Paradigma de Objetos

Paradigmas de Programación
2do Cuat. 2017

Nuestro enfoque

- Conceptual/Aplicado
    - Introducimos conceptos fundamentales del paradigma:
        - ★ Objetos y modelo de Cómputo
        - ★ Clasificación y herencia
        - ★ Method dispatch
- Ilustraremos esos conceptos en el lenguaje **Smalltalk**

Resultado

Objetos que **colaboran** entre sí enviándose **mensajes**. Definición del paradigma axiomática (minimalista) que permite construirlo sin influencias (ni injerencia) de otros paradigmas.

Objetos

- Def 1. Objeto: Representación de un ente[1] del dominio del problema.

  - ¿Cómo es dicha representación?

  - ¿De qué manera?

[1]ente: elementos tangibles o intangibles que identico lo largo del proceso de aprendizaje.

Colaboración

- Def 2. Colaboración: Instanciación de un envío de mensajes.

  - Características:
    - Dirigida: el receptor esta presente y es alcanzable (visible) al momento del envío.
    - Sincrónica: es bloqueante
    - Anónima: receptor desconoce al emisor
    - Siempre hay respuesta

Mensaje y Método

- Def 3. Mensaje: Un objeto. Es una solicitud para que un objeto lleve a cabo una de sus operaciones.

- Def 4. Método: Un conjunto de colaboraciones asociado a un mensaje que un objeto dado sabe responder.

  - Mensaje especifica el **QUÉ**

  - Método especifica el **CÓMO**

Relación de Conocimiento

- Colaboradores Internos: conocidos como atributos o variables de instancia. Es lo que necesita un objeto para ser lo que realmente representa.

- Colaboradores Externos: se conocen durante el tiempo de vida de la ejecución del método.

- Colaboradores Globales: entes que son accesibles desde cualquier lugar. Ejemplo: números, strings, fechas.

Sintaxis Smalltalk

- Unarios:
  - ★ unArray **size**
  - ★ 3 **factorial**

- Binarios:
  - ★ 1+3
  - ★ December, 2014

- Keyword:
  - ★ unArray **at:**1 **put:** 2
  - ★ miCuenta **transferir:** 300 **a:** otraCuenta

- Prioridad

  unarios > binarios > keywords (de izquierda a derecha)

  Ejemplo 1: ‘objetos uno’ size * 4 between: 6 negated and: 3 factorial * 5

Clasificación

Motivación: objetos polimórficos cuya implementación es la misma.

- Def 6. Clase: un objeto que representa un conceptos abstractos del dominio del problema
  - ★ Definen el comportamiento y la forma de un conjunto de objetos (sus instancias)
  - ★ Todo objeto es instancia de alguna clase

Componentes de una Clase

- Un nombre
- Definición de variables de instancia
- Métodos de instancia y de clase
- Por cada método se especifica
  - ★ su nombre
  - ★ parámetros formales
  - ★ cuerpo

Self

- self: Forma en la cual un objeto se refiere a si mismo.
- Pseudo-variable
  - Una de las palabras reservadas.
  - Siempre está y no se puede asignar.
  - Representa al objeto que recibió el mensaje y esta ejecutando el método.

Ejercicio 1

Definir una clase **Complex** que implemente la clase de los números complejos. Definir métodos para los siguientes mensajes:

- **+** (infjo): que devuelva la suma compleja de sus dos argumentos.

- **abs**: devuelve el valor absoluto.

Subclasificación

- Permite decir que hay conceptos más abstractos que otros.
- No suele haber realizaciones concretas de esas clases más abstractas.
- Una subclase es algo más concreto que una super clase.
- Una de las herramientas peor usadas del paradigma.
- La herencia es un mecanismo para construir teor´ıa, no para ahorrar código.

Herencia

- Hay dos tipos de herencia
  - ★ Simple: una clase tiene una única clase padre (salvo la clase raíz object)
  - ★ Múltiple: una clase puede tener m´as de una clase padre
- La gran mayoría de los lenguajes OO utilizan herencia simple
- La herencia múltiple complica el proceso de method dispatch

Detrás del modelo de cómputo: Method dispatch

- La interacción entre objetos se lleva a cabo a través de envío de mensajes
- Al recibir un mensaje se activa el método correspondiente
- Para poder procesar este mensaje es necesario hallar la declaración del método que se pretende ejecutar
- El proceso de establecer la asociación entre el mensaje y el método a ejecutar se llama **method dispatch**
- Si el method dispatch se hace en tiempo de
  - ★ compilación (i.e. el método a ejecutar se puede determinar a partir del código fuente): se habla de method dispatch estático
  - ★ ejecución: se habla de method dispatch dinámico

Super

- Def. Super: pseudovariable que me permite usar la implementación de un método de mi superclase.

- super = self

Algoritmo

- input: mensaje m, objeto receptor r

```
1. class := r class
2. buscar m en class
3. si lo encuentro, evalúo el método en el
contexto de r.
4. si no class := class superclass
5. si class no es nil, go to 2
6. r doesnotunderstand: m
```

Ejercicio 2

```
Object subclass: #C1
"Métodos de instancia"
 m1
 ^self m2

 m2
  ^13
```

```
C1 subclass: #C2
"Métodos de instancia"
 m1
  ^22

 m2
  ^23

 m3
  ^super m1
```

```
C2 subclass: #C3
"Métodos de instancia"
 m1
  ^32

 m2
  ^33
```

(c2 new) m3. ----> Qué valor devuelve?