AYUDANTÍA

PREPARACIÓN I1

IIC2143 - INGENIERÍA DE SOFTWARE

RELATOS DE USUARIO

CONCEPTOS IMPORTANTES

- Épicas
- Relatos de Usuario
- Condiciones de Satisfacción

TÍTULO

Yo, como <rol de usuario>
quiero <meta> para
<beneficio>.

CONDICIONES DE SATISFACCIÓN

Dado <contexto> cuando <una acción es realizada por el usuario> entonces <el resultado observable de la acción>.

NOTAR: En este caso, el formato es una recomendación y puede variar

PLANIFICACIÓN Y ESTIMACIÓN

TAMAÑO, DURACIÓN, Y ESFUERZO

- Tamaño: puntos de relato, relatos de usuario.
- Duración: días, semanas.
- Esfuerzo: horas/hombre, semanas/hombre.

AJUSTE Y NEGOCIACIÓN

- Por alcance: dejar fuera algunos relatos de usuario.
- Por tiempo: extender el plazo de entrega.
- Por calidad: comprometer la calidad para cumplir con el tamaño completo en el tiempo deseado (Nunca debe hacerse).

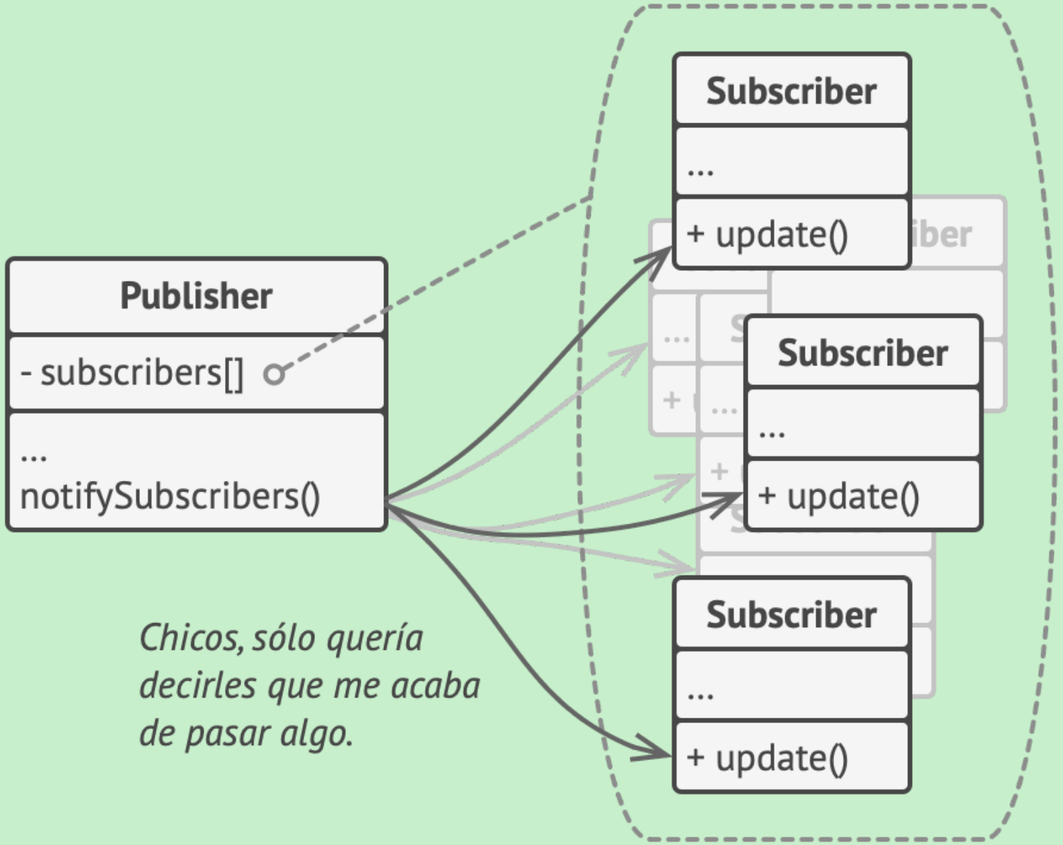
PATRONES DE DISEÑO

TRES CATEGORÍAS

- Comportamiento
- Creacional
- Estructural

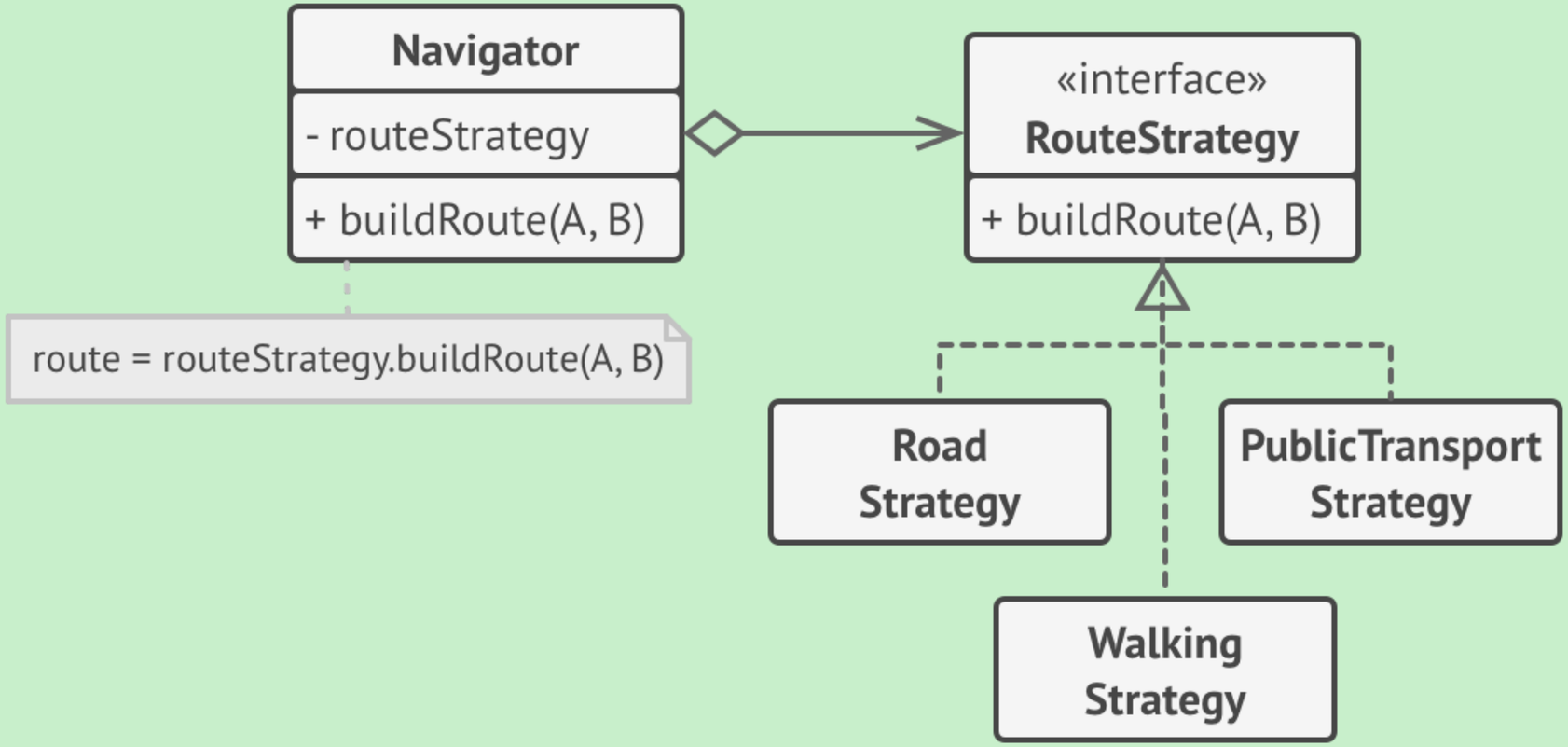
COMPORTAMIENTO

Observer



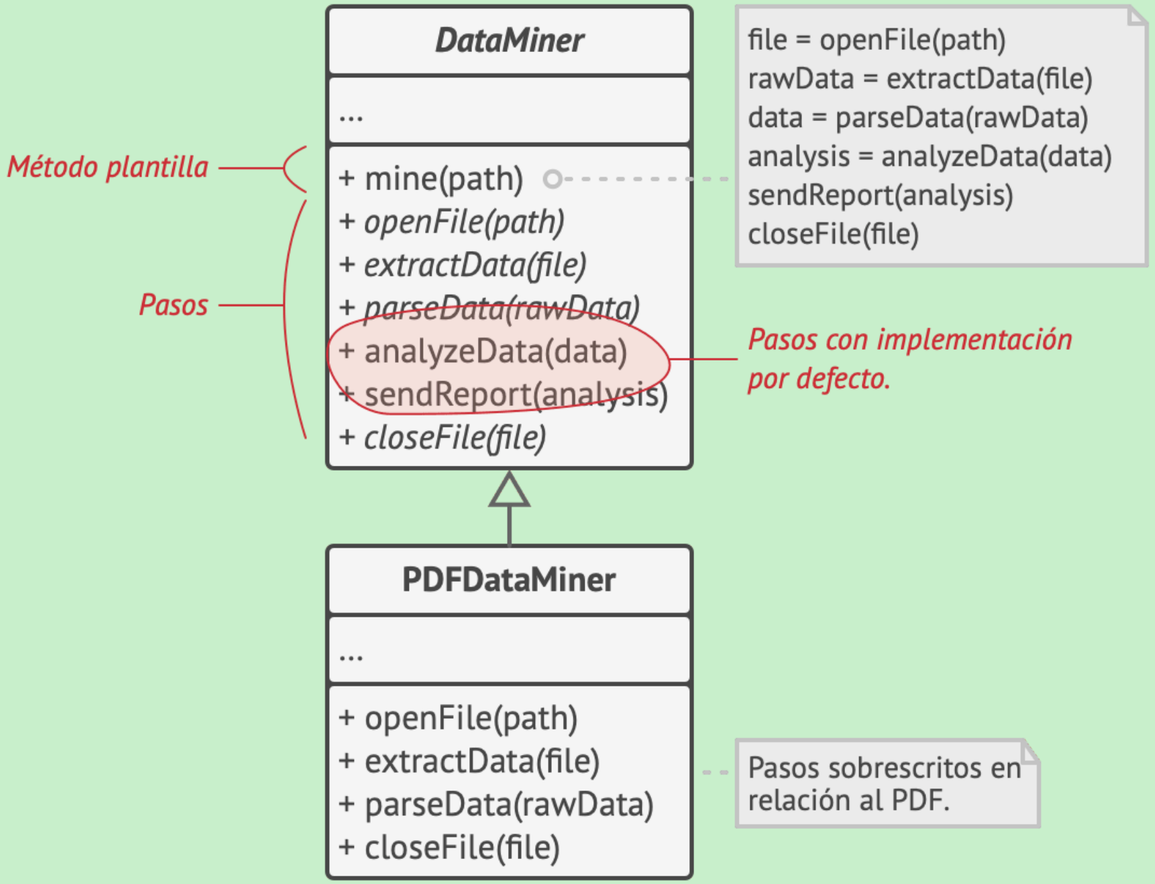
COMPORTAMIENTO

Strategy



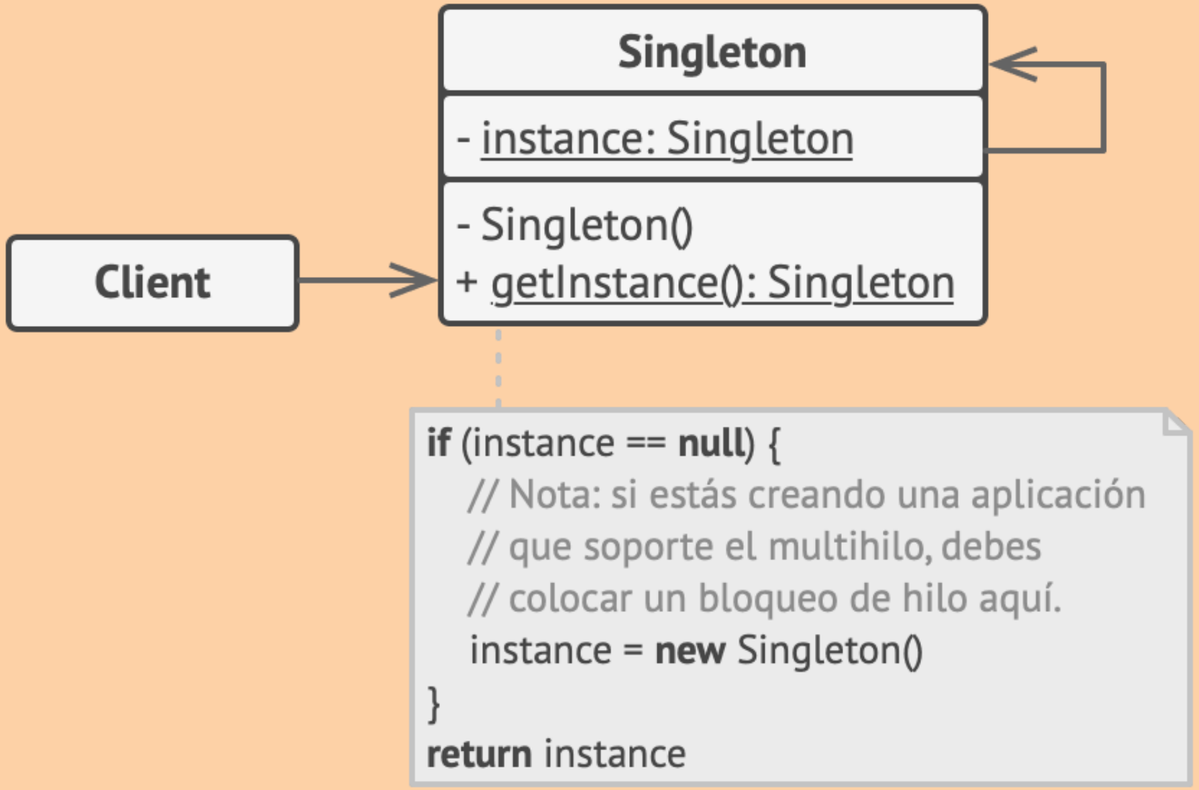
COMPORTAMIENTO

Template



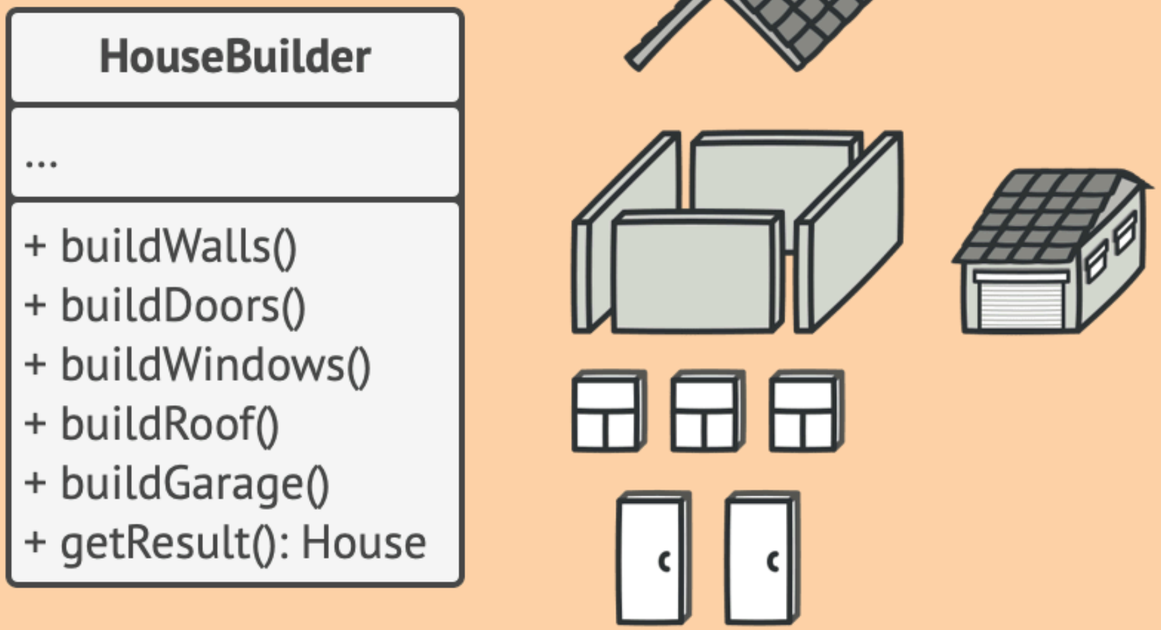
CREACIONAL

Singleton



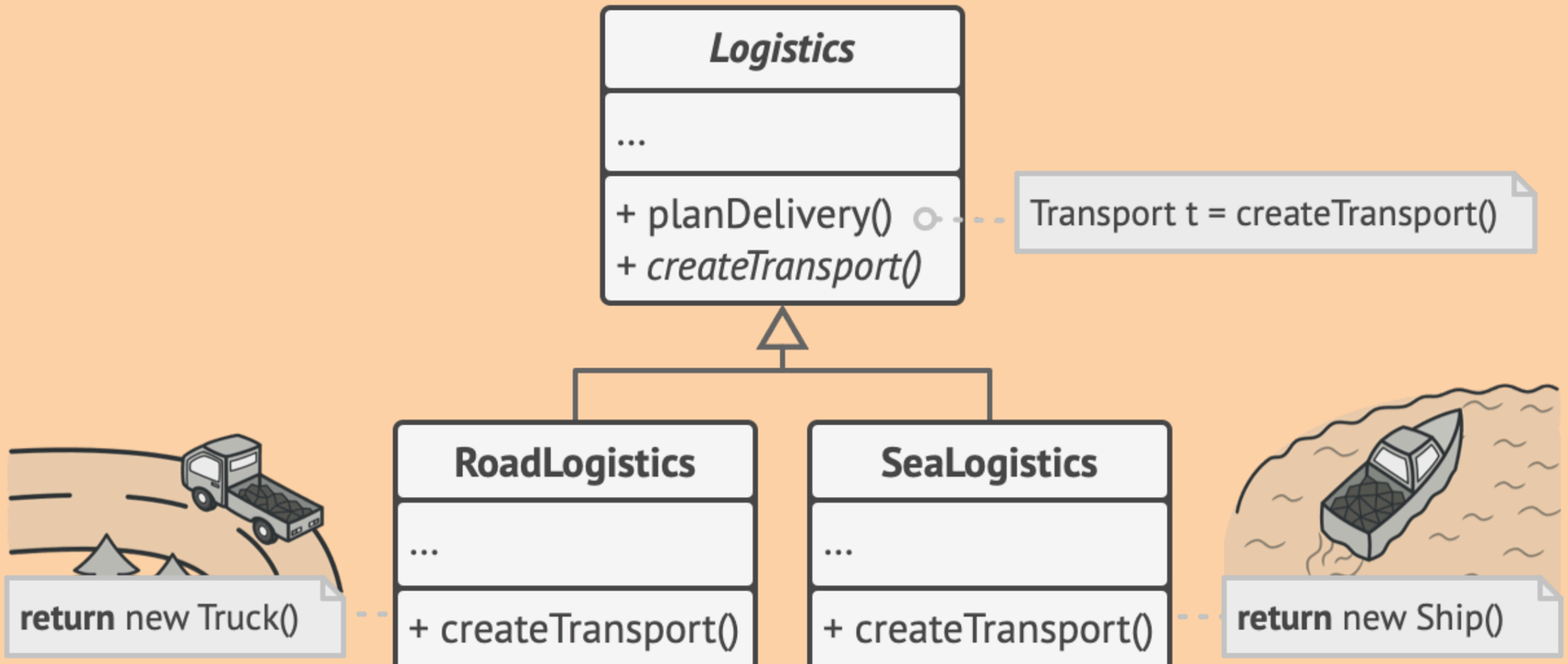
CREACIONAL

Builder



