Explicación práctica

Semana 17 de septiembre de 2018

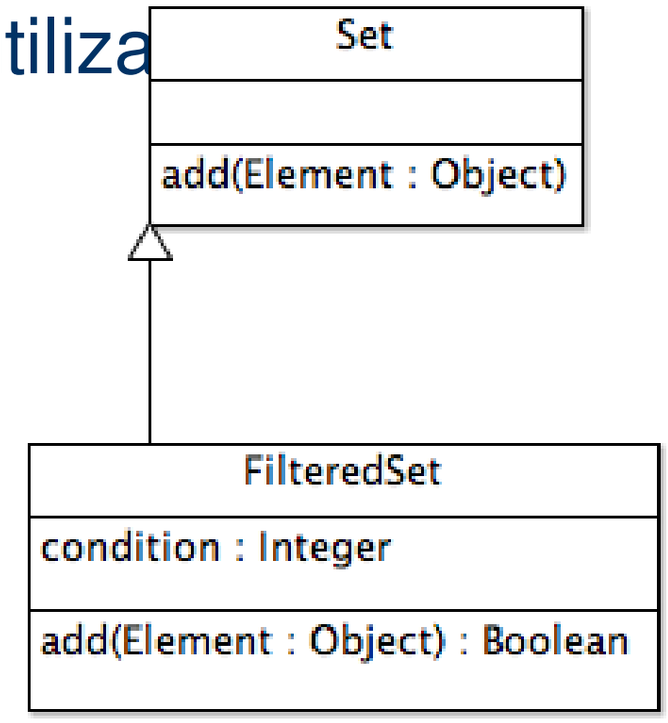
Temas

- Herencia - (práctico - no suplanta la teoría)
- Method lookup
- Initialize
- Test case - Asserts

Herencia

•Es el mecanismo por el cual las subclases reutiliza

comportamiento y estructura de su superclase.



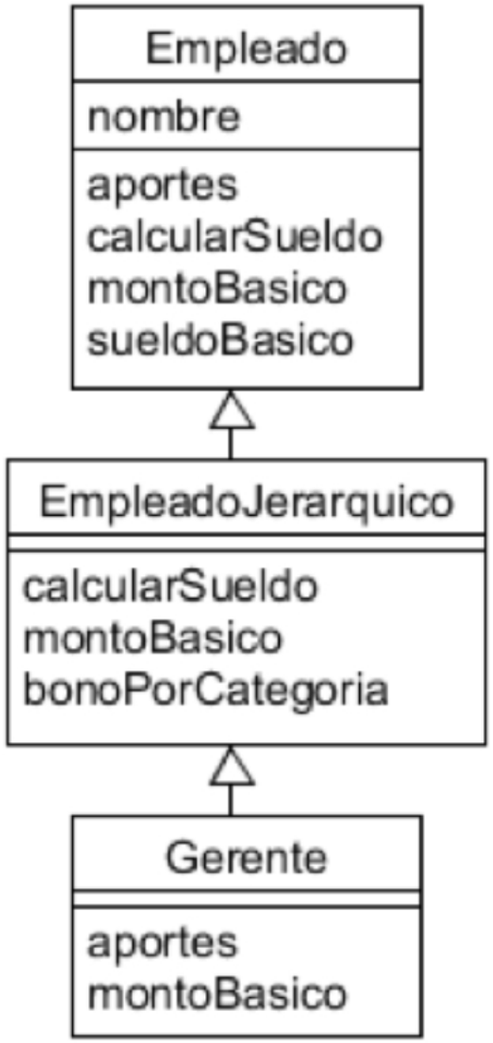
•La herencia permite:

• Crear una nueva clase como refinamiento de otra.

• Diseñar e implementar sólo la diferencia que presenta la nueva clase.

• Factorizar similitudes entre clases.

