



1.- HOJA DE PRESENTACION:

**MATERIA: ESTRUCTURA DE DATOS
INGIENERIA INFORMATICA**

T1EDEQUIPO1 (UNIDAD 1)

EQUIPO #1: Castro Ramón David

Alejandro 20010329

Martínez Ramos Rodrigo 20010347

GRUPO:

Fecha de entrega: 22-ABRIL-2023

INDICE DEL REPORTE

1) UNA PÁGINA DE PRESENTACIÓN (O PORTADA)	Pág. 1
2) INTRODUCCIÓN (RESUMEN):	Pág. 1
3) COMPETENCIA ESPECIFICA:	Pág. 1
4) MARCO TEÓRICO:	Pág. 1
5) MATERIAL Y EQUIPO:	Pág. 1
6) DESARROLLO DE LA PRACTICA:	Pág. 1
7) RESULTADOS:	Pág. 1
8) CONCLUSIONES:	Pág. 1
9) BIBLIOGRAFÍA:	Pág. 1

2) INTRODUCCIÓN (RESUMEN):

En este reporte de la unidad 1 se explicarán todos los códigos/programas que en este caso abarcan los temas de Datos Simples o arreglos y también vimos ejemplos fáciles como lo son los números Armstrong, una Suma de Divisores y una Validación entre Amigos.

Este reporte tiene el propósito de que pongamos y documentemos todo lo que se hizo dentro de las diversas unidades en la materia de estructura de datos (en este caso es la Unidad 1) en el cual se mostrara paso a paso como fue que fuimos evolucionando en temas además de nuestro avance con los códigos que se presentaron por parte de la maestra.

Además, nos beneficiara en la parte de que repasaremos todo lo visto en las unidades anteriores y recalcar nuestros conocimientos prácticos para resolver problemas por nuestra propia cuenta basándonos en apuntes y temas/documentos previos para saber si los resultados son los esperados al finalizar no solo cada unidad sino la materia de estructura de datos en el presente semestre

Se explicará a manera detallada cada una de las partes o apartados que se nos presenta en este documento y ver nuestros avances en esta materia el cual se presentaran apartados donde se incluya:

- 1) Una página de presentación (o portada)
- 2) Introducción (Resumen):
- 3) Competencia específica:
- 4) Marco teórico:
- 5) Material y Equipo:
- 6) Desarrollo de la practica:
- 7) Resultados:
- 8) Conclusiones:
- 9) Bibliografía:

3) COMPETENCIA ESPECIFICA:

INTRODUCCIÓN A LAS ESTRUCTURAS DE DATOS

- 1.1 Clasificación de las estructuras de datos
- 1.2 Tipos de datos abstractos (TDA)

1.3 Ejemplos de TDA's

1.4 Manejo de memoria

1.4.1 Memoria estática-Operaciones básica

✓ Programa-Proyecto Datos desordenados

✓ Programa-Proyecto Datos desordenados

4) MARCO TEÓRICO:

ARREGLOS:

Es una Colección ordenada de elementos de un mismo tipo, que se almacenan en memoria de manera contigua, que se referencian utilizando un nombre común. Ordenada significa que cada elemento tiene una ubicación determinada dentro del arreglo (hay un primer elemento, un segundo elemento, un tercer elemento, y así sucesivamente) y debemos conocerla para accederlo.

Para diferenciar y acceder a un elemento en particular de un arreglo se usa un índice.

Estos índices van después del nombre del arreglo y encerrados por [] (corchetes).

El elemento 5° (quinto) de un arreglo, es representado por el índice [4], ya que los índices comienzan en 0, esto significa que un arreglo de 10 elementos tendría los índices del 0 al 9: [0...9]

Una forma de visualizar un arreglo es como un grupo de cajas, una detrás de otra. Arreglo
←índices

Donde cada caja representa un dato del arreglo o un elemento.

