|  |  |  |
| --- | --- | --- |
|  | **Carátula para entrega de prácticas** | |
| Facultad de Ingeniería | | Laboratorio de docencia |

Laboratorios de computación

salas A y B

|  |  |
| --- | --- |
| *Profesor:* | Ing. Guadalupe Lizeth Parrales Romay |
| *Asignatura:* | Programación Orientada a Objetos |
| *Grupo:* | 04 |
| *No de Práctica(s):* | 2 |
| *Integrante(s):* | 1 |
| *No. de Equipo de cómputo empleado:* | Equipo personal. |
| *No. de Lista o Brigada:* | 14 |
| *Semestre:* | 2021-1 |
| *Fecha de entrega:* | 15 de octubre del 2020 |
| *Observaciones:* |  |
|  |  |

CALIFICACIÓN: \_\_\_\_\_\_\_\_\_\_

**Objetivo de la practica**

Crear programas que implementen variables y constantes de diferentes tipos de datos, expresiones y estructuras de control de flujo.

**Introducción**

* **Scanner en Java**

Como parte de la inducción a Java y al entorno de programación tenemos una clase denominada Scanner que sirve para la introducción de valores por medio de teclado, de tal forma que para almacenar un valor en una variable de determinado tipo primitivo es necesario indicar el tipo de la variable, a continuación se muestra la forma de cómo se ajusta a cada tipo de dato (Imagen 1)

* Scanner scanner = new Scanner( text ); //declaración del objeto que scanea.
* byte b = new Scanner("16").nextByte( );
* int n = new Scanner("42").nextInt( );
* long l = new Scanner("99999999999").nextLong( );
* float f = new Scanner("4.2").nextFloat( );
* double d = new Scanner("99.99999999").nextDouble( );
* boolean b = new Scanner("true").nextBoolean( );
* String description = scanner.next( );

Generalizando: [Tipo de dato] [identificador] =[new] [Scanner] [.next][Tipo de dato]();

A modo de aclaración se tiene que la declaración de un objeto Scanner se puede hacer separado como se indica en el primer ejemplo, y posteriormente asignar el tipo de dato en el next.

Siguiente de la declaración del tipo de dato, sigue el identificador, después se tiene la definición new Scanner como la obtención del valor que se tiene como argumento, y por último se define la expresión nextInt, la cual viene acompañada del tipo de dato, y define el valor que se tendrá que guardar.

La utilidad Scanner se introdujo en Java 5.0, por lo que la utilidad puede leer elementos individuales de una cadena de estos, dependiendo del tipo que sean, se aplica la forma correspondiente que permite el almacenado del tipo, además de poder utilizar una base como un argumento opcional de funcionamiento que en primera instancia no es fundamental, ya que sin este puede funcionar.

* **For Each-Java**

La estructura iterativa for each es una versión del for, que esta mejorada porque no es necesario el conocer el fin de la lista únicamente se necesita una lista o arreglo que recorrer, y una variable que la recorra, esta variable almacenara los valores de cada posición que vaya visitando en el arreglo o lista.

La forma es la siguiente:

For(tipo\_dato var : arreglo/lista ){

//instrucciones;

}

Algo importante a considerar es que no utiliza índices únicamente una variable que tendrá que ser del mismo tipo del arreglo o lista, de lo contrario no podrá almacenar los valores del tipo, además que se tiene que considerar el uso que tiene. Ya que es mas de recolección de datos en una única variable, por lo que es necesario evaluar bien las condiciones de trabajo para que no se cometan errores en la selección del tipo de ciclo de iteración.

* **Método Print()**

Este método se encuentra dentro de la clase PrintStream, el cual esta sobrecargado, lo cual significa que es un método que tiene el mismo nombre que otros métodos, pero que la diferencia que existe entre estos es que reciben diferentes argumentos, lo que da una ilusión de que el mismo método realiza distintas operaciones.

En base a esto tenemos algo bastante interesante de este método, ya que la línea siguiente esconde un secreto que se relaciona con el sobrecargar un método: System.out.print( argument ), en esta línea de código tenemos al print, debido a que el Output, lo que realiza es la impresión en pantalla por medio de la clase PrintStream.

Esto significa que es un objeto de la clase PrintStream, y que al mandar a imprimir algo, se están mandando argumentos a los distintos métodos de print(), que como sabemos cambian por que reciben distintos argumentos al imprimir, la clase esta definida de la siguiente forma:

class PrintStream {

void print( Object arg ) { ... }

void print( String arg ) { ... }

void print( char [] arg ) { ... }

...

}

Estos funcionan por el emparejamiento adecuado de los argumentos y en caso de que no existiera de esta forma, sería el necesario recordar cada método creado y para que tipo estaba definido así que en este método es posible ver la efectividad de la sobrecargar.

**Ejercicios de la practica:**

* Ejercicio 1:

El primer ejercicio consistía en tener tres valores, los cuales van a ser comparados, de esta manera se va a determinar cual de ellos es el mayor, cual es el segundo mayor o el de en medio y menor, para decidir las posiciones se decide por la implementación de la estructura if-else.

* Dificultades en el código

En este primer ejercicio lo difícil fue el realizar, el planteamiento del funcionamiento de las decisiones o condicionales que van dentro del if. Otro elemento del código que se hizo confuso fue el utilizar la clase Scanner porque en un determinado momento me di cuenta de que no se podía asignar a un elemento estático, pero no tenía ningún estático, y esto va más allá porque el método que se utiliza de la clase Scanner para poder recibir valores se desconoce si es estático o no, por lo que decidí guiarme por los ejercicios realizados y por los ejemplos del manual de práctica.