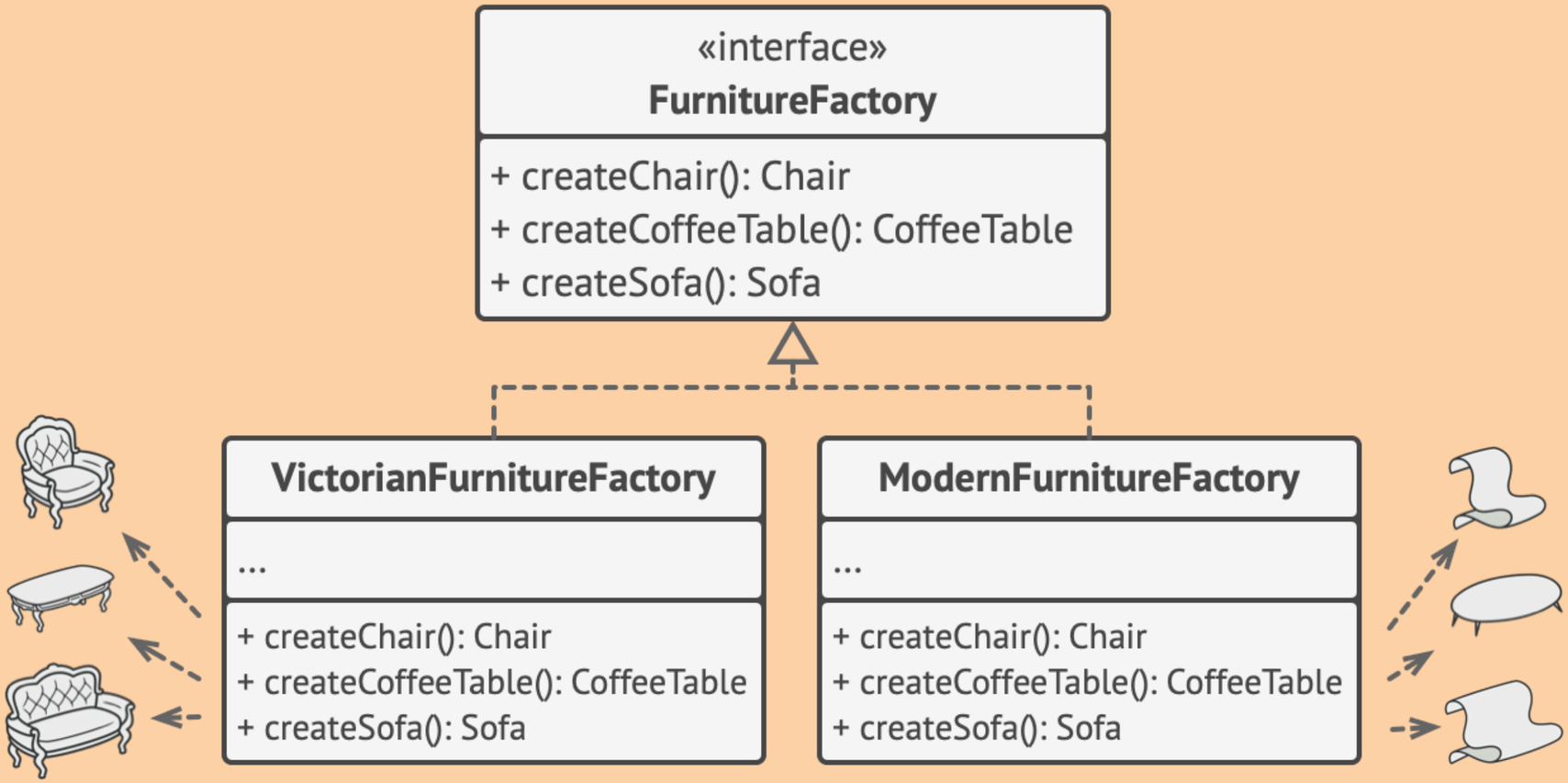
CREACIONAL

Factory



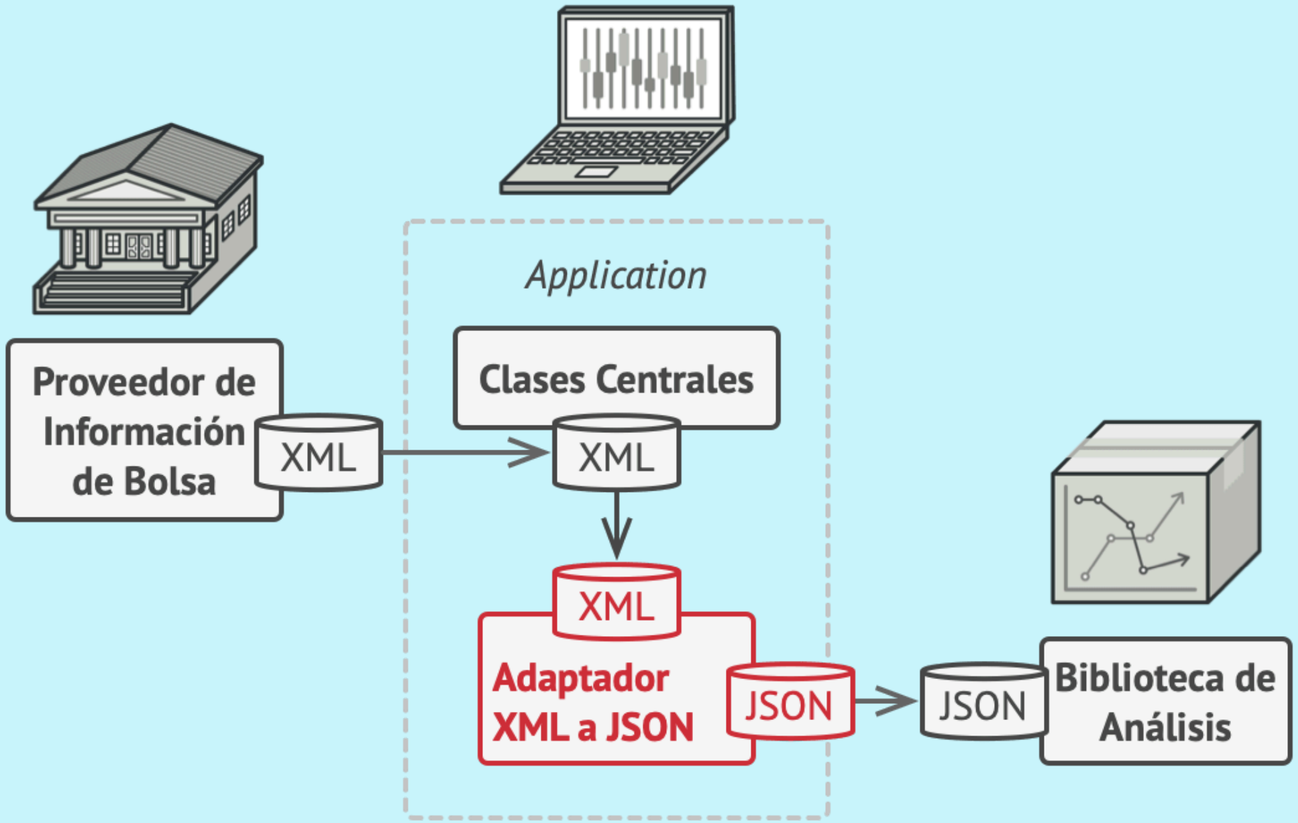
CREACIONAL

Abstract Factory



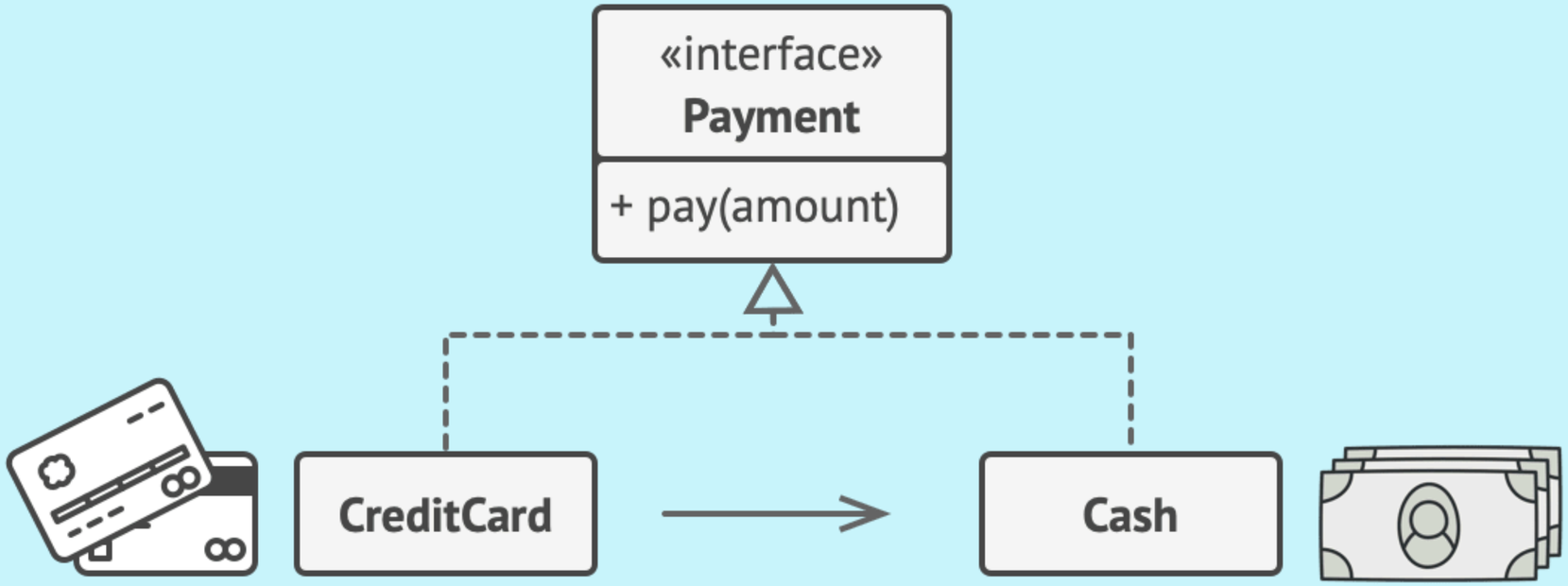
ESTRUCTURAL

Adapter



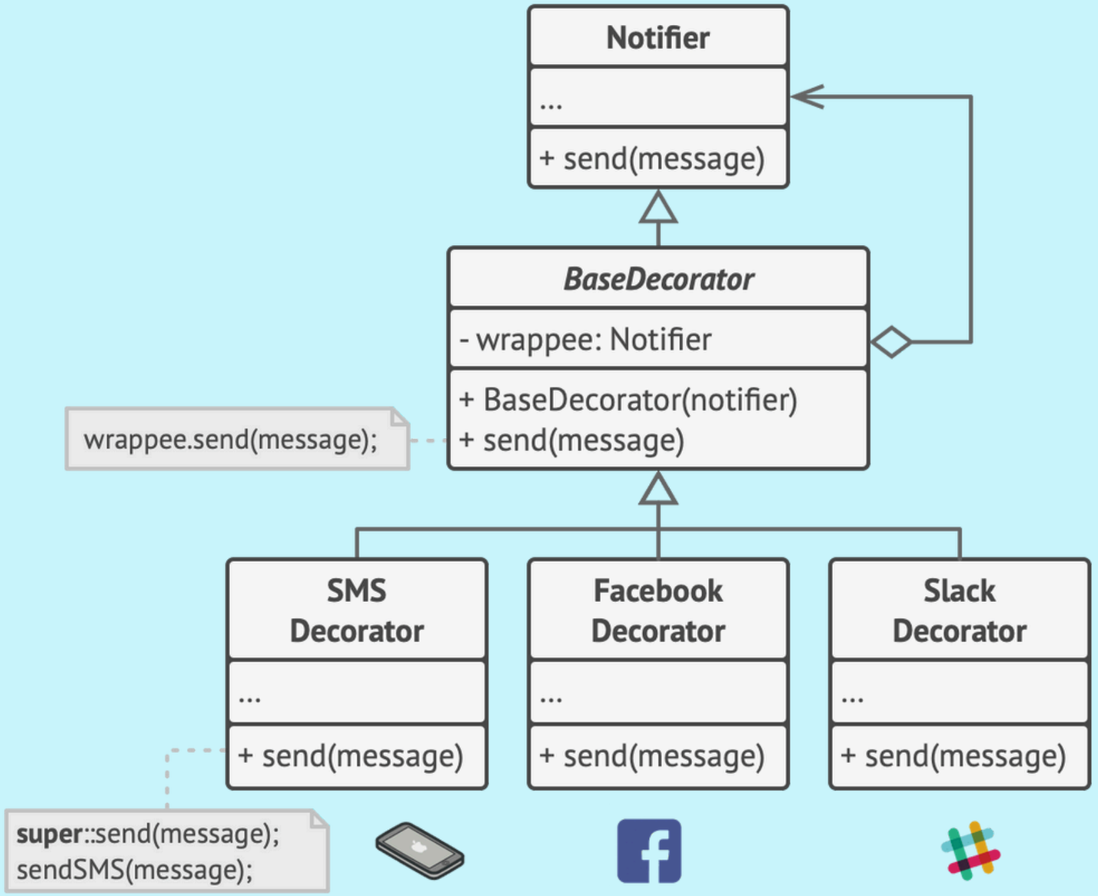
ESTRUCTURAL

Proxy



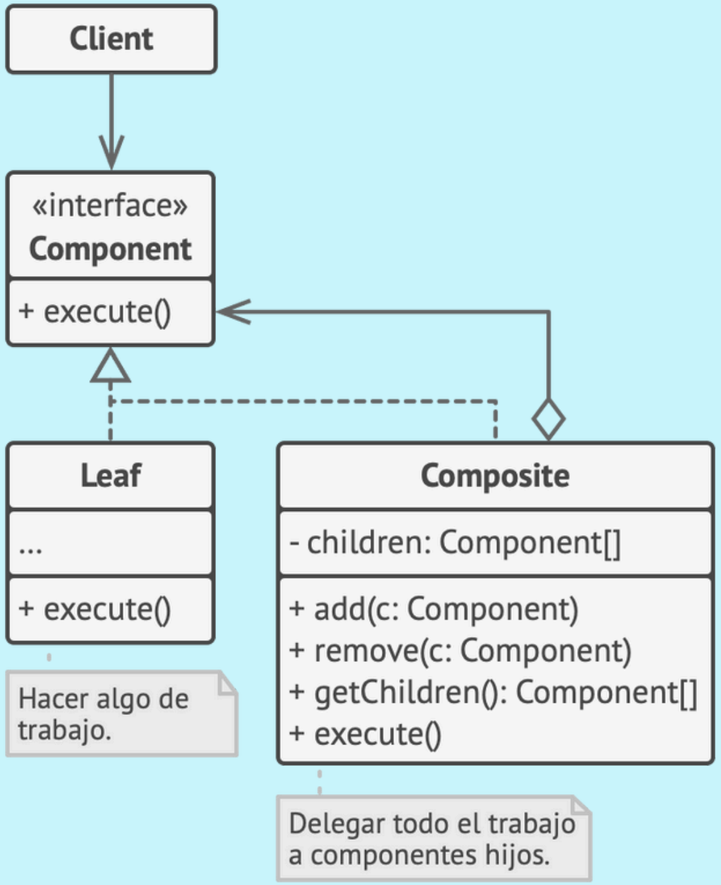
ESTRUCTURAL

Decorator



ESTRUCTURAL

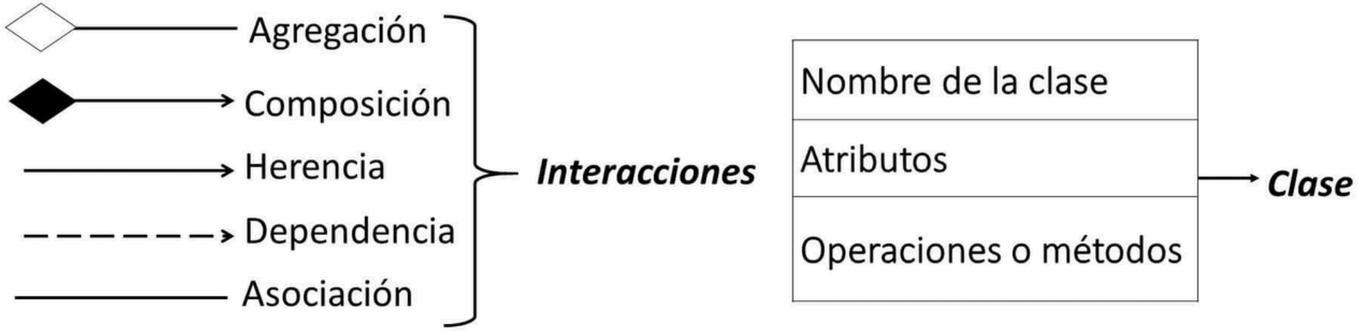
Composite

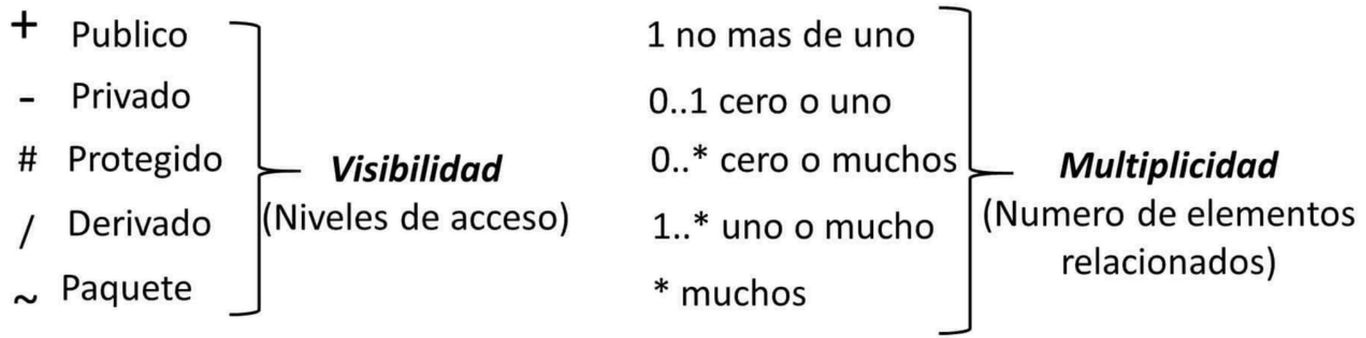


PARA MÁS INFORMACIÓN

Web de patrones

Elementos y símbolos en los diagramas de clases UML





EJERCICIO ESTIMACIÓN

Tu equipo de desarrollo ha realizado un levantamiento de requerimientos funcionales para un software de control de activo fijo de un cliente. Los relatos de usuario se han priorizado y catalogado en 3 niveles de complejidad (se usará una escala exponencial de story points comenzando con 1 punto para los relatos de complejidad baja, 2 y 4 puntos para mediana y alta respectivamente). La tabla muestra para cada relato, su nivel de complejidad y la prioridad que le ha asignado el product owner.

Se ha comprometido la entrega en 8 semanas. Se acaba de completar el primer sprint (de 2 semanas) en que el equipo, formado por 5 desarrolladores, logró implementar los relatos US01, US03, US05 y US10.

Relato	Complejidad	Prioridad
US01	mediana	2
US02	mediana	8
US03	baja	1