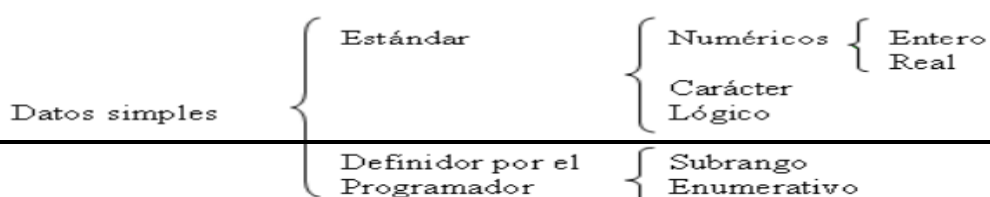
Un arreglo puede tener una o varias dimensiones, las dimensiones indican como están organizados los elementos dentro del grupo.

Los arreglos se clasifican de acuerdo a las dimensiones que tengan, así hay arreglos de una dimensión, de dos dimensiones, de tres dimensiones, etc.

Podemos concluir que un arreglo tiene:

- Tamaño: cuantas cajas va a tener, el número de datos o elementos.
- Tipo: cual es el tipo de todos los elementos (enteros, reales, etc.)
- Nombre: es el único nombre bajo el cual vamos a dirigirnos al arreglo.

Tipos de datos



ESTRUCTURA DE DATOS ESTATICAS:

Las estructuras de datos estáticas son aquellas en las que el tamaño ocupado de la memoria se define antes de que el programa se ejecute lo cual permite que n se pueda modificar durante la ejecución del programa.

Estas estructuras están implementadas en casi todos los lenguajes de programación e cual se representan como:

- Arreglos (Vector/Matriz)
- Registros
- Ficheros
- Archivos Adjuntos

ESTRUCTURA DE DATOS DINAMICAS:

Las estructuras de datos dinámicas no tienen limitaciones o restricciones dentro del arreglo en el tamaño de memoria ocupada que son propias de las memorias estáticas.

Mediante el uso de tipo de un tipo de dato específico denominado puntero es posible construir estructuras de datos dinámicas, el cual son soportados en la mayoría de lenguajes.

Estas estructuras se representan principalmente mediante:

- Listas (Enlazadas, Pilas, Colas)
- Arboles (binarios, árbol-b)
- Grafos

5) MATERIAL Y EQUIPO:

✓ **Computadora:** Cualquier dispositivo electrónico (Computadora o Laptop) que tenga buen funcionamiento en sistema y que pueda ser compatible con un lenguaje de programación para que se pueda instalar, además de una buena memoria de almacenamiento.

