UNIVERSIDAD DE EL SALVADOR

FACULTAD MULTIDISCIPLINARIA DE OCCIDENTE
DEPARTAMENTO DE INGENIERÍA Y ARQUITECTURA.

GUIA 2

"CONCEPTOS PARTE 1, DESARROLLO DE GUIA 2"

ASIGNATURA: PROGRAMACION II

Catedrático:

Ing. William Virgilio Zamora Girón

Instructor:

Br. Sigfrido Ernesto Gómez Guinea.

Alumnos:

Br. Marcello Ernesto Solórzano Flores.

Br. Jonathan Otoniel González Aguirre.

Br. Antonieta Carolina Aragón Cruz.

Grupo: #2

Ciclo I -2018

Guía número dos, programación dos ingeniería.

Instrucciones: Resuelva los cuestionamientos presentados a continuación de manera concisa, lógica y ordenada.

Objetivo: Realizar un repaso de la programación orientada a objetos (POO) profundizando en el manejo de vectores(arreglos) y operaciones básicas con ellos así como el manejo de variables, dando un enfoque a la resolución de problemas cotidianos de ingeniería, guiando al alumno a hacer uso de la programación para resolverlo.

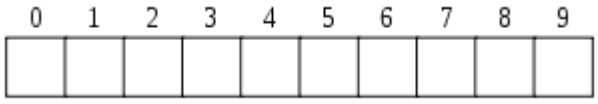
Parte I:

I. ¿Qué es un vector dentro de JAVA?

Según Wikipedia un vector es una zona de almacenamiento continuo, que contiene una serie de elementos del mismo tipo.

Los vectores se utilizan para almacenar información de una manera más ordenada y de esta forma será más fácil y rápido de acceder a la información cuando la necesitemos.

Un vector es una estructura estática, que se maneja por índices, cada uno de estos índices es un espacio donde se puede almacenar un valor. Si visualizamos un vector en una imagen sería así:



Como se aprecia en la imagen, los índices del vector empiezan en cero, y cada cuadro contendría adentro un valor específico, en caso que se le hubiera dado uno, de lo contrario aparecerá por defecto como null.

La forma de crear un vector en java es sumamente sencilla, es suficiente con agregar los paréntesis cuadrados, "[ ]", al final del tipo de nuestra variable, como ejemplo vamos a crear un vector de Strings, uno de enteros y otro de Object, se procede de la siguiente manera:

```
1  String[] vectorDeStrings;
2  int[] vectorDeEnteros;
3  Object[] vectorDeObjetos;
```

Como ven la inicialización de nuestro vector no esta completa, porque necesitamos crear el nuevo vector y darle un tamaño, este tamaño será el número de elementos que el vector pueda contener.

Hay varias formas de darle el tamaño a un vector la más corriente es ponerla al final de la inicialización de la variable dentro de los paréntesis cuadrados, así:

```
String[] vectorDeStrings = new String[10];
int[] vectorDeEnteros = new int[256];
```

Como ven esta es la inicialización completa de un vector, ahora un repaso rápido:

- Se selecciona el tipo de variables que va a contener el vector.
- Se le da un nombre para identificar nuestra variable.
- Creamos la nueva variable y le asignamos un tamaño.

Ya nuestro vector está listo!, ahora tenemos que agregar valores a ese vector, esta tarea no es complicada en lo absoluto, en realidad es muy sencillo como veremos con un pequeño ejemplo, con un ciclo for llenaremos un vector de enteros de tamaño 10.

```
int[] numeros = new int[10];

for(int indice = 0; indice < numeros.length; indice++){

    numeros[indice] = indice + 1;

}
```

Con este sencillo for estamos agregando valores del cero al nueve a nuestro vector de números, explicaré más detalladamente este trozo de código:

- Inicializamos nuestro vector como se explicó anteriormente.
- Creamos una variable llamada índice, la función de este índice es llevar el control de los campos de nuestro vector, para que agregue cada número en una posición diferente y siguiente de la anterior para que no queden espacios nulos en medio de nuestro vector.
- Condición **indice<numeros.length**, esto es para que a la hora de agregar los números al vector no se sobrepase del tamaño del mismo, porque si no nuestro programa se "caeria", la función del .length es medir el tamaño del vector como empieza a contar desde uno el tamaño seria diez pero la posición diez en nuestro vector no existe (porque va del cero al nueve) por eso se pone el "menor que" para que no se desborde debido a que no buscará una posición inexistente en el vector.
- Se agrega un número al vector en la posición que tenga el valor de índice, como índice empieza en cero, sería en la posición cero del vector agregue el valor de índice más uno y así sucesivamente.

