Programación III

Clase XVI

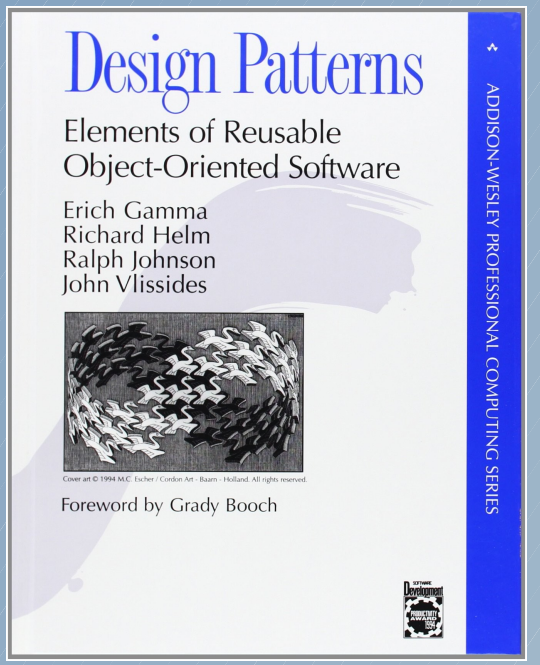
Andrés Fortier

Pregunta de la Rhodesia

- ¿Qué es *super*? ¿Cómo se resuelve un envío de mensaje que se envía a *super*?

Patrones de Diseño

- La idea original surge en el contexto de la arquitectura
  - A Pattern Language: Towns, Buildings, Construction - Christopher Alexander (1977).
- Adaptado a la POO (y popularizado) por Erich Gamma, Richard Helm, Ralph Johnson y John Vlissides (The "Gang of Four").



Ejemplo en la arquitectura

- Contexto
  - Las personas naturalmente prefieren las habitaciones con iluminación natural de mas de un lado.
  - Si hay dos habitaciones, una con sólo una ventana y otra con dos ventanas en distintas paredes, las personas van a preferir estar en la segunda.
- Luego
  - Todas las habitaciones deberían tener iluminación natural proveniente de mas de una zona.

