**RESUMEN UNIDADES 2.3. Y 2.4.**

**2.3. GESTION DE EXCEPCIONES**

En la mayoría de programas en algún momento hay que interaccionar con los soportes de datos, de archivos de la red o de un dispositivo. El funcionamiento de flujos de datos, no depende del dispositivo hardware.

Los tipos de datos pueden ser: carácter o binarios.

10.1. Excepciones

Los errores en los programas son prácticamente inevitables, la manipulación de errores suele ser confusa, lo que nos lleva a un código el cual no está claro.

Esto obliga a adoptar un enfoque distinto, para evitar dichos errores, implementando medios para ello, un error es Throwable, esto lo que quiere decir un mensaje diciendo que tipo de error tenemos y donde está.

Hay errores graves que pueden hacer que el programa quede inservible. Ha y también otro errores menos grabes los cuales son excepciones o una subclase.

Cuando sabemos de un determinado código con un error hay que examinarlo, para saber de dónde procede y cuál es la solución a dicho error.

Por ejemplo: una división entre 0, o un String con valor null.

Esto se puede evitar utilizando el siguiente Codigo:

Try {

C=a/b;

} Catch (Exception e) {

System.out.println (“Error: división por cero”);

}

10.1.1. Requisitos de captura o especificación

Al principio distinguimos entre una Exception y error normal, pero Exception que son predecibles y fácil de evitar si estamos atentos, manipulando el código como es debido, por ejemplo con: try-catch.

Este grupo de ex cesiones son:

Comprobadas: son las más comunes, que generalmente tiene un origen fuera del programa; Las más importantes son: IOException, NumberFormatException, etc.

No Comprobadas: son errores aritméticos, que se producen cuando nos salimos de los límites de una tabla.

10.1.2. Excepciones de usuario

Hasta ahora todas las excepciones binen predefinidas por java. Pero podemos encontrar otras las cuales están causadas por el usuario, esto lo arreglamos con un condicional o lanzar una Exception nueva con throw:

Public class ExceptionEdadNegativa extends Exception {

Public String toString () {

Return “Edad negativa”;

}

}

10.2. Flujos de entrada

Los flujos de entrada de tipo texto heredan de la clase InputSteamReader.Siempre tienen un nombre que termina en Reader. El constructor es: FileReader. Al que le pasaremos un parámetro el cual lo que hará llevarnos a un archivo para deshacernos de esa excepción, podemos utilizar una barra invertida la cual lo que hace es convertir un erro en lo que nosotros queramos.

**2.4. ARRAYS**

Existe una forma de almacenar más de un valor simultáneamente en una variable, la respuesta es si mediante tablas.

5.1. Variables escalares versus tablas

Una tabla es una variable que permite más de un valor simultaneó.

5.2. Índices

El problema es cómo distinguir entre un elemento o un componente que constituye la tabla. La solución es asignar un orden por cada el elemento. La forma correcta es de acceder a ese valor es utilizando el numero junto a los [].

5.2.1. Índices fuera de rango

Los errores de fuera de rango es por culpad e falta de atención dentro del código o porque nos hemos pasado del límite del valor.

5.3. Construcción de tablas

En el momento de crear una tabla hay que tener en cuenta lo siguiente: definir el tipo, las variables, y crear la tuya propia.

5.3.1. Longitud y tipo

Una tabla se define mediante dos características fundamentales: su longitud y su tipo.

La longitud es el número de elementos que posee, y el tipo de una tabla define los element

5.3.2. Variables de tabla

El primer paso para declarar una tabla es [], y su tipo.

5.3.3. Operador new

Una ver hemos declarado una variable, creamos una tabla con la longitud requerida, y utilizando el operador new.

5.4. Referencias

En este punto hemos creado una tabla. El siguiente paso es asignar las variables correspondientes. Para que cada variable sepa su lugar en la tabla se le asigna una regencia a él. Las variables pueden verse como medios para acceder a las tablas a las que hace referencia. Para hecho esta tabla debe estar bien referenciada.

5.4.1. Recolector de basura

Lo primero y más obvio es que una tabla no referenciada es inútil, porque no hay forma de acceder a sus elementos, y además está ocupando espacio necesario. Esto se soluciona mediante el siguiente mecanismo: el recolector de basura, que compraba todas las tablas creadas.

5.4.2. Referencia null

Hemos visto la forma de asignar a una variable la referencia de una tabla, pero podemos hacer lo contrario utilizando la variable null, la cual significa vacío.

