INSTITUTO POLITÉCNICO NACIONAL

Escuela Superior de Cómputo

Unidad de aprendizaje

"Programación Orientada a Objetos"

Grupo:

2CM1

Profesor:

Daniel Cruz García

Tarea 9:

"Polimorfismo"

Alumna:

Luciano Espina Melisa

Fecha de entrega:

25/Mayo/2018

Ligadura estática

Cuando se crea un código mediante un lenguaje de alto nivel, en los procesos de compilación y enlace se deciden las posiciones que ocuparán en la memoria las llamadas a los métodos. Esta decisión se toma en tiempo de compilación y enlace, o sea, cuando se está generando el código ejecutable, antes de correr el programa. Esto es conocido como enlace anticipado o ligadura temprana o estática, (early binding)

Algunos lenguajes tienen una lista interna de métodos con sus direcciones y cuando se produce una llamada el compilador inserta la dirección de este método en el lugar de llamada. Este tipo de ligadura tiene sus ventajas, entre las que cabe destacar la rapidez de ejecución, ya que sólo se necesita el tiempo de paso de argumentos y almacenamiento en la pila de los parámetros a restaurar.

También presenta graves problemas, exige conocer de antemano todos los objetos que se usarán en todas las llamadas a los métodos. En algunas ocasiones esto es posible, pero existen otras en que no es posible saber a qué objeto llamar hasta el momento de la ejecución. [1]

Ligadura dinámica

Cuando existe un conflicto entre un método de una superclase y un método de la subclase, el comportamiento correcto es que el método de la subclase sobrescriba al de la superclase.

Si estamos llamando a un método de la subclase desde una variable que ha sido declarada del tipo de la superclase. ¿Cómo se consigue que funcione correctamente?

Significa que la forma dinámica del objeto determina la versión de la operación que se aplicará.

Esta capacidad de las operaciones para adaptarse automáticamente a los objetos a los cuales se aplican es una de las propiedades más importantes de la orientación a objetos.

- Consiste en realizar el proceso de ligadura en tiempo de ejecución siendo la forma dinámica del objeto la que determina la versión del método a ejecutar.
- Se utiliza en todos los métodos de instancia de Java que no son privados ni final.

Deben presentar ligadura dinámica solo aquellos que pueden ser redefinidos.

Por ejemplo, en Java, los métodos de clase y los métodos de instancia privados y/o finales no presentan ligadura dinámica.

En Java, si no se especifica nada se entenderá que el método puede ser redefinido y por tanto debe presentar ligadura dinámica. [2]

Polimorfismo

En programación orientada a objetos el polimorfismo se refiere a la posibilidad de definir clases diferentes que tienen métodos o atributos denominados de forma idéntica, pero que se comportan de manera distinta. El concepto de polimorfismo se puede aplicar tanto a funciones como a tipos de datos. Así nacen los conceptos de funciones polimórficas y tipos polimórficos. Las primeras son aquellas funciones que pueden evaluarse o ser aplicadas a diferentes tipos de datos de forma indistinta; los tipos polimórficos, por su parte, son aquellos tipos de datos que contienen al menos un elemento cuyo tipo no está especificado. [3]

Tipos de polimorfismo

Polimorfismo de sobrecarga

Para empezar con esta entrada, se ha de decir que el término "Polimorfismo" es una palabra de origen griego que significa "muchas formas". Este término se utiliza en la POO para "referirse a la propiedad por la que es posible enviar mensajes sintácticamente iguales a objetos de tipos distintos".

El polimorfismo de sobrecarga ocurre cuando las funciones del mismo nombre existen, con funcionalidad similar, en clases que son completamente independientes una de otra (éstas no tienen que ser clases secundarias de la clase objeto). Por ejemplo, la clase complex, la clase image y la clase link pueden todas tener la función "display". Esto significa que no necesitamos preocuparnos sobre el tipo de objeto con el que estamos trabajando si todo lo que deseamos es verlo en la pantalla.

Por lo tanto, el polimorfismo de sobrecarga nos permite definir operadores cuyos comportamientos varían de acuerdo a los parámetros que se les aplican. Así es posible, por ejemplo, agregar el operador + y hacer que se comporte de manera distinta cuando está haciendo referencia a una operación entre dos números enteros (suma) o bien cuando se encuentra entre dos cadenas de caracteres (concatenación).

El Polimorfismo es uno de los 4 pilares de la programación orientada a objetos (POO) junto con la Abstracción, Encapsulación y Herencia.

Polimorfismo paramétrico (también llamado polimorfismo de plantillas)

El polimorfismo paramétrico es la capacidad para definir varias funciones utilizando el mismo nombre, pero usando parámetros diferentes (nombre y/o tipo). El polimorfismo paramétrico selecciona automáticamente el método correcto a aplicar en función del tipo de datos pasados en el parámetro.

Por lo tanto, podemos por ejemplo, definir varios métodos homónimos de addition() efectuando una suma de valores.

- El método int addition (int,int) devolvería la suma de dos números enteros.

- El método float addition (float, float) devolvería la suma de dos flotantes.
- El método char addition (char, char) daría por resultado la suma de dos caracteres definidos por el autor.

Una signature es el nombre y tipo (estático) que se da a los argumentos de una función. Por esto, una firma de método determina qué elemento se va a llamar.

Polimorfismo de inclusión (también llamado redefinición o subtipado)

La habilidad para redefinir un método en clases que se hereda de una clase base se llama especialización. Por lo tanto, se puede llamar un método de objeto sin tener que conocer su tipo intrínseco: esto es polimorfismo de subtipado. Permite no tomar en cuenta detalles de las clases especializadas de una familia de objetos, enmascarándolos con una interfaz común (siendo esta la clase básica).[4]

Bibliografía

- [1] Programación (POO) | Programación Orientada a Objetos (POO) – Investigación | Manchoneria.es [Online] Available: http://www.manchoneria.es/colaboracion/tipo/1607/programacion-orientada-a-objetos-poo
- [2] s/a Herencia java | Dis.um.es [Online] Available: http://dis.um.es/~lopezquesada/documentos/IES_1213/IAW/curso/UT3/ActividadesAlumnos/13/ligadura.htm
- [3] Polimorfismo (Informática) – EcuRed | Ecured.cu [Online] Available: https://www.ecured.cu/Polimorfismo_(Inform%C3%A1tica)
- [4] S/A Polimorfismo en Java (Parte I), con ejemplos – Jarroba| Jarroba [Online] Available: https://jarroba.com/polimorfismo-en-java-parte-i-con-ejemplos/