	Carátula para entrega de prácticas
Facultad de Ingeniería	Laboratorio de docencia

Laboratorios de computación salas A y B

Profesor:	Edgar Tista Garcia
Asignatura:	Programación Orientada a Objetos.
Grupo:	Grupo 3
No de Práctica(s):	Práctica 3 – Utilerías y clases de uso general.
Integrante(s):	Edwin Jaret Santiago Díaz
No. de Equipo de cómputo empleado:	Trabajo en casa.
No. de Lista o Brigada:	
Semestre:	2022 - 2
Fecha de entrega:	26 febrero 2022
Observaciones:	

CALIFICACIÓN: _____

Utilerías y clases de uso general.

Objetivos

Objetivo: Utilizar bibliotecas propias del lenguaje para realizar algunas tareas comunes y recurrentes.
Objetivo de clase: conocer las clases de uso general y sus métodos más importantes y comprender la importancia de su uso como capa de abstracción para el programador.

Desarrollo

Ejemplos de la guía

Colecciones

El programa trabaja con un arreglo de tamaño variable de la clase *ArrayList*, se utiliza el método **add()** para agregar un número entero al arreglo, este puede recibir un parámetro (el número) o dos (la posición y el número).

```
1  arreglo.add(5);
2  // arreglo.add(valor);
3  arreglo.add(1, 9);
4  // arreglo.add(valor, posicion);
```

Otro método utilizado es el **size()** que nos retorna el tamaño del arreglo y el método **get()** que recibe como parámetro un índice y retorna el elemento de ese índice. Por último, se comprueba que con el ciclo for-each se puede recorrer los elementos de esa lista.

```
1  arreglo.size()
```

```
1  arreglo.get(3)
```

```
Tamano del array list 4
Elemento en la posicion 3: 5
1
9
8
5
```

Colecciones 2

Se utiliza la clase *Hastahble* las cuales guardan los valores con sus claves (asignadas por el programador o el usuario) Las claves o valores pueden ser de cualquier tipo de dato pero para este caso se trabaja con claves de tipo String y valores de tipo Integer.

Utilizamos los métodos:

- put(clave, valor) para añadir a la colección el elemento y su clave.
- containsKey(clave) para indicar si la clave existe o no en la colección.
- get(clave) para devolver el valor que tiene esa clave.

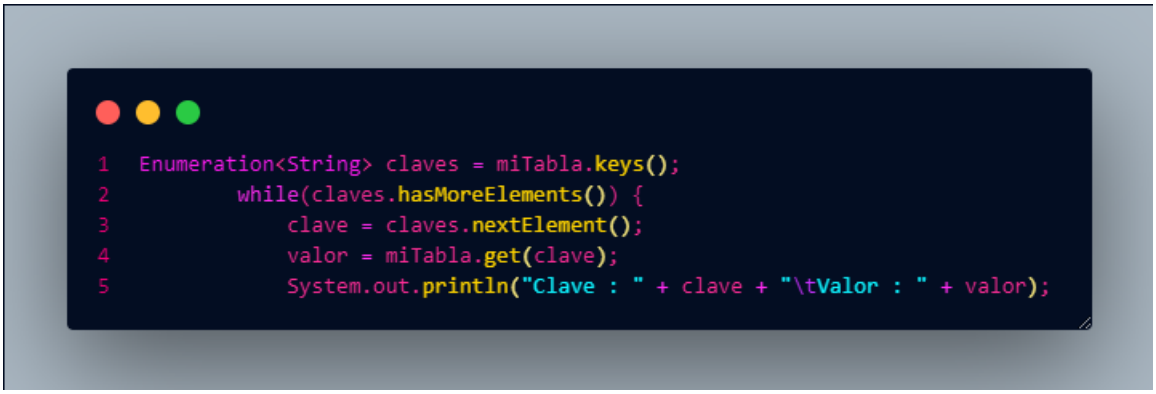
```
1  for(Integer valor:miTabla.values()) {
2              System.out.println(valor);
3          }
```

Y con los ciclos for-each recorremos la tabla, sin embargo, debemos de recorrerla por sus valores o por sus claves. Para recorrerlos por sus claves debemos de utilizar el método **keySet()** y para los valores **values().**

```
Contiene a cuatro? false
cinco
dos
uno
5
2
1
```

Colecciones 3

Para recorrer una tabla Hash con un ciclo while debemos de utilizar los métodos de la clase Enumeration. Esto es:



```
Enumeration<String> claves = miTabla.keys();
        while(claves.hasMoreElements()) {
            clave = claves.nextElement();
            valor = miTabla.get(clave);
            System.out.println("Clave : " + clave + "\tValor : " + valor);
```

1. Se guardan las claves en la variable **clave.**
2. Con el ciclo while se obtendrán las claves y los valores de la tabla.
3. Obtenemos la clave actual
4. Obtenemos el valor de la clave actual
5. Imprimimos los valores.