* Diagrama de funcionamiento:

Println(“El mayor es A”)

If(A>B && A>C)

1)

Se insertan tres valores:

A=10

B=15

C=5

Println(“El intermedio es B”)

Println(“El menor es C”)

If(B>C)

Println(“El intermedio es C”)

Println(“El menor es B”)

Println(“El intermedio es A”)

Println(“El menor es C”)

Else If(B>A && B>C)

If(A>C)

Println(“El mayor es B”)

2)

Si no se cumple la condición

Si no se cumple la condición

Println(“El intermedio es C”)

Println(“El menor es A”)

3)

Println(“El intermedio es A”)

Println(“El menor es B”)

If(A>B)

Else If(C>A && C>B)

Println(“El mayor es C”)

Si no se cumple la condición

Println(“El intermedio es B”)

Println(“El menor es A”)

Verifica si hay valores iguales y cuales son.

If(A==B && A==C)

Println(“Todos son iguales”)

4)

Si no se cumple la condición

Println(“A y B son iguales”)

Println(“El distinto es C”)

Else If(A==B)

Si no se cumple la condición

Println(“A y C son iguales”)

Println(“El distinto es B”)

Else If(A==C)

Si no se cumple la condición

Println(“By C son iguales”)

Println(“El distinto es A”)

Else If(B==C)

Como se distingue la parte donde se evalúa sin son iguales está separada de la parte donde se evalúan los valores, porque puede existir el caso de tener dos iguales y un distinto por lo que se aplicaría el if de mayores y menores y el if de iguales, para verificar cual es el mayor de todos y cuales son iguales.

* Relación con la teoría / Análisis:

Para poder resolver la principal dificultad tuve que plantear un primer filtro, en el cual la condición inicial tendría que definir si un valor es mayor a los otros dos, de esta forma se puede decidir, ya que si se cumple los valores siguientes son menores y ya solamente se comparan los valores para determinar cual es el valor de en medio y cual es el menor de todos (Imagen 1).

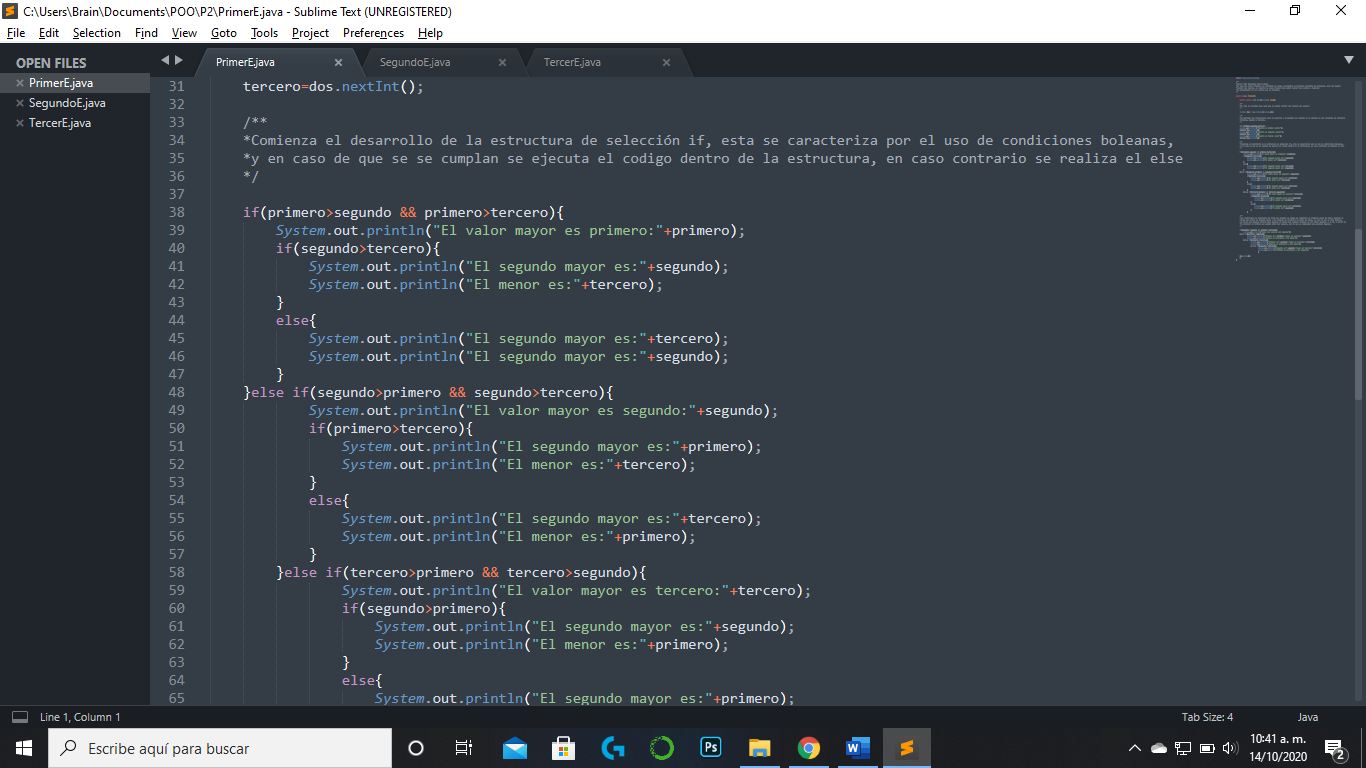


Imagen 1

Con esta condición inicial se tiene que aplicar un else if para poder aplicar la misma condición pero con el segundo termino y se repite el análisis anterior, y posteriormente se aplica al tercer término, en dado caso de que los valores sean iguales o que dos de ellos sean iguales se tiene el siguiente apartado del programa donde se compara que sean iguales. En este caso se verifica primero que todos sean iguales y si no se cumple se tiene que se comprueba de uno a uno, para poder determinar si existen pares iguales de valores (Imagen 2).

