TALLER LENGUAJE DE PROGRAMACION I

PROFESOR:

- ***PLINIO NEIRA***

REALIZADO POR:

- ***MARIA CAMILA RODRIGUEZ.***
- ***MIGUEL ANGEL GALLEGO.***

1.PUNTO

- ***Modificadores:***

Java ofrece varios modificadores que se pueden utilizar para controlar el acceso y el comportamiento de clases, métodos y variables. Los modificadores se utilizan para definir el alcance y la visibilidad de una clase, método o variable, lo que puede ayudar a garantizar la seguridad y la integridad del código.

A continuación, se describen los principales modificadores en Java:

1. Modificadores de acceso:

- public: el miembro es accesible desde cualquier lugar.

- private: el miembro solo es accesible dentro de la misma clase.

- protected: el miembro es accesible dentro de la misma clase y en las subclases.

2. Modificadores de no acceso:

- static: el miembro pertenece a la clase en lugar de a una instancia de la clase.

- final: el miembro no se puede modificar después de su inicialización.

- abstract: el miembro no tiene una implementación completa y debe ser implementado por subclases.

- synchronized: el miembro solo puede ser accedido por un hilo a la vez.

- native: el miembro es implementado en código nativo fuera de la JVM.

3. Modificadores de sobrescritura:

- override: el método sobrescribe el método de la superclase.

- super: el método de la superclase se llama desde el método de la subclase.

Es importante tener en cuenta que los modificadores en Java se aplican a nivel de clase, método o variable, y pueden ser combinados en cualquier orden, siempre y cuando sean válidos en ese contexto.

- ***Encapsulamiento:***

La encapsulación es uno de los conceptos fundamentales de la programación orientada a objetos y se refiere a la ocultación de los detalles internos de una clase y la exposición únicamente de una interfaz pública para interactuar con ella. En Java, la encapsulación se logra mediante el uso de modificadores de acceso para los miembros de la clase, como los campos y los métodos.

En Java, hay tres modificadores de acceso que se pueden utilizar para definir el nivel de acceso de los miembros de una clase:

- public: un miembro público de una clase es accesible desde cualquier lugar, es decir, desde cualquier otra clase en cualquier paquete.

- protected: un miembro protegido de una clase es accesible desde cualquier clase en el mismo paquete o desde cualquier subclase, ya sea en el mismo paquete o en otro.

- private: un miembro privado de una clase solo es accesible desde la propia clase y no puede ser accedido desde ninguna otra clase.

Al definir los miembros de una clase como privados, se ocultan los detalles internos de la clase y se obliga a los usuarios de la clase a interactuar con ella a través de la interfaz pública que se expone mediante métodos públicos. Esto permite al programador controlar cómo se utiliza la clase y protegerla contra cambios no deseados o errores de programación.

- ***Paquetes/API:***

En Java, un paquete es un mecanismo para organizar clases relacionadas y hacerlas más fácilmente accesibles y reutilizables. Un paquete es una colección de clases relacionadas y subpaquetes que proporcionan una estructura jerárquica para las clases relacionadas.

Los paquetes en Java se identifican por su nombre de paquete, que es una cadena de caracteres separados por puntos. Por ejemplo, el paquete java.util contiene varias clases útiles relacionadas con la programación de utilidades.

La API de Java (Application Programming Interface) es un conjunto de clases, interfaces y paquetes predefinidos que se utilizan para desarrollar aplicaciones Java. La API de Java proporciona una amplia gama de clases y métodos que se pueden utilizar para realizar tareas comunes, como entrada/salida, redes, bases de datos, gráficos, seguridad, etc.

La API de Java está organizada en varios paquetes que se encuentran en la biblioteca de clases de Java (classpath). Algunos de los paquetes más comunes incluyen:

- java.lang: contiene clases fundamentales, como Object, String y las clases de excepción.
- java.util: contiene clases para la manipulación de colecciones, fechas, tiempos, etc.
- java.io: contiene clases para la entrada/salida de datos.
- java.net: contiene clases para la programación de redes.
- java.sql: contiene clases para la programación de bases de datos.

Para utilizar las clases y métodos de la API de Java en su código, debe importar los paquetes relevantes en su código fuente.

- ***Herencia:***

La herencia en Java es uno de los conceptos fundamentales de la programación orientada a objetos (OOP). La herencia permite que una clase herede las propiedades (atributos y métodos) de otra clase, lo que puede ayudar a simplificar el código y hacer que sea más fácil de entender.

