Instituto Tecnológico Superior de Calkiní en el Estado de Campeche.

Carrera:

Ingeniería en Sistemas Computacionales.

Asignatura:

Tópicos Avanzados de Programación.

Docente:

Ing. José Luis Lira Turriza.

Actividad:

Práctica 1.

Alumno:

Felipe de Jesús Balan Martín 7005.

Fecha:

12/04/2021

Ciclo escolar:

2020-2021P

PRÁCTICA No. 1.- Generación de Aplicaciones con UI y eventos

-INTRODUCCIÓN

Programación orientada a eventos.

La programación dirigida por eventos es un paradigma de la programación en el que tanto la estructura como la ejecución de los programas van determinados por los sucesos que ocurran en el sistema, definidos por el usuario o que ellos mismos provoquen.

Para entender la programación dirigida por eventos, podemos oponerla a lo que no es: mientras en la programación secuencial (o estructurada) es el programador el que define cuál va a ser el flujo del programa, en la programación dirigida por eventos será el propio usuario o lo que sea que esté accionando el programa, el que dirija el flujo del programa. Aunque en la programación secuencial puede haber intervención de un agente externo al programa, estas intervenciones ocurrirán cuando el programador lo haya determinado, y no en cualquier momento como puede ser en el caso de la programación dirigida por eventos.

El creador de un programa dirigido por eventos debe definir los eventos que manejarán su programa y las acciones que se realizarán al producirse cada uno de ellos, lo que se conoce como el administrador del evento. Los eventos soportados estarán determinados por el lenguaje de programación utilizado, por el sistema operativo e incluso por eventos creados por el mismo programador.

En la programación dirigida por eventos, al comenzar la ejecución del programa se llevarán a cabo las inicializaciones y demás código inicial y a continuación el programa quedará bloqueado hasta que se produzca algún evento. Cuando alguno de los eventos esperados por el programa tenga lugar, el programa pasará a ejecutar el código del correspondiente administrador de evento. Por ejemplo, si el evento consiste en que el usuario ha hecho clic en el botón de play de un reproductor de películas, se ejecutará el código del administrador de evento, que será el que haga que la película se muestre por pantalla.

Un ejemplo claro lo tenemos en los sistemas de programación Léxico y Visual Basic, en los que a cada elemento del programa (objetos, controles, etcétera) se le asignan una serie de eventos que generará dicho elemento, como la pulsación de un botón del ratón sobre él o el redibujado del control.

La programación dirigida por eventos es la base de lo que llamamos interfaz de usuario, aunque puede emplearse también para desarrollar interfaces entre componentes de Software o módulos de núcleos.

-OBJETIVO

El estudiante deberá de identificar los elementos de programación de eventos además del desarrollo de aplicaciones de software, en base al uso de diagramas UML y programación en el lenguaje Java.

LUGAR

AULA

-SEMANA DE EJECUCIÓN

SEMANA DOS (Parcial 1)

- MATERIAL Y EQUIPO

• Sistema Operativo

• Procesador de Textos

- • Software para el desarrollo de aplicaciones “eclipse”.
- • Cañón
- • Plumones
- • Pizarrón.

• Equipos de cómputo para todos los estudiantes de la asignatura.

- DESARROLLO DE LA PRÁCTICA.

