	Carátula para entrega de prácticas
Facultad de Ingeniería	Laboratorio de docencia

Laboratorios de computación salas A y B

Profesor:	TISTA GARCÍA EDGAR
Asignatura:	PROGRAMACIÓN ORIENTADA A OBJETOS
Grupo:	3
No de Práctica(s):	PRACTICA #3 – UTILERIAS Y CLASES DE USO GENERAL
Integrante(s):	CARRILLO CERVANTES IVETTE ALEJANDRA
No. de Equipo de cómputo empleado:	TRABAJO EN CASA
No. de Lista o Brigada:	08
Semestre:	2022 - 1
Fecha de entrega:	28 SEPTIEMBRE 2021
Observaciones:	

CALIFICACIÓN: _____

PRACTICA #3 – UTILERIAS Y CLASES DE USO GENERAL

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.

Antes de comenzar la práctica, se activó la página de códigos 65001 desde la terminal, con el fin de que en los programas muestren los caracteres.

EJEMPLOS DE LA GUIA

Ejemplo 1

Este programa tiene como función principal suministrar parámetros al método main a través de la línea de comandos, esto se logra especificando estos parámetros a continuación del nombre de la clase al momento de ejecutar el programa. Dentro de este programa, se imprimen 3 elementos de un arreglo, es decir, los 3 parámetros necesarios que se escribieron desde consola para que el programa funcione.

```
C:\Users\aleja\OneDrive\Escritorio\FACULTAD\2022-1\POO\Prácticas\P3 POO\Carrillo Ce
rvantes Ivette Alejandra G3 P3 V1\Ejemplos>java Ejemplo Hola, que tal?

Hola,
que
tal?
```

./Ejemplos/Ejemplo.java

Ejemplo 2

Este programa tiene como función principal crear un arreglo a través del método "*ArrayList*" de la clase "*Arrays*". Ya creado el arreglo, se le van agregando el valor a cada uno de sus índices en la posición siguiente o bien, especificando en que indice se requiere agregar el valor con el método "*add*". Finalmente se imprime el tamaño que tiene el arreglo (con el método "size"), que elemento se encuentra en la posición 3 (Con el método "*get*") y el arreglo completo mediante un ciclo de repetición "*for each*".

```
C:\Users\aleja\OneDrive\Escritorio\FACULTAD\2022-1\POO\Prácticas\P3 POO\Carrillo Ce
rvantes Ivette Alejandra G3 P3 V1\Ejemplos>java Colecciones

Tamaño del array list 4
Elemento en la posición 3: 5
1
9
8
5
```

./Ejemplos/Colecciones.java

Ejemplo 3

Este programa tiene como función principal conocer los métodos más relevantes que tiene la clase "*Hashtable*", para ello se creó una tabla hash la cual contiene elementos de tipo String (como claves) e Integer (como valores), esta tabla se va llenando mediante el método "*put*" y después con el método "*containsKey*" se busca en la tabla si se contiene la llave "cuatros", este último método devuelve un tipo de dato booleano "True" o "false" si se encuentra en la tabla el dato a buscar. Finalmente, se imprime cada clave de la tabla, al igual que cada valor de ella, mediante un ciclo "*Foreach*".

```
C:\Users\aleja\OneDrive\Escritorio\FACULTAD\2022-1\POO\Prácticas\P3 POO\Carrillo Ce
rvantes Ivette Alejandra G3 P3 V1\Ejemplos>java Colecciones2

Contiene a cuatros? false
Clave: cinco
Clave: dos
Clave: uno
Valor: 5
Valor: 2
Valor: 1
```

./Ejemplos/Colecciones2.java

Ejercicio4

Este programa tiene como función principal imprimir las claves y valores de una tabla hash, mediante un ciclo de repetición "*while*" cuya condición repetitiva es el método "*hasMoreElements*" el cual comprueba si una enumeración tiene más elementos.

```
C:\Users\aleja\OneDrive\Escritorio\FACULTAD\2022-1\POO\Prácticas\P3 POO\Carrillo Ce
rvantes Ivette Alejandra G3 P3 V1\Ejemplos>java Colecciones3

Clave: cinco    Valor: 5
Clave: dos      Valor: 2
Clave: uno      Valor: 1
```