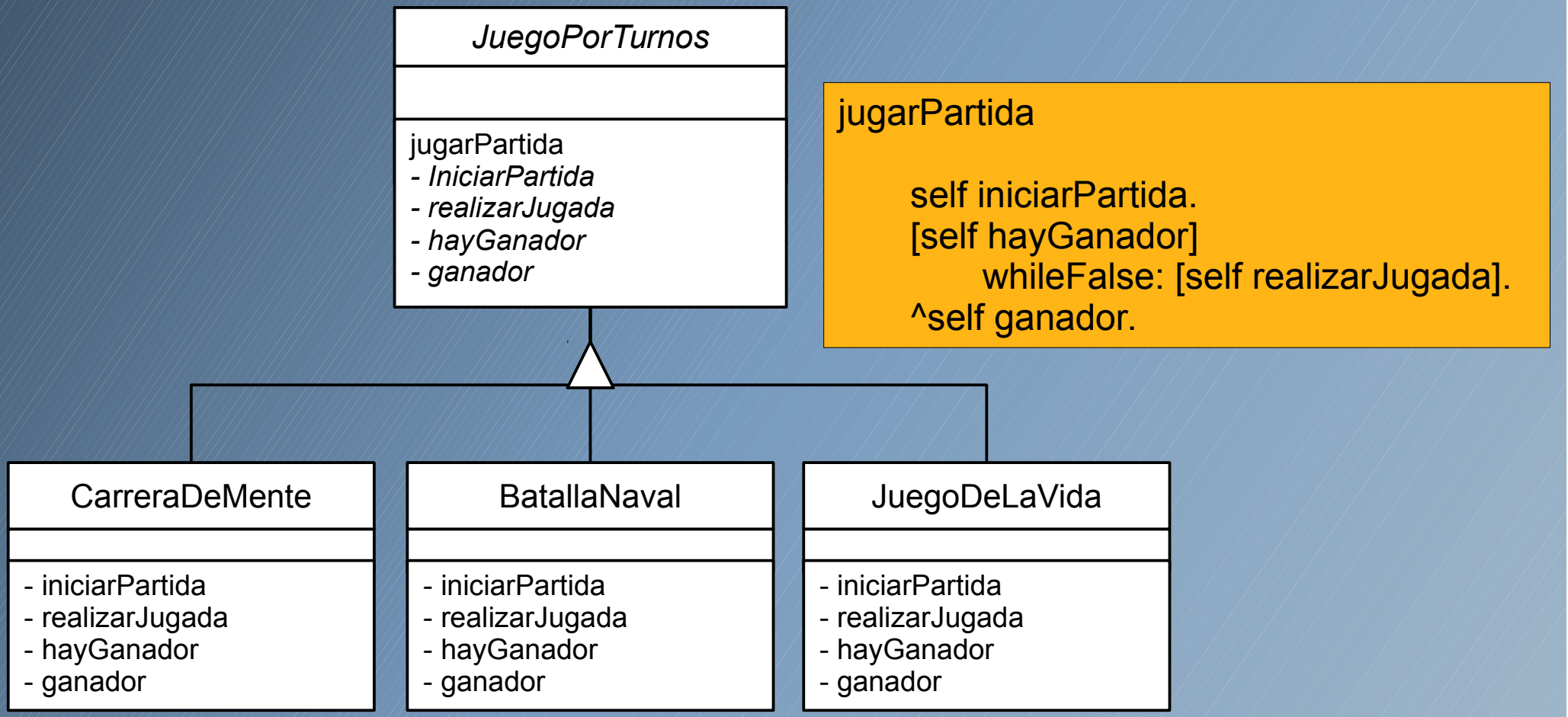
Ejemplo en la arquitectura

- Es independientes del tipo de construcción (casa, departamento, etc) o de los materiales utilizados.
- Describe una idea, no es un producto específico.