En Java, una clase puede heredar de una clase existente utilizando la palabra clave extends. La clase que se hereda se llama la clase base o clase padre, y la clase que hereda se llama la clase derivada o clase hija.

- ***Polymorfismo:***

El polimorfismo es uno de los conceptos fundamentales de la programación orientada a objetos en Java. Se refiere a la capacidad de un objeto de tomar múltiples formas. En Java, esto se puede lograr mediante dos mecanismos: la herencia y las interfaces.

En la herencia, una clase puede heredar los métodos y propiedades de otra clase. Si una clase hija tiene un método con el mismo nombre y firma que un método en la clase padre, se dice que la clase hija está sobrescribiendo el método de la clase padre. Cuando un objeto de la clase hija se utiliza en lugar de un objeto de la clase padre, se puede llamar al método de la clase hija y se ejecutará en lugar del método de la clase padre.

Por otro lado, las interfaces son una colección de métodos abstractos que una clase puede implementar. Una clase que implementa una interfaz debe proporcionar una implementación para todos los métodos definidos en la interfaz. Si varias clases implementan la misma interfaz, se

pueden tratar todas estas clases como objetos de la misma interfaz y se pueden utilizar de manera intercambiable.

El polimorfismo en Java permite escribir código más flexible y reusable. En lugar de escribir código específico para cada clase, se puede escribir código que funcione con cualquier objeto que cumpla con una interfaz o una clase base específica.

- ***Clases inner(Anidadas):***

En Java, una clase interna (también conocida como clase anidada) es una clase que está definida dentro de otra clase. Hay cuatro tipos diferentes de clases internas en Java: clases internas regulares, clases internas anónimas, clases internas locales y clases internas estáticas.

Clases internas regulares: Estas son clases que se definen dentro de otra clase y tienen acceso a todos los miembros de la clase externa. Pueden ser públicas, protegidas, privadas o tener acceso de paquete.

Clases internas anónimas: Son clases internas que no tienen nombre y se crean de forma anónima. Se utilizan comúnmente para implementar interfaces o clases abstractas de forma rápida y sencilla.

Clases internas locales: Son clases que se definen dentro de un método y tienen acceso a los miembros de la clase externa y a los miembros del método. Solo se pueden acceder desde el método donde se definen y se utilizan para realizar tareas específicas.

Clases internas estáticas: Son clases internas que se definen como estáticas y solo pueden acceder a miembros estáticos de la clase externa. No pueden acceder a miembros no estáticos de la clase externa.

Las clases internas se utilizan comúnmente para encapsular la funcionalidad relacionada y mejorar la legibilidad y el modularidad del código. También permiten acceder a los miembros privados de la clase externa y proporcionan una forma elegante de implementar ciertos patrones de diseño, como el patrón de fábrica.

- ***Clase abstracta:***

n Java, una clase abstracta es una clase que no puede ser instanciada y que sirve como una plantilla para definir clases hijas. La clase abstracta puede tener métodos abstractos, que son métodos que no tienen una implementación y deben ser definidos en las clases hijas. También puede tener métodos concretos, que tienen una implementación y pueden ser heredados por las clases hijas.

Para definir una clase abstracta en Java, se utiliza la palabra clave "abstract" antes de la palabra clave "class" en la definición de la clase.

Las clases abstractas se utilizan comúnmente en Java para definir clases base que encapsulan la funcionalidad común y proporcionan una interfaz consistente para las clases hijas. También se utilizan para implementar polimorfismo y para evitar la duplicación de código en las clases hijas.

Interfaces:

Una interfaz es una especie de plantilla para la construcción de clases. Normalmente una interfaz se compone de un conjunto de declaraciones de cabeceras de métodos (sin implementar, de forma similar a un método abstracto) que especifican un protocolo de comportamiento para una o varias clases. Además, una clase puede implementar una o varias interfaces: en ese caso, la clase debe proporcionar la declaración y definición de todos los métodos de cada una de las interfaces o bien declararse clase abstract. Por otro lado, una interfaz puede emplearse también para declarar constantes que luego puedan ser utilizadas por otras clases.

Una interfaz puede parecer similar a una clase abstracta, pero existen una serie de diferencias entre una interfaz y una clase abstracta:

Todos los métodos de una interfaz se declaran implícitamente como abstractos y públicos.