Method lookup



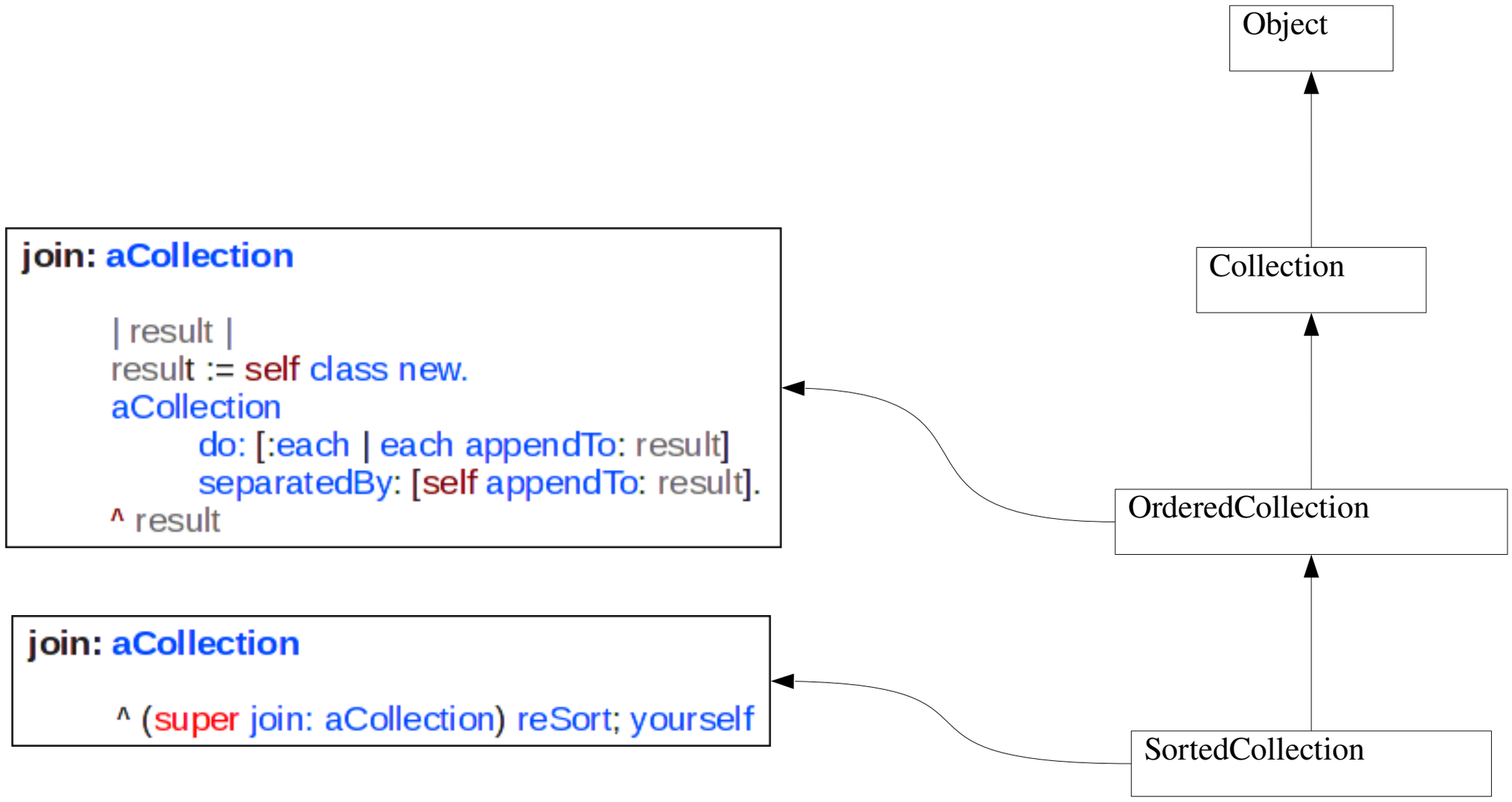
Empleado	EmpleadoJerarquico	Gerente
sueldoBasico ^self montoBasico - self aportes	**calcularSueldo** ^super sueldoBasico + self bonoPorCategoria	**aportes** ^self montoBasico * 0.05
montoBasico ^ 700	**montoBasico** ^ 850	**montoBasico** ^1000
aportes ^10	**bonoPorCategoria** ^ 200	

```
| gerente |

gerente := Gerente new.

gerente aportes.

gerente calcularSueldo
```



- Si evaluasemos: **asortedCollection join: anotherCollection**

super = OrderedCollection → … → Object
self = asortedCollection

#initialize

- Cuando se crea una instancia en Pharo (usando el mensaje new), automáticamente la instancia recibe el mensaje initialize.

- Esto significa que hay que definir el método #initialize en nuestras nuevas clases para que podamos asignarles los valores iniciales a nuestras variables de instancia, por ejemplo:

  - Que la variable de encendido de una farola tenga el valor de apagada. (estaEncendida:= false).

  - Que las vecinas de una farola sean una colección vacía:

    - vecinas:= Set new / OrderedCollection new.

- Veamos un ejemplo.

TestCase - #asserts

- Leer el capitulo 7 del libro de Pharo por Ejemplo
- Sobre los asserts
  - Existen diferentes alternativas

asserts

- #assert:
- #assert: equals:
- #assert: description:
- #deny:
- #deny:description:
- #fail
- #fail: