fiuba

algo3

RTTI – reflexión Smalltalk y Java avanzados

Carlos Fontela

cfontela@fi.uba.ar

RTTI – Reflexión - Lenguajes

Temario

RTTI

Reflexión

Java y Smalltalk avanzados

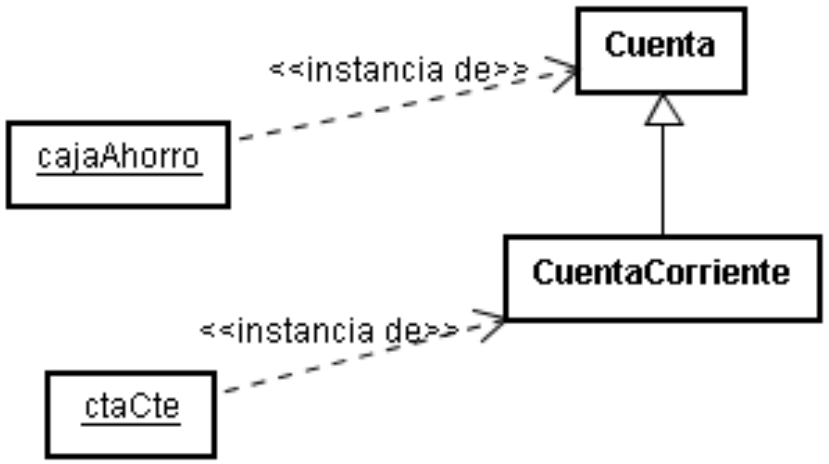
Ejemplo marco RTTI

Java

```
Cuenta cajaAhorro = new Cuenta( );
Cuenta ctaCte = new CuentaCorriente( );
```

Smalltalk

```
cajaAhorro := Cuenta new.
ctaCte := CuentaCorriente new.
```



RTTI

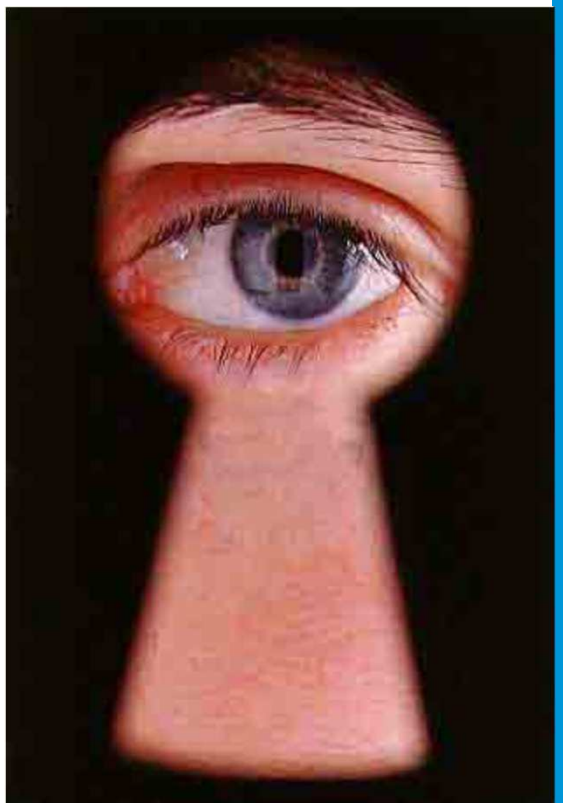
Información de tipos en tiempo de ejecución

2 situaciones

- Conocer la clase exacta de un objeto
- O la familia de la clase del objeto

Veremos:

- Cómo lograrlo
- Inconvenientes de RTTI
- Cómo está implementado



fiuba algo3

RTTI: conocer la clase

Java

```
assertTrue (cajaAhorro.getClass( ) == Cuenta.class);
assertTrue (ctaCte.getClass ( ) ==
  CuentaCorriente.class);
assertFalse (ctaCte.getClass ( ) == Cuenta.class);
```

Smalltalk

```
self assert: (cajaAhorro class = Cuenta).
self assert: (ctaCte class = CuentaCorriente).
self deny: (ctaCte class = Cuenta).
```

RTTI: conocer la familia

Java

```
assertTrue (cajaAhorro instanceOf Cuenta);
assertTrue (ctaCte instanceOf Cuenta);
assertTrue (ctaCte instanceOf CuentaCorriente);
```

Smalltalk

```
self assert: (cajaAhorro isKindOf: Cuenta).
self assert: (ctaCte isKindOf: CuentaCorriente).
self assert: (ctaCte isKindOf: Cuenta).
```

