

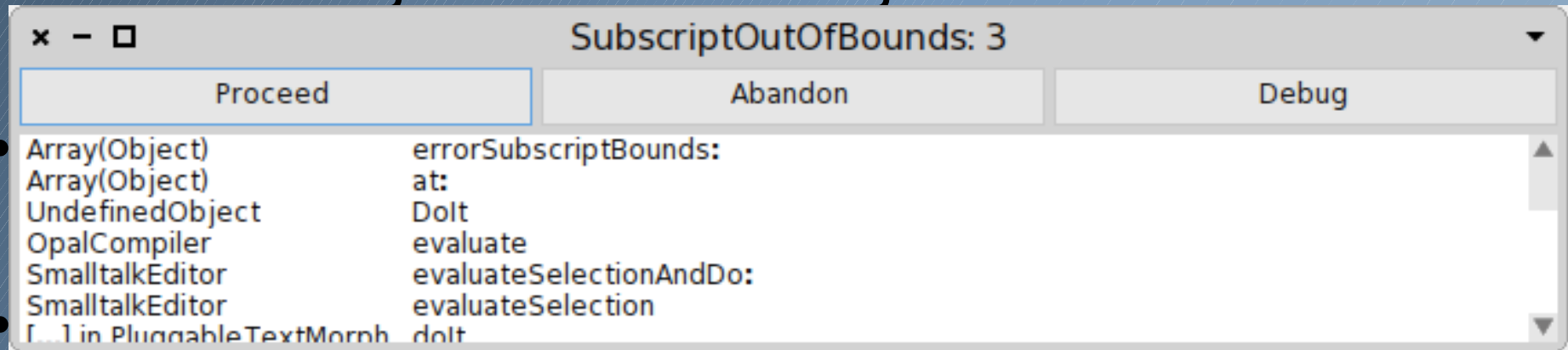
# Laboratorio de Computación III

Clase XX

*Andrés Fortier*

# Excepciones

- Una excepción indica una condición en particular que altera el flujo normal de la ejecución de un



```
| numeros |  
numeros := #(33 25).  
numeros at: 3.
```