US04	mediana	5
US05	alta	4
US06	baja	6
US07	alta	10
US08	mediana	9
US09	alta	7
US10	baja	3
US11	mediana	12
US12	mediana	11
US13	alta	16
US14	mediana	14
US15	mediana	15
US16	baja	13
US17	mediana	17
US18	mediana	18
US19	alta	20
US20	alta	19

1. Haz una planificación para las iteraciones que restan antes del release. En caso que sea necesario hacer ajustes, diseñe una estrategia de negociación basada en ajuste de calendario y otra basada en ajuste por alcance. ¿Cuál estrategia es mejor? Justifique su respuesta.
2. Cuantifica tamaño y duración del proyecto en el supuesto de que se acepte un ajuste por calendario.

Relato	Complejidad	Prioridad
US01	mediana	2
US02	mediana	8
US03	baja	1
US04	mediana	5
US05	alta	4
US06	baja	6
US07	alta	10
US08	mediana	9
US09	alta	7
US10	baja	3
US11	mediana	12
US12	mediana	11
US13	alta	16
US14	mediana	14
US15	mediana	15
US16	baja	13
US17	mediana	17
US18	mediana	18
US19	alta	20
US20	alta	19

Tu equipo de desarrollo ha realizado un levantamiento de requerimientos funcionales para un software de control de activo fijo de un cliente. Los relatos de usuario se han priorizado y catalogado en 3 niveles de complejidad (se usará una escala exponencial de story points comenzando con 1 punto para los relatos de complejidad baja, 2 y 4 puntos para mediana y alta respectivamente). La tabla muestra para cada relato, su nivel de complejidad y la prioridad que le ha asignado el product owner.