El recorrido del ciclo while es desde el final hacia el inicio.

Fechas

El programa importa las clases java.textSimpleDateFormat, java.util.Calendar, java.util.Date, para obtener el formato de la fecha, la fecha actual y modificar la estructura de fechas.

Para esto, se instancia una variable de la clase *Date* y con esto nos da la fecha actual, se puede modificar el formato con la clase *SimpleDateFormat("dd-MM-yyyy")* y con el parámetro del formato deseado y se imprime la fecha actual con dicho formato.
Por último, instanciamos otra variable con el método **getInstance()** de la clase *Calendar()* con el cual podemos obtener con los métodos **get(Calendar.DAY_OF_MONTH)** el día, **get(Calendar.MONTH)** el mes y **get(Calendar.YEAR)** el año actual.

Este es el resultado del programa:

```
Fri Feb 25 19:02:47 CST 2022
25-02-2022
Hoy es dia 25 del mes 2 de 2022
```

Ejercicios de la clase

Ejercicio1

El programa nos demuestra qué es lo que sucede en el momento de escribir más de 1 palabra (lo que sea, pero en este caso un String) después del nombre del archivo cuando se va a ejecutar. Funciona para algunos casos y para otros no (dependerá de algoritmo del programa). A primera vista, es un programa en donde se imprime un "Hola" con una variable y "Tu nombre es" con la misma variable.

Si ejecutamos el programa sin modificar nada y escribo mi nombre separado con un espacio del nombre del programa, imprimirá dos mensajes en la pantalla utilizando mi nombre.

```
o Díaz Edwin Jaret Practica 3> java Ejercicio1 Edwin
Hola Edwin
Tu nombre es: Edwin
```

Y si quito los comentarios del programa y al momento de ejecutar el programa escribo mi

nombre, mi apellido y mi edad separados por espacios, imprimirá 4 mensajes en la pantalla.

```
o Díaz Edwin Jaret Practica 3> java Ejercicio1 Edwin Santiago 19
Hola Edwin
Tu nombre es: Edwin
tu apellido es:Santiago
Tu edad: 19
```

Y, por último, si en el momento de ejecutar el programa solo escribo mi nombre y mi apellido, me aparece un error.

```
o Díaz Edwin Jaret Practica 3> java Ejercicio1 Edwin Santiago
Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: Index 2 out of bounds for length 2
        at Ejercicio1.main(Ejercicio1.java:5)
```

Esto nos da a entender que la función **main()** puede recibir parámetros de cualquier tipo de dato y estos se utilizan cuando se va a ejecutar el código, los parámetros se escriben después del nombre del archivo separados por espacios. Esto funciona como cualquier otra función donde si el algoritmo espera 3 parámetros, pero se reciben 2, notifica un error que esto no es posible.

Ejercico2

En el programa se creó una lista de números enteros (del 1 al 10) y se utiliza un método en donde sumará todos los números de la lista excepto el número mayor y el menor (10 y 1).

```
Resultado de la suma: 44
Los numeros sumados son: 2
Los numeros sumados son: 3
Los numeros sumados son: 4
Los numeros sumados son: 5
Los numeros sumados son: 6
Los numeros sumados son: 7
Los numeros sumados son: 8
Los numeros sumados son: 9
El numero menor: 1 y el mayor: 10
```

Después, en otros programas se van a utilizar 7 métodos de la clase *java.utils.Arrays*, cada método está escrito en distintos programas.

- **asList():** Archivo "*MetodoAsList.java*", transforma un arreglo (de cualquier tipo) a una lista del mismo tipo y lo guarda en una variable. Esto tiene como fin obtener los valores del arreglo y trabajar con ellos de manera de una lista.
  Para crear una lista y copiar el arreglo se necesita la clase *java.util.List* y se crea de esta forma:

```
1  List<Integer> lista1 = Arrays.asList(arreglo1);
2  // List<TipoDeDato> NombreVariable = Arrays.asList(NombreArreglo);
```

  Se trabajó como ejemplo un arreglo de tipo entero, se copió en una lista y se imprimió el primer elemento del arreglo y el contenido de la lista:

```
Primer elemento del arreglo: 10
Lista: [10, 40, 50, 20, 90]
```

- **copyOf():** Archivo "*MetodoCopyOf.java"* hace una copia de un arreglo (de cualquier tipo) y se puede definir el tamaño que tenga la copia, este puede ser más grande o más pequeño que el tamaño del arreglo original, si es más pequeño, se conservarán los elementos hasta el tamaño deseado, si es más grande, los elementos que no se encuentran en el arreglo original tendrán un valor nulo y después podemos definirle su valor. La copia se guarda en un arreglo.