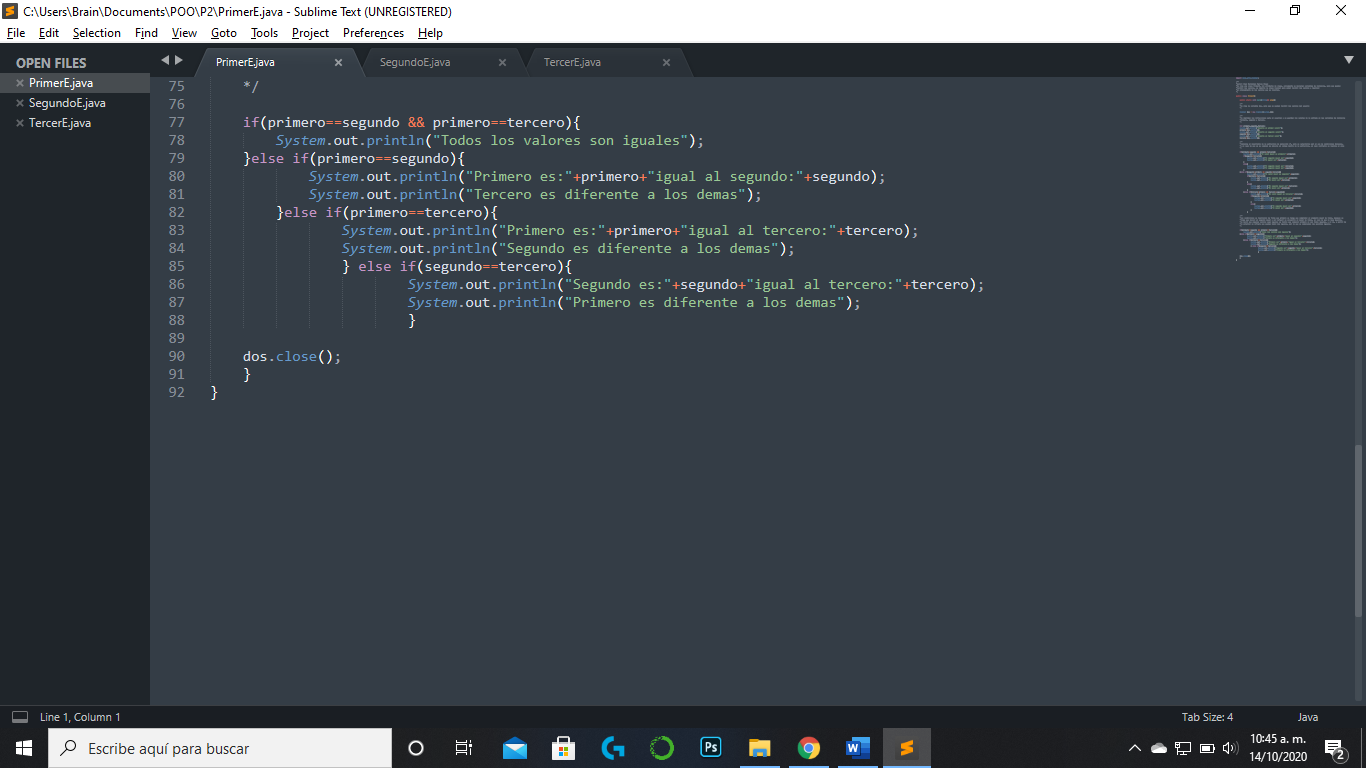
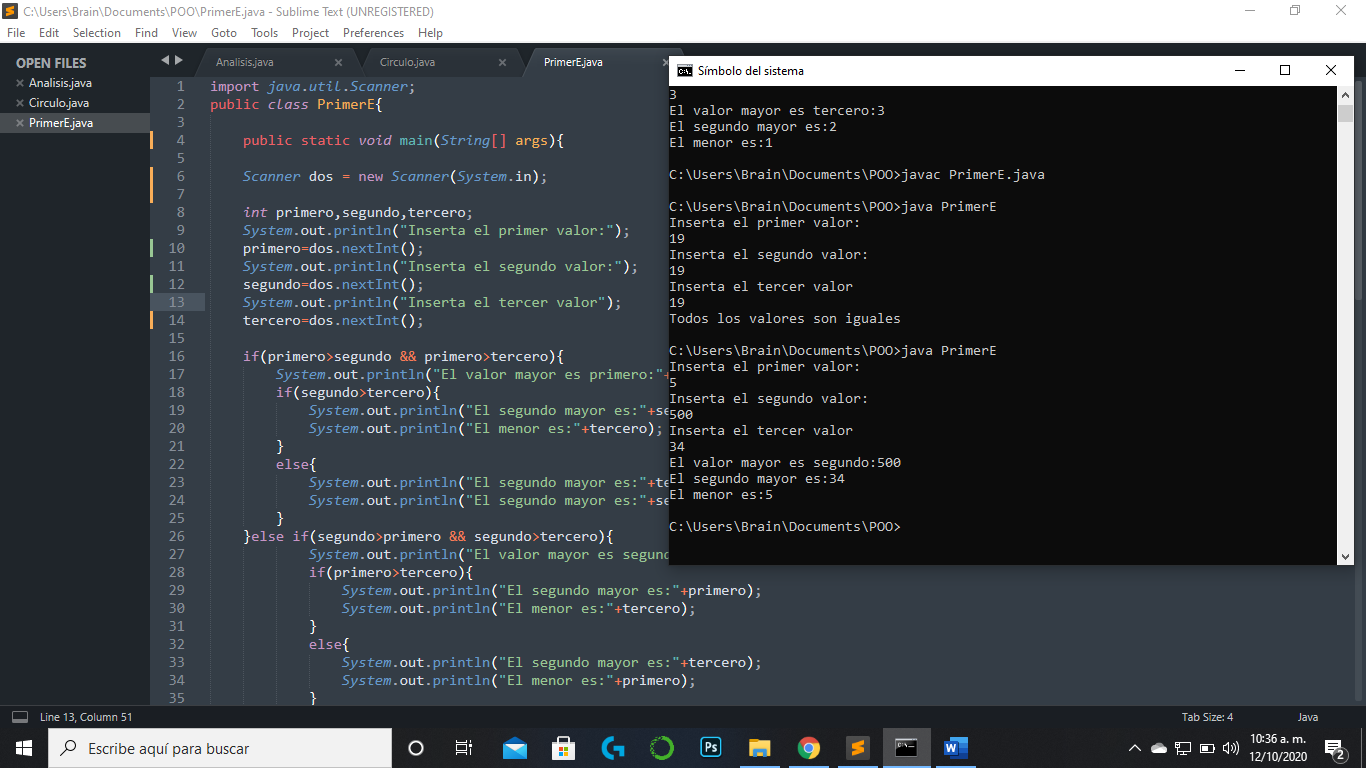