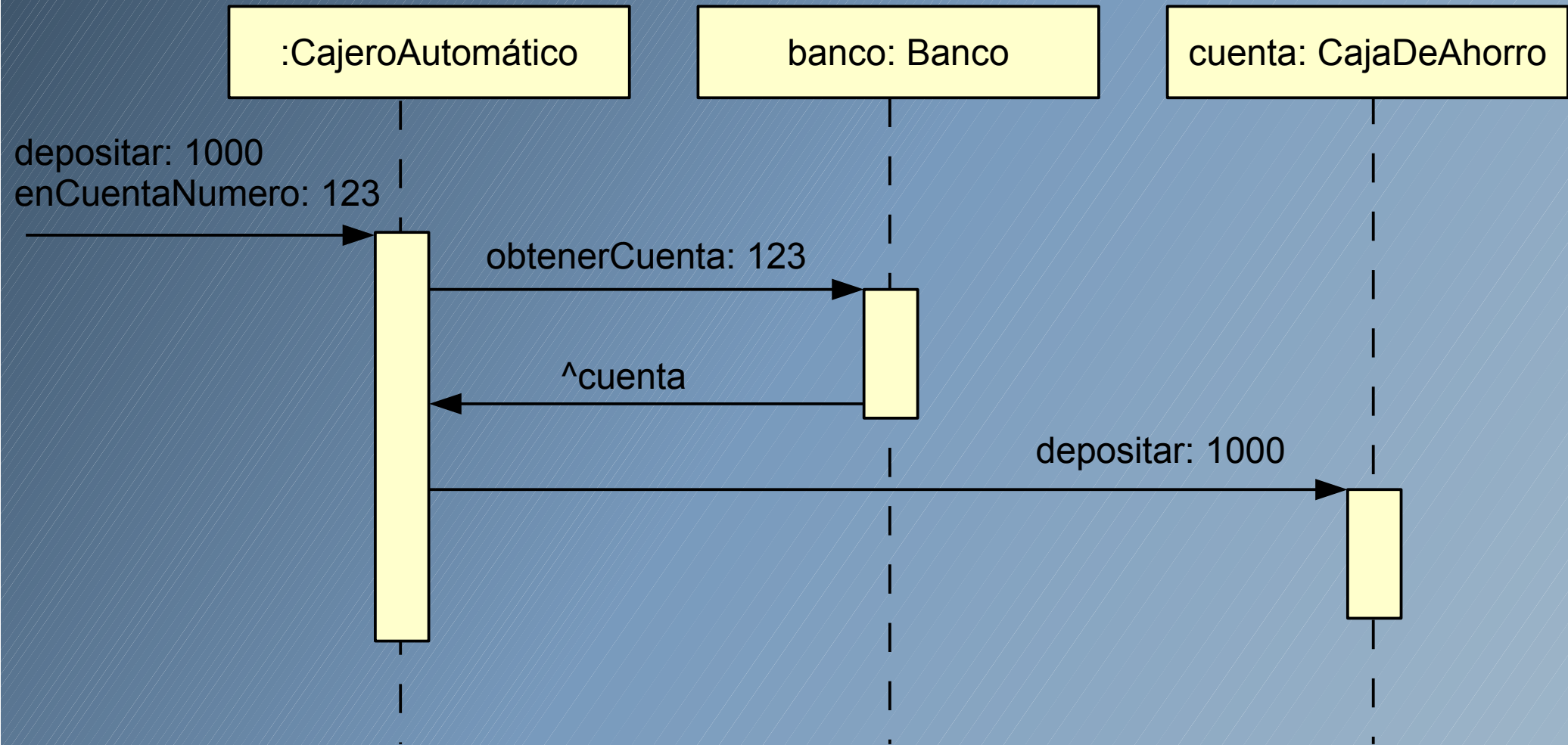


# Excepciones

- Problema:
  - Intentamos acceder a una posición de la colección que no existe.
  - El objeto instancia de Array no sabe cómo resolver dicho error o qué significa en el contexto en el que lo están usando.
- Solución:
  - Generar una excepción indicando el error.
  - Propagar la excepción al que envió el mensaje.
  - Permitir a quien tenga mas información manejar el error.



# Ejemplo 1



# Ejemplo 1

```
Banco>>obtenerCuenta: numeroDeCuenta
```


```
  ^self cuentas detect:  
    [:cuenta | cuenta numero = numeroDeCuenta].
```



```
Cajero>>depositar: unMonto enCuentaNumero: numeroDeCuenta
```