  Para copiar el arreglo y guardarlo en otro arreglo, se hace de esta forma:

```
int[] copiaMenor = Arrays.copyOf(arreglo1, 4);
// tipo[] nuevoArreglo = Arrays.copyOf(arregloOriginal, Tamaño);
```

Para este ejemplo, se trabajó con arreglos de tipo entero y estos fueron los resultados:

```
Arreglo original:
1 2 3 4 5 6
Copiamos el arreglo con tamaño a 10 indices:
1 2 3 4 5 6 0 0 0 0
Copaimos el arreglo con tamaño a 4 indices:
1 2 3 4
Modifamos el arreglo de mayor tamano:
1 2 3 4 5 6 964 12 121 532
```

- **copyOfRange():** Archivo "*MetodoCopyOfRange.java*", hace una copia de un arreglo (de cualquier tipo de dato) y se define desde qué índice X hasta que índice Y se hace la copia (el índice Y es exclusivo). Se guarda en un arreglo (del mismo tipo).

  El formato para realizar la copia es:

```
int[] arreglo2 = Arrays.copyOfRange(arreglo1, 1, 3);
// tipoDato[] arrelgoCopia = Arrays.copyOfRange(arregloOriginal, incio, final);
```

Para este ejemplo, se trabajó con arreglos de tipo entero y estos fueron los resultados:

```
Arreglo original:
1 2 3 4 5
Copia del arreglo desde el indice 1 hasta el 3:
2 3
```

- **equals():** Archivo "*MetodoEquals.java*", compara dos objetos y nos da un valor booleano.

  Funciona de la siguiente manera:

```
lista1.equals(lista2);
// objeto1.equals(obejto2);
```

Para este ejemplo, se trabajó con arreglos, listas y cadenas. Estos fueron los resultados:

```
[Ljava.lang.Integer;@5b2133b1 y [Ljava.lang.Integer;@72ea2f77 son DIFERENTES
[1, 2, 3, 4, 5] y [1, 2, 3, 4, 5] son IGUALES
Chocolate y Vainilla son DIFERENTES
```

Llegué a la conclusión de que con los arreglos se compara la dirección de memoria y no los elementos, con las listas y las cadenas se comparan los elementos/datos.

- **sort():** Archivo "*MetodoSort.java*", ordena cualquier tipo de array que le pasemos como argumento y este modifica el arreglo original.

  Funciona de la siguiente manera:

```
Arrays.sort(paises);
// Arrays.sort(arreglo);
```

Se obtuvieron este resultado:

```
Arreglo original:
MEXICO COLOMBIA EUA UCRANIA RUSIA CHINA
Arreglo ordenado:
CHINA COLOMBIA EUA MEXICO RUSIA UCRANIA
```

- **toString():** Archivo "*MetodoToString.java*", devuelve la variable o el parámetro recibido en formato de String.

  Funciona de las dos siguientes maneras:

```java
System.out.println(Integer.toString(917));
// System.out.println(Integer.toString(parametro));

// O:
// Se crea una lista
List<Integer> lista1= Arrays.asList(1,2);
System.out.println(lista1.toString());
// System.out.println(variable.toString());
```

  Y el resultado es:

```
917
[1, 2]
```

- **binarySearch():** Archivo "*MetodoBinarySearch.java*", busca en el arreglo a través de la búsqueda binaria un elemento deseado, sin embargo, el arreglo debe de estar ordenado para que funcione. Retorna el índice de ese elemento.

```java
Arrays.binarySearch(paises, pais);
// Arrays.binarySearch(arreglo, elemento/variable);
```

  Funciona de la siguiente manera:

  El resultado es:

```
Indice 0 y pais CHINA
Indice 1 y pais COLOMBIA
Indice 2 y pais EUA
Indice 3 y pais MEXICO
Indice 4 y pais RUSIA
Indice 5 y pais UCRANIA
El indice de Ucrania es: 5
```

En mi experiencia, es más complicado usar los métodos de java pues he trabajado con métodos en el lenguaje Python y son más fáciles y accesibles de usar.