Imagen 2

* Evidencia de implementación



* Ejercicio 2:

El segundo ejercicio al igual que el primero consistía en comparar tres términos de los cuales se tenia que determinar el valor de mayor, el intermedio y el menor, para esto se utilizaría el operador ternario.

* Dificultades en el código

En un principio vi que el problema de la no se podría resolver con pocas variables, por lo que decidí implementar la estructura if-else y el operador ternario, esto porque tener variables de un solo uso no es eficiente porque al final no es un buen uso de la memoria, por ello es por lo que decidí en implementar ambas.

* Diagrama de funcionamiento:

Se realizan las operaciones del ternario y después se verifican los valores en el if

m1=(A>B)?A: B;

m2=(m1>C)?m1:C;

md=(B>C)?B:C;

If(m1==m2 && m1==A)

Println(“El mayor es A”)

1)

Se insertan tres valores:

A=10

B=15

C=5

If(md==B)

Println(“El intermedio es B”)

Println(“El menor es C”)

Println(“El intermedio es C”)

Println(“El menor es B”)

Si no se cumple la condición

m1=(A>B)?A: B;

m2=(m1>C)?m1:C;

md=(B>C)?B:C;

If(md>A)

Else If(m1==m2 && m1==C)

Println(“El mayor es B”)

2)

Println(“El intermedio es A”)

Println(“El menor es C”)

Si no se cumple la condición

Si no se cumple la condición

Println(“El intermedio es C”)

Println(“El menor es A”)

m1=(A>B)?A: B;

m2=(m1>C)?m1:C;

md=(B>C)?B:C;

If(md==A)

Else If(m1==m2 && m2==C)

3)

Println(“El mayor es C”)

Println(“El intermedio es A”)

Println(“El menor es B”)

Si no se cumple la condición

Println(“El intermedio es B”)

Println(“El menor es A”)

Verifica si hay valores iguales y cuales son.

m1=(A>B)?A: B;

m2=(m1>C)?m1:C;

Println(“Todos son iguales”)

If(m1==m2 && A==C)

4)

Si no se cumple la condición

Println(“A y B son iguales”)

Println(“El distinto es C”)

Else If(A==B)

Si no se cumple la condición

Println(“A y C son iguales”)

Println(“El distinto es B”)

Else If(A==C)

Si no se cumple la condición

Println(“By C son iguales”)

Println(“El distinto es A”)

Else If(B==C)

* Relación con la teoría / Análisis:

Para poder resolver lo que decidí fue crear tres variables secundarias, dos de ellas tendrían que almacenar el valor de la variable que cumpliera la condición, ya que el operador ternario toma el valor de dos elementos que se comparan dentro de este, así que si se cumple la condición toma el valor del primer elemento y si no se cumple toma el del segundo valor, contemplando la el proceso que lleva a cabo se tiene que es necesario el implementar dos variables, para comparar, entonces se realiza el mismo proceso del primer ejercicio, solo que ahora no se comparan en los if, solo se comparan los valores asignados y se ahí se verifican los valores (Imagen 1).