./Ejemplos/Colecciones3.java

ACTIVIDADES PARA EL DESARROLLO DE LA PRÁCTICA

Ejercicio 1. String[] args

El objetivo principal de este programa es suministrar parámetros al método main a través de la línea de comandos, especificando estos parámetros seguida del nombre de la clase al momento de ejecutarse. En este programa a diferencia del ejemplo de la guía, se crean dos variables tipo String y una de tipo int, ocupando el método "*parseInt*" con el fin de convertir una cadena de texto a un número entero, en las cuales se van a guardar los datos que se pasaron como parámetros al momento de ejecutarse.

```
String cadena1 = args[0];
String cadena2 = args[1];
int entero =Integer.parseInt(args[2]);
```

Tipo de datos

Para probar el funcionamiento de este programa se realizaron 3 ejecuciones:

En la primera ejecución solo se paso 1 parámetro cuyo valor es el nombre de una persona y la salida fue la siguiente:

```
C:\Users\aleja\OneDrive\Escritorio\FACULTAD\2022-1\POO\Prácticas\P3 POO\Carrillo
 Cervantes Ivette Alejandra G3 P3 V1>java Ejercicio1 Alejandra

Hola Alejandra (:
Tu nombre es: Alejandra
```

Ejecución 1 ./Ejercicio1.java

Para la segunda ejecución se pasaron los tres parametros con los cuales se va a trabajar, los cuales son el nombre de la persona, su apellido y su edad. Al igual que en la primera ejecución se imprimieron los datos correctamente.

```
C:\Users\aleja\OneDrive\Escritorio\FACULTAD\2022-1\POO\Prácticas\P3 POO\Carrillo
 Cervantes Ivette Alejandra G3 P3 V1>java Ejercicio1 Alejandra Carrillo 19

Hola Alejandra (:
Tu nombre es: Alejandra
Tu apellido es: Carrillo
Tu edad: 19
```

Ejecución 2 ./Ejercicio1.java

Finalmente, al ejecutarlo por tercera vez tomando como parámetros solo el nombre y el apellido de la persona, se marca el siguiente error, el cual indica que se esta accediendo a un indice del arreglo el cual no existe, o bien, no están ocupándose todos los valores del arreglo.

```
C:\Users\aleja\OneDrive\Escritorio\FACULTAD\2022-1\POO\Prácticas\P3 POO\Carrillo
 Cervantes Ivette Alejandra G3 P3 V1>java Ejercicio1 Alejandra Carrillo
Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: Index 2 out
 of bounds for length 2
        at Ejercicio1.main(Ejercicio1.java:5)
```

