Resumen LABORATORIO 4

Una variable pasa por un constructor cuando se instancia

los constructores tienen que ser públicos y tener el nombre de la clase.

para mostrar el contenido de un constructor:

system.out.println("ID:"+art.getId()+" )

¿¿Para qué sirve el método ToString??

para mostrar una variable?

Resumen video 2:

para que una variable se crea con la clase hay que ponerle STATIC

esa variable no va a pertenecer al objeto sino a la clase por lo tanto ya no hace falta instanciar. Para que un cambio afecte a todos los objetos.

Resumen video 3:

Metodos ESTATICOS para que un metodo sea estatico y pertenesca a la clase y no al objeto se le debe poner STATIC ej:



de este modo el metodo ya no pertence a los objetos sino a la clase. Evitando tener que crear un objeto para poder llamar a este metodo . Y para llamarlo lo hacemos de la siguente forma:



tambien lo podria mostrar mediante un array o vector



Resumen Video 4:

Para heredar métodos se usa Extends.

SUPER llama al constructor y a los métodos.

ToString para llamar al método padre.

Resumen Video 5:

"Final" para dejar una variable en tipo constante

con "final" nadie la puede heredar, corta la herencia.

una clase cuando tiene FINAL ya no permite la herencia en sus clases hijas.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Nuevo resumen 15/02/23

* Una variable de tipo constante solo puede recibir un valor.
* Cuando una variable usa FINAL y recibe un valor ya no puede modificar su valor.
* Cuando una clase usa FINAL ya no puede ser heredada por otra.
* Si al declararla no le asigno un valor, me va a obligar a hacerlo en los constructores.
* Si queremos que la variable constante sea de la clase y no del objeto anteponemos la palabra reservada "Static".



Resumen Video 6:

Concepto de encapsulamiento

El método se va adaptando y cambia de forma toString (Polimorfismo) según la clase que sea.

Puedo hacer guardar un objeto de la clase hija en la clase padre.

Resumen Video 7:

Concepto de Abstracción.

una CLASE ABSTRACTA

Deja crear Objetos, pero no crear Instancias

Método Abstracto. Como forzar a que una clase hija desarrolle un metodo, haciendolo un metodo abstracto (posiciono el cursor en la clase hija y selecciono ADD\_UNIMPLEMENTED\_METHODS) asi desarrollo los metodos implementados.

Resumen Video 8:

Interfaces

se pueden implementar varias interfaces pero sola una herencia

siempre una interfaz tiene metodos abstractos, por mas que no lo diga en la declaracion

los atributos seran Publicos, estaticos y constantes

se pueden implementar varias interfaces

La interfaz no se puede hacer un NEW

Simpre en las clases hijas se pueden guardar en la clase padre.

Resumen Video 9:

Dentro de una Interfaz se puede guardar un objeto que implemente esa interfaz

Los objetos dejan de tener dependencia con las clases por trabajar con interfacez.

Nos facilita el no tener que modificar la clase solo inyectar desde afuera las variables.

Resumen Video 10:

Para comparar dos objetos ir a la clase del objeto y generar los equals en la clase, lo castea

Equal compara objetos No direcciones de memoria.

compara por contenido

Resumen de video 10 bis:

List set y queue son metodos COLLECTION

Sirve para guardar muchas instancias juntas

No se puede crear un objeto de una interfaz

Puedo crear objetos de las CLASES de esa interfaz

En List tenemos Colleccion ARRAYlIST y LINKEDLIST

podes trabajar con datos repetidos repetidos

trabaja con sub indice

ArrayList se guardan uno al lado del otro (contigua)

si no puede ubicar de manera contigua busca un lugar de espacio mas amplio en memoria libre moviendo el vector en un nuevo lugar donde pueda entra el nuevo objeto de manera contigua.

Lento para agregar y rapido para leer.

Si elimino una posicion va a mover toda la lista hasta dejarla contigua (lento).

LinkedList

Al agregar un nuevo elemento lo ubica en el primer lugar que encuentre pero conserva la referecia de quien es el que le sigue ( Lista enlazada)

Para leer lento, agregar y eliminar rapido.

Una lista Doblemente enlazada mantiene la referencia del que le sigue y el anterior.

Resumen de video 11:

HashCode es un numero que se puede obtener de cada objeto.