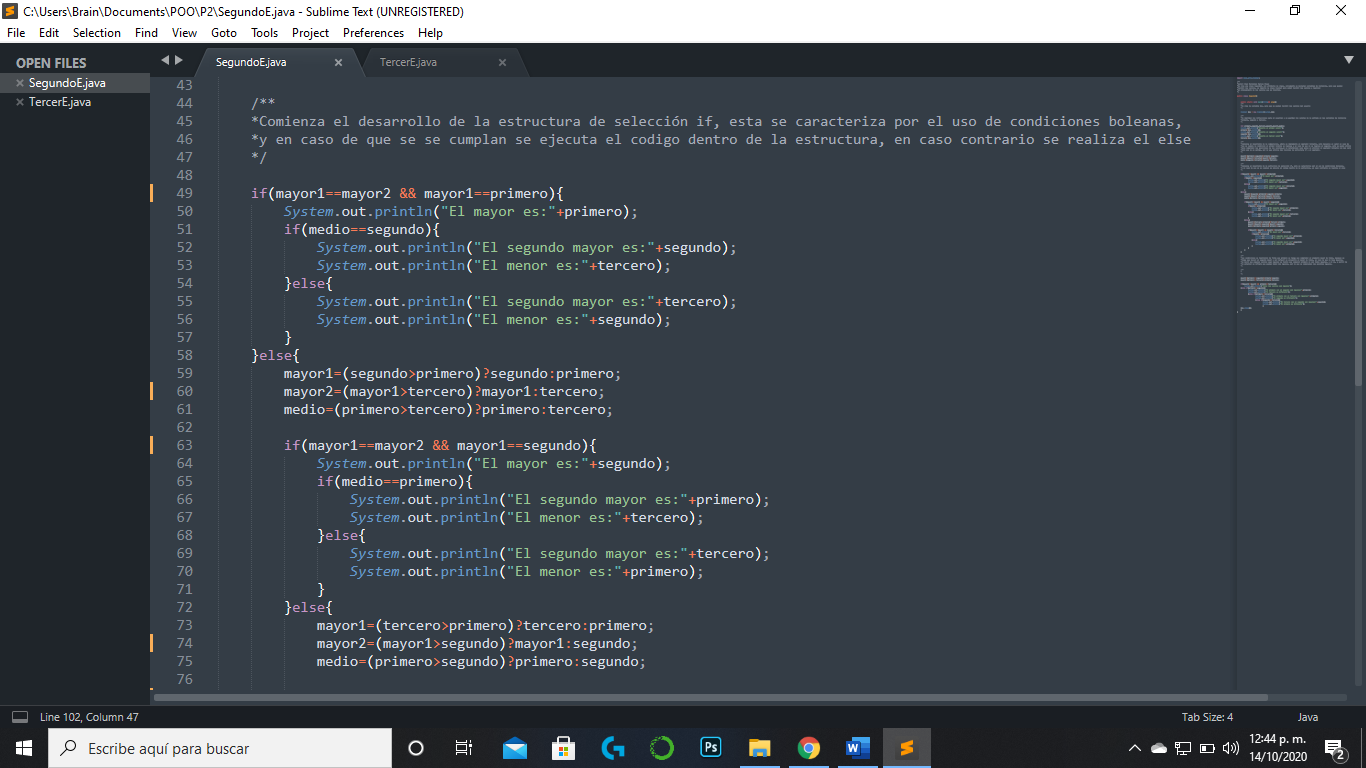
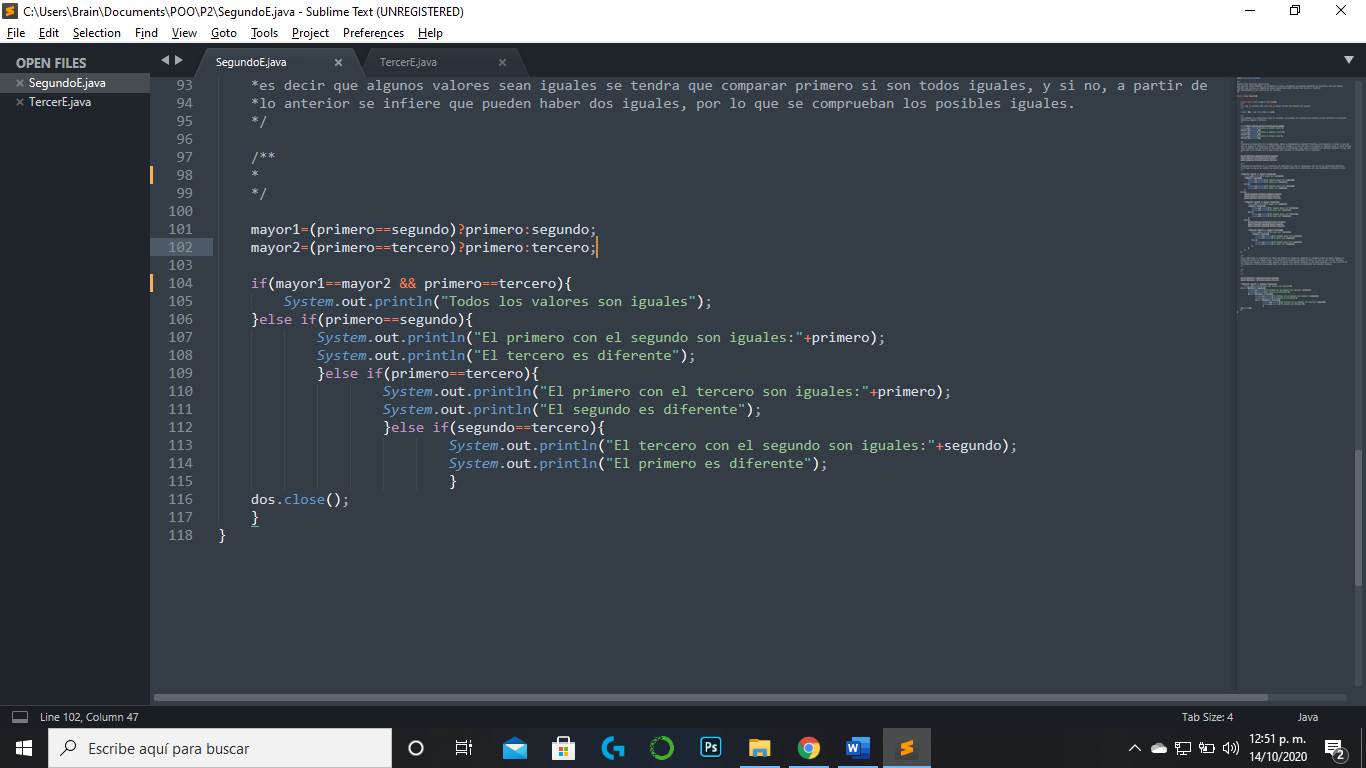


