



Universidad Nacional Autónoma de México

Facultad de Ingeniería



Laboratorio de docencia
Laboratorios de computación
Laboratorios de computación A y
B

Práctica #3 Fundamentos y Sintaxis del lenguaje

Profesor: Tista García Edgar

Asignatura: Programación Orientada a Objetos

Grupo: 3

No de Práctica(s): 3

Integrante(s): Félix Flores Paul Jaime

Semestre: 2019-2

Fecha de entrega: 25-02-2019

Observaciones:

CALIFICACIÓN: _____

OBJETIVO.

Utilizar bibliotecas propias del lenguaje para realizar algunas tareas comunes y recurrentes.

DESARROLLO.

Ejercicio 1) Manual de laboratorio.

En el primer ejercicio suministramos parámetros al método main a través de la línea de comandos. Para ello, los valores a pasar deberán especificarse a continuación del nombre de la clase separados por un espacio:

>> *java NombreClase arg1 arg2 arg3*

```
C:\> Símbolo del sistema

Microsoft Windows [Versión 10.0.17134.590]
(c) 2018 Microsoft Corporation. Todos los derechos reservados.

C:\Users\andro>cd Desktop

C:\Users\andro\Desktop>javac Ejemplo1.java

C:\Users\andro\Desktop>java Ejemplo1 hola que tal
hola
que
tal

C:\Users\andro\Desktop>
```

Ejercicio 2) Manual de laboratorio.

Creamos un objeto ArrayList con la siguiente sintaxis:

```
ArrayList<TipoDato> nombreVariable = new ArrayList<TipoDato>( );
```

En el segundo ejemplo creamos un ArrayList:

```
ArrayList<Integer> arreglo = new ArrayList<Integer>( );
```

En este caso se creó un ArrayList llamado arreglo, el cual podrá contener elementos enteros (Integer) y usamos la palabra add de la colección para añadir los enteros.

```
C:\Users\andro\Desktop>javac Ejemplo2.java
C:\Users\andro\Desktop>java Ejemplo2
Tamaño del array list 4
Elemento en la posición 3: 5
1
9
8
5
C:\Users\andro\Desktop>_
```

Ejercicio 3) Manual de laboratorio.

En este ejercicio vimos el uso de Hashtable, que dice La clase Hashtable representa un tipo de colección basada en claves, donde los elementos almacenados en la misma (valores) no tienen asociado un índice numérico basado en su posición, sino una clave que lo identifica de forma única dentro de la colección. Para poder usarlo usamos la siguiente sintaxis :

```
Hashtable<TipoDatoClave, TipoDatoElemento> nombreVariable = new  
Hashtable<TipoDatoClave, TipoDatoElemento>( );
```

Aquí creamos un objeto llamado miTabla de la clase Hashtable, para este ejercicio usamos la colección de “containsKey(clave)” que nos Indica si la clave especificada existe o no en la colección (devuelve un boolean)

```
C:\Users\andro\Desktop>javac Ejercicio3.java
C:\Users\andro\Desktop>java Ejercicio3
Cotiene a cuatro ?false
cinco
dos
uno
5
2
1
```

Ejercicio 4) Manual de laboratorio.

Este ejemplo es parecido al anterior, pero en aquí usaremos put y gets

put(clave, valor) – Añade a la colección el elemento valor, asignándole la clave especificada. En caso de que exista esa clave en la colección el elemento se sustituye por el nuevo valor.

get(clave) – Devuelve el valor que tiene asociado la clave que se indica. En caso de que no exista ningún elemento con esa clave asociada devuelve null.

```
C:\Users\andro\Desktop>javac Ejemplo4.java
C:\Users\andro\Desktop>java Ejemplo4
Clave: cinco    valor: 5
Clave: dos     valor: 2
Clave: uno     valor: 1
C:\Users\andro\Desktop>_
```

Ejercicio 5) Manual de laboratorio.