Problemas con RTTI

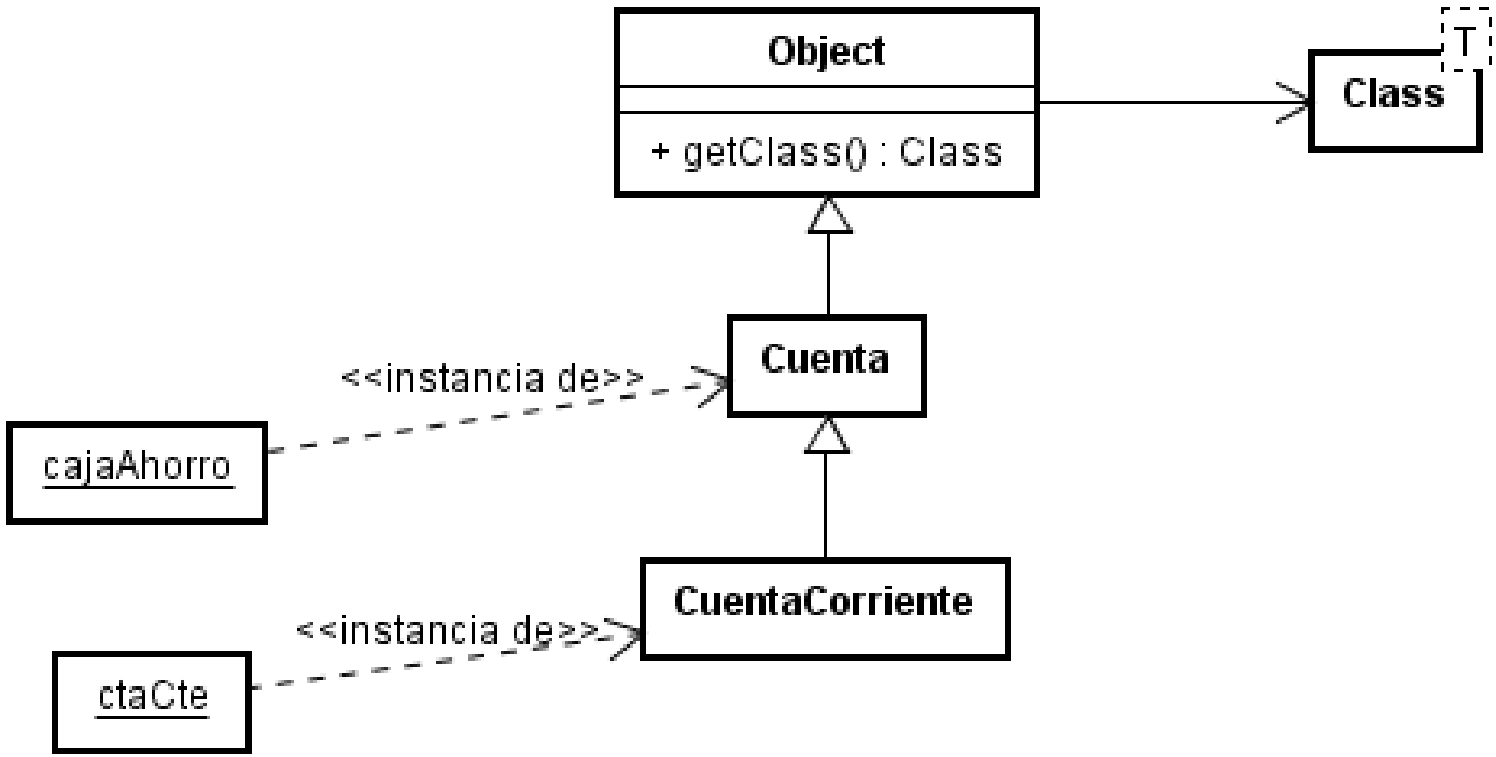
Compromete la extensibilidad

```
public void extraer ( int monto ) {
  if (this instanceof CajaAhorro)
      disponible = saldo;
  else
      disponible = saldo + descubierto;
  if (disponible < monto)
      throw new SaldoInsuficiente( );
  saldo -= monto;
}
```

Evita el polimorfismo

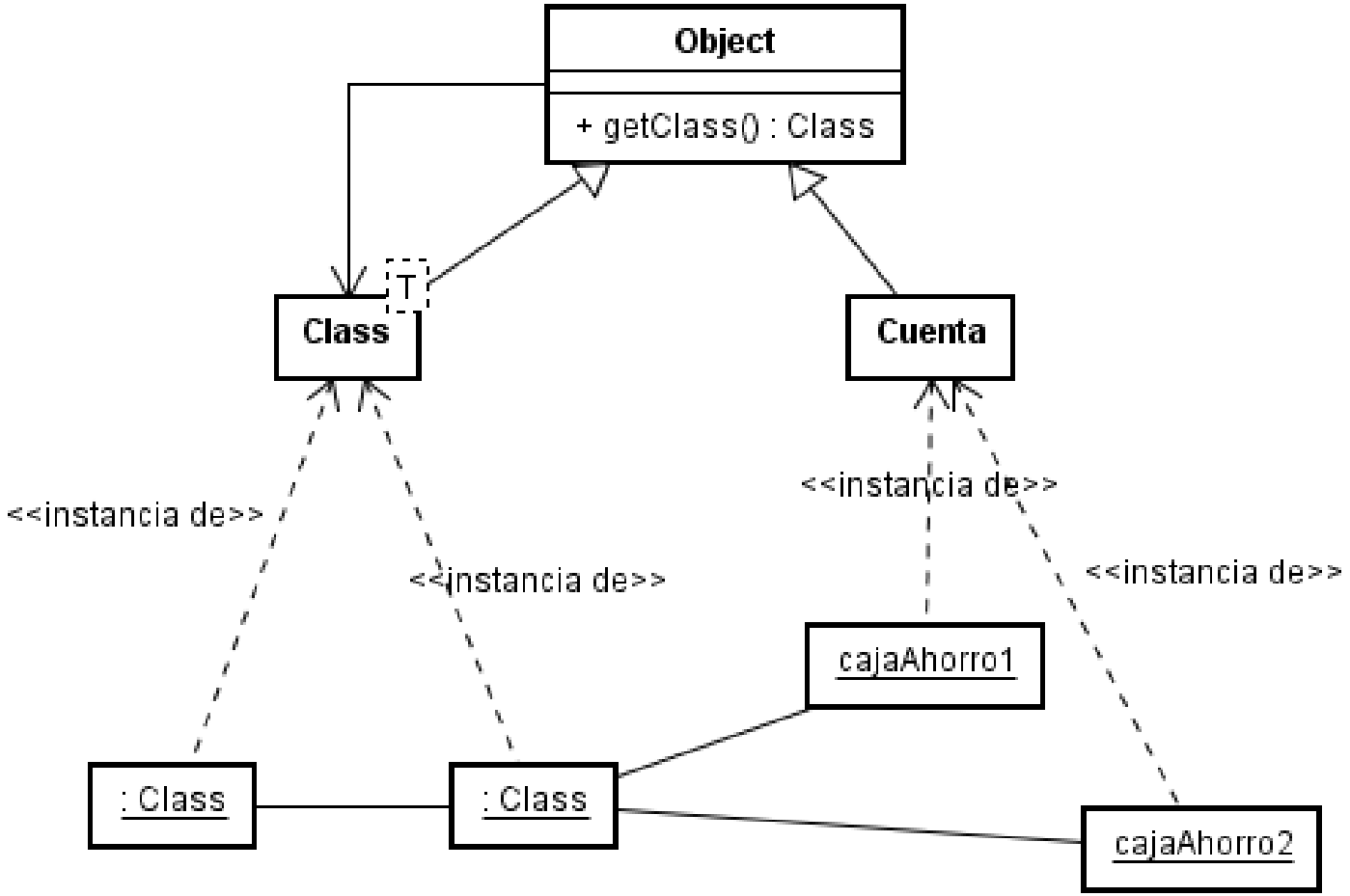
```
if (x instanceof CuentaCorriente)
  ((CuentaCorriente)x).setDescubierto(0);
```

RTTI en Java (1)



fiuba algo3

RTTI en Java (2)