Se ha comprometido la entrega en 8 semanas. Se acaba de completar el primer sprint (de 2 semanas) en que el equipo, formado por 5 desarrolladores, logró implementar los relatos US01, US03, US05 y US10.

Relato	Complejidad	Prioridad
US01	mediana	2
US02	mediana	8
US03	baja	1
US04	mediana	5
US05	alta	4
US06	baja	6
US07	alta	10
US08	mediana	9
US09	alta	7
US10	baja	3
US11	mediana	12
US12	mediana	11
US13	alta	16
US14	mediana	14
US15	mediana	15
US16	baja	13
US17	mediana	17
US18	mediana	18
US19	alta	20
US20	alta	19

Tu equipo de desarrollo ha realizado un levantamiento de requerimientos funcionales para un software de control de activo fijo de un cliente. Los relatos de usuario se han priorizado y catalogado en 3 niveles de complejidad (se usará una escala exponencial de story points comenzando con 1 punto para los relatos de complejidad baja, 2 y 4 puntos para mediana y alta respectivamente). La tabla muestra para cada relato, su nivel de complejidad y la prioridad que le ha asignado el product owner.

Se ha comprometido la entrega en 8 semanas. Se acaba de completar el primer sprint (de 2 semanas) en que el equipo, formado por 5 desarrolladores, logró implementar los relatos US01, US03, US05 y US10.