Hay que tener cuidado en que índice vamos a guardar la información porque si guardamos un valor en un índice que ya tiene algo contenido, el valor anterior sera reemplazado por el entrante.

Obtener valores de un vector, si vemos gráficamente nuestro vector anterior quedó de la siguiente manera:

[1] [2] [3] [4] [5] [6] [7] [8] [9] [10]

Para obtener el valor de determinada posición se escribe:
- El nombre de nuestro vector.
- Paréntesis cuadrados "[]"
- Dentro de ellos la posición de la cual queremos obtener el valor.

2. ¿Qué es un array dentro de JAVA?

Un array es un medio de guardar un conjunto de objetos del mismo tipo.

Se accede a cada elemento individual del array mediante un número entero denominado índice.

0 es el índice del primer elemento y *n-1* es el índice del último elemento, siendo *n*, la dimensión del array.

Para crear un array se hace así:

```
int[] numeros;
```

Con esa línea se crea la variable tipo Array.

Eso si, ahora hay que darle un índice, para saber de cuántos elementos se compone el Array.

```
numeros = new int[5];
```

Así "rellenamos" al Array de 5 elementos. Es decir, es como si tuvieramos 5 variables "int" todas juntas.

Veamos un ejemplo donde tenemos un Array de 5 elementos y le damos un valor a cada elemento, y los mostramos por pantalla...

```java
package aulafacil;

public class AulaFacil {

    public static void main(String[] args) {

        int[] numeros; //Un array de números enteros

        numeros = new int[5]; //Array de 5 elementos (del 0 al 4)

        //Le damos valor a cada elemento del Array
        numeros[0] = 23;
        numeros[1] = 5;
        numeros[2] = 300;
        numeros[3] = 12;
        numeros[4] = 79;

        //Se muestran por consola los valores de todos los elementos
         System.out.println(numeros[0]);
         System.out.println(numeros[1]);
         System.out.println(numeros[2]);
         System.out.println(numeros[3]);
         System.out.println(numeros[4]);
    }
}
```

AulaFacil.com

Así se le asignan valores a los elementos:

numeros[0] = 23; => Lo que hay entre [] es el índice. No puede ser mayor que 4, en este caso.

Si el Array fuera de 10 elementos, no podría ser mayor que 9 su índice.

Luego se muestran por consola con el System.out.println, como siempre.

Pero ¿qué problema hay en todo eso? Pues que imagina que tenemos un Array de 200 elementos...¿No sería un poco engorroso mostrar los elementos? Costaría mucho trabajo además.

Pues bien, la solución es utilizar un bucle For, para los Arrays son ideales.

Veamos como mostrar todos los elementos del Array de forma fácil.

```java
package aulafacil;

public class AulaFacil {

    public static void main(String[] args) {

        int[] numeros; //Un array de números enteros
        int i;

        numeros = new int[5]; //Array de 5 elementos (del 0 al 4)

        //Le damos valor a cada elemento del Array
        numeros[0] = 23;
        numeros[1] = 5;
        numeros[2] = 300;
        numeros[3] = 12;
        numeros[4] = 79;

        //Se muestran por consola los valores de todos los elementos
        for (i = 0; i < numeros.length; i++)
        {
            System.out.println(numeros[i]);
        }
    }
}
```

AulaFacil.com

Fijaos en el bucle For. La condición es: Si **"i"** es menor que numeros.length (números.length es un entero que nos dice el tamaño de ese Array, en este caso 5). O sea si "i" es menor que 5, se ejecuta el For.

Y se muestra numeros[i]...Ya que "i" va a empezar en 0, y luego va a ir subiendo hasta 4, por lo tanto, el bucle recorre todo el Array.