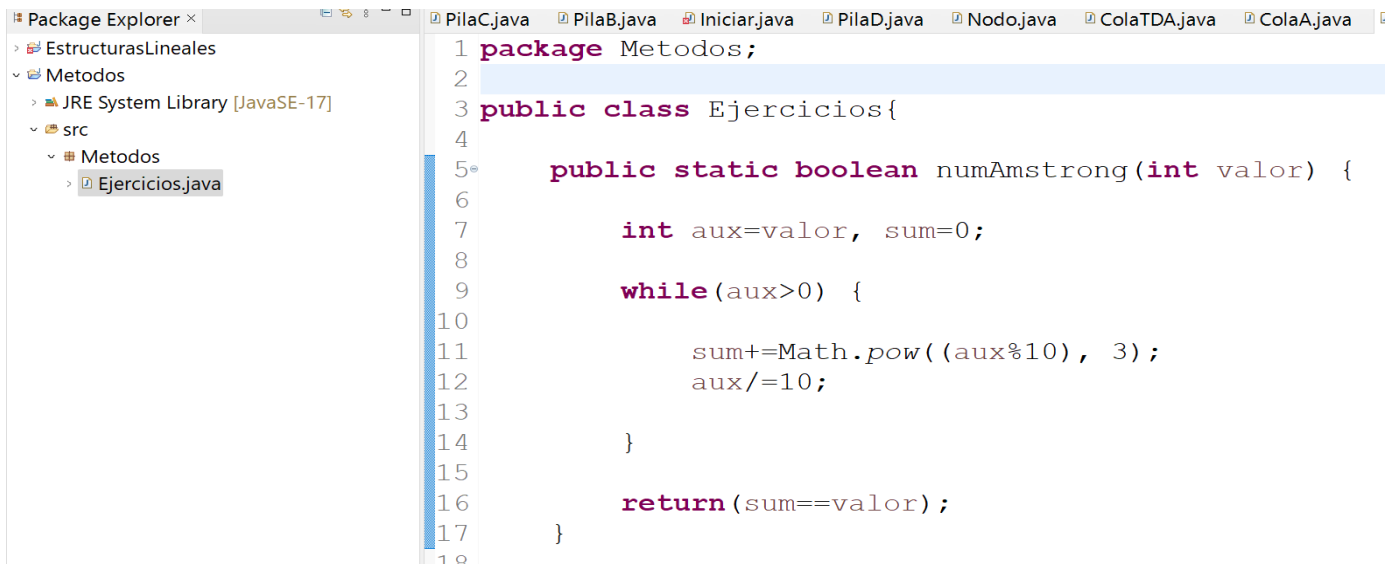
✓ **Software y versión usados:** En la creación de los programas se utilizó el programa/aplicación de Eclipse que se instala por medio de su JDK Y el IDE que se puede encontrar en la plataforma del lenguaje de programación

✓ **Materiales de apoyo para el desarrollo de la practica:** Documentos de apoyo para la parte teórica presentados por la maestra el cual contenía los códigos y ejemplos de los programas que vemos y pasarlos con nuestros conocimientos al lenguaje de programación que elegimos en este caso Eclipse Versión 2023

6) DESARROLLO DE LA PRACTICA:

EJERCICIOS DE PRACTICA

NUMERO AMSTRONG



```
1 package Metodos;
2
3 public class Ejercicios{
4
5     public static boolean numAmstrong(int valor) {
6
7         int aux=valor, sum=0;
8
9         while (aux>0) {
10
11             sum+=Math.pow((aux%10), 3);
12             aux/=10;
13
14         }
15
16         return (sum==valor);
17     }
18 }
```

1. El paquete llevará por nombre **Métodos** y a la clase se le pondrá Ejercicios
2. En la línea #5 se pone un valor estático booleano que se llamara numAmstrong y también se crea una variable de tipo entero llamado valor
3. En la línea #7 se agrega una variable llamada aux que va a ser igual a lo que tenga la variable valor que se creó en la línea #5.

Igual se creó la variable sum que va a ser igual a 0 (se inicializa en 0).
4. En la línea 9 se dice que mientras Aux sea mayor que 0 se mete dentro del ciclo while (mientras se cumpla la función se meterá y realizara lo siguiente)

5. En la línea #11 se hace la cuenta para sum el cual si sum el cual se sumara y tomara en cuenta a función `Math. pow ()` que retorna la base elevada al exponente, es decir que pide que el auxiliar (Aux) haga división en modulo con el 10 y en este caso se multiplicara hasta el exponente 3.
6. Después se tomará si el auxiliar o (aux) se dividirá entre 10.
7. Después se retorna solo todo lo que sum tenga igual al Valor en este caso todos los resultados que dieron desde la base 0 hasta la base 3 y se suman e imprime el resultado final.

Suma De Divisores

```
22
23 public static int SumDivisores(int valor) {
24
25     int sum=0;
26     for(int i=0; i<=valor; i++) {
27
28         if(valor%i==0) {
29
30             sum+=i;
31
32         }
33
34     }
35
36     return sum;
37 }
20
```

llamado SumDivisores y también una variable llamada valor igual de tipo entero.