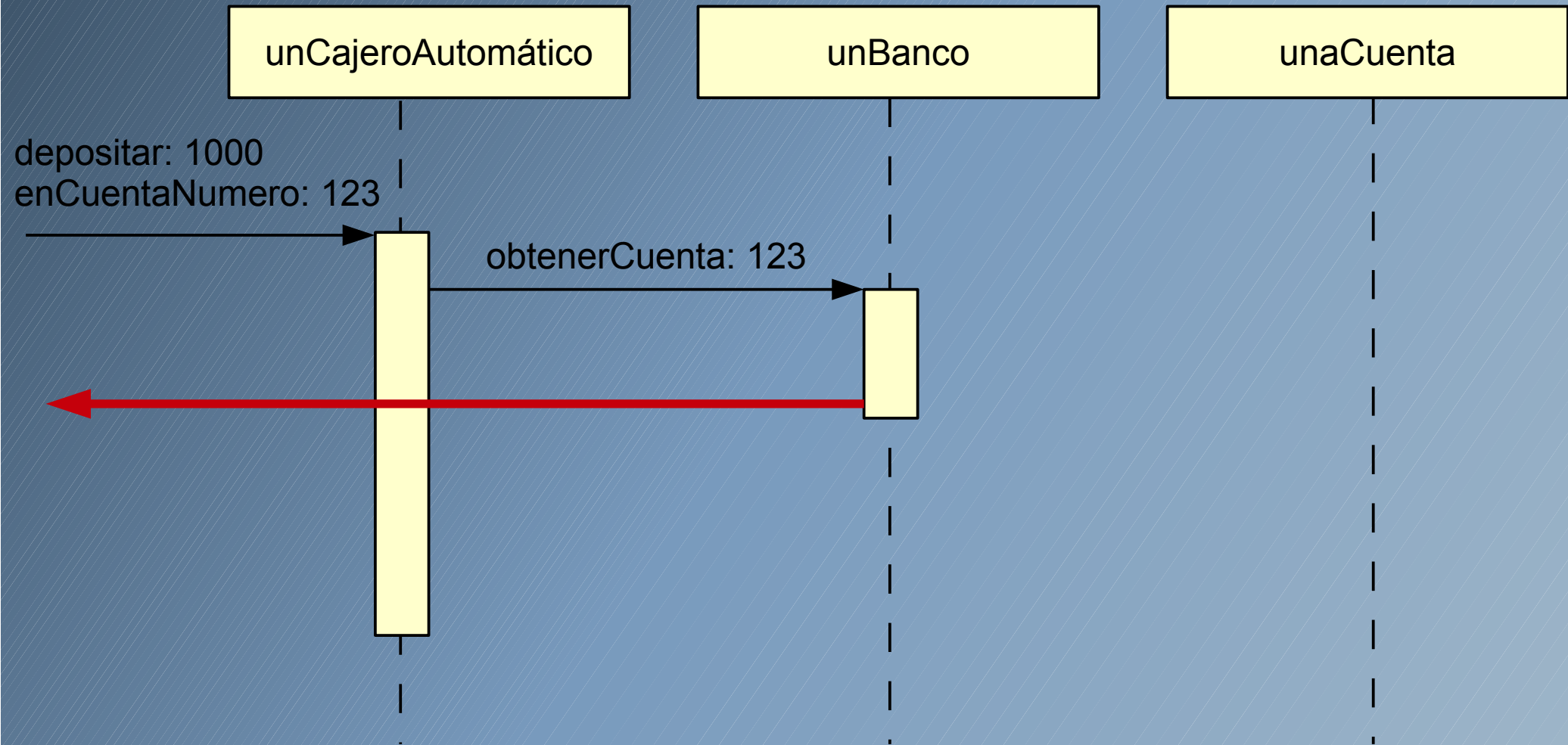
```
  | cuenta |
```

```
  cuenta := self banco obtenerCuenta: numeroDeCuenta  
  cuenta depositar: unMonto.
```



- ¿Qué sucede si el banco no tiene la cuenta buscada?
  - Se genera una excepción (NotFound).
  - La excepción se propaga buscando alguien que la maneje.

# Ejemplo 1



# Manejando Excepciones

```
Cajero>>depositar: unMonto enCuentaNumero: numeroDeCuenta  
  
    | cuenta |  
    cuenta := self banco obtenerCuenta: numeroDeCuenta.  
    cuenta depositar: unMonto.
```



# Manejando Excepciones

```
Cajero>>depositar: unMonto enCuentaNumero: numeroDeCuenta
```

```
[ | cuenta |
```

```
cuenta := self banco obtenerCuenta: numeroDeCuenta.
```

```
cuenta depositar: unMonto.
```

```
]
```



# Manejando Excepciones

```
Cajero>>depositar: unMonto enCuentaNumero: numeroDeCuenta
```

```
[ | cuenta |
```

```
cuenta := self banco obtenerCuenta: numeroDeCuenta.
```

```
cuenta depositar: unMonto.
```

```
]
```

```
  on:
```

```
  do:
```

