Presentado por: Brayan Esteven Marín Echeverri

Danna Isabella Rendón Sánchez

Consulta evaluable

1

Modificadores: Son palabras clave que se usan para modificar el acceso o el comportamiento de las clases, los métodos y los atributos. Por ejemplo, los modificadores de acceso son private, protected, public y default, y determinan quién puede acceder a los elementos de una clase. Otros modificadores son static, final, abstract, synchronized, etc. Por ejemplo, si declaramos un atributo como private int edad; significa que solo se puede acceder a él desde la misma clase donde se declara.

Encapsulación: Es el proceso de ocultar los detalles internos de un objeto y proporcionar solo una interfaz pública para interactuar con él. La encapsulación se logra usando los modificadores de acceso y los métodos get y set. La encapsulación permite proteger los datos de un objeto de manipulaciones externas y facilitar el mantenimiento del código. Por ejemplo, si tenemos una clase Persona con un atributo private String nombre; podemos definir un método public String getNombre() para obtener el valor del nombre y un método public void setNombre(String nombre) para asignarle un valor.

Paquetes/API: Un paquete es un conjunto de clases relacionadas que se agrupan bajo un mismo nombre. Los paquetes sirven para organizar el código y evitar conflictos de nombres entre clases. Una API (Application Programming Interface) es un conjunto de paquetes, clases, interfaces y métodos que proporcionan una funcionalidad específica para ser usada por otros programas. Java tiene muchas APIs integradas, como la API de colecciones, la API de entrada/salida, la API de gráficos, etc. Por ejemplo, si queremos usar la clase ArrayList que está en el paquete java.util, debemos importar el paquete con la sentencia import java.util.*; o importar solo la clase con la sentencia import java.util.ArrayList;.

Herencia: Es el mecanismo por el cual una clase puede heredar los atributos y los métodos de otra clase. La clase que hereda se llama subclase o clase derivada, y la clase que es heredada se llama superclase o clase base. La herencia permite reutilizar el código existente y establecer una relación jerárquica entre las clases. En Java, la herencia se indica con la palabra clave extends. Por ejemplo, si tenemos una clase Animal con un atributo private String especie; y un método public void comer(), podemos crear una subclase Perro que herede de Animal con la sentencia public class Perro extends Animal. Así, la clase Perro tendrá el atributo especie y el método comer, además de los que defina por sí misma.

Polimorfismo: Es la capacidad de que un objeto pueda tener diferentes formas o comportamientos según el contexto. El polimorfismo se manifiesta de dos formas: por sobrecarga o por sobreescritura. La sobrecarga ocurre cuando se definen varios métodos con el mismo nombre pero con diferentes parámetros en una misma clase. La sobreescritura ocurre cuando se redefine un método heredado en una subclase para darle una nueva

implementación. Por ejemplo, si tenemos una clase Figura con un método abstracto public double area(), podemos crear varias subclases que hereden de Figura y sobreescriban el método area() para calcular el área según la forma de cada figura. Así, si tenemos un objeto de tipo Figura que apunta a una instancia de tipo Circulo, al invocar el método area() se ejecutará el código definido en la clase Circulo.

Clases inner(anidadas): Son clases que se definen dentro de otras clases. Las clases anidadas pueden ser estáticas o no estáticas. Las clases anidadas estáticas se comportan como clases normales, pero solo se pueden acceder desde la clase externa o mediante su nombre completo. Las clases anidadas no estáticas se llaman clases internas y tienen una referencia implícita a la instancia de la clase externa que las contiene. Las clases internas pueden acceder a los atributos y métodos de la clase externa, incluso si son privados. Por ejemplo, si tenemos una clase externa Auto con un atributo privado private String modelo; y una clase interna no estática llamada Motor, podemos crear una instancia de la clase interna con la sintaxis: Auto auto = new Auto("Ford"); Auto.Motor motor = auto.new Motor();

Clases abstractas: Son clases que no se pueden instanciar porque tienen al menos un método abstracto, es decir, un método sin implementación que solo tiene la firma. Las clases abstractas sirven para definir una estructura común para las subclases que heredan de ellas, pero delegan la implementación de los métodos abstractos a las subclases. Una clase abstracta puede tener también métodos concretos (con implementación) y atributos. Por ejemplo, si tenemos una clase abstracta llamada Empleado con un método abstracto llamado public abstract double calcularSueldo();, ninguna subclase que herede de ella podrá instanciarse a menos que sobreescriba el método abstracto.

Interface: Es una colección de métodos abstractos (sin implementación) y constantes (atributos finales y estáticos) que definen un contrato o una especificación para las clases que la implementan. Una interfaz no se puede instanciar ni tiene constructor. Una clase puede implementar varias interfaces usando la palabra clave implements. Una interfaz puede extender otra interfaz usando la palabra clave extends. Por ejemplo, si tenemos una interfaz llamada Volador con un método abstracto llamado public abstract void volar();, cualquier clase que implemente esta interfaz deberá definir el comportamiento del método volar.