En este ejercicio usamos a Calendar es una clase que surgió para cubrir las carencias de la clase Date en el tratamiento de las fechas. Para crear el objeto de Calendar se usa la siguiente sintaxis:

Calendar calendario = Calendar.getInstance();

Con esto pedimos el día del mes, el mes y el año para que fuera mostrado en pantalla y aprendimos a usar toString que obtiene la representación en forma de cadena de la fecha.

```
C:\Users\andro\Desktop>javac Ejemplo5.java
C:\Users\andro\Desktop>java Ejemplo5
Thu Feb 21 08:16:25 CST 2019
21-02-2019
miFecha
```

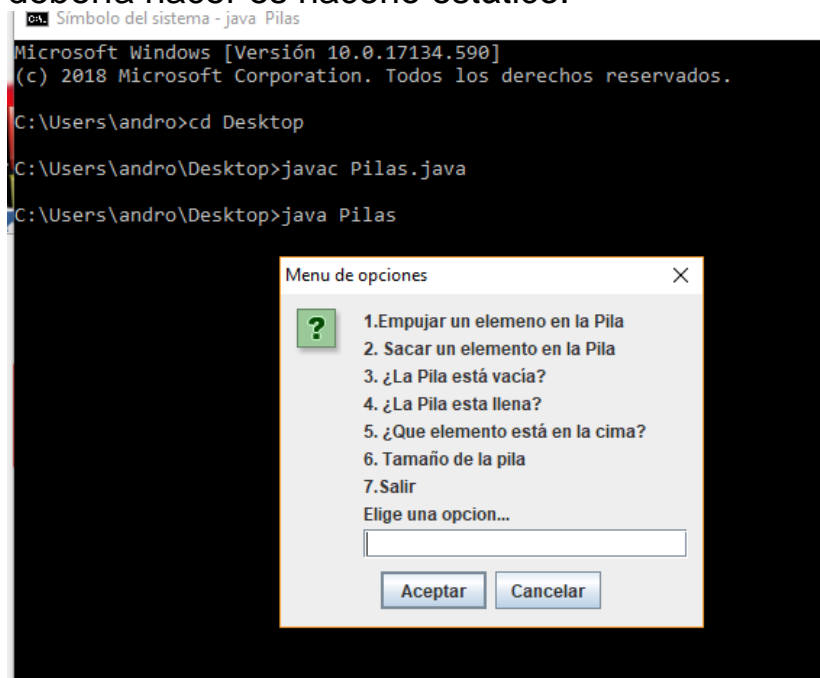
Ejercicio 1. Simulando una pila

Creando una clase para simular una pila, que tendrá como atributos un valor entero para indicar el tope de la pila y un arreglo que va a funcionar como la lista asociada a la pila, y como métodos las funciones push, pop, isEmpty y top para una pila. Recuerda que una pila solo se puede ingresar elementos mientras haya espacio en ella y al extraer algún elemento deberá ser el último que la ingreso.

- Crear una primera pila donde el usuario ingresara los valores deseados (10)
- Mediante dos pilas adicionales, y utilizando solo los métodos de la pila, encuentre el valor más grande ingresado por el usuario

Para esta primera parte hice un menú interactivo con el JOptionPane, para hacerlo un poco más visual bueno ahí hay un menú dónde podemos poner los datos de nuestra pila, para este programa hice mi clase principal y una clase secundaria donde puse todos los métodos que se muestran en el menú y los mandé a llamar desde la clase principal.

Para este primer problema tuve la dificultad de encontrar el número mayor entre mis números al ingresarlos para la pila, ya que no sé cómo implementar dos pilas al mismo tiempo. Busqué información en internet, pero no encontré nada para poder guiarme. También para este ejercicio decidí subir la dificultad al dejar que el usuario ingresara el número de elementos, ya que no pude hacer el último elemento, aunque para el elemento estático lo único que se debería hacer es hacerlo estático.



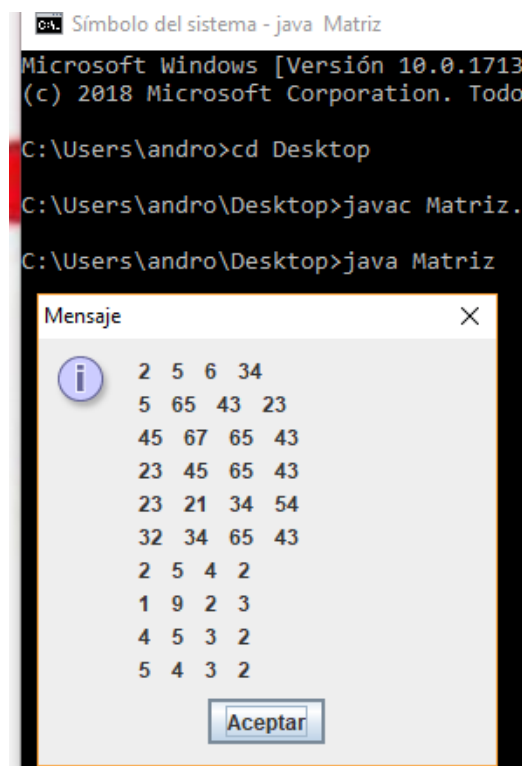
Ejercicio 2) Arreglos Irregulares

Escribe una clase que permita la creación de arreglos irregulares.

