Consulta evaluable #2

Estudiantes:
Brayan Baena Ramirez
Brayam Alexander Chica Betancur

Profesor: Plinio Neira

Materia: Lenguaje de Programación

Uniremington

Ingeniería de sistemas

Rionegro – Antioquia

02/05/2023

Modificadores

En Java, los modificadores son palabras clave que se utilizan para modificar la accesibilidad y el comportamiento de las clases, métodos, variables, constructores y interfaces.

Hay cuatro tipos de modificadores en Java:

Modificadores de acceso:

Estos modificadores determinan la accesibilidad de una clase, método, variable o constructor. Hay tres tipos de modificadores de acceso en Java:

public: El elemento es accesible desde cualquier lugar, dentro o fuera de la clase.

protected: El elemento es accesible dentro de la misma clase y dentro de las clases derivadas (subclases) de esa clase.

private: El elemento es accesible solo dentro de la misma clase.

Modificadores de no acceso:

Estos modificadores no afectan a la accesibilidad de un elemento, sino que se utilizan para modificar su comportamiento. Los dos modificadores de no acceso en Java son:

static: El elemento pertenece a la clase en lugar de a una instancia específica de la clase. Esto significa que se puede acceder al elemento sin crear una instancia de la clase.

final: El elemento no se puede modificar una vez que se ha inicializado.

Modificadores de herencia:

Estos modificadores se utilizan para controlar la herencia de una clase. El único modificador de herencia en Java es:

abstract: Una clase abstracta no se puede instanciar y se utiliza como una clase base para otras clases que la extienden.

Modificadores de sincronización:

Estos modificadores se utilizan para controlar el acceso a los métodos y bloques de código en entornos multi-hilo (múltiples hilos de ejecución). El único modificador de sincronización en Java es:

synchronized: Un método o bloque de código sincronizado solo puede ser accedido por un hilo a la vez.

Encapsulamiento

La encapsulación es un principio de programación orientada a objetos que se refiere a la ocultación de los detalles internos de una clase y la protección de su estado y comportamiento.

En Java, la encapsulación se logra mediante el uso de modificadores de acceso en las variables y métodos de una clase. Los modificadores de acceso son palabras clave que determinan la accesibilidad de una variable o método desde fuera de la clase. Hay tres tipos de modificadores de acceso en Java:

public: La variable o método es accesible desde cualquier lugar, dentro o fuera de la clase.

private: La variable o método solo es accesible dentro de la misma clase.

protected: La variable o método es accesible dentro de la misma clase y dentro de las clases derivadas (subclases) de esa clase.

Al definir una clase, es una buena práctica declarar las variables de instancia (también conocidas como campos) como privadas y proporcionar métodos públicos para acceder a ellas (también conocidos como getters y setters). De esta manera, los detalles internos de la clase están ocultos y solo se puede acceder a ellos a través de los métodos públicos.

Paquetes / Api

Un paquete es un conjunto de clases y subpaquetes relacionados que se utilizan para organizar y agrupar el código. Los paquetes se utilizan para evitar conflictos de nombres entre clases, para facilitar la navegación del código y para proporcionar un mecanismo de acceso controlado a las clases y recursos.

Por ejemplo, la clase Scanner se encuentra en el paquete java.util, que es un paquete estándar de Java que contiene clases y utilidades para trabajar con estructuras de datos, fechas, entrada/salida y otros recursos.

Por otro lado, una API (Application Programming Interface, Interfaz de Programación de Aplicaciones en español) es un conjunto de clases, interfaces y métodos que se proporcionan para que los desarrolladores puedan utilizarlos en sus aplicaciones. Las APIs se utilizan para proporcionar una interfaz clara y coherente para el uso de los recursos del sistema y para fomentar la reutilización del código.

Por ejemplo, la API de Java incluye clases y métodos para trabajar con estructuras de datos, entrada/salida, redes, gráficos y muchos otros recursos. Los desarrolladores pueden utilizar estas clases y métodos en sus aplicaciones para realizar tareas comunes.

Herencia

La herencia en Java es un mecanismo que permite que una clase adquiera las propiedades y comportamientos de otra clase. En términos sencillos, podemos pensar en la herencia como una relación de "es un/a" entre dos clases.