2

Paquetes de java para qr, barcode, bluetooth, whatsapp: Un paquete es un conjunto de clases relacionadas que se agrupan bajo un mismo nombre. Los paquetes sirven para organizar el código y evitar conflictos de nombres entre clases. Una API (Application Programming Interface) es un conjunto de paquetes, clases, interfaces y métodos que proporcionan una funcionalidad específica para ser usada por otros programas. Java tiene muchas APIs integradas, como la API de colecciones, la API de entrada/salida, la API de gráficos, etc. Para trabajar con códigos qr y barcode, puedes usar la API de Aspose.BarCode for Java123, que te permite generar y leer diferentes tipos de códigos de barras. Para trabajar con bluetooth, puedes usar la API de Java Bluetooth4, que te permite comunicarte con otros

dispositivos bluetooth. Para trabajar con whatsapp, puedes usar la API de WhatsApp Business, que te permite enviar y recibir mensajes a través de whatsapp.

Public, private, protected, final: Estas son palabras clave que se usan como modificadores de acceso o comportamiento en Java. Los modificadores de acceso determinan quién puede acceder a los elementos de una clase (atributos y métodos). Los modificadores de comportamiento determinan si los elementos de una clase se pueden cambiar o no. Los modificadores de acceso son:

public: El elemento es accesible desde cualquier clase.

private: El elemento solo es accesible desde la misma clase donde se declara.

protected: El elemento es accesible desde la misma clase, las subclases y las clases del mismo paquete.

default: El elemento es accesible desde las clases del mismo paquete. No se usa ninguna palabra clave para este modificador.

Los modificadores de comportamiento son:

final: El elemento no se puede modificar ni sobreescribir. Se usa para declarar constantes, métodos que no se pueden sobreescribir y clases que no se pueden heredar.

static: El elemento pertenece a la clase y no a las instancias. Se usa para declarar variables y métodos que se comparten entre todas las instancias de una clase.

abstract: El elemento no tiene una implementación definida. Se usa para declarar métodos que deben ser sobrescritos por las subclases y clases que no se pueden instanciar.

Cómo saber el tipo de dato de una variable en java: Una variable es un espacio en memoria que almacena un valor temporalmente. Cada variable tiene un tipo de dato que indica qué tipo de valor puede almacenar y qué operaciones se pueden realizar con él. En Java hay dos tipos de datos: primitivos y referenciados. Los tipos primitivos son los más básicos y se dividen en:

byte: Almacena un número entero de 8 bits con signo. Su rango es de -128 a 127.

short: Almacena un número entero de 16 bits con signo. Su rango es de -32768 a 32767.

int: Almacena un número entero de 32 bits con signo. Su rango es de -2147483648 a 2147483647.

long: Almacena un número entero de 64 bits con signo. Su rango es de -9223372036854775808 a 9223372036854775807.

float: Almacena un número decimal (real) de 32 bits con signo. Su precisión es de aproximadamente 6-7 dígitos decimales.

double: Almacena un número decimal (real) de 64 bits con signo. Su precisión es de aproximadamente 15 dígitos decimales.

char: Almacena un carácter Unicode de 16 bits. Su rango es de '\u0000' a '\uffff'.

boolean: Almacena un valor lógico (verdadero o falso).

Los tipos referenciados son los que se basan en clases o interfaces. Algunos ejemplos son:

String: Almacena una cadena de caracteres.

Object: Es la clase base de todas las clases en Java.

Integer: Es una clase envoltorio que almacena un valor primitivo int.

ArrayList: Es una clase que implementa la interfaz List y almacena una colección dinámica de objetos.

Para saber el tipo de dato de una variable en Java, podemos usar el operador instanceof, que devuelve verdadero si la variable es una instancia del tipo especificado o falso en caso contrario

Manejo de cadenas y regex en java: Una cadena es una secuencia ordenada e inmutable de caracteres. En Java, las cadenas se representan con la clase String, que ofrece varios métodos para manipularlas. Algunos ejemplos son:

length(): Devuelve la longitud (número de caracteres) de la cadena.

charAt(int index): Devuelve el carácter en la posición especificada por el índice.

substring(int beginIndex, int endIndex): Devuelve una subcadena desde el índice inicial hasta el final (excluido).

concat(String str): Devuelve una nueva cadena que resulta de concatenar la cadena original con la cadena especificada.

replace(char oldChar, char newChar): Devuelve una nueva cadena que resulta de reemplazar todas las ocurrencias del carácter antiguo por el nuevo en la cadena original.

toLowerCase(): Devuelve una nueva cadena que resulta de convertir todos los caracteres en minúscula en la cadena original.

toUpperCase(): Devuelve una nueva cadena que resulta de convertir todos los caracteres en mayúscula en la cadena original.

trim(): Devuelve una nueva cadena que resulta de eliminar los espacios en blanco al inicio y al final de la cadena original.

equals(Object obj): Devuelve verdadero si la cadena original es igual a la cadena especificada o falso en caso contrario.

compareTo(String str): Devuelve un número negativo si la cadena original es menor que la cadena especificada, cero si son iguales o un número positivo si es mayor.

Un regex (expresión regular) es una secuencia de caracteres que define un patrón para buscar o reemplazar cadenas o partes de cadenas. En Java, podemos usar las clases Pattern y Matcher para trabajar con regex. Un ejemplo es:

\d: Coincide con cualquier dígito decimal.