fiuba algo3

RTTI en Smalltalk: principios

Las clases son objetos

Las clases son instancias de una metaclase

Hay una jerarquía de metaclases paralela a la de las clases

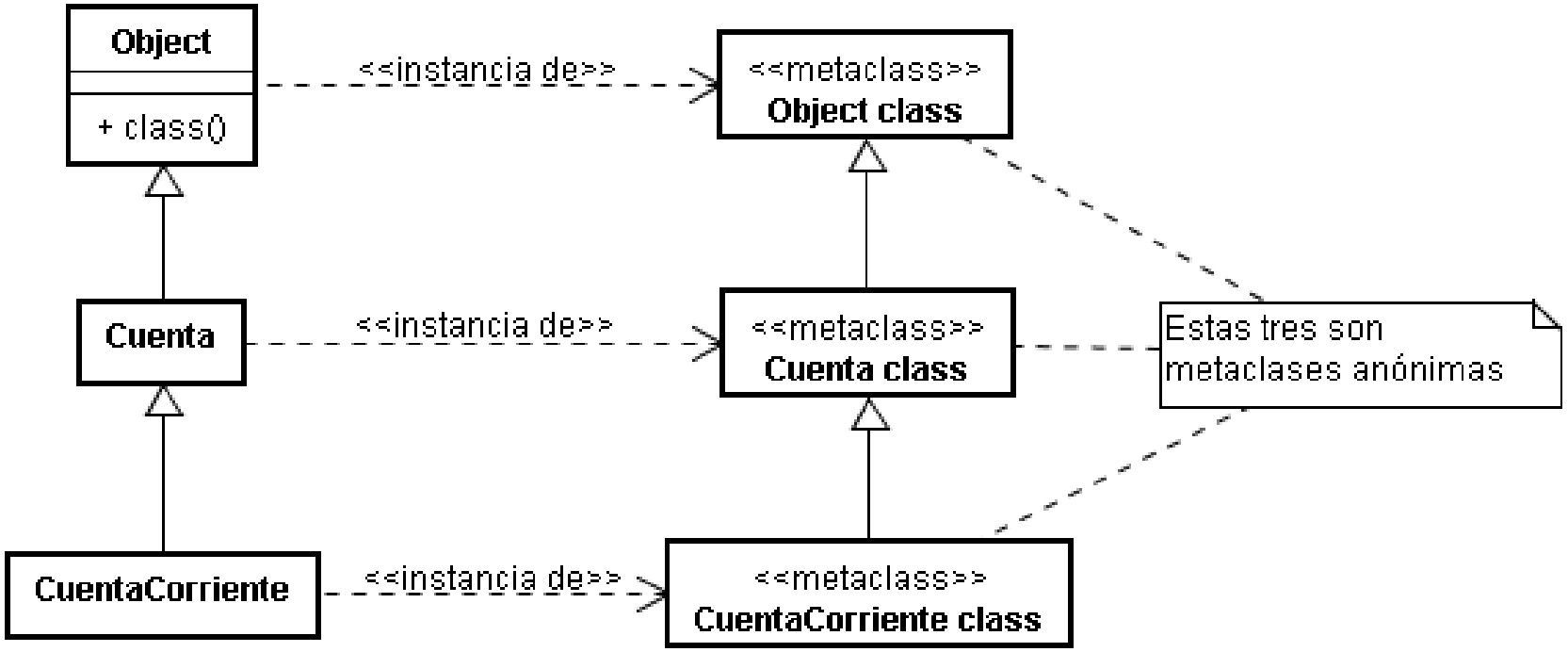
Cada metaclase hereda de Class

Class hereda de Behavior

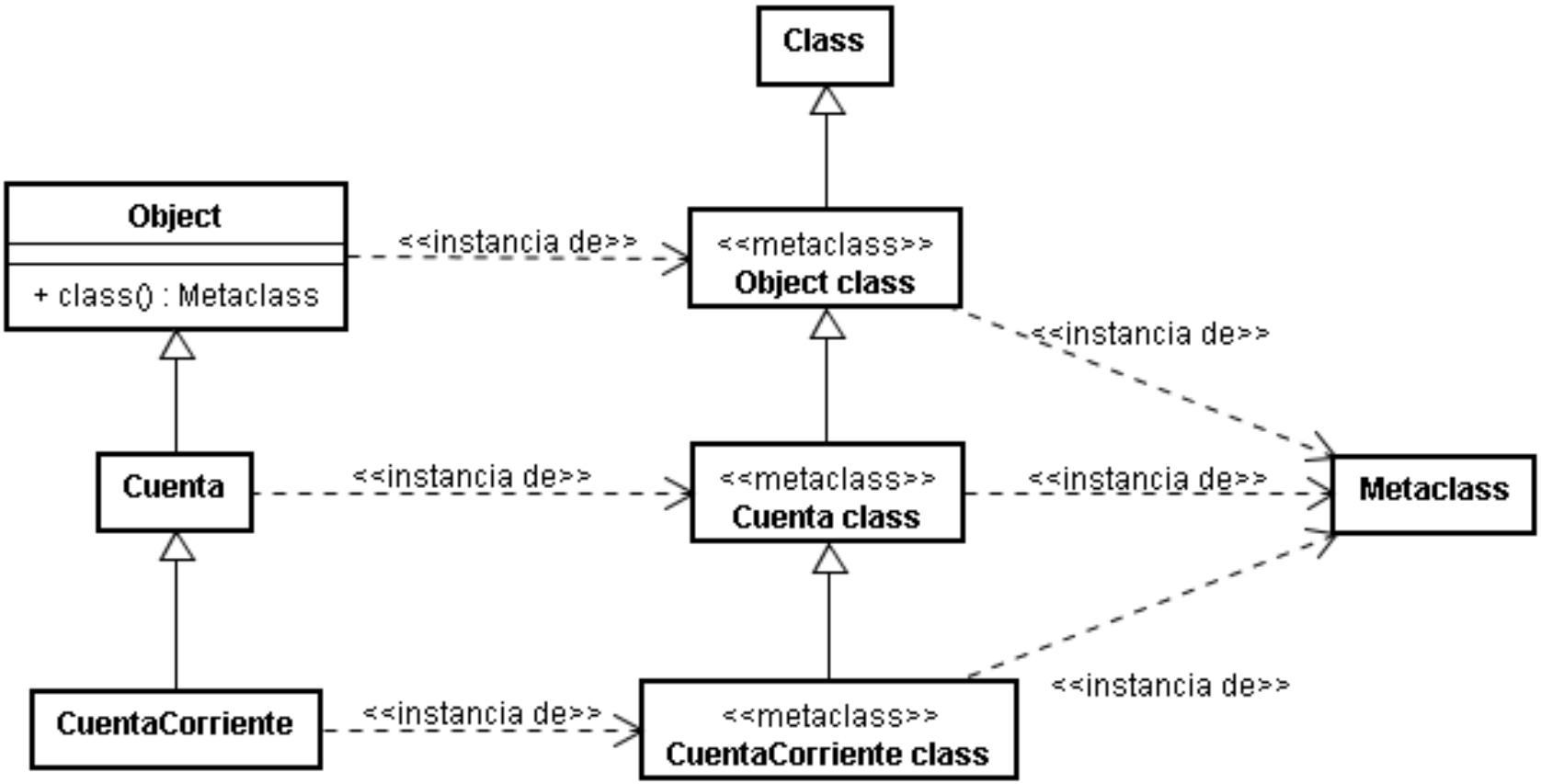
Las metaclases son instancias de Metaclass