En dicha clase deberá haber un método para crear el arreglo irregular, solicitando al usuario la cantidad de renglones y la cantidad de columnas por renglón.

También habrá un método para imprimir el arreglo, dicho método deberá ser genérico y funcionar para cualquier tamaño de arreglo irregular.

Para este programa cree un arreglo que me recibiría tanto las filas como las columnas y mediante j e i iba pidiendo los datos mediante un sencillo for realmente es muy fácil, porque no es tan complicado, al igual que el programa anterior lo hice con un JOptionPane para hacerlo un poco más visual, realmente no se me complicó, ya que había hecho algo así en estructuras de datos, solo use la lógica para hacerlo en orientado a objetos.



The screenshot shows a Java IDE window titled "Símbolo del sistema - java Matriz". Inside, a command prompt displays the following commands and output:

```
Microsoft Windows [Versión 10.0.17134.0]
(c) 2018 Microsoft Corporation. Todos los derechos reservados.

C:\Users\andro>cd Desktop
C:\Users\andro\Desktop>javac Matriz.java
C:\Users\andro\Desktop>java Matriz
```

Below the command prompt, a "Mensaje" dialog box is displayed, showing an irregular array of integers:

2	5	6	34
5	65	43	23
45	67	65	43
23	45	65	43
23	21	34	54
32	34	65	43
2	5	4	2
1	9	2	3
4	5	3	2
5	4	3	2

An "Aceptar" button is located at the bottom of the dialog box.

Ejercicio 3: Clases envolventes.

En esta parte vimos que Java debe contener objetos y las operaciones entre ellos. La única excepción a esto en un programa Java son los tipos de datos primitivos (int, double, char, etc.)

Los tipos de datos primitivos no son objetos, pero en ocasiones es necesario tratarlos como tales. Por ejemplo, hay determinadas clases que manipulan objetos (ArrayList, HashMap, ...). Para poder utilizar tipos primitivos con estas clases Java provee las llamadas clases envolventes también llamadas clases contenedoras o wrappers. Así mismo vimos como funcionaban con los ejemplos de esta práctica.

```
C:\Users\andro\Desktop>javac ClasesEnvolventes.java
C:\Users\andro\Desktop>java ClasesEnvolventes
1.- 43
2.- 24.32
3.- 1234
4.- 12.664
5.- 83
6.- 83
7.- 83
8.- 83.0
9.- 10.32
10.- y 11.- 10
12.- 10
13.- 50.25
14.- 1100
15.- 3.1416
```

Ejercicio 4)

Elabora una clase para el manejo de fechas que tenga los siguientes métodos

- Método que calcule la diferencia en días entre dos fechas ingresadas como cadenas con el formato dd/mm/yy
- Método que dada una fecha en formato dd/mm/yy, calcule la fecha a 50 días posterior
- Método que devuelvan el “epochTime” calculando hasta la fecha que ingrese el usuario *Agregue en tu reporte una descripción de que es el “epochTime”.

Para la primera parte del programa no había tanto problema ya que lo único que teníamos que hacer era hacer una simple resta entre estas dos fechas con la parte parse, hacíamos un cast para convertir la fecha y así poder restarlo las fechas.

Para la segunda parte lo único que tuvimos que hacer era:

`hoy.add(Calendar.DATE, +50); //Aumentamos los 50 días`

Usar la palabra add para poder añadir los 50 días después.