2 hashcode son iguales en base a lo que tiene adentro el objeto.

ejemplo de codigo. Ejemplo de IF:

Public int hashcode( ){

int result = 1;

result += ((apellido == null) ? 0 : apellido.hashCode( ));

result += ((dni==null) ? 0 : dni.hashCode ( ) );

result += ((nombre ==null) ? 0 : nombre.hasCode( ));

return result;

}

se puede obtener el valor hashCode solo de las variables de tipo string???

Un objeto solo va a tener un solo un codigo identificatorio. Se pueden evaluar todas las variables o algunas.

Resumen Video 12:

Set

una collection de tipo set no acepta datos repetidos y sin indices.

tambien es una interfaz.

si quiero crear una coleccion va ser de tipo HashSet, LinkedHashSet, TreeSet o EnumSet. Solo puedo crear instancia de estos tipos.

HashSet : Rapido, sin orden, no se puede acceder aleatoriamente y no hay datos duplicados.

Al instanciar un objeto se llama al metodo internamente (hashCode). El Hascode devuelve un numero que es evaluado e ingresa en algun rango de la coleccion Hashset. Si encuentra dos objetos iguales no los agrega(compara con un EQUAL).

La clase tiene que tener definido el metodo hashCode y el metodo Equals.

LinkedHashSet Orden segun entrada de datos, no es eficiente para agregar un elemento.

TreeSet: Orden obligatorio en los elementos mediante la Interfaz Comparable. No Permite datos duplicados.

EnumSet: Trabaja con tipo de datos enumerados.

Video 13:

Iterator

Para recorrer una coleccion se puede utilizar un for ech, y tambien se pueden utilizar un ListIterator y un Iterator.

El ListIterator solamente funciona con las colecciones de tipo List, mientras que el Iterator funciona para las demas colecciones como por ejemplo las colecciones de tipo Set.

El ListIterator tiene mas metodos: te permite agregar mientras recorre, tambien remover, ver cual es el indice.

En resumen te permite modificar la lista a medida que la vas recorriendo.

El ITERATOR sirve para lista de tipo set, tipo map o treeSet. Aunque tambien te deja remover mientra va leyendo los datos.

Video 14:

Errores

Existen 2, en tiempo de compilacion y en tiempo de ejecucion.

ee

ejemplo error de sintaxis COMPILACION

String nombre;

nombre = 88888885;

ejemplo error en tiempo de ejecucion

int num1 = 9;

int num2 = 0;

int res = num1/num2;

Las excepciones son errores en tiempo de ejecucion.

EXCEPTION.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Video 15:

Tipos de excepciones

Los objetos de tipo ERROR se producen cuando hay errores de hardware, disco duro, etc, LOS PROGRAMADORES NO PUEDEN MANEJARLOS.

EXCEPTION son errores que se producen en tiempos de ejecucion.

IOException son todas las excepciones que se producen en la entrada y salida de datos. Cuando quiero acceder a una base de datos, ver y ultilizar imagenes, accedecer a un archivo, grabar o leer un archivo, etc.

RuntimeException: ERRORES de tipo aritmeticos o por objetos nulos.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Video 16:

IOEXCEPTION

siempre van a estar dentro de un bloque TRY CATCH.

Si no pasa por uno de los dos si o si va a pasar por FINALLY(pasa por ahi ocurra o no el error, siempre se va a ejecutar).

Ademas agregando en el CATCH la linea

e.printstackTrace();

nos va a dar una informacion mas exacta de donde esta el error y a que se debe.

Gracias al "throws IOException" java obliga a crear el bloque try catch-g

Video 17:

RunTimeException

java.lang.Object

java.lang.Throwable

java.lang.Exception

java.lang.RuntimeException

java.lang.IndexOutOfBoundsException

java.lang.ArrayIndexOutOfBoundsException

Voy a tener que hacer un TRY CATCH para poder acceder a estas clases.

Se lanza para indicar que se ha accedido a una matriz con un índice ilegal. El índice es negativo o mayor o igual que el tamaño de la matriz.

No hace falta poner

ArrayIndexOutOfBoundsException todo completo ya que poniendo solo el nombre EXCEPTION (que es la clase padre) se podria acceder de todas formas, aunque siempre es mejor ser lo mas especifico posible.

casi siempre se soluciona agregandole lo que indica el error del cartel de consola en el CATH(ERROR){}

puede haber varios CATCH

" ArithmeticException e " : sirve para validar si hay un error matematico como por ejemplo el usuario dividio por cero.

" NumberFormatException e " : sirve para validar si en vez de agregarse un caracter numerico se agrego una palabra o letra.

IMPORTANTE! : podes usar varios CATCH en un try catch.