Relato	Story Points	Prioridad
US03	1	1
US01	2	2
US10	1	3
US05	4	4
US04	2	5
US06	1	6
US09	4	7
US02	2	8
US08	2	9
US07	4	10
US12	2	11
US11	2	12
US16	1	13
US14	2	14
US15	2	15
US13	4	16
US17	2	17
US18	2	18
US20	4	19
US19	4	20

NOTAR: Ordeno según prioridad

Se ha comprometido la entrega en 8 semanas. Se acaba de completar el primer sprint (de 2 semanas) en que el equipo, formado por 5 desarrolladores, logró implementar los relatos US01, US03, US05 y US10.

Relato	Story Points	Prioridad
US03	1	1
US01	2	2
US10	1	3
US05	4	4
US04	2	5
US06	1	6
US09	4	7
US02	2	8
US08	2	9
US07	4	10
US12	2	11
US11	2	12
US16	1	13
US14	2	14
US15	2	15
US13	4	16
US17	2	17
US18	2	18
US20	4	19
US19	4	20

Se ha comprometido la entrega en 8 semanas. Se acaba de completar el primer sprint (de 2 semanas) en que el equipo, formado por 5 desarrolladores, logró implementar los relatos US01, US03, US05 y US10.

---

8 semanas
2 semanas/sprint

Relato	Story Points	Prioridad
US03	1	1
US01	2	2
US10	1	3
US05	4	4
US04	2	5
US06	1	6
US09	4	7
US02	2	8
US08	2	9
US07	4	10
US12	2	11
US11	2	12
US16	1	13
US14	2	14
US15	2	15
US13	4	16
US17	2	17
US18	2	18
US20	4	19
US19	4	20

Se ha comprometido la entrega en 8 semanas. Se acaba de completar el primer sprint (de 2 semanas) en que el equipo, formado por 5 desarrolladores, logró implementar los relatos US01, US03, US05 y US10.

