Diseño orientado a objetos: la forma de Go

Vamos a ver cómo se compara Go con los pilares de la programación orientada a objetos: encapsulación, herencia y polimorfismo. Esas son las características de los lenguajes de programación basados en clases, que son los lenguajes de programación orientados a objetos más populares.

En lo fundamental, los objetos son construcciones de lenguaje que tienen un estado y un comportamiento que opera sobre el estado y lo expone selectivamente a otras partes del programa.

Encapsulado

Go encapsula las cosas a nivel de paquete. Los nombres que empiezan con una letra minúscula únicamente son visibles dentro de ese paquete. Puedes ocultar cualquier cosa en un paquete privado y solamente exponer tipos específicos, interfaces y funciones de fábrica.

Herencia

La herencia o subclases siempre fue un tema controversial. Hay muchos problemas con la herencia de la implementación (a diferencia de la herencia de la interfaz). La herencia múltiple, como se ha implementado por C++, Python y otros lenguajes, sufre el problema del diamante, pero incluso la herencia única no es tarea fácil con el problema frágil de la clase base.

Los lenguajes modernos y el pensamiento orientado a objetos ahora favorecen la composición sobre la herencia. Go se lo toma en serio y no tiene ningún tipo de jerarquía. Te permite compartir detalles de implementación a través de la composición. Pero Go, en un giro muy extraño (que posiblemente se originó por preocupaciones pragmáticas), permite la composición anónima mediante la incrustación.

En efecto, la composición mediante la incrustación de un tipo anónimo es equivalente a la herencia de la implementación. Una estructura incrustada es tan frágil como una clase base. También puedes incrustar una interfaz, lo que equivale a heredar de una interfaz en lenguajes como Java o C++. Incluso puede provocar un error de tiempo de ejecución que no se descubra en el momento de la compilación si el tipo de incrustación no implementa todos los métodos de interfaz.

Polimorfismo

El polimorfismo es la esencia de la programación orientada a objetos: la capacidad de tratar objetos de diferentes tipos de forma uniforme siempre que se adhieran a la misma interfaz. Las interfaces de Go brindan esta capacidad de una manera muy directa e intuitiva.