5.5. Uso de tablas

Existen distintas formas de utilizar las tablas, una vez ya creadas mantienen su longitud constante y no se puede cambiar. Para poder modificarlo debemos crea una nueva dándole la longitud que tenemos.

5.5.1. Tablas ordenadas

En ocasiones, interesa que los datos estén ordenados según un criterio dado.

5.5.2. Tablas + indicador

Otra forma de usar las tablas es hacer una subdivisión, una para elementos y otra parte vacía, así lo que hacemos es poder darle más valores hasta el final de la longitud dada.

5.6. Tablas como parámetros de funciones

Cuando utilizamos tablas como parámetros de un mecanismo mismo para una función. Esto lo que conlleva es copiar una tabla con distintas referencias, para poder llamarla.

5.7. Operaciones con tablas: la clase Arrays

La API de java proporciona herramientas que facilitan el trabajo del programador.

Arrays nos da la seguridad de un Codigo eficiente, sin errores y la comodidad que supone ahorrar el tiempo de escribir nuestra implementación, siempre que sea posible.

5.7.1. Obtención del número de elementos de una tabla

Java proporciona un mecanismo para conocer la longitud de una tabla: length

5.7.2. Inicialización

Si deseamos inicializar con un valor distinto que por el defecto, tendremos que asignar un valor determinado: Arrays.fill

5.7.3. Recorrido

Muchas operaciones con tablas implican recorrerlas. Para poder realizar esta acción debemos utilizar un bucle con un constructor que recorra dila tabla. Para ello es necesario declarar una variable del mismo tipo del cual es la tabla.

5.7.4. Mostrar una tabla

Para mostrar los elementos utilizaríamos el syso y el nombre de la tabla, dando los valores que queremos mostrar junto a una longitud.

5.7.5. Ordenación

Ordenar una tabla consiste en cambiar de posición los datos que contiene para que en su conjunto resulte ordenado. La clase Arrays permite realizar esto con el siguiente método: Arrays.sort

5.7.6. Búsqueda

Consiste en averiguar si entre los elementos de una tabla se encuentra, y en que posición se encuentra un valor determinado, llamado clase búsqueda. Todo esto depende de si la tabla esta ordenada o no.

No Ordenada

Es una búsqueda secuencial y consiste en ir recorriendo la tabla hasta encontrar el elemento requerido, y finalizar cuando lo hemos encontrado.

Ordenada

Aprovechamos la posición de los valores para darnos información extra: edad[n].

5.7.7. Copia

El procedimiento para realizar una copia de una tabla es crear una tabla con la misma longitud y tipo, y recorrer la tabla para que copie los valores en la tabla de destino. También podemos atizar el método: arrayscopy ()

5.7.8. Inserción

Para añadir un nuevo valor a una tabla depende de si esta ordenado o no.

No Ordenada

Aquí lo que ocurre es que se crea una nueva tabla para poder meter el nuevo valor, dando la sensación de que se ha modificado.

Ordenada

La inserción ordenada consiste en añadir un nuevo valor en una posición adecuada y que la tabla siga ordenada. Primeramente buscaríamos el lugar donde corresponde dicho valor y lo introduciríamos donde corresponde como un elemento extra.

5.7.9. Eliminación

Esta operación consiste en borrar un elemento determinado, esto depende de que si esta ordenada o no.

No Ordenada

Para poder eliminar en una tabla desordenada es necesario buscar el elemento a borrar, sustituirlo por el último dato de la tabla, Crear una copia con la modificación y la vieja borrarla.

Ordenada

El primer paso es buscar el elemento para borrarlo. Al estar ordenada sabemos solo hay que indicar el elemento a borrar con su posición y volver a crear la tabla pero sin ese elemento, desplazando los demás.

5.7.10. Comparación de dos tablas

El contenido de dos tablas no se puede realizar con el operador ==, sino con sus referencias, para considerar dos tablas iguales deben de estar con el mismo orden.: Arrays.equals ().

5.8. Tablas n-dimensionales

Hasta el momento las tablas que hemos utilizado han sido unidimensionales. Pero puede ocurrir que nuestros datos se refieran a entidades que se encuentra en más de una propiedad.

5.8.1. Tablas bidimensionales

Podemos ampliar el concepto de tabla haciendo que los elementos se extiendan en dos dimensiones utilizando los eje x, y.Arrays.deepToString ().

5.8.2. Tablas bidimensionales

Al igual que antes pero añadiendo una nueva dimensión mas x, y, z.

5.8.3. Tablas con más dimensiones

Imaginar una con más de tres dimensiones es posible pero algo complicado, para ello lo que hay que hacer es dividir la tabla en otras más simples.