De esta forma, sea el Array de 10, de 200, o de 56999 elementos, el For de este tipo, siempre recorrerá el Array completo.

3. ¿Cómo se diferencia un vector de un array?

Un vector es similar a un array, la diferencia estriba en que un vector crece automáticamente cuando alcanza la dimensión inicial máxima. Además, proporciona métodos adicionales para añadir, eliminar elementos, e insertar elementos entre otros dos existentes.

Los arrays unidimensionales se conocen con el nombre de **vectores**. Los arrays bidimensionales se conocen con el nombre de **matrices**.

4. Liste los diferentes tipos de vectores y arrays que pueden utilizarse en JAVA (int vector [];)

Se pueden crear arrays:
- ✓ De enteros de tipo int
- ✓ De reales de tipo float

Vector:
- ✓ Sentencia import.
- ✓ String
- ✓ Object

5. ¿Qué utilidad concreta tiene cada una de estas estructuras de datos dentro de la programación?

Un array es una estructura de datos que contiene una colección de datos del mismo tipo.

Ejemplos: Temperaturas mínimas de los últimos treinta días. Valor de las acciones de una empresa durante la última semana.

Propiedades de los arrays.

Los arrays se utilizan como **contenedores** para almacenar datos relacionados (En vez de declarar variables por separarado para cada uno de los elementos del array).

Todos los **datos** incluidos en el array son del mismo tipo. Se pueden crear arrays de enteros de tipo int o de reales de tipo float, pero en un mismo array no se pueden mezclar datos tipo int y datos de tipo float.

Para usar la clase *Vector* hemos de poner al principo del archivo del código fuente la siguiente sentencia **import**

```
import java.util.*;
```

Cuando creamos un *vector* u objeto de la clase *Vector*, podemos especificar su dimensión inicial, y cuanto crecerá si rebasamos dicha dimensión.

```
Vector vector=new Vector(20, 5);
```

Tenemos un vector con una dimensión inicial de 20 elementos. Si rebasamos dicha dimensión y guardamos 21 elementos la dimensión del vector crece a 25.

Al segundo constructor, solamente se le pasa la dimensión inicial.

```
Vector vector=new Vector(20);
```

Si se rebasa la dimensión inicial guardando 21 elementos, la dimensión del vector se duplica. El programador ha de tener cuidado con este constructor, ya que si se pretende guardar un número grande de elementos se tiene que especificar el incremento de la capacidad del vector, si no se quiere desperdiciar inútilmente la memoria el ordenador.

Con el tercer constructor, se crea un vector cuya dimensión inicial es 10.

```
Vector vector=new Vector();
```

La dimensión del vector se duplica si se rebasa la dimensión inicial, por ejemplo, cuando se pretende guardar once elementos.

6. ¿Qué es el tipo de datos object dentro de JAVA?

La clase **Object**, como ya se ha indicado anteriormente, es la clase raíz de todo el árbol de la jerarquía de clases Java, y proporciona un cierto número de métodos de utilidad general que pueden utilizar todos los objetos. La lista completa se puede ver en la documentación del API de Java, aquí solamente se tratarán algunos de ellos; por ejemplo, **Object** proporciona:

- Un método por el que un objeto se puede comparar con otro objeto
- Un método para convertir un objeto a una cadena
- Un método para esperar a que ocurra una determinada condición
- Un método para notificar a otros objetos que una condición ha cambiado
- Un método para devolver la clase de un objeto

La clase Object trae una serie de métodos que en algunos casos son complicados de usar y que en muchas ocasiones el usarlos es causa de un mal diseño. Entre los más importantes y los que posiblemente más usaremos son:

- El método *equals()* se utiliza para comparar si dos objetos son iguales. Este método devuelve true si los objetos son iguales, o false si no lo son. Entendemos iguales a aquellos objetos que guardan el mismo estado interno (caracterizado por el valor de sus atributos) no uno concreto.

