Programación orientada a objetos

Un programa en el paradigma orientado a objetos se define como *Objetos que colaboran entre sí enviándose mensajes*.

A continuación se detallan los conceptos más importantes del paradigma orientado a objetos.

Conceptos generales

Un **objeto** es la representación de un ente del dominio de problema. Donde un **ente** es cualquier cosa que podamos observar, hablar sobre, etc. La esencia del ente es modelado por los mensajes que el objeto sabe responder.

Un **mensaje** es una especificación sobre qué puede hacer un objeto. Los mensajes son un objeto. Por lo tanto representa un ente de la realidad, pero del dominio de la "comunicación".

Se debería poder decir qué representa un objeto a partir de los mensajes que sabe responder.

Una **colaboración** es el hecho por el cual dos objetos se comunican por medio de un mensaje.

En toda colaboración existe:

- Un emisor del mensaje.
- Un receptor del mensaje.
- Un conjunto de objetos que forman parte del mensaje (colaboradores externos o parámetros).
- Una respuesta.

Algunas carácteristicas de las colaboraciones son:

- Son dirigidas (no broadcast).
- Son sincrónicas, es decir, el emisor no continua hasta que obtenga una respuesta del receptor.
- El receptor desconoce al emisor. Su relación será siempre la misma no importa quién le envía el mensaje.
- Siempre hay respuesta.

Un **prototipo** es un objeto ejemplar que representa el comportamiento de un conjunto de objetos similares. Este mecanismo de representación de conocimiento se utiliza en lenguajes de prototipación o "Wittgenstein-nianos"

Una **clase** es un objeto que representa un concepto o idea del dominio de problema. Por ser un objeto, sabe responder mensajes.

Las clases existen como mecanismo de representación de conocimiento en lenguajes de clasificación o Aristotélicos. Python es un lenguaje de clasificación.

Un **método** es un objeto que representa un conjunto de colaboraciones. Es evaluado como resultado de la recepción de un mensaje por parte de un objeto. El método se busca utilizando el algoritmo *Method Lookup*.

En las clases se definen dos tipos de métodos: los *métodos de instancia* y los *métodos de clase*.

Los *métodos de instancia* son aquellos métodos que implementan los mensajes que se envían a los objetos que son instancias de una clase. En cambio, los *métodos de clase* son aquellos métodos que implementan los mensajes que se envían a la clase.

La **relación de conocimiento** es la única relación que existe entre objetos. Es la que permite que un objeto colabore con otro.

Una **variable** es un nombre contextual que se le asigna a una relación de conocimiento. Implica que el objeto conocido es llamado, en el contexto del conocedor, de acuerdo a dicho nombre.

En las clases se definen dos tipos de variables: las *variables de instancia* y las *variables de clase*.

Las *variables de instancia* representan las relaciones de conocimiento del objeto que es instancia de una clase. En cambio, las *variables de clase* representan las relaciones de conocimiento de la clase.

Polimorfismo: Dos o más objetos son polimórficos entre sí para un conjunto de mensajes, si responden a dicho conjunto de mensajes semánticamente igual. Donde semánticamente igual significa que:

- Hacen lo mismo.
- Reciben objetos polimórficos.
- Devuelven objetos polimórficos.

La **subclasificación** es la herramienta utilizada para organizar el conocimiento en ontologías. Es una relación "estática" entre clases. Permite organizar el conocimiento en clases abstractas (que representan conceptos abstractos) y clases concretas (que representan conceptos con realizaciones concretas).

Un **tipo** es un conjunto de mensajes. Los tipos no son únicamente un nombre, sino que define semántica.

Problemas de la clasificación

Los lenguajes de clasificación al representar conceptos (por medio de clases) nos obliga a conocer el concepto antes de crear la clase. Y como el desarrollo de software es representar conocimiento en una computadora y por lo tanto es un proceso de aprendizaje, en un primer instante no conozco suficientemente el

problema como para saber qué conceptos o ideas tengo que representar. Con lo cual, los lenguajes de clasificación nos obligan a tener una clase y por lo tanto su nombre antes del objeto concreto, lo cual es antinatural. Es decir, es inverso a nuestro modo de conocer y aprender. Ya que nosotros aprendemos desde lo concreto a lo general.

Problemas de la subclasificación

La subclasificación tiene un problema parecido, ya que la jerarquía de clases se especifica desde la clase más abstracta a la más concreta (de manera inversa a como se obtiene el conocimiento).

Además rompe el encapsulamiento ya que la subclase debe conocer la implementación de la superclase.

Method Lookup

Cuando se envía un mensaje a un objeto, se tiene que buscar el método que implementa el mensaje. El algoritmo que busca el método que implementa el mensaje recibido se conoce con el nombre de *Method Lookup*.

A continuación, se explica como funciona este algoritmo.

Cuando se envía un mensaje a un objeto, primero se busca el método que implementa el mensaje en la clase del objeto.

1. Si lo encuentra evalúa el método.
2. Si no lo encuentra, busca el método en la superclase. Si en la superclase tampoco lo encuentra busca el método en la superclase de esta. Y así siguiendo hasta no haber más superclases (es decir,cuando se llega a la clase *object*).

Si no se encuentra el método en toda la jerarquía de clases desde la clase del objeto que recibe el mensaje hasta la clase *object,* entonces se lanza una excepción de tipo *AttributeError*. Indicando que el objeto no sabe responder el mensaje.

La clase super (en otros lenguajes implementados como una pseudo variable) indica al algoritmo *Method Lookup* que empiece a buscar el método desde la superclase de la clase pasada como primer parámetro, evaluando el método con el objeto pasado en el segundo parámetro.