2. En la línea #25 se crea otra variable de tipo entero o int que se llamará sum y valdrá o su valor será igual que 0.
3. En la línea #26 es la que llevara a cabo el ciclo que en este caso será for que empezara o inicializara con la `i=0`; después el conteo se acabara hasta que la

i o el contador sea menor o igual que lo que valor tenga administrado o lo que valga y el ultimo o i++ es que el contador ósea i se va a ir incrementando de 1 en 1 cada vez que se repita o vuelva a pasar el ciclo hasta su fin.

4. En la línea #28 se entre módulo de lo realiza un sí o un if que dice o especifica si valor modulo (%) de lo que i valga o tenga en ese momento es igual(==) que 0 entonces Sum se le sumara 1.
5. Cuando termine el contador de realizar su trabajo a lo último en la línea #36 se retornara (return) todos los sum o en este caso sum contendrá todos los 1 que salieron y sumaron y se imprimirá el resultado de las sumas de todos los 1 que dieron como resultado de la operación en modulo que sea igual (=) a 0 y los demás números que el resultado no sea igual que 0 específicamente no se contarán dentro de sum .

VALIDAR AMIGOS

```
38
39 public static boolean ValidarAmigos(int dato1, int dato2) {
40     return(dato1==SumDivisores(dato2) && dato2==SumDivisores(dato1));
41 }
42 }
```

Aunque este código sea corto de todas formas explicare a continuación de que se trata y cómo funciona:

1. Se creo un método estático de tipo booleano que se llame Validar Amigos en la línea #39 y se crearan las variables dato1 y dato2 las 2 de tipo entero.
2. En la siguiente línea que es la línea #40 se retornara en donde el dato1 sea igual (==) al sum Divisores hecha anteriormente que contendrá en este caso el (dato2)

y al mismo tiempo se tiene que cumplir la condición de que igual (&&) el dato2 sea igual (==) SumDivisores de lo que contenga almacenado el dato1 y después de esto llamara a traer lo que se tenga almacenado en esas 2 variables.

DATOS SIMPLES

Datos Desordenados

```
1 package DatosDesordenados;
2
3 import ToolPanel.Tools;
4
5 public class DatoSimple {
6
7     private Object datos[];
8     private byte p;
9
10    public DatoSimple(byte tam)
11    {
12        datos=new Object[tam];
13        p=-1;
14    }
15
16    public boolean validaVacio()
17    {
18        return (p== -1);
19    }
20 }
```

1. Se crea el Paquete llamado DatosDesordenados en la línea 1 y en la línea 3 se importa la clase Tools dentro del paquete ToolPanel, en la línea 5 se crea la clase DatoSimple.
2. En la línea 7 se crea un Object (Objeto) de tipo privado llamado datos.
3. En la línea 8 se creade tipo privado, de tipo byte la p.
4. En la línea 10 se crea de tipo public DatoSimple y de tipo byte la variable tam que será el tamaño del arreglo.
5. En la línea 12 se pone datos y se crea un nuevo objeto que tendrá el tamaño o medida del arreglo.

6. En la línea 13 p se le asignara el valor de -1.
7. En la línea 16 se agrega un método de tipo booleano que se llamara ValidaVacio el cual validara si es que el arreglo tiene datos o no.
8. En la línea 18 se retornará a la p si es que es exactamente ==-1 ósea sino ha nada dentro del arreglo.

```
PilaC.java  PilaB.java  Ejercicios.java  *DatoSimple... x  *Tools.java  *DatoSimple...  *Tools.java
20
21 public void almacenarDato()
22 {
23     if(p<datos.length)
24     {
25         datos[p+1]=Tools.leerInt("Escribe un nombre de persona");
26         p++;
27     }
28     else Tools.imprimeMsg("Array lleno");
29 }
30
31
32 public void imprimeDatos()
33 {
34     String cad="";
35     for (int i = 0; i <=p; i++) {
36         cad+=i+"["+datos[i]+"]"+"\\n";
37     }
38     ToolsPanel.imprimePantalla("datos del array\\n"+cad);
39 }
40
```