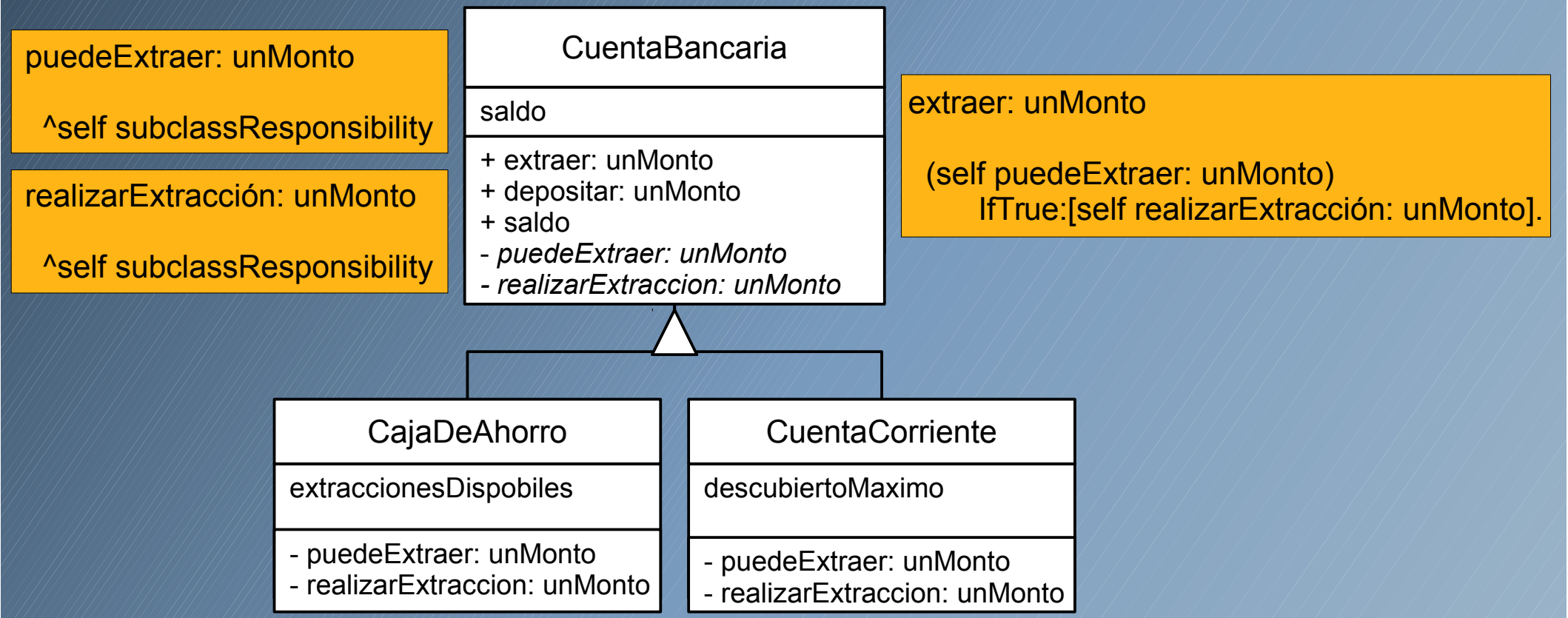
Ejemplo en la POO

- Nombre: Template Method.
- Objetivo
  - Definir el esqueleto de un algoritmo, delegando la implementación de pasos concretos a las subclases.
  - Permitir que las subclases redefinan el comportamiento predefinido de algún paso de un algoritmo, sin alterar la estructura del mismo.

Template Method - Ejemplo



Recordemos las cuentas bancarias




```
puedeExtraer: unMonto

  ^(saldo >= unMonto) & (extraccionesDisponibles > 0).
```

```
realizarExtracción: unMonto

  saldo := saldo - unMonto.
  extraccionesDisponibles := extraccionesDisponibles – 1
```

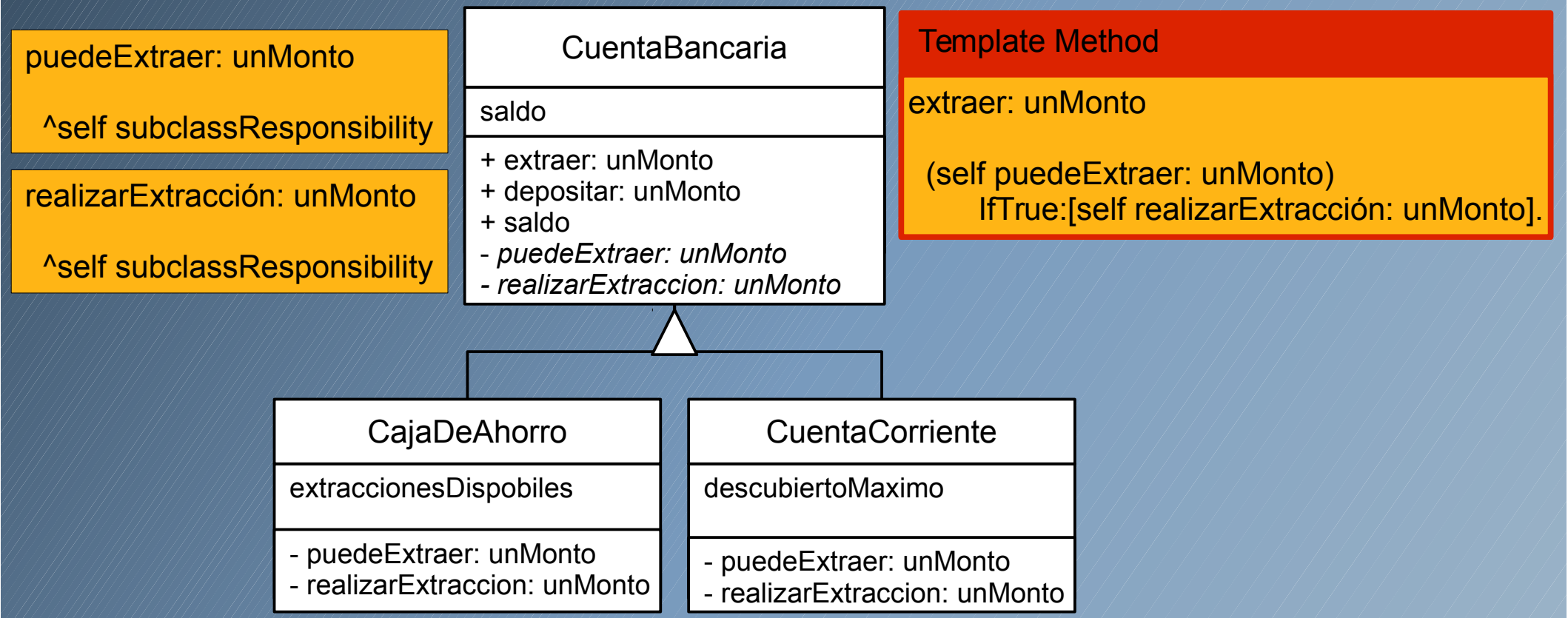
```
puedeExtraer: unMonto

  ^saldo + descubiertoMaximo >= unMonto.
```

```
realizarExtracción: unMonto

  saldo := saldo - unMonto.
```