La última parte me costó mucho, ya que no encontré mucha información de epochtime para guiarme, como que en cada parte hacían diferentes cosas en diferentes tipos de lenguaje, después leí y era que se tenía que contar a partir de la fecha de 1970 y así contar el tiempo desde esa fecha.

```
C:\Users\andro\Desktop>java Fechas
1.-Entre las fechas : 2019-02-14 & 2019-01-20
Hay: -25 días de diferencia
2.-Calculando 50 días despues apartir de la fecha de hoy
Mon Apr 15 23:30:26 CDT 2019
Tiempo actual = feb 24 2019 23:30:26.648 CST
Tiempo en Epoch: 1551072626648

C:\Users\andro\Desktop>
```

En esta parte fue el programa de epoch, para pedir una fecha, tuve un problema en pedir la fecha y convertirla, hice ambos programas, pero no logre hacer el match de hacerlo con epochTime, por mis pocos conocimientos en el programa de tiempo en java.

```
C:\Users\andro\Desktop>java Pedirfecha
Introduzca la fecha con formato dd/mm/yyyy
23/02/1991
Ahora hemos creado un objeto date con la fecha indicada, Sat Feb 23 00:00:00 CST 1991
Fecha validada
```


EpochTime

Two layers of encoding make up Unix time. The first layer encodes a point in time as a scalar real number which represents the number of seconds that have passed since 00:00:00 UTC Thursday, 1 January 1970. The second layer encodes that number as a sequence of bits or decimal digits.

As is standard with UTC, this article labels days using the Gregorian calendar, and counts times within each day in hours, minutes, and seconds. Some of the examples also show International Atomic Time (TAI), another time scheme which uses the same seconds and is displayed in the same format as UTC, but in which every day is exactly 86400 seconds long, gradually losing synchronization with the Earth's rotation at a rate of roughly one second per year.

Unix time is a single signed number which increments every second, without requiring the calculations to determine year, month, day of month, hour and minute required for intelligibility to humans.

The Unix epoch is the time 00:00:00 UTC on 1 January 1970. There is a problem with this definition, in that UTC did not exist in its current form until 1972; this issue is discussed below. For brevity, the remainder of this section uses ISO 8601 date and time format, in which the Unix epoch is 1970-01-01T00:00:00Z.

The Unix time number is zero at the Unix epoch, and increases by exactly 86400 per day since the epoch. Thus 2004-09-16T00:00:00Z, 12677 days after the epoch, is represented by the Unix time number $12677 \times 86400 = 1095292800$. This can be extended backwards from the epoch too, using negative numbers; thus 1957-10-04T00:00:00Z, 4472 days before the epoch, is represented by the Unix time number $-4472 \times 86400 = -386380800$.

Within each day, the Unix time number is calculated as in the preceding paragraph at midnight UTC (00:00:00Z), and increases by exactly 1 per second since midnight. Thus, 2004-09-16T17:55:43.54Z, 64543.54 seconds since midnight on the first day in the example above, is represented by the Unix time number $1095292800 + 64543.54 = 1095357343.54$. On dates before the epoch, the number still increases, thus becoming less negative as time moves forward.

Because Unix time is based on the Unix epoch, it is sometimes referred to as epoch time.

Conclusiones

Después de realizar la presente práctica, se puede concluir que se cumplieron los objetivos, debido a que se logró identificar el uso de arreglos, así como el uso de clases envolventes vistas en clase. Se logró comprenderlos al implementarlos en Java.

Con los ejercicios vistos en clases y al aplicarlos al lenguaje de programación orientado a objetos, los conceptos quedaron más claros.

En cuanto la dificultad de los ejercicios, tuve complicación en la implementación de pilas, al usar doble pilas, no recuerdo haber visto en eda 1, la implementación de una doble pila, no llegue a entender los conceptos de internet.

En cuanto al ejercicio 2 de arreglo multidimensionales, se me hizo más fácil entenderlo, ya que era había hecho algo similar en eda1 y no me costó tanto comprender ese concepto.

El de clases envolventes era más fácil ya que solo era entender los conceptos vistos en clase y bastaba con repasar los apuntes y poder entenderle mejor. Para el último ejercicio, fue lo más complicado ya que todavía no entiendo bien a cómo manejar el tiempo en java y al buscar información en red, los programas son escasos, casi nulos incluso buscando en otros idiomas, sobre todo epochtime, donde la información se vuelve muy escasa. Al final los pude resolver, pero no del modo que hubiese querido.

Los ejercicios de la práctica fueron muy buenos para comprender de una mejor manera el funcionamiento de los algoritmos de arrays y diferentes tipos de clases existentes en java, los ejercicios fueron buenos para poner en práctica los conocimientos adquiridos en la materia de Programación Orientada a Objetos.