**Consulta lenguaje programación 1**

**Nombres: Julian Quintero, Santiago**

**Modificadores en java**

En Java, un modificador es una palabra clave que se utiliza para modificar el comportamiento de una clase, un método o una variable. Los modificadores se pueden utilizar para restringir el acceso a los miembros de una clase, para modificar la herencia o para definir el ámbito de una variable.

Los modificadores en Java se dividen en dos categorías principales: modificadores de acceso y modificadores no de acceso.

Los modificadores de acceso se utilizan para controlar el acceso a los miembros de una clase y se dividen en cuatro tipos:

**Public:** los miembros marcados como "public" son accesibles desde cualquier parte del programa.

**Private:** los miembros marcados como "private" solo son accesibles desde dentro de la clase en la que se definen.

**Protected:** los miembros marcados como "protected" son accesibles desde dentro de la clase en la que se definen, así como desde las clases que heredan de ella.

**Default:** los miembros que no tienen ningún modificador se consideran "default" y solo son accesibles desde dentro del mismo paquete.

Los modificadores no de acceso se utilizan para modificar el comportamiento de una clase o de un miembro y se dividen en varios tipos, entre ellos:

**Static:** indica que un miembro pertenece a la clase en lugar de a una instancia de la misma.

**Final:** indica que una variable no puede ser modificada una vez que ha sido asignada un valor y que un método no puede ser sobrescrito.

**Abstract:** indica que una clase o un método no tienen una implementación concreta y que deben ser definidos en una subclase o en una implementación concreta.

**Synchronized:** se utiliza para garantizar la sincronización de los hilos de ejecución.

**Volatile:** se utiliza para indicar que una variable puede ser modificada por varios hilos de ejecución

**Encapsulamiento**

El encapsulamiento es un concepto fundamental de la programación orientada a objetos que se utiliza para proteger la integridad de los datos y ocultar los detalles de implementación de una clase. En Java, el encapsulamiento se logra mediante el uso de los modificadores de acceso y los métodos de acceso (getters y setters).

La idea detrás del encapsulamiento es que una clase debe tener control completo sobre sus datos internos, y que otros objetos solo deben ser capaces de acceder a ellos a través de métodos específicos. En otras palabras, los datos de una clase deben ser privados y solo accesibles a través de métodos públicos.

**Api**

API significa "Application Programming Interface" (Interfaz de Programación de Aplicaciones) y se refiere a un conjunto de clases, interfaces, métodos y constantes que proporcionan una funcionalidad específica y que pueden ser utilizados por otros programas para interactuar con un sistema o plataforma. En Java, las API son muy importantes ya que proporcionan una forma estándar y consistente de interactuar con el lenguaje de programación y sus librerías

**Herencia**

La herencia es un concepto fundamental de la programación orientada a objetos que permite crear nuevas clases a partir de clases existentes, reutilizando y extendiendo su funcionalidad. En Java, la herencia se logra mediante la declaración de una clase como "subclase" o "clase derivada" de otra clase existente, que se conoce como "superclase" o "clase base".

La subclase hereda los atributos y métodos de la superclase y puede añadir nuevos atributos y métodos propios.

**Polimorfismo**

En Java, el polimorfismo se logra a través de la herencia y las interfaces, permitiendo que un objeto pueda tomar diferentes formas o comportarse de diferentes maneras según el contexto en el que se utilice.

Existen dos tipos de polimorfismo en Java: el polimorfismo estático y el polimorfismo dinámico.

El polimorfismo estático se refiere al uso de la sobrecarga de métodos o la sobrecarga de operadores para proporcionar diferentes implementaciones de una misma funcionalidad en una clase. Esto permite que la misma operación pueda ser realizada con diferentes tipos de datos.

El polimorfismo dinámico, por otro lado, se refiere al uso de la herencia y las interfaces para permitir que los objetos de diferentes clases puedan ser tratados como si fueran del mismo tipo. En este caso, un objeto puede tomar diferentes formas según el contexto en el que se utilice, lo que permite una mayor flexibilidad y modularidad en el diseño de programas

**Clases anidadas**

Una clase anidada es una clase definida dentro de otra clase. Las clases anidadas se utilizan principalmente para encapsular una clase dentro de otra, lo que puede mejorar la legibilidad y modularidad del código. Existen dos tipos de clases anidadas en Java: las clases internas y las clases locales.

Las clases internas, también conocidas como "clases anidadas no estáticas", son clases que se definen dentro de otra clase y pueden acceder a los miembros privados de la clase contenedora. Las clases internas pueden ser declaradas como públicas, privadas, protegidas o sin modificador de acceso y se acceden a ellas a través de una instancia de la clase contenedora.

Las clases locales, por otro lado, son clases que se definen dentro de un método y sólo pueden ser accedidas dentro del método. Las clases locales son útiles para encapsular la lógica de un método y mejorar la modularidad del código.

**Clases abstractas**

Una clase abstracta es una clase que no puede ser instanciada directamente, sino que debe ser extendida por otra clase. Las clases abstractas se utilizan para definir una estructura general o abstracta para otras clases que extienden la clase abstracta.

Una clase abstracta puede contener métodos abstractos y métodos concretos (con cuerpo), así como campos y constructores. Los métodos abstractos son métodos que no tienen una implementación y que deben ser implementados por cualquier subclase que extienda la clase abstracta. Los métodos concretos, por otro lado, tienen una implementación definida y pueden ser heredados y utilizados directamente por las subclases.

**Interfaz**

Una interfaz es una colección de métodos abstractos y constantes que pueden ser implementados por una clase. Una interfaz proporciona una especificación para los métodos que una clase debe implementar, pero no proporciona una implementación concreta de los métodos.

**Paquetes QR**

QRcoder

QRGen

Zxing

**Paquetes Bluetooth**

javax.bluetooth

java.awt.Robot

javax.obex

javax.microedition.io.Connector

Bluecove

**Paquetes WhatsApp**

Javax

Junit

Gson

Android

Okhttp

**Paquetes bar code**

Zxing

Barbecue

Java barcode generador

Aspose.BarCode

**Como saber el tipo de dato de una variable en java**

Para conocer el tipo de una variable lo puedes realizar mediante el método . getClass(). getSimpleName() (también puedes usar . getName() ) y realizando un casting de (Object) para la variable de la cual deseas conocer su tipo.

**Cadenas en java**

El paquete java.lang contiene dos clases de cadenas: String y StringBuffer. La clase String se utiliza cuando se trabaja con cadenas que no pueden cambiar. Por otro lado, StringBuffer, se utiliza cuando se quiere manipular el contenido de una cadena. El entorno de desarrollo Java proporciona dos clases para manipular y almacenar datos del tipo carácter: String, para cadenas constantes, y StringBuffer, para cadenas que pueden cambiar.