Ejecución 3 ./Ejercicio1.java

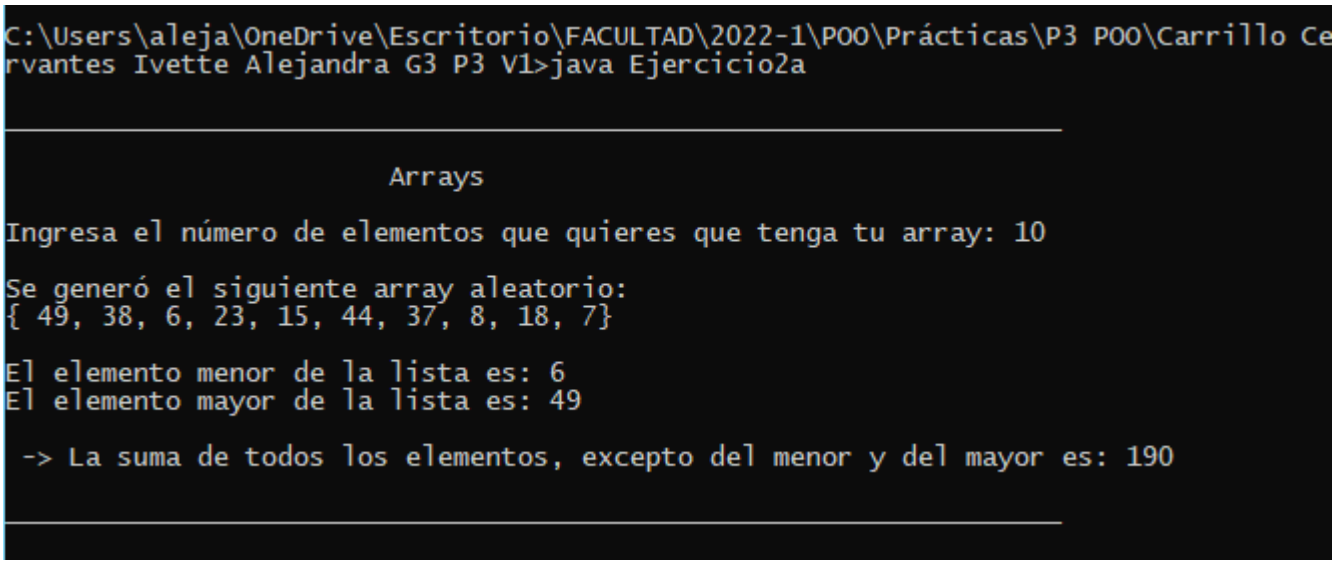
Ejercicio 2. Arreglos

Ejercicio2a

Este programa tiene como función principal sumar todos los elementos de un arreglo excepto el mayor y el menor de sus elementos, esto se logró realizando un método el cual devuelve la suma de los elementos mencionados anteriormente, dentro de este método se realizó un ciclo de repetición for el cual recorrerá toda la lista y determinara cual es el número mayor y cual es el número menor del arreglo, estos datos se guardará en variables auxiliares que se ocuparán dentro de otro ciclo de repetición "*for*" el cual también recorrerá esta lista, pero a diferencia del anterior este irá comparando cada elemento del arreglo con las variables auxiliares donde se guarda el valor mayor y menor del arreglo, en caso de que sean iguales, el programa continuará sin hacer ninguna modificación, de lo contrario (si estos elementos son diferentes) con ayuda de una variable "*suma*" previamente inicializada, ira sumando los elementos del arreglo.

La principal dificultad que se me presentó en este programa fue al momento de obtener el menor y el mayor elemento del arreglo, ya que estaba comparando el elemento actual con el siguiente con el fin de obtener estos valores; sin embargo, marcaba un error, ya que el último elemento no tiene un siguiente y por ende no se puede hacer la comparación. Para solucionar este problema, las variables auxiliares llamadas “menor” y “mayor” fueron inicializadas con el primer elemento del arreglo y así poder ir comparando este elemento con el elemento en el indice correspondiente en el ciclo “*for*”.

La salida de este programa es la siguiente:



./Ejercicio2a.java

Ejercicio2b

Este programa tiene como objetivo probar los siguientes métodos de la clase *java.util.Arrays*, usando la API de java para obtener información.

- asList(T…a)

Devuelve una lista de tamaño fijo respaldada por el arreglo especificado.

- copyOf()

Copia el arreglo especificado, truncando o rellenando con ceros/nullos/falsos (si es necesario) para que la copia tenga la longitud especificada.

- copyOfRange()

Copia el rango del arreglo especificado en un nuevo arreglo.

- Equals()

Devuelve verdadero si dos arreglos especificados son iguales entre sí.

- Sort()

Ordena el arreglo especificado en orden ascendente.

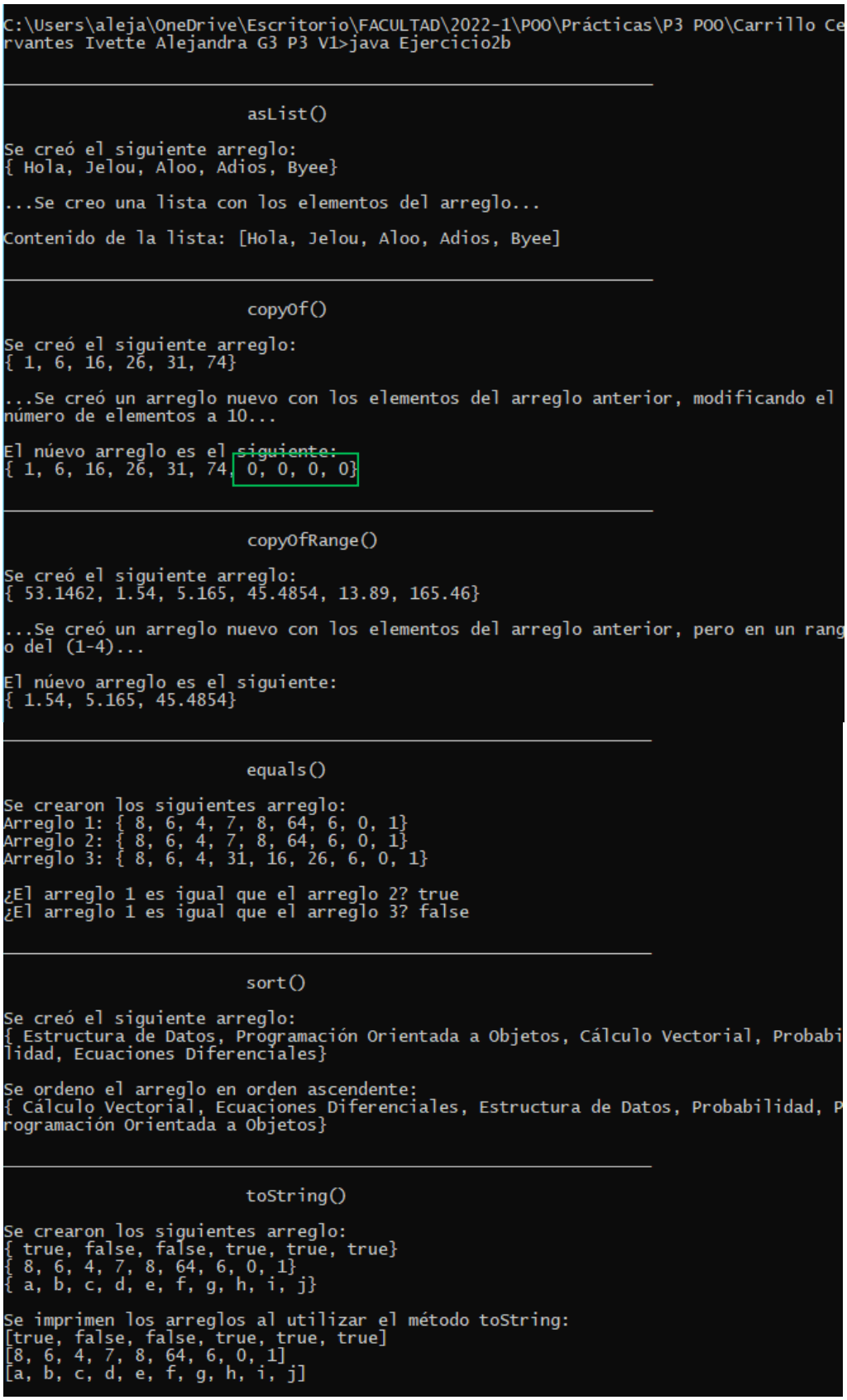
- toString()