Recordemos las cuentas bancarias




```
puedeExtraer: unMonto

  ^(saldo >= unMonto) & (extraccionesDisponibles > 0).
```

```
realizarExtracción: unMonto

  saldo := saldo - unMonto.
  extraccionesDisponibles := extraccionesDisponibles – 1
```

```
puedeExtraer: unMonto

  ^saldo + descubiertoMaximo >= unMonto.
```

```
realizarExtracción: unMonto

  saldo := saldo - unMonto.
```

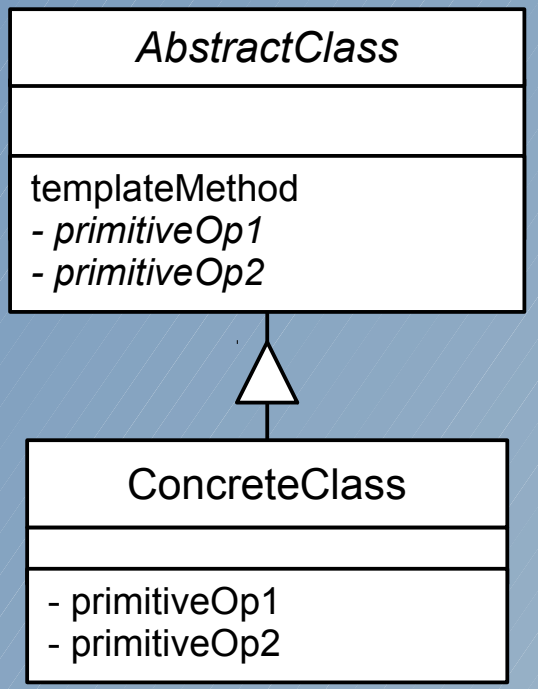
Template Method

- Objetivo
  - Definir el esqueleto de un algoritmo, delegando la implementación de pasos concretos a las subclases.
  - Permitir que las subclases redefinan el comportamiento predefinido de algún paso de un algoritmo, sin alterar la estructura del mismo.
- Aplicación
  - Implementar el invariante de un algoritmo, delegando a las subclases la parte variable.
  - Cuando encontramos partes de código repetido en métodos de las subclases y queremos factorizarlo.

Template Method - Participantes

```
AbstractClass>>templateMethod

    self primitiveOp1.
    ...
    self primitiveOp2.
```



- Clase abstracta
  - Define el esqueleto del algoritmo en términos de “operaciones primitivas”.
  - Define las “operaciones primitivas” como mensajes abstractos.
- Clase concreta
  - Implementa las “operaciones primitivas”.

Template Method – Abstract vs. Hooks

- Métodos abstractos: las subclases ***deben*** redefinirlos.
- Métodos “hooks”: la clase abstracta provee implementaciones predefinidas. Las subclases pueden redefinirlas para implementar comportamientos particulares.