Una clase abstracta no puede implementar los métodos declarados como abstractos, una interfaz no puede implementar ningún método (ya que todos son abstractos).

Una interfaz no declara variables de instancia.

Una clase puede implementar varias interfaces, pero sólo puede tener una clase ascendiente directa.

Una clase abstracta pertenece a una jerarquía de clases mientras que una interfaz no pertenece a una jerarquía de clases. En consecuencia, clases sin relación de herencia pueden implementar la misma interfaz.

2.PUNTO

- ***QR***

Existen varios paquetes de Java para generar y decodificar códigos QR, como ZXing, QrGen, QR Code Generator, QR Code Reader y Apache PDFBox. Cada uno proporciona una API para generar y decodificar códigos QR, y tienen sus propias características y limitaciones. Es importante investigar y elegir el que mejor se adapte a las necesidades específicas del proyecto.

- ***Bar***

Existen varios paquetes de Java para generar y decodificar códigos de barras, como ZXing, Barcode4J, Barbecue, JBarcode y Zint Barcode Generator. Cada uno proporciona una API para generar y decodificar códigos de barras, y soporta diferentes formatos y opciones de personalización. Es importante investigar y elegir el que mejor se adapte a las necesidades específicas del proyecto.

- ***Bluetooth***

Hay varios paquetes de Java para trabajar con Bluetooth, incluyendo Bluecove, Bluecove-gpl, TinyB, Avetana Bluetooth Framework y BlueNRG. Cada uno proporciona una API para descubrir dispositivos Bluetooth, conectarse a ellos y comunicarse con ellos. Es importante investigar y elegir el que mejor se adapte a las necesidades específicas del proyecto.

- ***WhatsApp***

No existe un paquete oficial de Java para trabajar con la API de WhatsApp. Sin embargo, existen algunos paquetes de terceros que proporcionan una API para interactuar con WhatsApp en Java, como WABlas, WhatsAPI y Yowsup. Estos paquetes permiten enviar y recibir mensajes de WhatsApp utilizando una API de Java, pero deben utilizarse con precaución, ya que el uso de la API de WhatsApp por parte de terceros no está permitido y podría violar los términos de servicio de WhatsApp. Además, es importante tener en cuenta que la API de WhatsApp puede cambiar sin previo aviso, lo que podría afectar la funcionalidad de estos paquetes de terceros.

3.PUNTO

En Java, los modificadores de acceso public, private y protected y la palabra clave final se utilizan para definir el nivel de acceso y la capacidad de modificaciones en los miembros de una clase, tales como variables, métodos y clases.

- public: El modificador public indica que un miembro de la clase (variable, método, clase) es accesible desde cualquier otra clase en el programa, independientemente de su paquete.

- private: El modificador private indica que un miembro de la clase solo es accesible dentro de la propia clase, y no puede ser accedido desde otra clase.

- protected: El modificador protected permite el acceso a un miembro de la clase desde la propia clase, las subclases y las clases del mismo paquete. Sin embargo, este miembro no es accesible desde clases que se encuentren en otro paquete.

- final: La palabra clave final se utiliza para hacer que un miembro de la clase sea constante e inmutable, lo que significa que su valor no puede ser modificado después de su inicialización. Se utiliza para definir constantes, y también para asegurar que los métodos o clases no sean modificados o sobrescritos.

4.PUNTO

En Java, se puede obtener el tipo de dato de una variable utilizando el operador "typeof" o el método "getClass()". El operador "typeof" se utiliza en tiempo de compilación y devuelve una cadena que representa el tipo de dato de la variable. El método "getClass()" se utiliza en tiempo de ejecución y devuelve un objeto de la clase Class que representa el tipo de dato de la variable. Es importante destacar que estos métodos sólo funcionan para variables de referencia y no para variables de tipo primitivo.

5.PUNTO

Java proporciona una amplia variedad de herramientas para el manejo de cadenas de caracteres y expresiones regulares. Se pueden utilizar métodos como length(), charAt(), substring() y indexOf() para manipular cadenas de caracteres y obtener información sobre ellas. Para el manejo de expresiones regulares, se utilizan las clases Pattern y Matcher, que permiten definir una expresión regular y buscar coincidencias dentro de una cadena. En resumen, el manejo de cadenas y expresiones regulares en Java es una tarea sencilla y eficiente gracias a las herramientas proporcionadas por el lenguaje.