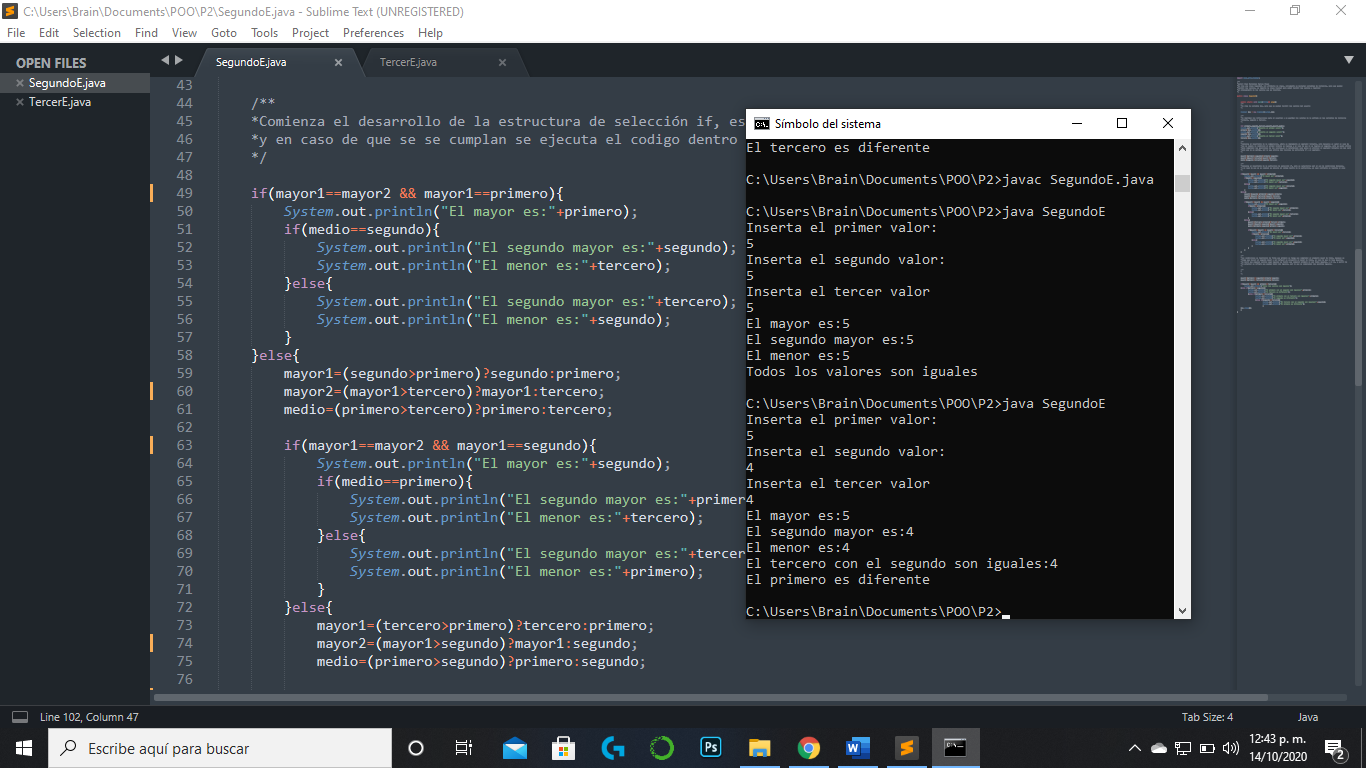
Imagen 1.

En cuanto a la parte de la comparación para determinar si son valores iguales se ejecuta la misma operación y únicamente se verifica que los mayores sean iguales, y también se hace una consideración para cuando hay dos valores que son iguales, en el caso de que el segundo y tercero pueden sean iguales y ser diferentes al primero, y en caso de tener solo un parámetro puede imprimir que todos son iguales por el valor que regresa el ternario. Entonces se agrega un segundo elemento de comprobación en cada if principal, porque puede haber casos como el anterior y regresar el mismo valor, pero no estar bien con respecto a las instrucciones de impresión (Ejercicio 2)



Ejercicio 2

* Evidencia de implementación



* Ejercicio 3:

El último ejercicio consistía en la elaboración de un programa que recibiera cadenas, tal que si recibe la cadena “stop”, el programa debe de detenerse la ejecución y si se recibe un “seguir ” puede continuar a la siguiente iteración, esto porque se debe de implementar con la estructura de repetición, do-while y for.

* Dificultades en el código

La mayor dificultad fue el encontrar una forma de comparar las cadenas, para esto tuve que buscar en un libro y encontré el método para poder comparar las cadenas, que en caso de que sea cierto o se cumpla la condición pues permite la elaboración de la siguiente línea de código contenida en caso de que se tenga en un condicional if. De ahí en fuera lo demás fue algo mas intuitivo, pero igual tuve que analizar donde colocar los breaks para detener cada ciclo en caso de que se insertara la cadena “stop”

* Diagrama de funcionamiento:

Scanner A=new Scanner(System.in)

do

String A=”Stop”

String B=”Seguir”

for(int i=0;i<1000;i++)

Println(“Inserta tu cadena”)

String E=A.next()

Lo primero que se hace dentro de la estructura es crear el objeto A de la clase Scanner, enseguida se crea el for, y se imprime la instrucción de insertar, después se comprueba que sea igual a “seguir”, sino se imprime lo insertado y al final se comprueba que sea igual a “stop”, y en ese caso el primer break rompe el for y el segundo el do.

If(E.equals(B))

Continue

Else

Println(“Insertaste:”+E)

If(E.equals(A))

Break

Break

* Relación con la teoría / Análisis:

Para poder recibir las cadenas fue necesario en crear tres variables de tipo String, una que recibiera las cadenas que el usuario fuera insertando, y las otras dos que se inicializaron con el propósito de poder compararlas en caso de que se presentara la entrada de un “seguir” o de un “stop”. De la misma forma que en los ejercicios anteriores fue necesario el importar la clase Scanner para poder leer los valores, y por medio del tipo String utilizar la comparación de cadenas equals, en caso de que se insertaran las cadenas condicionales (Imagen 1).