La metaclase de la clase Metaclass es una instancia de Metaclass

Smalltalk: las clases son objetos

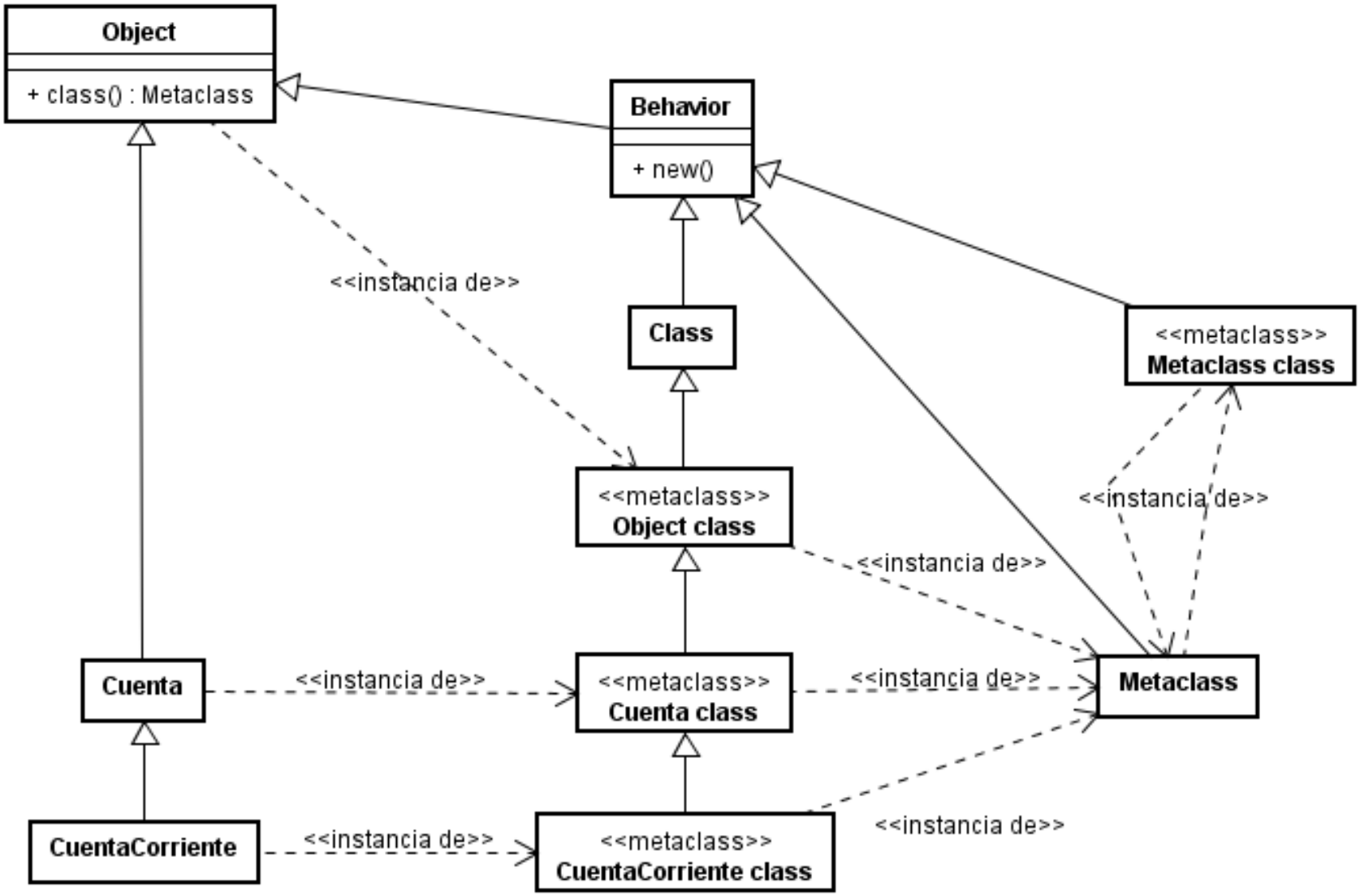


Smalltalk: jerarquía de metaclases



fiuba algo3

Smalltalk: todo cierra



fiuba algo3

Recapitulación

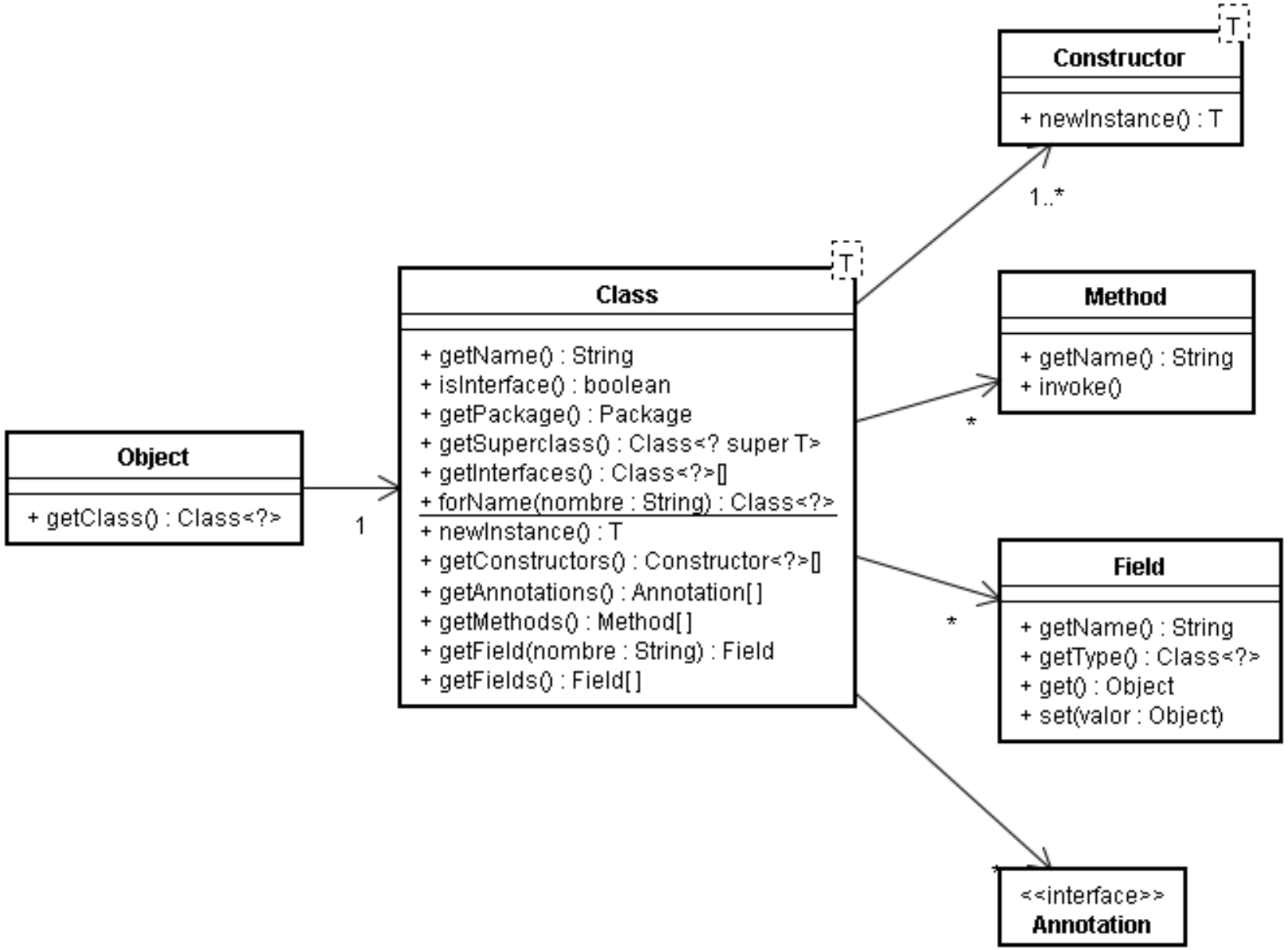
fiuba algo3

Recapitulación: preguntas

¿Qué es la clase Class de Java?

¿Qué es una metaclase en Smalltalk?

Reflexión en Java (1)



fiuba algo3

Reflexión en Java (2)

RTTI: el compilador debe conocer los tipos

¿Qué pasa si recibo un objeto por una red?

Puedo obtener el objeto Class y preguntar por métodos, atributos, interfaces, etc.

Paquete java.lang.reflect

La información debe estar en tiempo de ejecución

En general se maneja en forma automática

- Interrogación a componentes para saber qué eventos soporta

Ejemplo: creación de objetos

```
public class FabricaObjetos {
  public Object crearObjeto ( ) throws Exception {
       String nombre = leer("configuracion.txt");
       Class<?> aFabricar = Class.forName(nombre);
       Object nuevo = aFabricar.newInstance();
       return nuevo;
  }
}
```

Luego:

```
FabricaObjetos f = new FabricaObjetos();
Object x = f.crearObjeto();
```

fiuba algo3

Reflexión: ¿ya la usamos?

Framework Junit, SUnit

- ¿Cómo funciona?

Usa polimorfismo

- Métodos setUp() y tearDown()

Y reflexión

- Métodos “public void testXxx()”

Algo bastante común en todos los frameworks

Reflexión en Smalltalk

Más potente que en Java

También más compleja

2 niveles:

- Introspección
  - Similar a Java, con agregados
- Intersección
  - Actuación sobre el entorno de ejecución
  - Admite la metaprogramación

Ver “Pharo By Example”, capítulo 14

Introspección

Hay más control sobre el entorno de ejecución que en Java

Todos son objetos

- Workspace, Debugger, Inspector, Transcript

Hay más información

- Subclases
- Instancias existentes
- Toda la jerarquía de herencia
- Uso de objetos desde métodos
- Referencias cruzadas entre métodos

Intersección y metaprogramación

Control total de los objetos durante la ejecución

Permite la metaprogramación

Cambios de comportamiento en forma dinámica

- Por ejemplo, doesNotUnderstand
- O generar métodos que no existen
  - Ejemplo del Proxy