En la herencia, la clase que adquiere las propiedades y comportamientos se llama "subclase" o "clase hija", y la clase que proporciona las propiedades y comportamientos se llama "superclase" o "clase padre". La subclase hereda todos los miembros no privados de la superclase, incluyendo variables de instancia, métodos y otros constructores.

Por ejemplo, podemos tener una superclase Vehiculo que contiene métodos y variables de instancia comunes a todos los vehículos, como la velocidad y la dirección. Luego, podemos crear una subclase Coche que herede de la clase Vehiculo y agregue métodos y variables de instancia específicos para los coches, como el número de puertas y el tipo de combustible.

La herencia en Java es una herramienta poderosa que nos permite reutilizar código y crear jerarquías de clases que reflejen la realidad del problema que estamos intentando resolver. Con la herencia, podemos crear subclases más específicas que heredan las propiedades y comportamientos de sus superclases y que, al mismo tiempo, agregan funcionalidades y comportamientos específicos para cada subclase.

Polimorfismo

El polimorfismo en Java es un concepto fundamental de la programación orientada a objetos que nos permite utilizar un objeto de una clase hija de manera transparente en lugar de un objeto de su clase padre, lo que nos da flexibilidad y nos permite escribir código más genérico y reutilizable.

En Java, el polimorfismo se logra a través de la herencia y de las interfaces. Cuando una clase hija hereda de una clase padre, puede utilizar los métodos y variables de instancia de la clase padre. Sin embargo, también puede redefinir los métodos heredados o agregar nuevos métodos, lo que le permite tener un comportamiento propio y específico.

Cuando utilizamos polimorfismo, podemos crear una referencia a un objeto de una clase padre, pero asignarle un objeto de una clase hija en tiempo de ejecución.

El polimorfismo también se puede lograr a través de interfaces. Las interfaces son un conjunto de métodos que una clase puede implementar, y una clase puede implementar múltiples interfaces. Cuando una clase implementa una interfaz, podemos utilizar una referencia de la interfaz para referirnos a un objeto de la clase que la implementa.

Clases Anidadas

Las clases anidadas en Java son clases definidas dentro de otra clase. Hay cuatro tipos de clases anidadas en Java: clases internas estáticas, clases internas no estáticas, clases locales y clases anónimas.

Las clases internas estáticas son clases que se definen dentro de otra clase y que tienen el modificador static. Estas clases no dependen de una instancia de la clase externa y se pueden utilizar de manera independiente. Las clases internas no estáticas, por otro lado, dependen de una instancia de la clase externa y pueden acceder a sus métodos y variables de instancia.

Las clases locales son clases definidas dentro de un método, y su alcance está limitado a ese método. Las clases locales se utilizan principalmente para encapsular la lógica de un método y para evitar que se comparta con otros métodos o clases.

Las clases anónimas son clases sin nombre que se definen e instancian al mismo tiempo. Las clases anónimas se utilizan a menudo en Java para implementar interfaces de manera rápida y concisa.

Clase Abstracta

En programación orientada a objetos, una clase abstracta es una clase que no puede ser instanciada directamente, sino que solo sirve como base para que otras clases hereden sus atributos y métodos.

Una clase abstracta define métodos abstractos que deben ser implementados por las subclases. Los métodos abstractos son aquellos que solo definen la firma, pero no tienen una implementación concreta. Al implementar un método abstracto en una subclase, se proporciona la implementación específica de ese método para esa subclase.

En muchos lenguajes de programación orientada a objetos, se utiliza la palabra clave abstract para definir una clase abstracta o un método abstracto.

Interface

Una interfaz en Java es una colección de métodos abstractos y constantes estáticas, que se utiliza para definir un conjunto de comportamientos que las clases deben implementar. En otras palabras, una interfaz es un contrato que una clase debe cumplir para ser considerada compatible con una determinada API o conjunto de funcionalidades. Las interfaces en Java son completamente abstractas, lo que significa que no pueden contener implementaciones de métodos. En cambio, las clases que implementan una interfaz deben proporcionar una implementación para cada método definido en la interfaz. Las interfaces son ampliamente utilizadas en Java para definir especificaciones de API, y son una herramienta poderosa para lograr la abstracción y la modularidad en la programación orientada a objetos.