9. Se Agrega otro método en la línea 21 el cual lleva por nombre almacenarDato el cual guardara la información que el usuario agregara dentro del arreglo
10. Después en la línea 23 este método hará su función por medio de una condición if el cual comparara que si la p es menor que el tamaño del arreglo el cual se guarda en datos. lenght.
11. En la línea 25 se les anexara a datos [p+1] el cual se le ira agregando en espacio al arreglo hasta que llegue al tamaño elegido y por medio del

Tools.LeerInt se le ira poniendo el nombre de la persona a guardar dentro de la matriz

12. En la línea 26 p se ira incrementando de 1 en 1 (p++)
13. De lo contrario debido a un else en la línea 28 aparecerá un mensaje que imprima o que diga ("Array Lleno").
14. Después se creará otro método void de tipo público se llame imprimeDatos en la línea 32.
15. En la línea 34 se pondrá una cadena de tipo Sting el cual es donde podrás poner a los nombres de las personas.
16. En la línea 35 se agregará un for o contador en ciclo que contendrá la variable de tipo entero-int el cual empezara en 0 (se inicializa el contador), y se detendrá cuando el contador ósea i sea menor o igual que p, el contador ira incrementando de 1 en 1 con cada pasada en el ciclo hasta que se cumpla la condición en la línea 36.
17. En la línea 37 se concatena todos los nombres que se ingresaron dentro del arreglo.
18. En la línea 39 se manda el mensaje de que se impriman todos los nombres que se han concatenado en el paso anterior y te muestra el arreglo con los nombres.

```

42= public byte buscarSecuencial(Object dat) {
43
44     byte i=0;
45     while(i<=p && !dat.equals(datos[i]))
46         i++;
47
48     return (dat.equals(datos[i]))?i:-2;
49 }
50
51= public void eliminaDatos(byte pos) {
52     byte j=0;
53     for(j=pos; j<=p; j++) {
54         datos[j]=datos[j+1];
55     }
56     p--;
57 }
58
59= public void editaDato(byte dat) {
60     datos[dat]=Tools.leerString("Ingresa el nuevo nombre");
61 }
62
63 }
64

```

19. Se crea otro Método de tipo byte llamado buscar Secuencial y un objeto de tipo Object con el nombre de dat en la línea 42.

20. En la línea 44 se inicializa la i=0.

21. En la línea 45 se crea un ciclo while (mientras) se cumpla la función de i sea menor o igual y (&&) dat.equals (mientras los caracteres sean los mismos a los ingresados en el arreglo) se irá incrementando de 1 en 1.

22. En la línea 48 se retornará en este caso el nombre que buscamos mientras sea os mismos caracteres en el arreglo y en la búsqueda sino mandará un mensaje que diga ("Nombre no encontrado").

23. En la línea 51 se hará otro método de tipo void llamado eliminaDatos y una variable de tipo byte llamada pos.

24. En la línea 52 se inicia j de tipo byte igual a 0.

25. En la línea 53 se pone un for que dice que j va a ser igual a pos y parara hasta que j sea menor o igual que p, y este se incrementara de 1 en 1 ósea j++.
26. En la línea 54 se igualará si los datos de j son iguales y se le agregara 1 posición a j.
27. En la línea 56 es donde se eliminará el nombre que se quiere quitar del arreglo en este caso p--.
28. Se crea el ultimo método que se encargara de editar el nombre dentro del arreglo y el cual funciona que se busca una posición el cual es el nombre que modificaremos se buscara y se encontrara si es que existe o coincide con la búsqueda, también te podrá dar la oportunidad de ingresar un nuevo nombre al arreglo.

7) RESULTADOS:

Resultado de Numero amstrong

Inserta un entero

i

Escribe un numero

125

OK

Cancel

No es un numero amstrong

Resultado de valida amigos

Inserta un entero

i

Escribe el primer numero

123

OK

Cancel

Inserta un entero

i

Escribe el segundo numero

345

OK

Cancel

No son amigos

Resultado de Agregar desordenado

MENU

?

