**Taller realizado por:**

**Dylan Cardona Varilla**

**Juan Pablo Acevedo López**

**consulta evaluativa**

1.) **Significados en Java.**

**modificadores**

los modificadores son palabras clave que se utilizan para modificar el comportamiento de las clases, variables, métodos y otros elementos de un programa. Los modificadores se dividen en cuatro categorías: acceso, no acceso, estático y final. A continuación se describen brevemente cada uno de ellos:

Modificadores de acceso: estos modificadores controlan el nivel de acceso de los elementos de un programa. Los modificadores de acceso en Java son: public, private, protected y default.

**public:** los elementos con este modificador pueden ser accedidos desde cualquier parte del programa.

**private:** los elementos con este modificador solo pueden ser accedidos desde la misma clase.

**protected:** los elementos con este modificador solo pueden ser accedidos desde la misma clase o desde clases derivadas de ella.

**default:** también conocido como "sin modificador", los elementos con este modificador solo pueden ser accedidos desde la misma clase o desde clases del mismo paquete.

Modificadores no acceso: estos modificadores no controlan el nivel de acceso de los elementos, sino que modifican otras características. Los modificadores no acceso en Java son: abstract, final y static.

**abstract:** se utiliza para declarar clases abstractas o métodos abstractos. Una clase abstracta no se puede instanciar, mientras que un método abstracto no tiene implementación y debe ser definido en una subclase.

**final:** se utiliza para declarar constantes, métodos que no pueden ser sobreescritos o clases que no pueden ser subclases.

**static:** se utiliza para declarar elementos de clase que no dependen de una instancia de la misma, como variables o métodos estáticos.

Modificadores estáticos: estos modificadores se utilizan para declarar elementos que pertenecen a la clase en lugar de una instancia de la misma. Los modificadores estáticos en Java son: static y final.

**static:** se utiliza para declarar variables y métodos que pertenecen a la clase en lugar de a una instancia de la misma.

**final:** además de ser un modificador no acceso, también se utiliza para declarar variables que no pueden ser modificadas después de su inicialización.

Modificadores finales: estos modificadores se utilizan para declarar elementos que no pueden ser modificados después de su inicialización. Los modificadores finales en Java son: final y volatile.

final: como se mencionó anteriormente, se utiliza para declarar variables, métodos o clases que no pueden ser modificados después de su inicialización.

**volatile:** se utiliza para declarar variables que pueden ser modificadas por múltiples hilos de ejecución al mismo tiempo, asegurando que las lecturas y escrituras a esta variable sean atómicas.

**Encapsulacion**

la encapsulación es un mecanismo que se utiliza para proteger los datos de una clase y restringir su acceso desde fuera de la misma. Esto se logra mediante el uso de modificadores de acceso, como public, private y protected, para controlar la visibilidad de los campos y métodos de una clase.

La encapsulación permite ocultar la complejidad interna de una clase y exponer solo su interfaz pública, lo que facilita el mantenimiento y la evolución del código. Además, permite establecer restricciones sobre el acceso y modificación de los datos de una clase, lo que mejora la seguridad y la integridad de los mismos.

Para aplicar la encapsulación en Java, se suelen seguir las siguientes prácticas:

Declarar los campos de una clase como privados (private).

Proporcionar métodos de acceso (getters) y métodos de modificación (setters) para los campos privados, utilizando los modificadores de acceso adecuados (public, private, protected).

Utilizar validaciones en los métodos de modificación para asegurar que los valores asignados a los campos sean válidos.

No exponer los detalles de implementación de una clase en su interfaz pública.

**Paquetes / Api**

los paquetes son un mecanismo que se utiliza para organizar clases y otros elementos relacionados en un espacio de nombres común. Los paquetes en Java se representan mediante directorios en el sistema de archivos y se definen mediante la palabra clave package al comienzo de un archivo de código fuente.

**Herencia**

la herencia es un mecanismo que permite crear nuevas clases a partir de clases existentes, aprovechando y extendiendo sus características y comportamientos. La clase existente se llama clase base o superclase, mientras que la nueva clase creada se llama clase derivada o subclase.

**Polymorfismo**

el polimorfismo es un concepto que se refiere a la capacidad de los objetos de diferentes clases para responder de manera distinta a un mismo mensaje o método. En otras palabras, el polimorfismo permite que una variable de un tipo base pueda contener un objeto de una subclase, y que la invocación de un método sobre esa variable pueda tener un comportamiento diferente dependiendo del tipo real del objeto.