- **equals()** se utiliza para comparar si dos objetos son iguales. Este método devuelve **true** si los objetos son iguales, o **false** si no lo son. Observe que la igualdad no significa que los objetos sean el mismo objeto. Consideremos este código que compara dos enteros.

```
Integer uno = new Integer(1), otroUno = new Integer(1);

if (uno.equals(otroUno))
    System.out.println("Los objetos son Iguales");
```

- Este código mostrará **Los objetos son Iguales** aunque **uno** y **otroUno** referencian a dos objetos distintos. Se les considera iguales porque su contenido es el mismo valor entero.
- Las clases deberían sobreescribir este método proporcionando la comprobación de igualdad apropiada. Un método **equals()** debería comparar el contenido de los objetos para ver si son funcionalmente iguales y devolver **true** si es así.

- El método *getClass()* devuelve una representación en tiempo de ejecución de la clase del objeto. Este método devuelve un objeto Class con el que se puede obtener más información sobre el tipo del objeto.

- El método **getClass()** es un método final (no puede sobreescribirse) que devuelve una representación en tiempo de ejecución de la clase del objeto. Este método devuelve un objeto Class al que se le puede pedir varia información sobre la clase, como su nombre, el nombre de su superclase y los nombres de los interfaces que implementa. El siguiente método obtiene y muestra el nombre de la clase de un objeto.

```
void PrintClassName(Object obj) {
    System.out.println("La clase del Objeto es " +
          obj.getClass().getName());
}
```

- Un uso muy manejado del método **getClass()** es crear un ejemplar de una clase sin conocer la clase en el momento de la compilación. Este método de ejemplo, crea un nuevo ejemplar de

la misma clase que **obj** que puede ser cualquier clase heredada desde Object (lo que significa que podría ser cualquier clase).

```
Object createNewInstanceOf(Object obj) {
    return obj.getClass().newInstance();
}
```

- El **método *toString()*** devuelve una cadena de texto que representa al objeto. Este método se suele utilizar para dos cosas: comparar dos objetos con esa salida o bien mostrar esa salida al usuario.
  - Este método devuelve una cadena de texto que representa al objeto. Se puede utilizar **toString** para mostrar un objeto. Por ejemplo, se podría mostrar una representación del Thread actual de la siguiente forma.

```
System.out.println(Thread.currentThread().toString());
System.out.println(new Integer(44).toString());
```

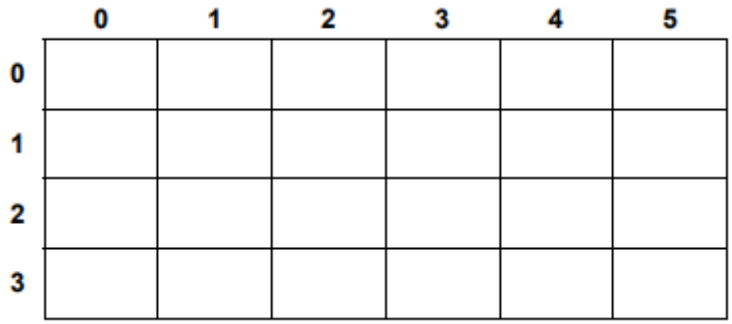
  - La representación de un objeto depende enteramente del objeto. El String de un objeto entero es el valor del entero mostrado como texto. El String de un objeto Thread contiene varios atributos sobre el thread, como su nombre y prioridad. Por ejemplo, las dos líneas anteriores darían la siguiente salida.

```
Thread[main,5,main]
4
```

  - El método **toString()** es muy útil para depuración y también puede sobreescribir este método en todas las clases.

7. ¿Qué es una matriz y que tipo de datos puede manejar?

Una matriz es una estructura de datos, o más técnicamente, un espacio de memoria que permite almacenar una colección de elementos, todos del mismo tipo. La diferencia con los arreglos está en que, en las matrices, los elementos no están organizados linealmente sino que su organización es bidimensional, es decir, en filas y columnas. Conviene imaginar una matriz como una organización de celdas de memoria, o casillas, en cada una de las cuales se puede guardar un elemento de la colección. Además, es usual dibujarla como lo ilustra la figura siguiente:



Esta figura representa un matriz de cuatro filas (numeradas verticalmente de 0 a 3) y seis columnas (numeradas horizontalmente de 0 a 5). En cada una de las 24 celdas o casillas se puede guardar un dato. La dimensión o tamaño de una matriz es el número filas por el número de columnas. Debe ser claro entonces que la figura anterior es la gráfica de una matriz de dimensión 4x6.