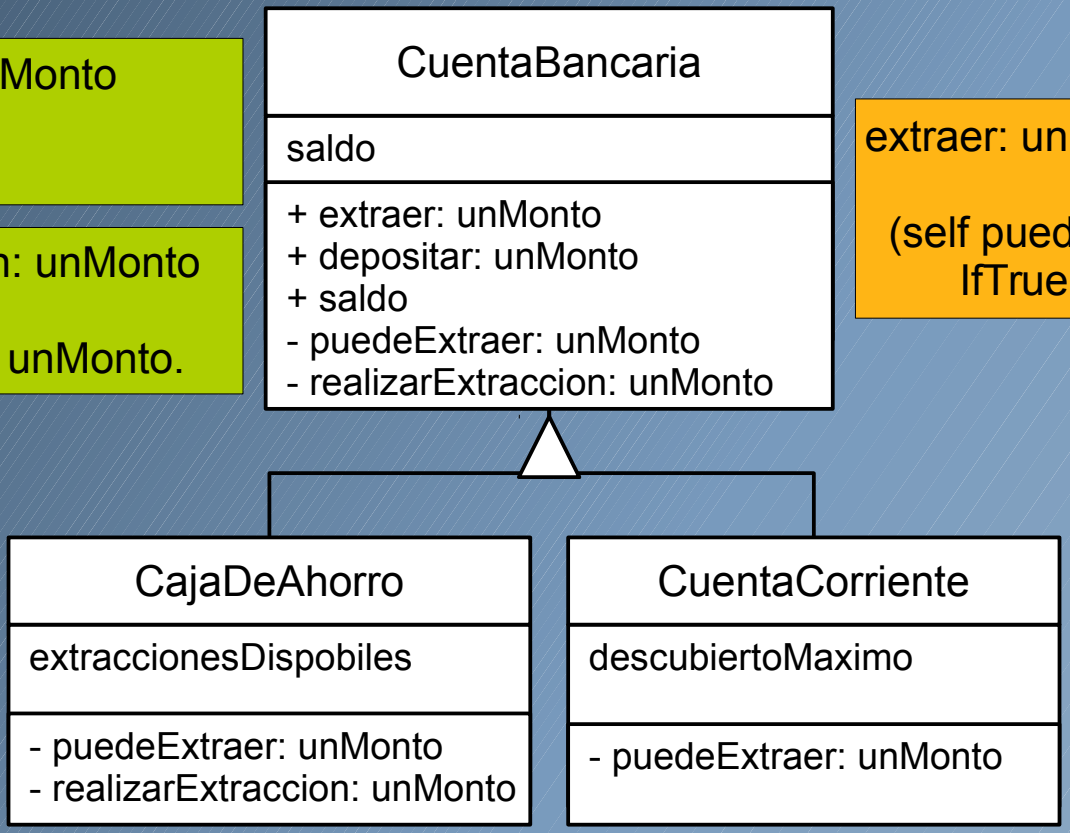
Cuentas bancarias – opción 2 – variación 1

```
puedeExtraer: unMonto

  ^true
```

```
realizarExtracción: unMonto

  saldo := saldo – unMonto.
```




```
extraer: unMonto

  (self puedeExtraer: unMonto)
        IfTrue:[self realizarExtracción: unMonto].
```

```
puedeExtraer: unMonto

  ^(saldo >= unMonto) & (extraccionesDisponibles > 0).
```

```
realizarExtracción: unMonto

  super realizarExtracción: unMonto.
  extraccionesDisponibles := extraccionesDisponibles – 1
```

```
puedeExtraer: unMonto

  ^saldo + descubiertoMaximo >= unMonto.
```

Patrones de diseño

- Soluciones reutilizables a problemas recurrentes.
- No son diseños terminados o implementaciones concretas.
- Son descripciones que pueden utilizarse en diferentes contextos.

Para tener en cuenta

- No buscar en los ejercicios de la práctica un patrón para resolverlos.
- Evitar la “fiebre de los patrones”.
- Los patrones no son soluciones estrictas. Admiten variantes.

Patrones de diseño

- Design Patterns: Elements of Reusable Object-Oriented Software

  Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides.
- The Design Patterns Smalltalk Companion
  - Sherman Alpert, Kyle Brown, Bobby Woolf.
- Head First Design Patterns
  - Eric Freeman, Bert Bates, Kathy Sierra, Elisabeth Robson.
- http://www.oodesign.com/
- http://stackoverflow.com/questions/tagged/design-patterns

Práctica 3

- Ya está disponible en la página.