El polimorfismo se puede implementar en Java a través de dos mecanismos: el polimorfismo de subtipos y el polimorfismo de sobrecarga.

El polimorfismo de subtipos se basa en la herencia y se produce cuando una subclase reemplaza o redefine un método heredado de su superclase. En este caso, la subclase puede proporcionar una implementación diferente del método y, al mismo tiempo, seguir siendo compatible con el tipo de la superclase.

**Clases inner (Anidadas)**

una clase interna o clase anidada es una clase definida dentro de otra clase. Hay cuatro tipos de clases internas en Java:

Clase interna estática (Static nested class): es una clase definida dentro de otra clase, pero que tiene el modificador static. Esto significa que no depende del objeto de la clase externa para ser instanciado. Se accede a la clase interna estática utilizando el nombre de la clase externa: ClaseExterna.ClaseInterna.

Clase interna no estática (Non-static nested class o Inner class): también conocida como clase interna anidada, es una clase definida dentro de otra clase sin tener el modificador static. Se accede a la clase interna no estática utilizando una instancia de la clase externa: ClaseExterna.ClaseInterna instancia = claseExterna.new ClaseInterna().

Clase interna local (Local class): es una clase interna definida dentro de un bloque de código, como un método o una declaración condicional. La clase interna local solo se puede acceder dentro del bloque de código en el que se define.

Clase interna anónima (Anonymous class): es una clase interna que no tiene nombre y se define en una expresión en el momento de la ejecución. La clase interna anónima se usa a menudo como una implementación de una interfaz o una clase abstracta.

**Clases abstractas**

una clase abstracta es una clase que no se puede instanciar directamente, sino que se utiliza como una plantilla para definir subclases. Se define mediante la palabra clave abstract.

Las clases abstractas se utilizan para proporcionar una implementación básica de una clase que se espera que sea extendida por sus subclases. Una clase abstracta puede contener métodos abstractos, que son métodos que no tienen implementación y que deben ser implementados por las subclases. También puede contener métodos concretos, que son métodos con una implementación concreta.

**Interface**

una interfaz es una colección de métodos abstractos y constantes que se pueden utilizar como un tipo de datos en una clase. Se define mediante la palabra clave interface.

Una interfaz se utiliza para definir un conjunto de métodos que una clase debe implementar. Una clase puede implementar múltiples interfaces, lo que le permite adquirir comportamiento de varias fuentes. Las interfaces también se utilizan para proporcionar una especificación para las clases que implementan la interfaz.

2.)  **Paquetes java para : QR, Bar Code, Bluetooth, WHatsapp.**

**QR:** Para generar y leer códigos QR, puedes utilizar las siguientes bibliotecas:

**ZXing:** Es una biblioteca de código abierto para generar y leer códigos de barras y códigos QR. Proporciona una API fácil de usar y es compatible con varios formatos de códigos de barras y QR. Puedes encontrar más información en el sitio web: https://github.com/zxing/zxing

**QREncoder:** Es una biblioteca de código abierto para generar códigos QR en Java. Proporciona una API sencilla para generar códigos QR con varias opciones de configuración. Puedes encontrar más información en el sitio web: https://github.com/kenglxn/QREncoder

Códigos de barras: Para trabajar con códigos de barras, puedes utilizar las siguientes bibliotecas:

**ZXing:** Como se mencionó anteriormente, esta biblioteca también es compatible con la generación y lectura de códigos de barras. Puedes encontrar más información en el sitio web: https://github.com/zxing/zxing

**Barbecue:** Es una biblioteca de código abierto para generar códigos de barras en Java. Proporciona una API fácil de usar para generar códigos de barras en varios formatos, como UPC, EAN, Code 128, etc. Puedes encontrar más información en el sitio web: http://barbecue.sourceforge.net/

**Bluetooth:** Para trabajar con Bluetooth en Java, puedes utilizar las siguientes APIs:

**Bluecove:** Es una biblioteca de código abierto para trabajar con Bluetooth en Java. Proporciona una API fácil de usar para buscar, conectar y enviar datos a dispositivos Bluetooth. Puedes encontrar más información en el sitio web: https://bluecove.github.io/bluecove/

JSR-82: Es una especificación Java para trabajar con Bluetooth en dispositivos móviles. Proporciona una API para buscar, conectar y enviar datos a dispositivos Bluetooth. Puedes encontrar más información en el sitio web: https://jcp.org/en/jsr/detail?id=82