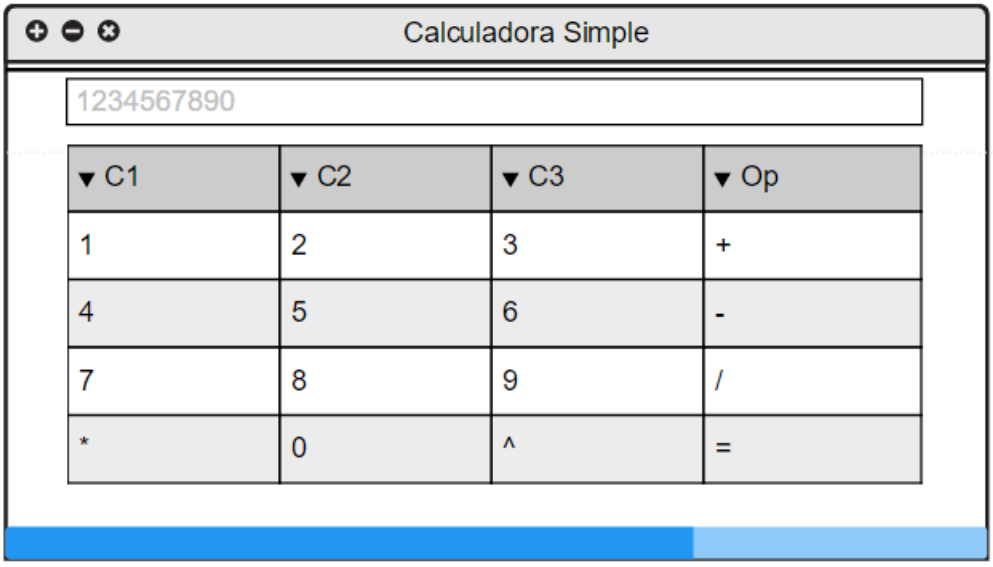


Figura 1. Calculadora Simple

Se desea manipular un conjunto de elementos para implementar la funcionalidad de las operaciones de una calculadora simple de acuerdo con lo siguiente:

1.- Se requiere realizar una aplicación de escritorio que realice los cálculos simples de números enteros de hasta 20 dígitos. En ella solamente se podrá introducir un número por vez seguido de una operación matemática; las operaciones matemáticas tienen prioridades para realizarse que deben aplicarse en la calculadora; el resultado final deberá mostrarse cuando se presione el operador =. En el caso de que la operación pueda realizarse se debe mostrar el resultado en el mismo espacio donde se introducen los números, en caso de algún error, éste se debe mostrar en el mismo espacio. Los errores que se deben considerar son los de división entre cero y desborde de memoria en caso de exceder los 20 dígitos. En esta tarea se deben escribir los casos de uso de la aplicación y los diagramas de caso de uso. También se debe representar la aplicación al menos en los diagramas UML de clases, de secuencia y de casos de uso.

2.- Una vez que se ha diseñado la aplicación se deberá desarrollar la aplicación dividida en dos partes: la interfaz de usuario y la lógica.

2.1 La interfaz gráfica de usuario (GUI por sus siglas en inglés) debe ser construido a partir del siguiente esqueleto:

```
public class CalculadoraGUI extends JFrame
{
// Los atributos pueden ser componentes de la interfaz como
//botones y campos de texto
// En el constructor podemos tener un parámetro que sea el
//título que le queremos dar al marco, dicho parámetro se lo
//pasaremos al constructor
public CalculadoraGUI (String nombre)
{
super (nombre);
// A continuación se definen y añaden las componentes de la
//interfaz
}
}
```

En esta etapa se deben inicializar cada uno de los elementos que para esta práctica serán 16 botones distribuidos dentro de un grid, y un campo de texto para introducir los números y mostrar los resultados. Configura los elementos de la siguiente manera:

Ventana principal con título “Calculadora Simple” de tamaño en pixeles de 500 x 400.

Campo de Texto con el texto inicial de “1234567890” en color gris.

16 botones en formato plano como se muestra en la figura 1, cuyo texto sean los números del 1 al 9, 0, +, -, *, / y ^, como los números naturales, el cero, suma, resta, multiplicación, división y potencia respectivamente. La primera fila contiene etiquetas que hacen referencia a las columnas C1, C2, C3 y Op., incluye éstos con ese texto.

Utiliza para la distribución general de la calculadora un administrador de componentes de tipo BorderLayout y para páneles internos un administrador GridLayout.

Implementa la visibilidad de la ventana en la creación de instancia de la clase. Al finalizar ejecuta tu aplicación y evidencie su resultado con una imagen de su escritorio.

2.2 La lógica de la aplicación deberá estar diseñada de acuerdo con un patrón. En nuestro caso utilizaremos el Modelo-Vista-Controlador (MVC) cuya vista ha sido desarrollada en el punto 2.1 y por lo que deberemos construir el Controlador y el Modelo de acuerdo con el diagrama de la figura 2.

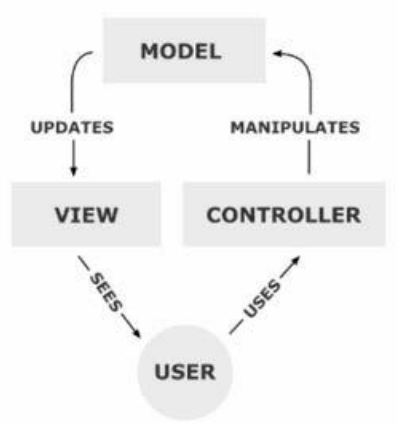


Figura 2. Patrón Modelo-Vista-Controlador (MVC).