# Manejando Excepciones

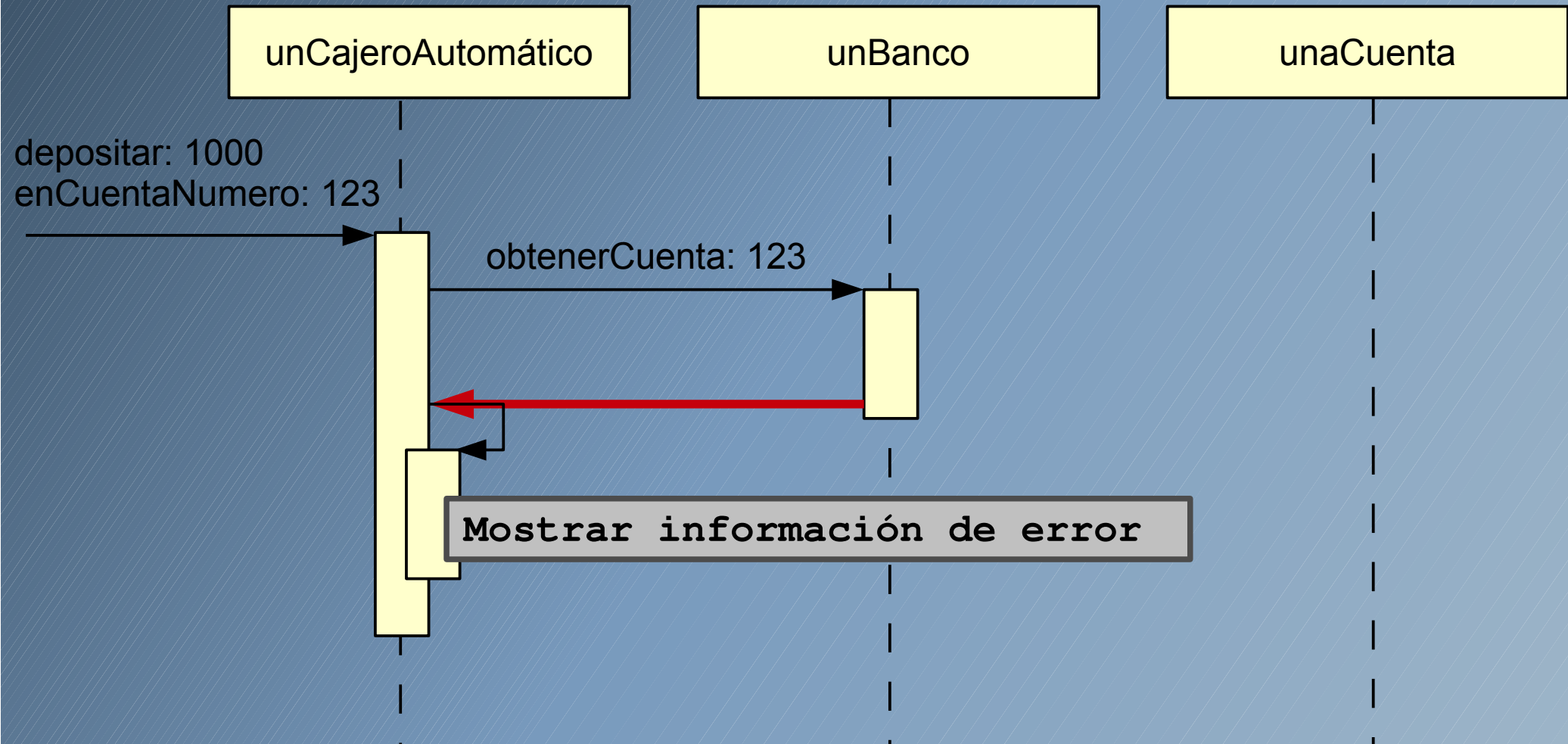
```
Cajero>>depositar: unMonto enCuentaNumero: numeroDeCuenta  
  
[ | cuenta |  
  cuenta := self banco obtenerCuenta: numeroDeCuenta.  
  cuenta depositar: unMonto.  
]  
  on: NotFound  
  do:
```