WhatsApp: No existe una API oficial para enviar mensajes a través de WhatsApp en Java, pero puedes utilizar las siguientes opciones:

Twilio API para WhatsApp: Es una API de terceros que permite enviar y recibir mensajes de WhatsApp utilizando Java. Proporciona una API fácil de usar y se integra con otras APIs de Twilio. Puedes encontrar más información en el sitio web: https://www.twilio.com/whatsapp/api

WhatsApp Web: Es posible enviar mensajes de WhatsApp a través de la versión web de WhatsApp utilizando Java y herramientas de automatización como Selenium. Esto no es una solución oficial y puede no estar permitido por WhatsApp.

3.) **Public, private, protect, final.**

**Public:** Los miembros de una clase marcados como public son accesibles desde cualquier parte del programa. Por ejemplo, si una variable es pública, se puede acceder a ella desde cualquier clase.

**Private:** Los miembros de una clase marcados como private son accesibles solo dentro de la misma clase. Por ejemplo, si una variable es privada, solo se puede acceder a ella dentro de la misma clase y no desde otra clase.

**Protected:** Los miembros de una clase marcados como protected son accesibles dentro de la misma clase y también desde las subclases que la heredan. Por ejemplo, si una variable es protegida, se puede acceder a ella desde la clase que la define y desde las subclases.

El modificador "final" se utiliza para indicar que un elemento no se puede modificar. Esto se puede aplicar a variables, métodos o clases enteras. Por ejemplo, si una variable es final, su valor no puede cambiar una vez que se ha inicializado. Si un método es final, no se puede sobrescribir en una subclase. Y si una clase es final, no se puede heredar de ella.

4.) **Como saber el tipo de dato de una variable en java.**

puedes determinar el tipo de datos de una variable en tiempo de compilación y en tiempo de ejecución. Aquí te presento algunas formas de hacerlo:

En tiempo de compilación: Si estás editando tu código en un IDE como Eclipse o IntelliJ, puedes simplemente colocar el cursor sobre la variable y verás su tipo de datos en el tooltip que aparece.

En tiempo de ejecución: Puedes usar el método getClass() de la clase Object para obtener el tipo de datos de una variable en tiempo de ejecución. Este método devuelve un objeto Class que representa el tipo de datos de la variable. Por ejemplo, si tienes una variable llamada "numero" de tipo int,La salida en la consola será "Tipo de dato de numero: class java.lang.Integer", lo que indica que el tipo de datos de la variable "numero" es Integer.

También puedes usar el operador instanceof para verificar si un objeto es de un tipo de datos específico. Por ejemplo, si tienes un objeto llamado "objeto" y quieres verificar si es del tipo String,Ten en cuenta que si intentas obtener el tipo de datos de una variable que no ha sido inicializada o que es nula, obtendrás una excepción NullPointerException en tiempo de ejecución.

5.) **Manejo de cadenas y RegEx Java.**

El manejo de cadenas (String) y las expresiones regulares (RegEx) son dos aspectos importantes en Java que te permiten trabajar con texto y patrones de texto.

El manejo de cadenas te permite crear, manipular y analizar cadenas de caracteres en Java. Java proporciona una clase llamada String que es utilizada para representar cadenas de caracteres. La clase String proporciona muchos métodos para manipular cadenas, como concatenar, comparar, dividir y buscar subcadenas, entre otros.

Las expresiones regulares, por otro lado, son patrones utilizados para buscar y manipular subcadenas dentro de una cadena más grande. Java proporciona una API de expresiones regulares que permite crear patrones de texto y buscar y manipular subcadenas que coinciden con esos patrones. Los patrones de expresiones regulares se definen mediante una sintaxis especial y pueden incluir caracteres comodín, grupos de caracteres, cuantificadores y mucho más.

La combinación del manejo de cadenas y las expresiones regulares te permite realizar operaciones avanzadas de procesamiento de texto en Java. Puedes buscar y reemplazar subcadenas específicas dentro de una cadena, validar que una cadena cumpla con un patrón específico, dividir una cadena en una matriz utilizando un delimitador específico, y mucho más.

En resumen, el manejo de cadenas y las expresiones regulares son herramientas esenciales en Java para trabajar con texto y patrones de texto, y te permiten realizar una variedad de operaciones avanzadas de procesamiento de texto.