2.2.1 El Modelo. Estará compuesto por la implementación de la lógica de negocio en la que se implementarán las operaciones tales como las CRUD o alguna otra necesaria como por ejemplo validarNumero; las clases de tipo Value Object(VO) en el que representamos los objetos con sus atributos y sus métodos set y get; y las clases de tipo data access object (DAO) que serán implementados como clases Hlp con un método action de cada tipo de operación de acceso a los datos.

2.2.2 El Controlador. Esta parte del patrón define la lógica de administración del sistema, establece la conexión entre la vista y el modelo.

Para la implementación del controlador se creará una clase Principal única y una clase Controlador por cada interfaz gráfica de la aplicación. Estos últimos deben contener la relación entre el modelo y las vistas.

En la clase Principal implementar el método main donde se instanciarán cada una de las vistas, del modelo y del controlador.

En la clase Controlador se deben tener instancias de la vista y de la lógica del negocio y métodos que permitan realizar las acciones definidas en las dos partes, como por ejemplo sumar o multiplicar.

Para terminar el armado del patrón, se modificará la vista incluyendo la implementación del ActionListener a la clase para obtener los datos de la interfaz gráfica y vaciarlos al objeto y viceversa a través de la implementación de los métodos dataToUserInterface y userInterfaceToData así como el método actionPerformed.

3. Realiza pruebas del prototipo con datos límite de acuerdo con el planteamiento del problema, evidencie esta etapa con el diseño de una prueba con el siguiente formato:

Nombre de la prueba:	Suma.
Descripción:	Se realizará la suma y resta de dos datos.
Datos de entrada de prueba:	1234+4321 y 4321-1234
Datos de salida esperado:	5555 y 3087

Nombre de la prueba:	Multiplicación.
Descripción:	Se multiplicarán dos números cualquiera.
Datos de entrada de prueba:	25*25
Datos de salida esperado:	625

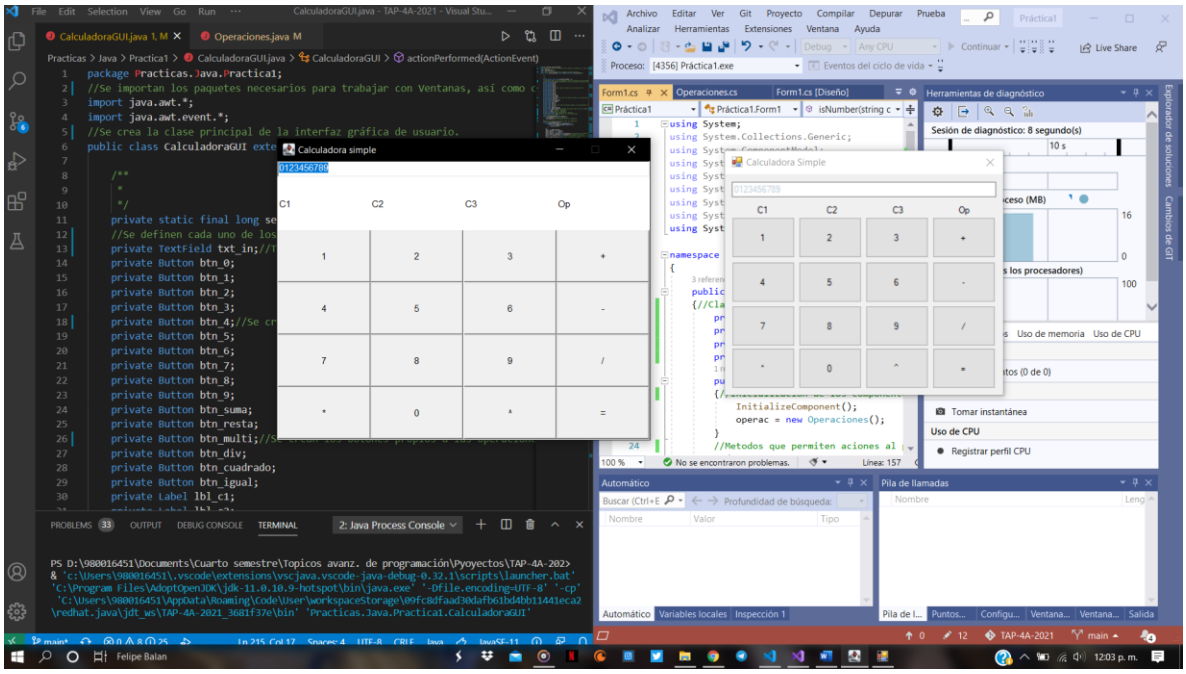
Nombre de la prueba:	División.
Descripción:	Se realizará la división entre 0.
Datos de entrada de prueba:	20/0
Datos de salida esperado:	Error.