Devuelve una representación de cadena del contenido de la matriz especificada.

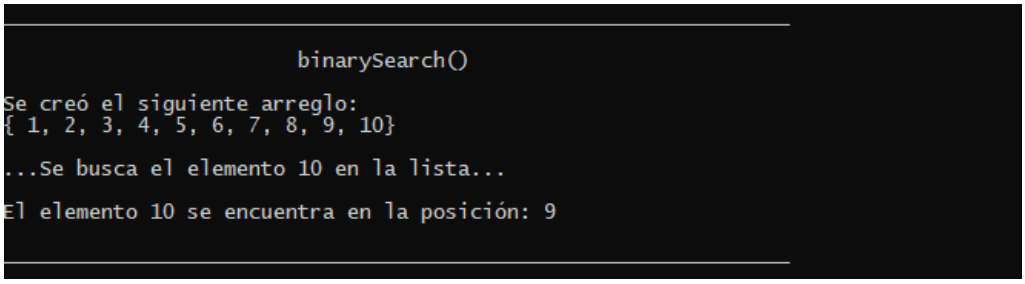
- binarySearch()

Busca en un arreglo especificado el valor especificado mediante el algoritmo de búsqueda binaria.

Cada uno de los métodos mencionados anteriormente se implemento de la siguiente manera:



Se relleno con ceros para que tenga la longitud especificada



./Ejercicio2b.java

Ejercicio 3. Clases envolventes

Se tuvo que corregir el código empleado para este programa, ya que tenía los siguientes errores:

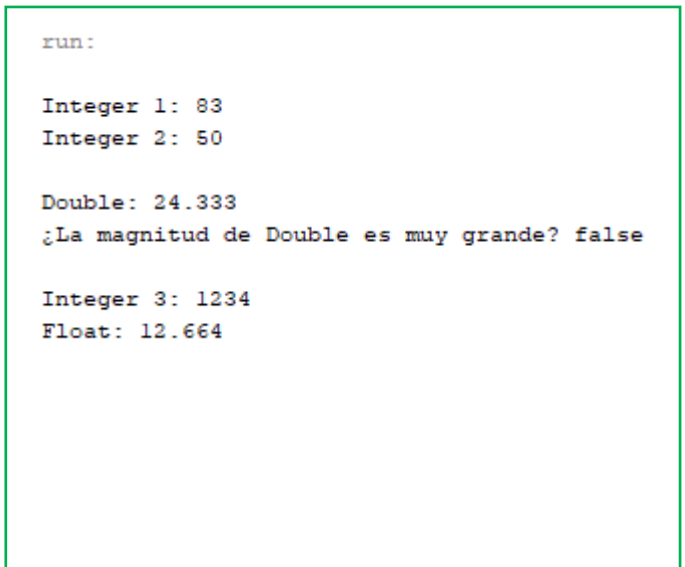
- Integre <- Este error surge de que esta mal escrita la clase envolvente. Se solucionó escribiéndola de manera correcta
- Integer entero3 = new Integer("1234"); <- Para realizar una clase envolvente se necesita la palabra reservada "*new*"
- short e = envolvente.shortValue() <- El valor que devolvía el método shortValue no se guardaba en ninguna variable, por lo que se creó la variable "e".