---

8 semanas
2 semanas/sprint
4 sprints

Sprint 1:
Sprint 2:
Sprint 3:
Sprint 4:

Relato	Story Points	Prioridad
US03	1	1
US01	2	2
US10	1	3
US05	4	4
US04	2	5
US06	1	6
US09	4	7
US02	2	8
US08	2	9
US07	4	10
US12	2	11
US11	2	12
US16	1	13
US14	2	14
US15	2	15
US13	4	16
US17	2	17
US18	2	18
US20	4	19
US19	4	20

Se ha comprometido la entrega en 8 semanas. Se acaba de completar el primer sprint (de 2 semanas) en que el equipo, formado por 5 desarrolladores, logró implementar los relatos US01, US03, US05 y US10.

---

8 semanas
2 semanas/sprint
4 sprints

Sprint 1:
Sprint 2:
Sprint 3:
Sprint 4:

NOTAR: Cuento con un historial para hacer la planificación y estimación

Relato	Story Points	Prioridad
US03	1	1
US01	2	2
US10	1	3
US05	4	4
US04	2	5
US06	1	6
US09	4	7
US02	2	8
US08	2	9
US07	4	10
US12	2	11
US11	2	12
US16	1	13
US14	2	14
US15	2	15
US13	4	16
US17	2	17
US18	2	18
US20	4	19
US19	4	20

Se ha comprometido la entrega en 8 semanas. Se acaba de completar el primer sprint (de 2 semanas) en que el equipo, formado por 5 desarrolladores, logró implementar los relatos US01, US03, US05 y US10.

---

8 semanas
2 semanas/sprint
4 sprints

Sprint 1: US01, US03, US05, US10 = 8 SPs
Sprint 2:
Sprint 3:
Sprint 4:

NOTAR: Cuento con un historial para hacer la planificación y estimación

Relato	Story Points	Prioridad
US03	1	1
US01	2	2
US10	1	3
US05	4	4
US04	2	5
US06	1	6
US09	4	7
US02	2	8
US08	2	9
US07	4	10
US12	2	11
US11	2	12
US16	1	13
US14	2	14
US15	2	15
US13	4	16
US17	2	17
US18	2	18
US20	4	19
US19	4	20

Se ha comprometido la entrega en 8 semanas. Se acaba de completar el primer sprint (de 2 semanas) en que el equipo, formado por 5 desarrolladores, logró implementar los relatos US01, US03, US05 y US10.

---

8 semanas
2 semanas/sprint
4 sprints

Sprint 1: US01, US03, US05, US10 = 8 SPs
Sprint 2:
Sprint 3:
Sprint 4:

Relato	Story Points	Prioridad
US03	1	1
US01	2	2
US10	1	3
US05	4	4
US04	2	5
US06	1	6
US09	4	7
US02	2	8
US08	2	9
US07	4	10
US12	2	11
US11	2	12
US16	1	13
US14	2	14
US15	2	15
US13	4	16
US17	2	17
US18	2	18
US20	4	19
US19	4	20

Se ha comprometido la entrega en 8 semanas. Se acaba de completar el primer sprint (de 2 semanas) en que el equipo, formado por 5 desarrolladores, logró implementar los relatos US01, US03, US05 y US10.

---

8 semanas
2 semanas/sprint
4 sprints

Sprint 1: US01, US03, US05, US10 = 8 SPs
Sprint 2: US04, US06, US09 = 7 SPs
Sprint 3:
Sprint 4:

NOTAR: ±1 es un desfase aceptable

Relato	Story Points	Prioridad
US03	1	1
US01	2	2
US10	1	3
US05	4	4
US04	2	5
US06	1	6
US09	4	7
US02	2	8
US08	2	9
US07	4	10
US12	2	11
US11	2	12
US16	1	13
US14	2	14
US15	2	15
US13	4	16
US17	2	17
US18	2	18
US20	4	19
US19	4	20

Se ha comprometido la entrega en 8 semanas. Se acaba de completar el primer sprint (de 2 semanas) en que el equipo, formado por 5 desarrolladores, logró implementar los relatos US01, US03, US05 y US10.

---

8 semanas
2 semanas/sprint
4 sprints

Sprint 1: US01, US03, US05, US10 = 8 SPs
Sprint 2: US04, US06, US09, US02 = 9 SPs
Sprint 3:
Sprint 4:

NOTAR: ±1 es un desfase aceptable

Relato	Story Points	Prioridad
US03	1	1
US01	2	2
US10	1	3
US05	4	4
US04	2	5
US06	1	6
US09	4	7
US02	2	8
US08	2	9
US07	4	10
US12	2	11
US11	2	12
US16	1	13
US14	2	14
US15	2	15
US13	4	16
US17	2	17
US18	2	18
US20	4	19
US19	4	20

Se ha comprometido la entrega en 8 semanas. Se acaba de completar el primer sprint (de 2 semanas) en que el equipo, formado por 5 desarrolladores, logró implementar los relatos US01, US03, US05 y US10.

---

8 semanas
2 semanas/sprint
4 sprints

Sprint 1: US01, US03, US05, US10 = 8 SPs
Sprint 2: US04, US06, US09, US02 = 9 SPs
Sprint 3: US08, US07, US12 = 8 SPs
Sprint 4:

Relato	Story Points	Prioridad
US03	1	1
US01	2	2
US10	1	3
US05	4	4
US04	2	5
US06	1	6
US09	4	7
US02	2	8
US08	2	9
US07	4	10
US12	2	11
US11	2	12
US16	1	13
US14	2	14
US15	2	15
US13	4	16
US17	2	17
US18	2	18
US20	4	19
US19	4	20

Se ha comprometido la entrega en 8 semanas. Se acaba de completar el primer sprint (de 2 semanas) en que el equipo, formado por 5 desarrolladores, logró implementar los relatos US01, US03, US05 y US10.

---

8 semanas
2 semanas/sprint
4 sprints

Sprint 1: US01, US03, US05, US10 = 8 SPs
Sprint 2: US04, US06, US09, US02 = 9 SPs
Sprint 3: US08, US07, US12 = 8 SPs
Sprint 4: US11, US16, US14, US15 = 7 SPs

Relato	Story Points	Prioridad
US03	1	1
US01	2	2
US10	1	3
US05	4	4
US04	2	5
US06	1	6
US09	4	7
US02	2	8
US08	2	9
US07	4	10
US12	2	11
US11	2	12
US16	1	13
US14	2	14
US15	2	15
US13	4	16
US17	2	17
US18	2	18
US20	4	19
US19	4	20

Se ha comprometido la entrega en 8 semanas. Se acaba de completar el primer sprint (de 2 semanas) en que el equipo, formado por 5 desarrolladores, logró implementar los relatos US01, US03, US05 y US10.

---

8 semanas
2 semanas/sprint
4 sprints

Sprint 1: US01, US03, US05, US10 = 8 SPs
Sprint 2: US04, US06, US09, US02 = 9 SPs
Sprint 3: US08, US07, US12 = 8 SPs
Sprint 4: US11, US16, US14, US15, US13 = 11 SPs

Relato	Story Points	Prioridad
US03	1	1
US01	2	2
US10	1	3
US05	4	4
US04	2	5
US06	1	6
US09	4	7
US02	2	8
US08	2	9
US07	4	10
US12	2	11
US11	2	12
US16	1	13
US14	2	14
US15	2	15
US13	4	16
US17	2	17
US18	2	18
US20	4	19
US19	4	20

Se ha comprometido la entrega en 8 semanas. Se acaba de completar el primer sprint (de 2 semanas) en que el equipo, formado por 5 desarrolladores, logró implementar los relatos US01, US03, US05 y US10.

---

8 semanas
2 semanas/sprint
4 sprints

Sprint 1: US01, US03, US05, US10 = 8 SPs
Sprint 2: US04, US06, US09, US02 = 9 SPs
Sprint 3: US08, US07, US12 = 8 SPs
Sprint 4: US11, US16, US14, US15 = 7 SPs

Relato	Story Points	Prioridad
US03	1	1
US01	2	2
US10	1	3
US05	4	4
US04	2	5
US06	1	6
US09	4	7
US02	2	8
US08	2	9
US07	4	10
US12	2	11
US11	2	12
US16	1	13
US14	2	14
US15	2	15
US13	4	16
US17	2	17
US18	2	18
US20	4	19
US19	4	20

Se ha comprometido la entrega en 8 semanas. Se acaba de completar el primer sprint (de 2 semanas) en que el equipo, formado por 5 desarrolladores, logró implementar los relatos US01, US03, US05 y US10.

---

8 semanas
2 semanas/sprint
4 sprints

Sprint 1: US01, US03, US05, US10 = 8 SPs
Sprint 2: US04, US06, US09, US02 = 9 SPs
Sprint 3: US08, US07, US12 = 8 SPs
Sprint 4: US11, US16, US14, US15 = 7 SPs

NOTAR: No será posible cumplir con todos los relatos en 8 semanas

Relato	Story Points	Prioridad
US03	1	1
US01	2	2
US10	1	3
US05	4	4
US04	2	5
US06	1	6
US09	4	7
US02	2	8
US08	2	9
US07	4	10
US12	2	11
US11	2	12
US16	1	13
US14	2	14
US15	2	15
US13	4	16
US17	2	17
US18	2	18
US20	4	19
US19	4	20

Se ha comprometido la entrega en 8 semanas. Se acaba de completar el primer sprint (de 2 semanas) en que el equipo, formado por 5 desarrolladores, logró implementar los relatos US01, US03, US05 y US10.