fiuba algo3

Ojo con reflexión

Podemos terminar generando cualquier cosa

Difícil de testear

Difícil de leer

No cualquiera la usa bien

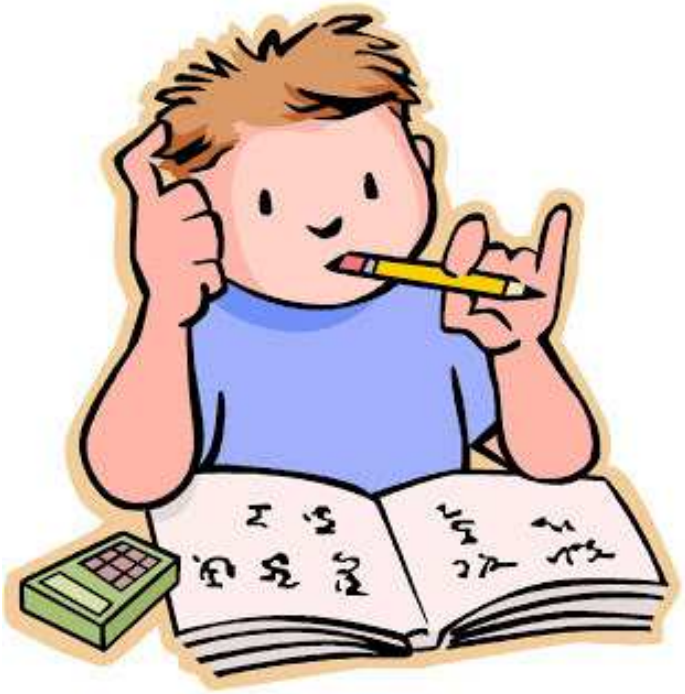
Recapitulación

fiuba algo3

Recapitulación: preguntas

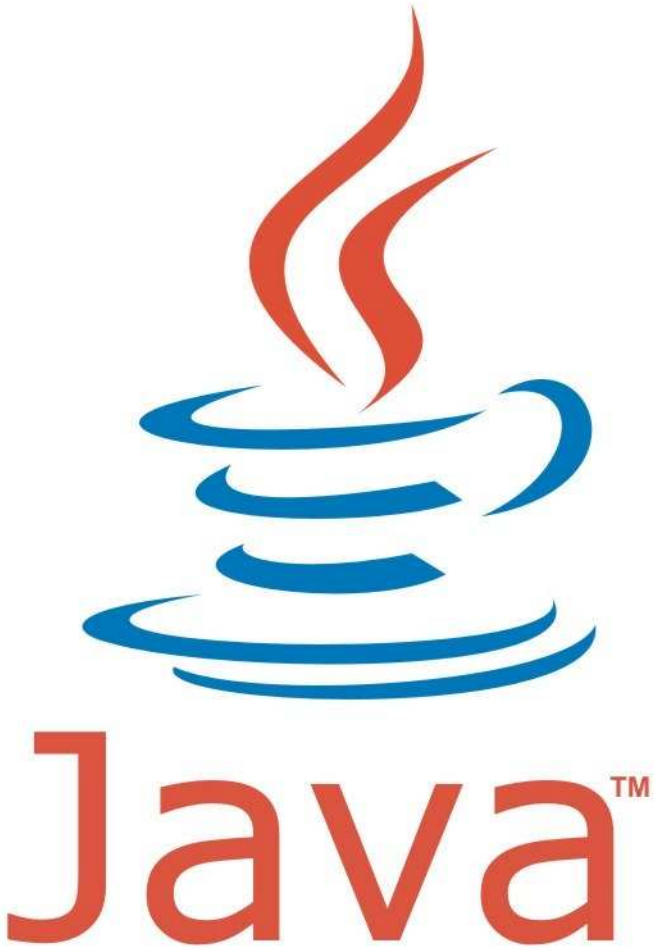
¿Para qué sirve entonces preguntar a una instancia cuál es su clase?

¿Qué otras cosas le puedo pedir a una instancia en tiempo de ejecución?



fiuba algo3

Trabajando en Java…



Java: historia

1991: Oak, para televisión interactiva

1995: Java 1.0

- Primera liberación real
- ¿Para software embebido?
- Java Applets y “el lenguaje de la web”