# Manejando Excepciones

```
Cajero>>depositar: unMonto enCuentaNumero: numeroDeCuenta

[ | cuenta |
  cuenta := self banco obtenerCuenta: numeroDeCuenta.
  cuenta depositar: unMonto.
]
  on: NotFound
  do: [:exception | "Mostrar información de error"]
```

# Ejemplo 1





# Generando Excepciones

```
CajaDeAhorro>>extraer: unMonto  
  
    (self puedeExtraer: unMonto)  
    ifTrue: [saldo := saldo - unMonto].
```

- ¿Qué sucede si la extracción no se puede realizar?
- Debemos generar una excepción.

# Generando Excepciones

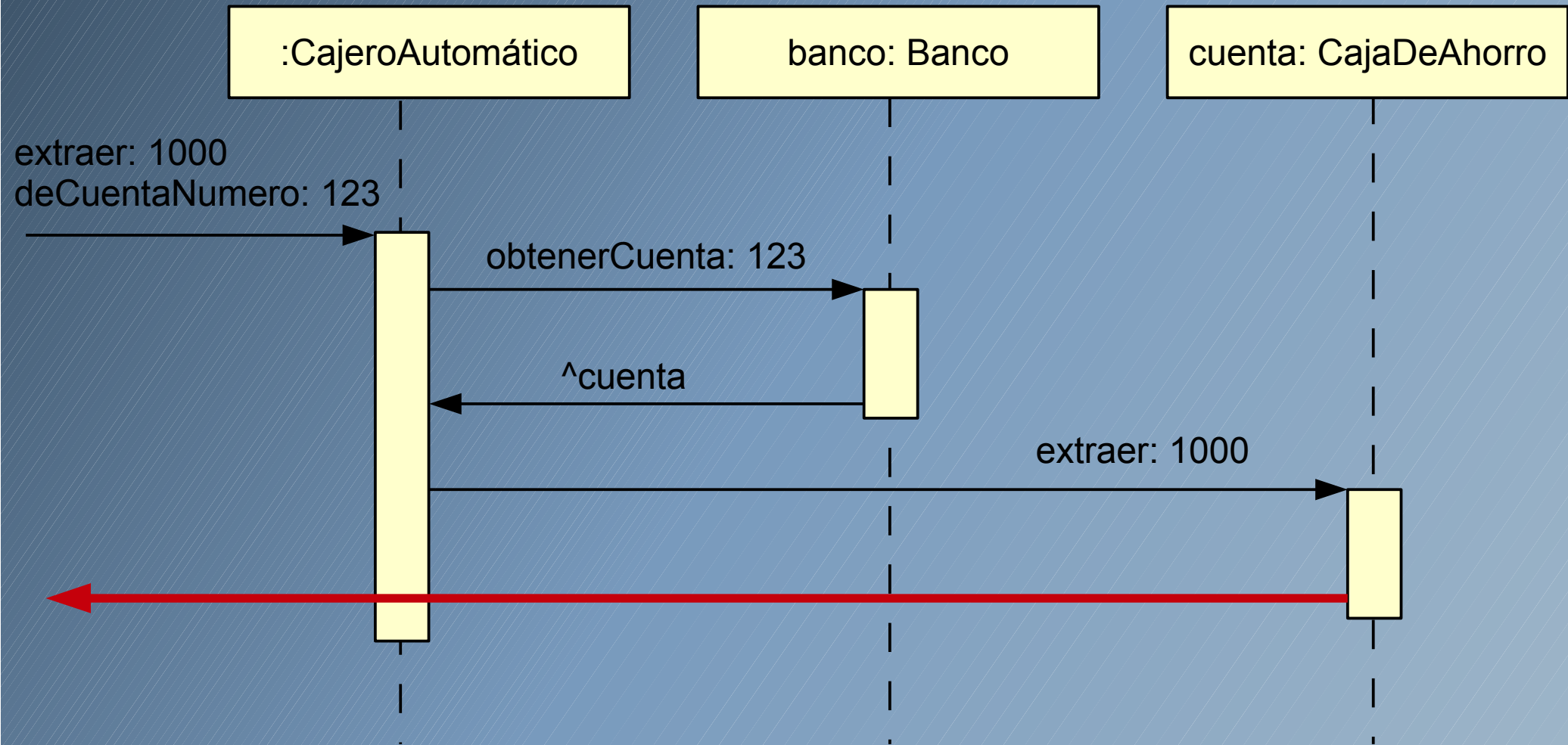
- Primer paso: Crear una clase que represente dicha excepción.

```
Error subclass: #ExtraccionNoRealizable  
  instanceVariableNames: ''  
  classVariableNames: ''  
  category: ''
```