Ejercicio3

Se utiliza el programa "*Ejercicio4Envolventes.java*" propuesto por el profesor, se corrigieron algunos errores como escribir bien "integer", escribir bien la asignación de valores en las variables de tipo int, float, double, short, byte y al querer guardar el valor del resultado del método **shortValue(),** no se había asignado el nombre de la variable (decidí ponerle C).

Para usar el método **compareTo()** (que compara cualquier tipo de dato) con dos números enteros ("83" y "50") se obtiene este resultado:

```
Al comparar 83 y 50 se obtiene: 1
```

El método **compareTo()** puede regresar 3 opciones:

- Si regresa 0, los 2 valores son iguales
- Si regresa un número menor a 0 es porque el primer elemento es menor que el segundo (X < Y).
- Si regresa un número mayor a 0 es porque el primer elemento es mayor que el segundo (X > Y).

La diferencia entre los métodos **valueOf()** y **ParseXx()** es que **valueof()** devuelve un **new integer()** (integer object) y almacena en caché valores de -128 a 127 , puede aceptar string e int, mientras que **parseXx()** solo acepta string y devuelve el tipo de dato que maneje en Xx (int, double, byte,short,long, etc).

Ejercicio4

En el archivo "*Ejercicio4.java*" se utilizan las clases *SimpleDateFormat* (para asignar el formato de la fecha), *Date* (instanciar variables con formato de la fecha), *Scanner* (para obtener datos de entrada) y *TimeUnit* (para utilizar unidades de tiempo).

Para el funcionamiento del programa, se creó un menú en donde el usuario escoge una opción de las 4, cada opción ejecuta una función y el objetivo de cada función es:

- Se le solicita al usuario que ingrese dos fechas en formato **dd-mm-yy** y se calculará la diferencia entre las fechas en días. Este es el resultado:

```
Ingrese dos fechas con el formado dd-mm-yy para comparar la diferencia de estas
Primera fecha:
06-01-14
Segunda fecha:
21-04-21
Los dias de diferencia entre 06-01-14 y 21-04-21 son: 2661
```

- Se le solicita al usuario que ingrese una fecha en formato **dd-mm-yy** y se calculará la fecha 40 días posterior a esta. La fecha que ingresa es explícita (no se toma en cuenta dentro de los 40 días). Este es el resultado:

```
Ingrese una fecha con el formado dd-mm-yy para calcular la fecha en 40 dias:
01-01-22
La fecha en 40 dias es: 10-02-22
```

- Se le solicita al usuario que ingrese una fecha en formato **dd-mm-yy** y se calculará los días faltan o han pasado con respecto a la fecha actual. Este es el resultado:

```
Ingrese una fecha con el formado dd-mm-yy para calcular la diferencia de dias con la fecha actual:
01-01-22
La diferencia de diascon la fecha actual 25-02-22 es de: 55
```

- Se le solicita al usuario que ingrese una fecha en formato **dd-mm-yy** y se calculará el epochTime. Este es el resultado:

```
Ingrese una fecha con el formado dd-mm-yy para calcular el epochTime:
25-02-22
El tiempo epoch con 25-02-22 es: 1645747200000
```

El tiempo epoch son las fechas de las computadoras expresado en segundos desde un punto de inicio (00:00:00), esto varía depende de los sistemas operativos como en MacOS comienza desde el 1 de enero de 1904, para Windows inicia el 1 de enero de 1601 y para Unix desde el 1 de enero de 1970.

El epoch se basa en Unix y, por lo tanto, el epoch son los segundos que han pasado desde el 1 de enero de 1970.

Conclusiones

Existen varias clases para trabajar con los arreglos y fechas y la implementación de estas dependerá del objetivo del programa. Hubo programas que se ocuparon más de 2 clases para el manejo de arreglos y más de 4 clases de fechas.

Es muy convencional pasar de un arreglo a una lista pues así se puede trabajar con la lista de manera eficaz, pero es mejor trabajar con los arreglo-listas.

Es de las prácticas que más tiempo me ha llevado para realizarla pues implicó que investigara mucho sobre clases, métodos, parámetros, sintaxis y analizar la lógica de mis programas para cumplir sus objetivos. Es una práctica un poco tediosa, pero me ayudó mucho y valió la pena.

Se completó todos los ejercicios de la práctica.

Bibliografía.

- *Oracle , Class Arrays*, Recuperado de: https://docs.oracle.com/javase/7/docs/api/java/util/Arrays.html

- Oracle, *Class Instant*. Recuperado de: https://docs.oracle.com/javase/8/docs/api/java/time/Instant.html