Estudiantes

11-10199 - Jonnathan Ng

11-10866 - Joel Rivas

Respuestas

> Los lenguajes de programación orientados a objetos que poseen herencia simple están limitados a incorporar comportamientos de un solo ancestro al momento de definir una clase.

[No] No necesariamente, ya que si se dispone de una clase A que hereda de una clase B y esta a su vez de una clase C (A < B) y (B < C), A incluye el comportamiento tanto de su ancestro mas proximo (B) como de sus ancestros superiores (C).

> Lenguajes de POO con un sistemas de tipos estático (C++, Java, C#) no tienen la posibilidades de elegir la implementación de un método a tiempo de ejecución (despacho dinámico).

[No] Ya que lenguajes como Java poseen despacho dinámico por defecto cuando se hace una sobreescritura (Override) de métodos, además lenguajes como C++/C# es posible elegir la implementación de un método a tiempo de ejecución si se utilizan métodos virtuales en las clases.

> La introspección y reflexibidad son conceptos que se manejan en la POO pero no guardar ninguna relación entre sí.

[No] Ambos permiten que un objeto o clase informe sobre la clase a la que pertenece (un objeto pregunte por si mismo). Sin embargo, la reflexividad puede modificar la estructura del mismo objeto, tambien puede alterar las reglas del codigo a tiempo de ejecucion, y por ende, alterar elementos de la clase existentes dinámicamente.

> En un lenguaje con un sistema de tipos dinámico la sobrecarga de métodos es innata y representa una comodidad dado que permite implementar un mismo método para distintos tipos.

[Si] Redefinir operaciones y sobrecargar nuevos operadores es una tarea facil por la forma en la

que operan los tipos dinámicos.

> En los lenguajes POO existen los términos interfaz, módulo, clase abstracta, rol, etc; definidos como objetos que pueden encapsular definiciones de clases o implementaciones concretas de métodos.

[Si] Ya que para el caso de las interfaces sólo se definen las firmas de los métodos de las clases y para el caso de los módulos se encuentran las implementaciones de los métodos.

> Los métodos virtuales permiten asociar, al momento de compilar, una implementación de un método sobrecargado con una llamada del mismo; eliminando el **overhead** del despacho dinámico.

[No] Los métodos virtuales permiten asociar a tiempo de ejecución la implementación de un método sobrecargado, utilizando estructuras adicionales como los vtables, que facilitan elegir la version adecuada, pero aun existe un overhead ya que esta decision aun debe realizarse a tiempo de ejecución.

> Cuando un lenguaje de POO tiene herencia simple no ocurre el problema del diamante pero de igual forma pueden existir llamadas ambiguas de métodos, dado que incorporar interfaces, módulos, protocolos, etc, no evita colisión de nombres.

[Si] En lenguajes como Java o C# se permite la implementacion de varias interfaces. Esto puede ocasionar conflictos de nombres que en el caso de C# la solución es utilizar lo que se denomina la implementación explícita de interfaces, en donde se precede con el nombre de la interfaz, el método a ser implementado. En lenguajes como Ruby sucede algo similar con los mixins. Otros casos como Java no permiten esta facilidad, por lo que es necesario recurrir a otras técnicas como el uso del patrón de diseño de delegación.

> El paso de mensaje es un término que se maneja en modelos concurrentes, también de POO y es equivalente a la llamada de una función.

[Si] Una llamada de método también se conoce como el paso de mensajes, se suele implementar como llamadas a los métodos de otros objetos.

> Sin importar la herencia del lenguaje de POO, una clase podría tener más de un ancestro.

[Si] En el caso de la herencia simple si se considera la clase padre (el padre del padre), tendría más

de un ancestro, por lo tanto la clase heredará los comportamientos de cada uno de ellos. Por otro lado, en la herencia múltiple, ésta tendrá mas de un ancestro inmediato, además de tomar en cuenta el caso anterior.