1997: Java 1.1 y compiladores JIT

- Mejora significativa de desempeño
- Comienza el auge industrial y académico

1999: Java 1.2 y plataformas J2SE, J2EE y J2ME

- Surgen los Servlets y mantienen el mito

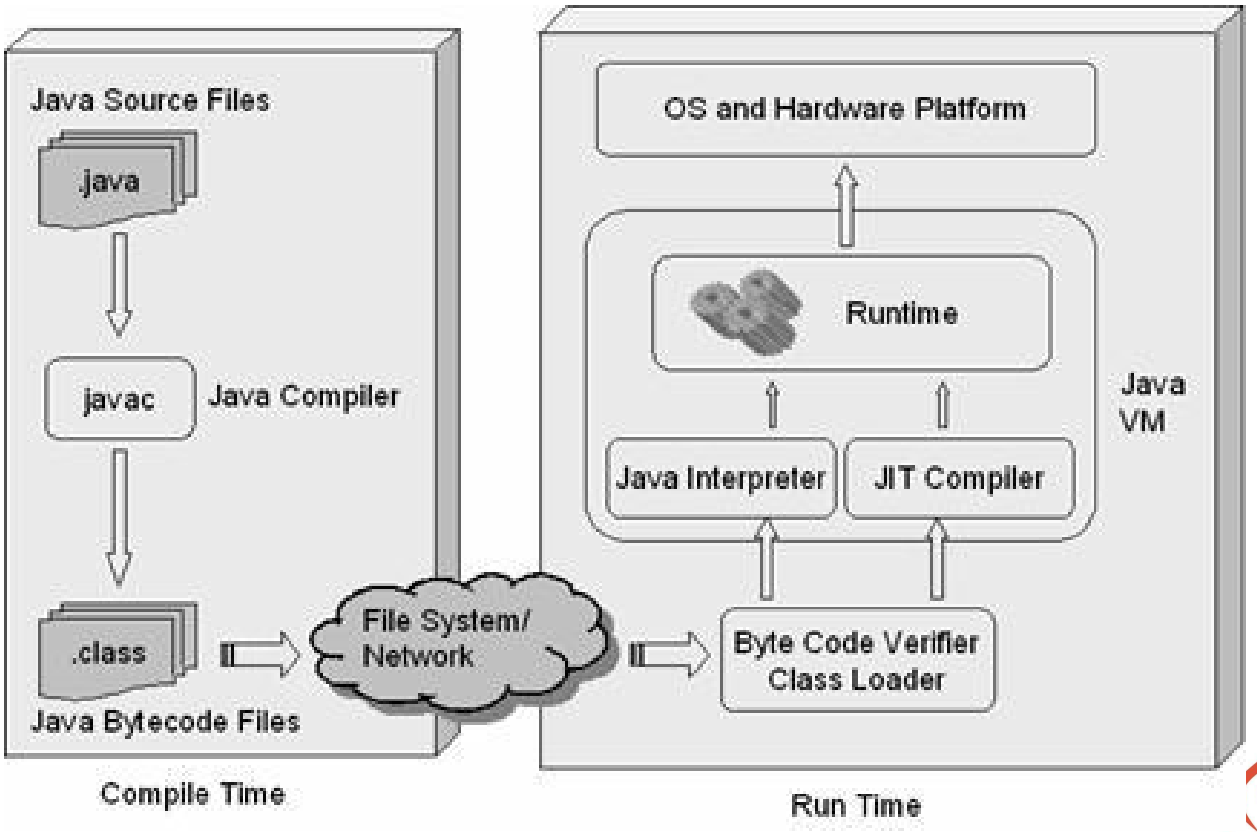
2007: se licencia Java bajo GNU GPL

2009-2010: Oracle absorbe Sun

Java: objetivos

- Disminuir la dependencia de plataformas específicas
  - Sin recompilación
    - Sólo requiere la JVM
  - Plataforma = hardware + sistema operativo
  - Celulares hasta supercomputadoras
  - Incluso hardware específico si SO ni JVM
    - Cuestiones de desempeño
- Sin grandes pretensiones de tiempo real y control explícito de memoria
- Foco en bajo costo de desarrollo

Java: funcionamiento



Java™

fiuba algo3

Java: implementaciones

- Oficial de Oracle
  - Linux, Windows, Solaris y Mac OS X
  - No es un estándar
- Hay muchas otras aceptadas
- Microsoft
  - Disputa legal llevó a sacar Java de Windows y aparición de .NET
- Android (Google)
  - Desarrollado en C
  - Java para aplicaciones
  - Usa Dalvik VM y bibliotecas diferentes
  - Hay disputa legal por uso “no estándar”

Lenguajes y JVM

Lenguajes creados para JVM	Adaptaciones de otros
Java	Erjang (Erlang)
BBj	Rhino (JavaScript)
Clojure	Free Pascal (Pascal)
Fantom	Quercus (PHP)
Groovy	Jython (Python)
MIDletPascal	NetRexx (REXX)
Scala	JRuby (Ruby)
Kawa	Jacl (TCL)

fiuba algo3

Trabajando en Smalltalk…

fiuba algo3

Smalltalk: historia

- 1971: Smalltalk-71, como desarrollo interno de Xerox PARC
  - Para la simbiosis humano-computadora que venía
  - Con fines educativos
- 1976: Smalltalk-76
  - Mejor desempeño
  - IDE incluido
- 1980: Smalltalk-80
  - Sale del ámbito exclusivo de Xerox
  - Incluye metaclases y todo POO
- 1996: surge Java
  - Varias “empresas Smalltalk” se pasan a Java
- 1998: Smalltalk estándar ANSI
- 2010s: revalorización académica

Smalltalk: ecosistema innovador

- Interfaces GUI / WIMP
- IDE integrado
- Herramientas de refactorización integradas
- Multimedia e hipertextos
- Prototipado rápido de aplicaciones
- Problema de la persistencia
  - Resuelto mediante la persistencia de la imagen
- Primeros patrones de diseño
  - MVC especialmente
- Métodos ágiles
  - XP especialmente
- Visión de la tablet por Alan Kay

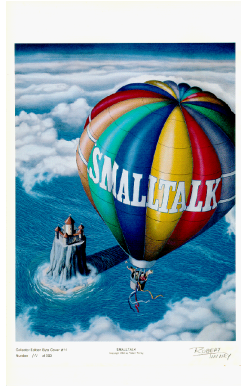
Smalltalk: implementaciones

Impulsado por muchos actores

- Xerox, IBM, Cincom, HP, Apple, universidades

Más de 40 implementaciones, entre ellas:

- Smalltalk/V
- Gemstone/S
- Visual Works
- GNU Smalltalk
- Visual Age
- Object Arts
- Squeak
- Pharo, derivado de Squeak
- Potato, Squeak escrito en Java



Claves

Suele ser posible saber la clase de una instancia en tiempo de ejecución

¡Usar con cuidado!

En Smalltalk todo es un objeto: ¡ya lo sabíamos!

Lectura obligatoria

Carlos Fontela, “Estado del arte y tendencias en Test-Driven development”

http://web.fi.uba.ar/~cfontela/Fontela_EstadoDelArteTDD_UNLP_EIS.pdf

(ojo que es largo)

Lecturas opcionales

Pharo By Example

- Capítulo 13, “Classes and Metaclasses”
- Capítulo 14, “Reflection”

Thinking in Java, Bruce Eckel

- Capítulo 4, “Initialization & Cleanup”
- Capítulo 12, “Run-time Type Identification”
- Apéndice A, “Passing & Returning Objects”
- Está en biblioteca
- Hay versión castellana

Orientación a objetos, diseño y programación, Carlos Fontela 2008

- Capítulo 20 “Los datos, los tipos y la memoria”

Qué sigue

Cierre de la materia

Terminar el TP final

fiuba algo3