La numeración de las filas y las columnas determina que cada una de las casillas de una matriz tiene asociados dos números que la identifican de manera única. A estos números se les llama índice de fila e índice de columna, respectivamente. En el seudolenguaje, y también en C y C++, las filas y las columnas se numeran desde 0.

Los lenguajes como C y C++, permiten que el programador declare matrices de cualquier tipo y prácticamente de cualquier tamaño. En el seudolenguaje, un matriz se declara usando el siguiente formato:

<NOMBRE> : matriz [<N>][<M>] de <TIPO>

En este formato aparecen en mayúsculas y entre los caracteres < y > los componentes que el programador puede determinar. Así por ejemplo, si se quiere declarar una matriz con nombre mat, de dimensión 15x4 y que pueda almacenar datos de tipo caracter, se debe escribir la siguiente línea.

mat : matriz [15][4] de caracter

Según el formato anterior, el programador debe bautizar la matriz (ponerle un nombre significativo), debe decir cuál es su dimensión, y también debe decir de qué tipo son los elementos que almacenará.

Enseguida se dan algunos ejemplos de declaraciones de matrices.

- Si se necesita guardar la información relacionada con el tablero de un juego de tic tac toe (el tradicional triqui), se puede declarar la siguiente matriz:

tablero : matriz [3][3] de caracter

- Si se requiere guardar las notas que han sacado 35 estudiantes en los 5 talleres y en los 5 laboratorios del curso de Programación de Computadores se pueden declarar las siguientes matrices.

talleres : matriz [35][5] de real
laboratorios : matriz [35][5] de real

Note que, en ambas matrices, cada fila guarda las notas de un estudiante del curso.

- Si se quiere guardar las letras que conforman una sopa de letras, como aquellas que vienen en los pasatiempos, se puede declarar la siguiente matriz.

sopa : matriz [10][15] de caracter

Note que la sopa de letras más grande que se puede guardar es de 10 filas por 15 columnas.

Los índices se crearon para permitir que el programador se pueda referir, de forma específica y directa, a una cualquiera de las casillas de la matriz, tanto para guardar un dato en esa casilla, como para obtener el dato almacenado en ella. En el seudolenguaje, para referirse a una casilla particular de una matriz se debe seguir el siguiente formato:

<NOMBRE>[<INDICE-DE-FILA>][<INDICE-DE-COLUMNA>]

Es decir, se debe escribir el nombre de la matriz seguido por el índice de fila y por el índice de columna, ambos entre paréntesis cuadrados, de la casilla que se quiere consultar. Para los siguientes ejemplos, suponga que se declara la matriz montos, de la siguiente manera:

Para los siguientes ejemplos, suponga que se declara la matriz montos, de la siguiente manera:

montos : matriz [6][10] de real

- La siguiente instrucción asigna o guarda el número 10,4 en la casilla de la esquina superior izquierda de esta matriz:

```
montos[0][0]:= 10,4
```

- La siguiente instrucción iterativa guarda 5,5 en cada una de las casillas de la última fila de esta matriz:

```
k:=0
MIENTRAS (k<10) HACER
  montos[5][k]:= 5,5
  k:=k+1
FIN-MIENTRAS
```

La siguiente figura muestra la matriz montos después de ejecutadas las instrucciones de los dos ejemplos anteriores. Las casillas vacías no tienen valores definidos.

	0	1	2	3	4	5	6	7	8	9
0	10,4									
1										

Matriz (array bidimensional):

```
matriz[índice1][índice2]
```

Una matriz, en realidad, es un vector de vectores:

- ✓ En Java, el índice de la primera componente de un vector es siempre 0, por lo que `matriz[0][0]` será el primer elemento de la matriz.
- ✓ El tamaño del array puede obtenerse utilizando la propiedad `array.length`:
  - `matriz.length` nos da el número de filas
  - `matriz[0].length` nos da el número de columnas
- ✓ Por tanto, el último elemento de la matriz es `matriz[matriz.length-1][matriz[0].length-1]`