Nombre de la prueba:	Potencia.
Descripción:	Se elevará un numero a una potencia cualquiera.
Datos de entrada de prueba:	25^3
Datos de salida esperado:	15625

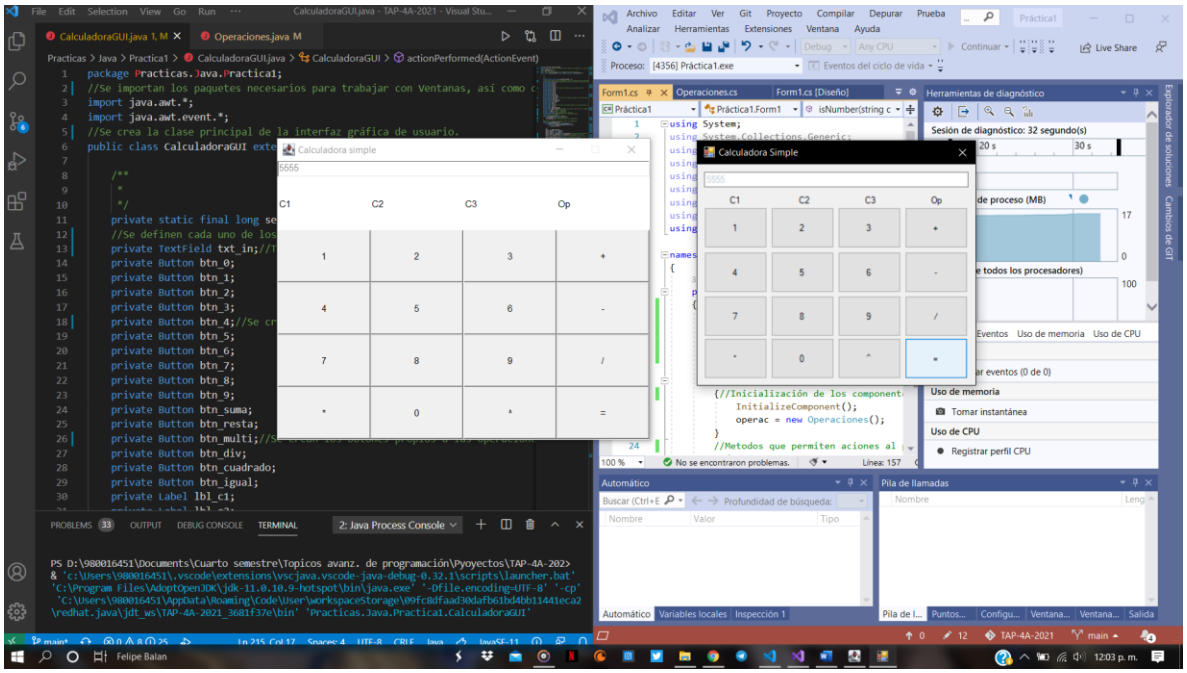
Nombre de la prueba:	Letras.
Descripción:	Se escribirán números, letras y cantidades mayores a 20 dígitos.
Datos de entrada de prueba:	123abcd123abcd123abcd
Datos de salida esperado:	Error.

4. Realice un informe de la práctica en donde se muestren los resultados y presente una discusión analizando los beneficios del patrón en la aplicación desarrollada.