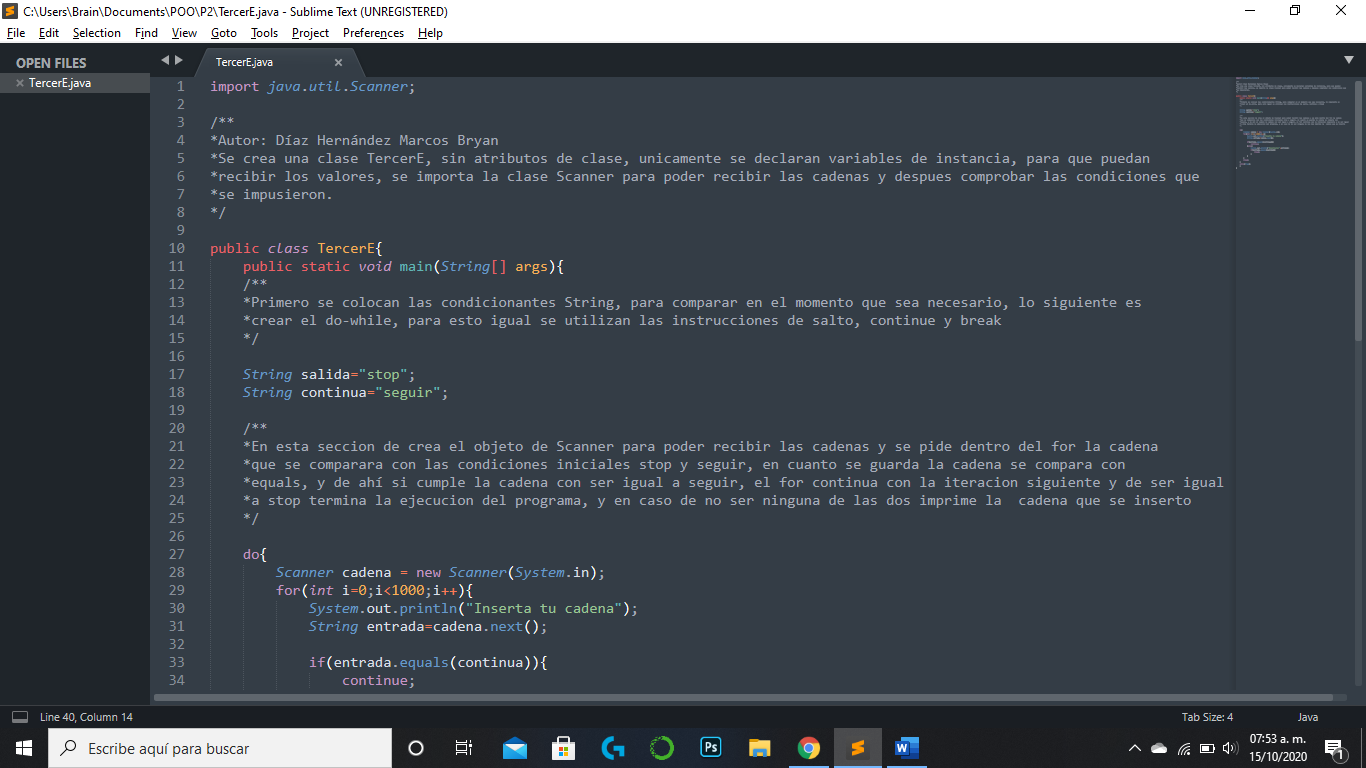


Imagen 1

Dentro del análisis realizado se tenían que realizar las implementaciones de un ciclo for y de un do-while, lo primero que hice fue el crear el do-while, para poder contener dentro de este el for, tal que el for se pueda repetir n veces mientras el do-while se cumpla, y únicamente se pueda romper cada estructura con un break (Imagen 2).

