credentials(email, password)
    @user.email = email
    @user.password = password
  end

  def user
    @user
  end
end
```

y luego

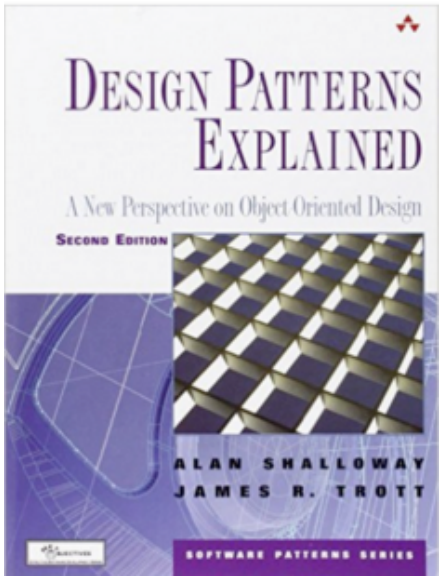
```
juan = UserBuilder.build do |theBuilder|
    theBuilder.set_name('Juan', 'Perez')
    theBuilder.set_as_on_hold
end

puts ("#{juan.first_name} #{juan.last_name} esta #{juan.status}")
```

- Observaciones
  - juan es un objeto User no un objeto UserBuilder (por eso se puede usar métodos first_name, last_name y status)
  - UserBuilder.build le pasa el control al bloque que viene después del do pasandole el builder asociado a ese usuario
  - Métodos set_name y ser_as_on_hold del UserBuilder terminan de construir al User juan

Patrones Estudiados

Libros


Design Patterns
Elements of Reusable Object-Oriented Software
Erich Gamma
Richard Helm
Ralph Johnson
John Vlissides
Foreword by Grady Booch
ADDISON-WESLEY PROFESSIONAL COMPUTING SERIES


DESIGN PATTERNS EXPLAINED
A New Perspective on Object Oriented Design
SECOND EDITION
ALAN SHALLOWAY
JAMES R. TROTT
SOFTWARE PATTERNS SERIES

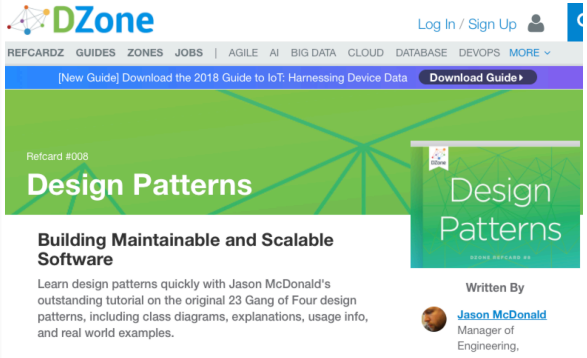

O'REILLY
10th Anniversary Updated for Java 8
Head First
Design Patterns
A Brain-Friendly Guide
Avoid those embarrassing coupling mistakes
Learn why everything your friends know about Factory pattern is probably wrong
Discover the secrets of the Patterns Guru
Load the patterns that matter straight into your brain
Find out how Starbuzz Coffee doubled their stock price with the Decorator pattern
See why Jim's love life improved when he cut down his inheritance
Eric Freeman & Elisabeth Robson
with Kathy Sierra & Bert Bates


Addison-Wesley Professional Ruby Series
DESIGN PATTERNS IN RUBY
Foreword by Obie Fernandez
Professional Ruby Series Editor
RUSS OLSEN

Links

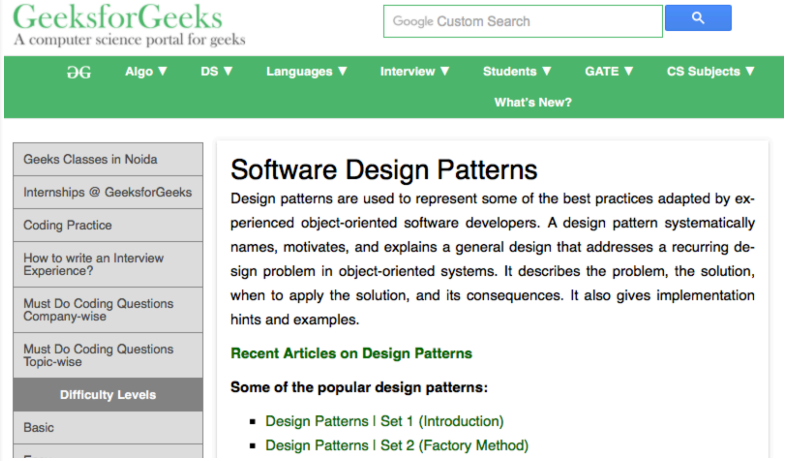
OODesign.com

http://www.oodesign.com



DZone.com

https://dzone.com/refcardz/design-patterns



Geeks for Geeks

http://www.geeksforgeeks.org/software-design-patterns/