Además, no se pudo compilar este programa desde la consola, ya que marcaba que no era necesario crear un objeto en la clase envolvente, pues el autoboxing trabaja convirtiendo los tipos primitivos en objetos y viceversa. Por lo tanto, este programa se realizó en el IDE "netbeans", en este IDE solo marcó una advertencia al crear clases envolventes, pero se pudo ejecutar.

Se agregaron las instrucciones necesarias para mostrar en pantalla cada valor de las variables que se ocuparon al crear clases envolventes; además se agregaron otras instrucciones para verificar el método "*compareTo()*" con ayuda de los enteros de las primeras líneas del programa, cabe recalcar que la función principal de este método es comparar dos variables, indicando cual es mayor según la siguiente consideración: Si el método devuelve un número positivo, la variable 1 es mayor que la variable 2; si devuelve un 0 es por que ambas variables son iguales; finalmente, si devuelve un número negativo, la variable 1 es menor que la variable 2.

Los métodos más importantes que se vieron durante la implementación de este programa fueron: valueOf y ParseXx, ambos convierten un tipo de dato "String" a algún tipo de dato numérico especificado; sin embargo, a pesar de que ambas tienen la misma función, valueOf devuelve una clase envolvente, mientras que ParseXx devuelve un tipo de dato primitivo.

Este programa tiene la siguiente salida:



```
Envolvente 1: 83
Convertimos Envolvente 1 en:
Byte: 83
short: 83
double: 83.0

Envolvente 2: 10.32
Convertimos Envolvente 2 en:
short: 10

Parse
Int: 10
Double: 50.25

ValueOf
Integer: 1100
Double: 3.141519

El entero 1 se comparo con el entero 2: 1
El entero 1 se comparo con envolvente: 0
El entero 2 se comparo con el entero 1: -1

BUILD SUCCESSFUL (total time: 0 seconds)
```

./Ejercicio4Envolventes

Ejercicio 4. Fechas en Java

Clases para utilizar:

- import java.text.SimpleDateFormat

Es una clase concreta para formatear y analizar fechas de una manera sensible a la configuración regional. Permite formatear (fecha -> texto), analizar (texto -> fecha) y normalizar.

- import java.util.Date

representa un instante específico en el tiempo, con precisión de milisegundos.

- import java.util.Calendar

es una clase abstracta que proporciona métodos para convertir entre un instante específico en el tiempo y un conjunto de calendar fields, como YEAR, MONTH, DAY_OF_MONTH, HOUR, y así sucesivamente, y para manipular los campos de calendario, como obtener la fecha de la próxima semana.

- import java.util.time

Es una extensión a las clases java.util.Date y java.util.Calendar, los cuales se ven un poco limitado para manejo de fechas, horas y locación. Las clases definidas en este paquete representan los principales conceptos de fecha – hora, incluyendo instantes, fechas, horas, periodos, zonas de tiempo, etc. Están basados en el sistema de calendario ISO, el cual el calendario mundial de-facro que sigue las reglas del calendario Gregoriano.

Conclusiones

Se cumplió la mayoría de los objetivos de esta práctica, ya que se conocieron las clases de uso general, así como sus métodos más importantes; también, se comprendió la importancia de su uso como capa de abstracción para el programador. Sin embargo, no se hizo uso de todas las clases propuestas para esta práctica, ya que en el ejercicio 4 se tuvieron que ver las clases principales que proporciona java para el manejo de fechas, pero no supe como implementar algunos métodos de cada clase, o bien, me hizo falta investigar un poco más acerca de dichas clases, con la finalidad de hacer más eficiente la implementación.

Como se mencionó anteriormente, no se cumplieron con todos los ejercicios de la práctica, a pesar de ello, considero que esta práctica tuvo los ejercicios necesarios para conocer los métodos y clases más importantes que se usan en lenguaje de programación Java.

Referencias

- Oracle (1993, 2020) Java SE Documentation. Oracle