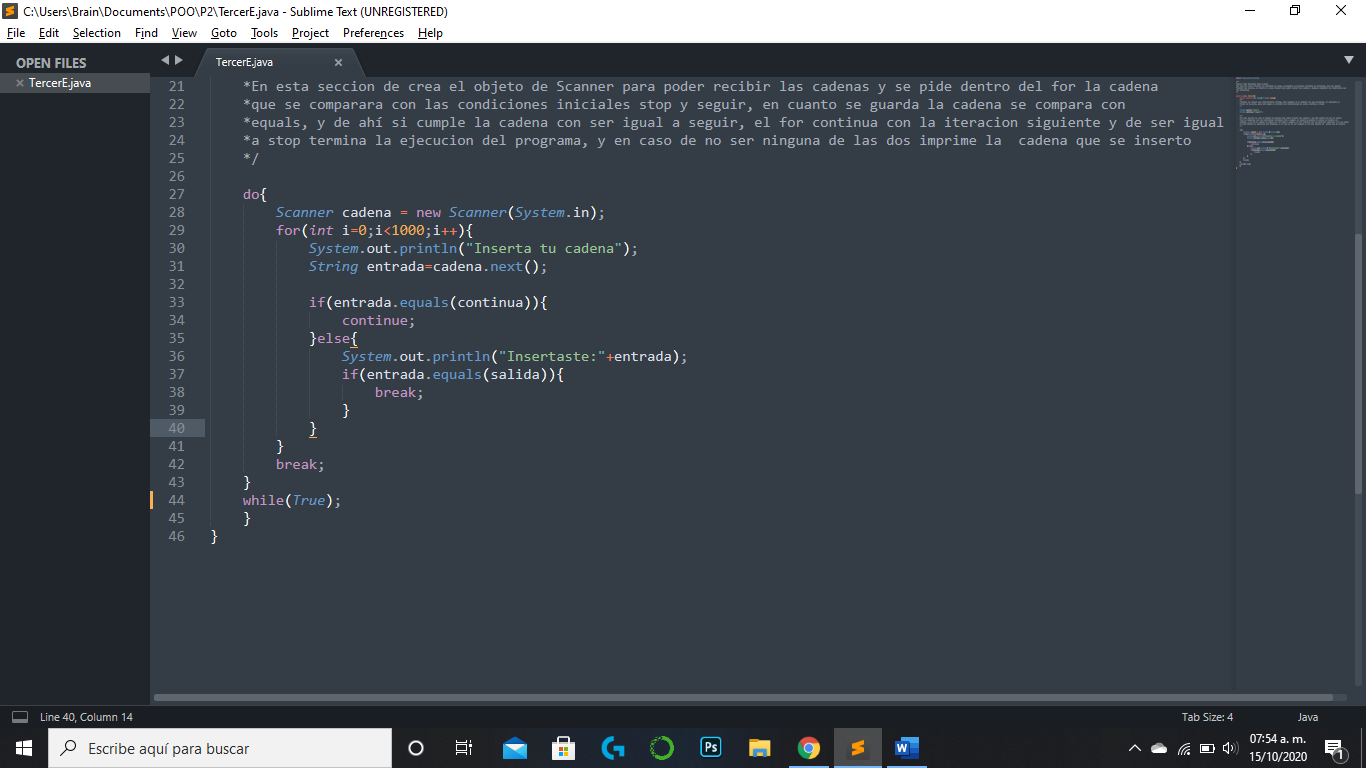
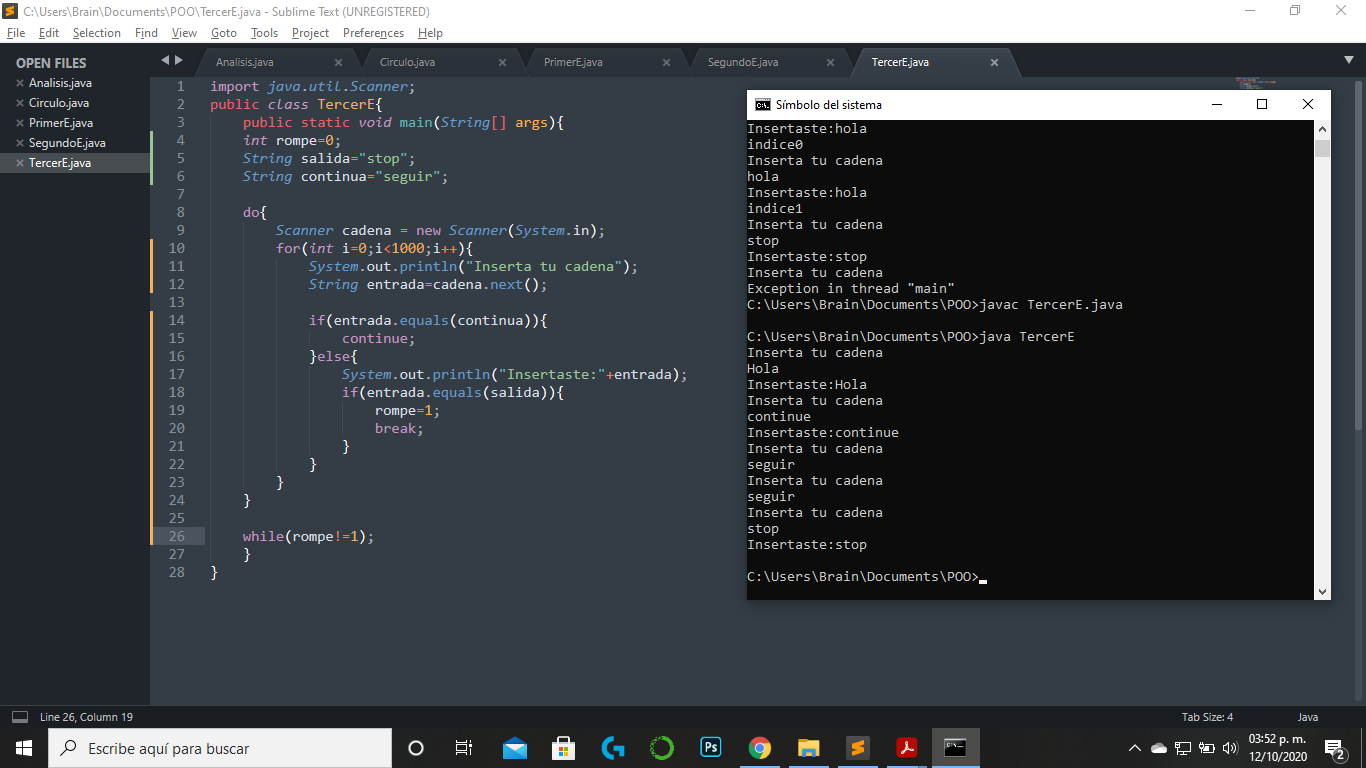


Imagen 2.

* Evidencia de implementación



**Conclusiones**

El objetivo de la practica el cual consistía en poder realizar programas que contuvieran estructuras repetitivas, de condición y la declaración de variables, se cumplió porque se pudieron implementar las estructuras de acuerdo con la sintaxis de java, e incluso se utilizaron clases adicionales como Scanner para poder insertar valores y ejecutar las instrucciones de acuerdo con lo que se solicitaba en cada uno de los ejercicios.

**Referencias:**

* Patrick N. (2005). Learning Java. USA: O’Reilly.