- “Levantar” la excepción cuando corresponda.

```
CajaDeAhorro>>extraer: unMonto  
  
  (self puedeExtraer: unMonto)  
    ifTrue: [saldo := saldo - unMonto]  
    ifFalse: [ExtraccionNoRealizable signal].
```

# Ejemplo 2



# Manejando Excepciones

```
Cajero>>extraer: unMonto deCuentaNumero: numeroDeCuenta  
  
| cuenta |  
cuenta := self banco obtenerCuenta: numeroDeCuenta.  
[  
cuenta extraer: unMonto.  
]
```



# Manejando Excepciones

```
Cajero>>extraer: unMonto deCuentaNumero: numeroDeCuenta  
  
| cuenta |  
cuenta := self banco obtenerCuenta: numeroDeCuenta.  
[  
cuenta extraer: unMonto.  
]  
    on:  
    do:
```

# Manejando Excepciones

```
Cajero>>extraer: unMonto deCuentaNumero: numeroDeCuenta  
  
| cuenta |  
cuenta := self banco obtenerCuenta: numeroDeCuenta.  
[  
cuenta extraer: unMonto.  
]  
    on: ExtraccionNoRealizable  
    do:
```

# Manejando Excepciones

```
Cajero>>extraer: unMonto deCuentaNumero: numeroDeCuenta  
  
| cuenta |  
cuenta := self banco obtenerCuenta: numeroDeCuenta.  
[  
cuenta extraer: unMonto.  
]  
    on: ExtraccionNoRealizable  
    do: [:exception | "Mostrar información de error"]
```

# Tests de unidad con excepciones

- Test básico de extraer:

```
CajaDeAhorroTest>>testExtraer  
  
| cuenta |  
  
cuenta := CajaDeAhorro saldo: 100.  
cuenta extraer: 50.  
self assert: cuenta saldo = 50.
```

- Pero sabemos que nuestro código puede generar una excepción.
  - Debemos testearlo!



# Tests de unidad con excepciones

```
CajaDeAhorroTest>>testExtraer
```

```
| cuenta |
```

```
cuenta := CajaDeAhorro saldo: 100.
```

```
cuenta extraer: 50.
```

```
self assert: cuenta saldo = 50.
```

```
self
```

```
    should:
```

```
    raise:
```

# Tests de unidad con excepciones

```
CajaDeAhorroTest>>testExtraer
```

```
| cuenta |
```

```
cuenta := CajaDeAhorro saldo: 100.
```

```
cuenta extraer: 50.
```

```
self assert: cuenta saldo = 50.
```

```
self
```

```
    should: [cuenta extraer: 200]
```

```
    raise:
```

# Tests de unidad con excepciones

```
CajaDeAhorroTest>>testExtraer
```

```
| cuenta |
```

```
cuenta := CajaDeAhorro saldo: 100.
```

```
cuenta extraer: 50.
```

```
self assert: cuenta saldo = 50.
```

```
self
```

```
    should: [cuenta extraer: 200]
```

```
    raise: ExtraccionNoRealizable.
```

# Práctica 5

- Va a estar disponible a partir de la semana que viene.