SELECCIONA DANDO CLICK

Agregar

Imprimir

Buscar

Eliminar

Salir

Lectura String

?

Escribe un nombre de persona

1

OK

Cancel

Lectura String

?

Escribe un nombre de persona

3

OK

Cancel

Lectura String

?

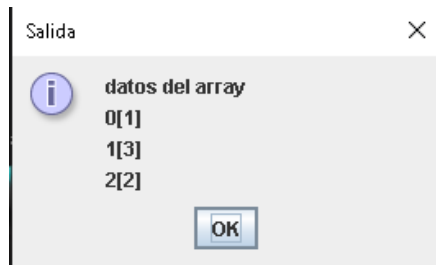
Escribe un nombre de persona

2

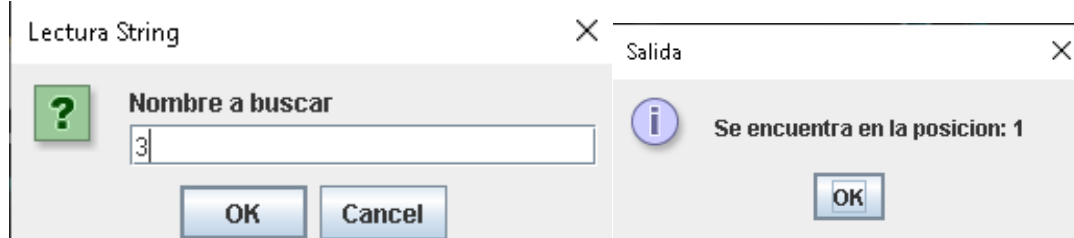
OK

Cancel

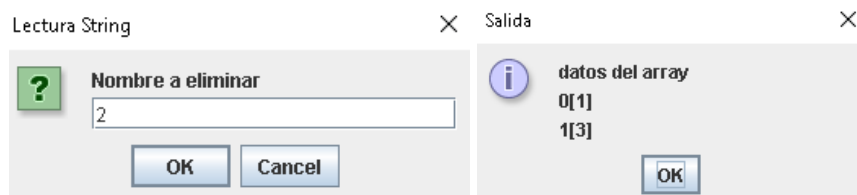
Mostrar desordenado



Buscar desordenado



Eliminar desordenado



8) CONCLUSIONES:

Los resultados esperados fueron los resultados obtenidos debido a que seguimos paso por paso las instrucciones prestadas y dadas por la docente a lo largo de la unidad 1.

Además, todo nos salió gracias a los trabajos de los ejercicios y actividades realizadas tanto como en el salón de clases como en el laboratorio de centro de computo en donde pudimos aclarar y aprender temas básicos a la introducción de la estructura de datos haciendo ejercicios básicos como:

Clasificación de las estructuras de datos:

- Tipos de datos abstractos (TDA)
- Ejemplos de TDA's (En este apartado hicimos varios programas que tienen que ver con los TDA'S O Tipos De Datos Abstractos en total hicimos 3 programas con esta característica representativa)

Memoria estática-Operaciones básica:

- Programa-Proyecto Datos desordenados (Este fue un proyecto para realizar un programa que mediante arreglos se puedan insertar, eliminar, modificar, validarVacio, ver todos los datos que insertamos mediante el usuario en los arreglos y que se muestren y se vayan agregando al arraylist al final se imprimirá el arreglo con los datos de manera desordenada).

9) Bibliografía:

<http://webdelprofesor.ula.ve/ingenieria/eliana/prog/unidad5.pdf>

EcuRed. (n.d.). *Arreglos (Informática)* -

EcuRed. [https://www.ecured.cu/Arreglos_\(Inform%C3%A1tica\)](https://www.ecured.cu/Arreglos_(Inform%C3%A1tica))