---

10 semanas
2 semanas/sprint
5 sprints

Sprint 1: US01, US03, US05, US10 = 8 SPs
Sprint 2: US04, US06, US09, US02 = 9 SPs
Sprint 3: US08, US07, US12 = 8 SPs
Sprint 4: US11, US16, US14, US15 = 7 SPs
Sprint 5: US13, US17, US18 = 8 SPs

Relato	Story Points	Prioridad
US03	1	1
US01	2	2
US10	1	3
US05	4	4
US04	2	5
US06	1	6
US09	4	7
US02	2	8
US08	2	9
US07	4	10
US12	2	11
US11	2	12
US16	1	13
US14	2	14
US15	2	15
US13	4	16
US17	2	17
US18	2	18
US20	4	19
US19	4	20

Se ha comprometido la entrega en 8 semanas. Se acaba de completar el primer sprint (de 2 semanas) en que el equipo, formado por 5 desarrolladores, logró implementar los relatos US01, US03, US05 y US10.

---

12 semanas
2 semanas/sprint
6 sprints

Sprint 1: US01, US03, US05, US10 = 8 SPs
Sprint 2: US04, US06, US09, US02 = 9 SPs
Sprint 3: US08, US07, US12 = 8 SPs
Sprint 4: US11, US16, US14, US15 = 7 SPs
Sprint 5: US13, US17, US18 = 8 SPs
Sprint 6: US20, US19 = 8 SPs

Relato	Story Points	Prioridad
US03	1	1
US01	2	2
US10	1	3
US05	4	4
US04	2	5
US06	1	6
US09	4	7
US02	2	8
US08	2	9
US07	4	10
US12	2	11
US11	2	12
US16	1	13
US14	2	14
US15	2	15
US13	4	16
US17	2	17
US18	2	18
US20	4	19
US19	4	20

AJUSTE:

Sprint 1: US01, US03, US05, US10 = 8 SPs
Sprint 2: US04, US06, US09, US02 = 9 SPs
Sprint 3: US08, US07, US12 = 8 SPs
Sprint 4: US11, US16, US14, US15 = 7 SPs

Sprint 1: US01, US03, US05, US10 = 8 SPs
Sprint 2: US04, US06, US09, US02 = 9 SPs
Sprint 3: US08, US07, US12 = 8 SPs
Sprint 4: US11, US16, US14, US15 = 7 SPs
Sprint 5: US13, US17, US18 = 8 SPs
Sprint 6: US20, US19 = 8 SPs

Sprint 1: US01, US03, US05, US10 = 8 SPs
Sprint 2: US04, US06, US09, US02 = 9 SPs
Sprint 3: US08, US07, US12 = 8 SPs
Sprint 4: US11, US16, US14, US15 = 7 SPs

Sprint 1: US01, US03, US05, US10 = 8 SPs
Sprint 2: US04, US06, US09, US02 = 9 SPs
Sprint 3: US08, US07, US12 = 8 SPs
Sprint 4: US11, US16, US14, US15 = 7 SPs
Sprint 5: US13, US17, US18 = 8 SPs
Sprint 6: US20, US19 = 8 SPs

AJUSTE:

Por alcance: posponer para un futuro los relatos que no se podrán realizar en 8 semanas.

Por tiempo: extender el desarrollo 4 semanas para agregar 2 sprints.

Sprint 1: US01, US03, US05, US10 = 8 SPs
Sprint 2: US04, US06, US09, US02 = 9 SPs
Sprint 3: US08, US07, US12 = 8 SPs
Sprint 4: US11, US16, US14, US15 = 7 SPs

Sprint 1: US01, US03, US05, US10 = 8 SPs
Sprint 2: US04, US06, US09, US02 = 9 SPs
Sprint 3: US08, US07, US12 = 8 SPs
Sprint 4: US11, US16, US14, US15 = 7 SPs
Sprint 5: US13, US17, US18 = 8 SPs
Sprint 6: US20, US19 = 8 SPs

AJUSTE:

Por alcance: posponer para un futuro los relatos que no se podrán realizar en 8 semanas.

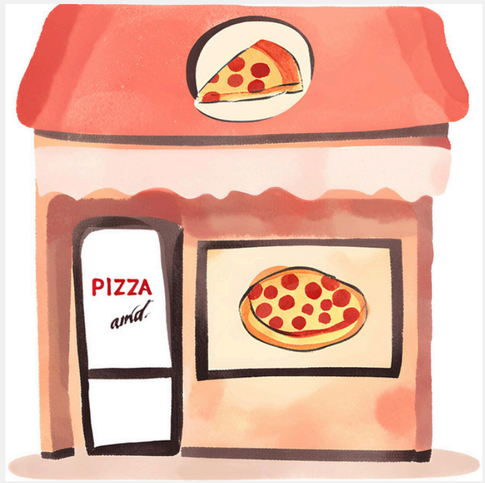
Por tiempo: extender el desarrollo 4 semanas para agregar 2 sprints.

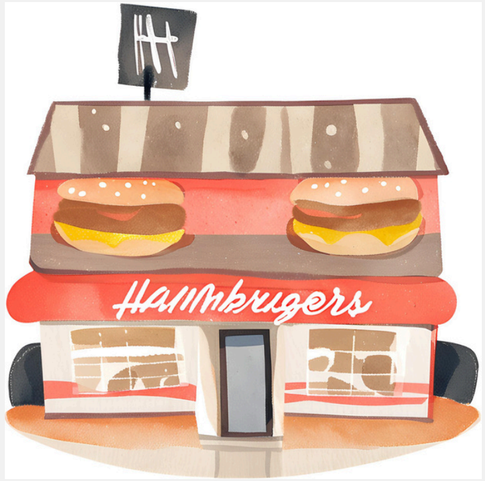
NOTAR: Si es que priorizamos bien, las funcionalidades más importantes para el cliente estarán incluídas, entonces un ajuste por alcance suele ser mejor recibido

EJERCICIOS PATRONES

1

Se quiere simular el funcionamiento de varios posibles negocios de venta de comida: una pizzería, una hamburguesería y una ensaladería. Para simplificar supongamos que en cada caso solo se fabrican dos productos:

















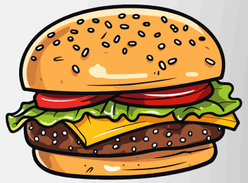


a) Definir **tres fábricas abstractas**:

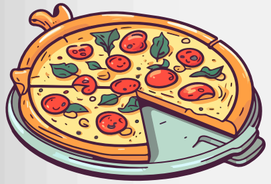
- Una para Pizzas
- Una para Hamburguesas
- Una para Ensaladas.

Además de escribir el código de las fábricas escriba el código de los productos (lo más simple posible).

b) Definir una clase **Negocio** con un método llamado **simular** que recibe una fábrica y el número de productos de cada tipo y procede a hacer la simulación como muestra el ejemplo. Su código debe funcionar exactamente de la forma que se muestra a continuación.



```
Negocio.new(Hamburgueseria.new,3,1).simular
hamburguesa regular 1 saliendo
hamburguesa regular 2 saliendo
hamburguesa regular 3 saliendo
hamburguesa not_meat 1 saliendo
Negocio.new(Pizzeria.new,2,3).simular
pizza peperoni 1 saliendo
pizza peperoni 2 saliendo
pizza vegetariana 1 saliendo
pizza vegetariana 2 saliendo
pizza vegetariana 3 saliendo
Negocio.new(Ensaladeria.new,4,1).simular
ensalada cesar 1 saliendo
ensalada cesar 2 saliendo
ensalada cesar 3 saliendo
ensalada cesar 4 saliendo
ensalada mediterranea 1 saliendo
```





Primero definimos las fábricas de comida (tiendas)

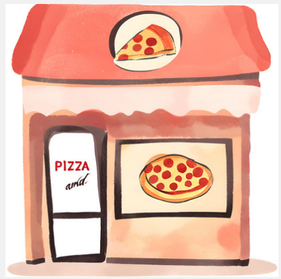
```
class FoodFactor
    def new_p1(number)
        raise NotImplementedError
    end
    def new_p2(number)
        raise NotImplementedError
    end
end
```

Primero definimos las fábricas de comida (tiendas)

```
class Pizzeria
    def new_p1(number)
        Pepperoni.new(number)
    end
    def new_p2(number)
        Vegetarian.new(number)
    end
end
```




```
class Ensaladeri
    def new_p1(number)
        Cesar.new(number)
    end
    def new_p2(number)
        Mediterranean.new(number)
    end
end
```




```
class Hamburgueseria
    def new_p1(number)
        Regular.new(number)
    end
    def new_p2(number)
        Not_meat.new(number)
    end
end
```