ATENCION! : al poner un catch Exception vas a impedir de que pase por los otros catch. Ya que solo va a entrar por esa clase que es la clase padre de las antes mencionadas.

poniendose al final seria lo mas optimo para que valide algo que no llegamos a catchear.

" e.printStackTrace(); " deja dividir x cero pero tambien te dice cual es el error.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Video 18 -

Creando Exceptions

para crear una EXCEPTION tiene que heredar de Exception.

Tambien deberias sobreescribir el metodo "getMessage()" el cual devuelve el mensaje del padre pero se puede devolver el mensaje que nosotros querramos.

Puede heredar o de EXCEPTION o de todas sus variantes por ejemplo de RUNTIMEEXCEPTION.

Para crear una exception en una funcion de otra clase podemos hacer ese llamado mediante una instancia de esa clase y a esa instancia la ejecutamos ante-poniendo la palabra THROW.

ejemplo:

throw new NombreClaseException();

o sino mas largo:

NombreClaseException exc1 =new NombreClaseException();

Y cuando un metodo contiene un THROW en su interior, en una excepcion tiene que declararlo en la parte de afuera del metodo tanto si tiene una como si tiene varias.

ejemplo:

public static boolean validarMail( String mail) throw NombreClaseException, SiHayMasClases{

throw

}

Con RunTimeException Java no me obliga a poner el TRY CATCH.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Video 19 -

Archivos I

Hay 2 formas de guardar infomacion

una como flujo de bytes(1 y 0) y para manipular los archivos hay que utilizar InputStream y OutputStream.

Tambien puedo guardar informacion como flujo de caracteres (texto, palabras, espacios, salto de linea, etc). Para manipular estos ultimos se utilizan las clases

FileReader / BufferedReader

File Writer/ BufferedWriter

el flujo de bytes se usa para leer archivo en un sistema de bajo nivel, caso contrario para sistemas de alto nivel.

Importando la clase FILE puedo crear una

instancia de tipo FILE.

En el constructor recibe la ruta.

Las rutas pueden ser ABSOLUTA(completa) O RELATIVA(ruta dentro del proyecto, en que carpeta esta solamente).

Las clases FILE usan la clase FileWrite para crear un archivo, esta se encuentra en el paquete Java.io.FileWriter;

Y esta instancia recibe dos parametros(RUTA,TRUE o FALSE).

El parametro TRUE significa que va a continuar escribiendo debajo de lo que ya estaba escrito, mientras que si este parametro fuera FALSE SOBRESCRIBE BORRA lo que estaba escrito copiando lo nuevo.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Video 20 -

Archivos II (FileReader - Filewriter)

Leen y escriben letra por letra los archivos

Las rutas relativas buscan el archivo dentro del proyecto.

ejemplo de metodo crear archivo:

public boolean crearArchivo()

{

Filewrite escritura;

try{

escritura = new FileWriter(ruta,true);

escritura.write("");

escritura.close();

return true;

}catch (IOException e) {

e.printStackTrace();

}

return false;

}

cuando llame a fileWriter hay que enviar la ruta y un true o un false.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Video 21 -

Archivos III (BufferedReader - BufferedWriter)

se crearon para acelerar el proceso de lectura y escritura.

Los Buffers permiten obtener informacion de a linea y no de forma secuencial.

Acelera tanto escritura como la lectura. Crea una memoria intermedia entre el archivo y eclipse.

BufferWriter permite escribir tanto letra como caracteres.

Con Buffer.Write se pueden escribir tanto char, int, string.

BufferRider

\n

Video 22 -

Threads

un me permite ejecutar una tarea en simultaneo, varias cosas al mismo tiempo.

En una clase declaro una variable texto.

genero un constructor recibiendo el texto como variable mas los getter y setter.

Para generar tres mensajes distintos

enviamos un mensaje por constructor o via set.

instancio un thread:

Thread hilo1 = new Thread();

Thread hilo2 = new Thread();

Thread hilo3 = new Thread();

como parametro me esta pidiendo algo que tenga Runnable.

Runnable es una interfaz que me permite trabajar con hilo, similar al trabajar con un set comparable.

al hilo le tengo que mandar los mensaje que se estan ejecutando.

eso tira error sino implementa la Interfaz Runnable

public class Clase implements Runnable{

Thread hilo1 = new Thread(msj1);

Thread hilo2 = new Thread(msj2);

Thread hilo3 = new Thread(msj3);

}

al implementar Runnabl la clase va a tener que desarrollar el metodo RUN que es lo que hilo va a estar ejecutando.