Luego definimos los alimentos en si (productos)

```
class Food
end

class Pizza < Food
end

class Burguer < Food
end

class Salad < Food
end
```

PIZZERÍA

```ruby
class Pepperoni < Pizza
    def initialize(number)
        @name = 'pizza peperoni ' + number
    end
    def reveal
        return @name + ' saliendo'
    end
end
```

```ruby
class Vegetarian < Pizza
    def initialize(number)
        @name = 'pizza vegetariana ' + number
    end
    def reveal
        return @name + ' saliendo'
    end
end
```





HAMGURGUESERÍA

```ruby
class Regular < Burguer
    def initialize(number)
        @name = 'hamburguesa regular ' + number
    end
    def reveal
        return @name + ' saliendo'
    end
end
```

```ruby
class Not_meat < Burguer
    def initialize(number)
        @name = 'hamburguesa not_meat ' + number
    end
    def reveal
        return @name + ' saliendo'
    end
end
```





ENSALADERÍA

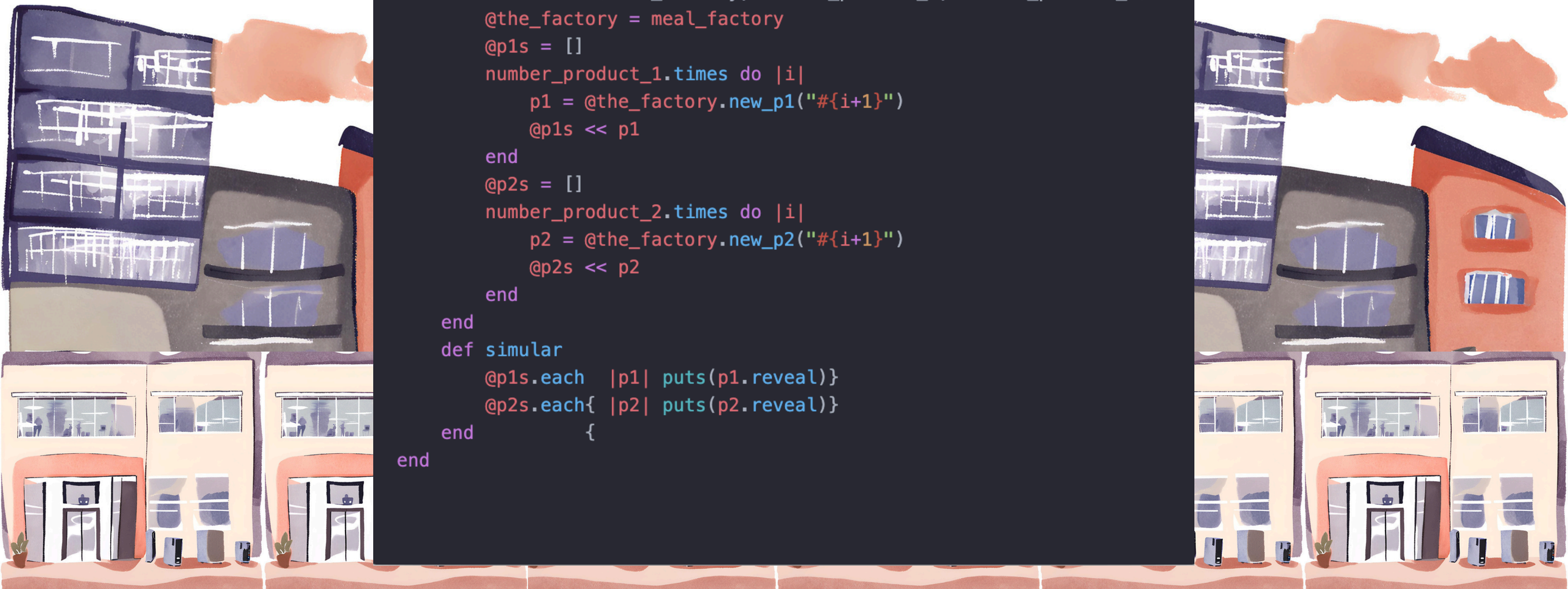
```ruby
class Cesar < Salad
    def initialize(number)
        @name = 'ensalada cesar ' + number
    end
    def reveal
        return @name + ' saliendo'
    end
end
```

```ruby
class Mediterranean < Salad
    def initialize(number)
        @name = 'ensalada mediterranea ' + number
    end
    def reveal
        return @name + ' saliendo'
    end
end
```





Finalmente definimos el negocio



```
class Negocio
    def initialize(meal_factory, number_product_1, number_product_2)
        @the_factory = meal_factory
        @p1s = []
        number_product_1.times do |i|
            p1 = @the_factory.new_p1("#{i+1}")
            @p1s << p1
        end
        @p2s = []
        number_product_2.times do |i|
            p2 = @the_factory.new_p2("#{i+1}")
            @p2s << p2
        end
    end
    def simular
        @p1s.each  |p1| puts(p1.reveal)}
        @p2s.each{ |p2| puts(p2.reveal)}
    end          {
end
```

```
Negocio.new(Hamburgueseria.new,3,1).simular
hamburguesa regular 1 saliendo
hamburguesa regular 2 saliendo
hamburguesa regular 3 saliendo
hamburguesa not_meat 1 saliendo
Negocio.new(Pizzeria.new,2,3).simular
pizza peperoni 1 saliendo
pizza peperoni 2 saliendo
pizza vegetariana 1 saliendo
pizza vegetariana 2 saliendo
pizza vegetariana 3 saliendo
Negocio.new(Ensaladeria.new,4,1).simular
ensalada cesar 1 saliendo
ensalada cesar 2 saliendo
ensalada cesar 3 saliendo
ensalada cesar 4 saliendo
ensalada mediterranea 1 saliendo
```

Ahora las personas pueden disfrutar de un buen almuerzo



2

Ivan analiza una parte de un sistema de banco encargada de notificar al dueño de la cuenta si existe algun cambio en su balance. Nota que esa parte del sistema está ilustrada en el código siguiente, y que por el momento solo envia correos al dueño de la cuenta cuando corresponde utilizando la siguiente instruccion.
Sin embargo, Ivan te comenta que muchos clientes del banco desean también recibir notificaciones a sus celulares mediante SMS. Para realizar este cambio, es necesario modificar el código actual **(agregar una linea en 2 metodos de la clase)** e Ivan no quiere hacer esto.

Ayuda a Ivan a refactorizar está parte del programa aplicando el **patron observer**, tal que pueda agregar las notificaciones según sea necesario:

- **Notificaciones por email**: Cada que el balance cambia, se debe imprimir en consola “Send email to owner: the actual balance is ….”.
- **Notificaciones por SMS**: Cada que el balance cambia, se debe imprimir en consola “Send SMS to owner: the actual balance is ….”.

```ruby
class Account
    def initialize()
        @balance = 0
    end
    def deposit(amount)
        @balance += amount
        puts "Send email to owner: the actual balance is #{@balance}"
    end
    def withdraw(amount)
        if @balance - amount >= 0
            @balance -= amount
            puts "Send email to owner: the actual balance is #{@balance}"
        end
    end
end
```

Comenzamos definiendo los observers

```ruby
class Notification
    def update(balance)
        raise NotImplementedError
    end
end

class SMSNotification < Notification
    def update(balance)
        puts "Send SMS to owner: the actual balance is #{balance}"
    end
end

class EmailNotification < Notification
    def update(balance)
        puts "Send email to owner: the actual balance is #{balance}"
    end
end
```

```
class Account
    def initialize()
        @balance = 0
        @observers = []
    end

    def addObserver(obs)
        @observers.push(obs)
    end

    def notifyAll()
        for obs in @observers
            obs.update(@balance)
        end
    end

    def deposit(amount)
        @balance += amount
        notifyAll()
    end

    def withdraw(amount)
        if @balance - amount >= 0
            @balance -= amount
            notifyAll()
        end
    end
end
```

```
class Account
    def initialize()
        @balance = 0
        @observers = []
    end

    def addObserver(obs)
        @observers.push(obs)
    end

    def notifyAll()
        for obs in @observers
            obs.update(@balance)
        end
    end
```

```
    @balance = 0
    @observers = []
end

def addObserver(obs)
    @observers.push(obs)
end

def notifyAll()
    for obs in @observers
        obs.update(@balance)
    end
end

def deposit(amount)
    @balance += amount
    notifyAll()
end

def withdraw(amount)
    if @balance - amount >= 0
```

```
    def notifyAll()
        for obs in @observers
            obs.update(@balance)
        end
    end

    def deposit(amount)
        @balance += amount
        notifyAll()
    end

    def withdraw(amount)
        if @balance - amount >= 0
            @balance -= amount
            notifyAll()
        end
    end
end
```

Ahora las personas pueden recibir sus notificaciones



MUCHAS GRACIAS!



José Tomás Valdivia y Jorge De Goyeneche

FUENTES

→ Pauta I1 2023-1 y pauta i1 2022-2
→ atlassian.com/agile/project-management
→ Clases y apuntes de los profesores: Alison Fernandez, Jaime Navón, Juan Sandoval