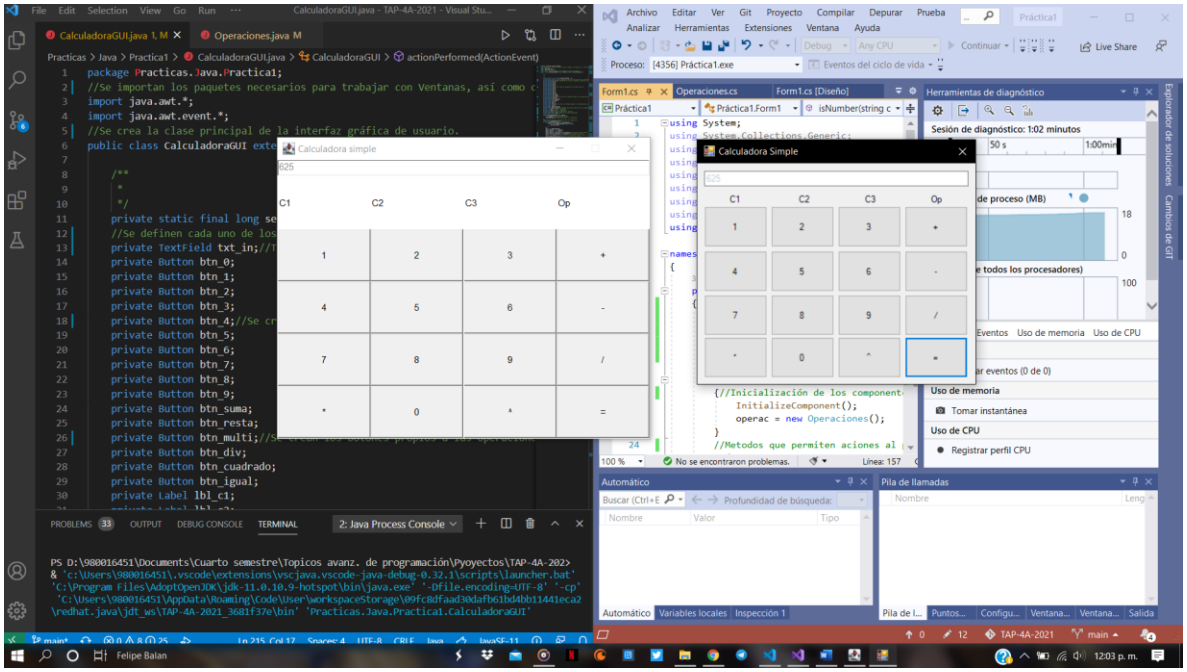
Algunos resultados de las operaciones propuestas anteriormente así como la interfaz gráfica de usuario tanto en Java como en C#.



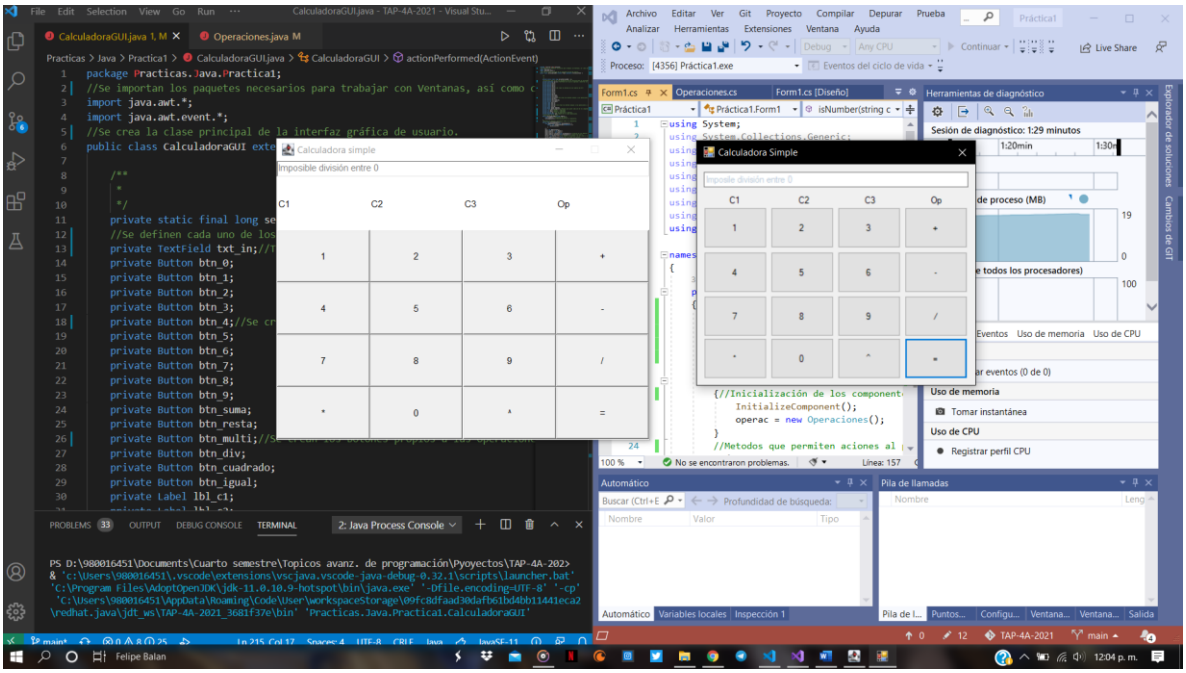
Ejecución del programa.



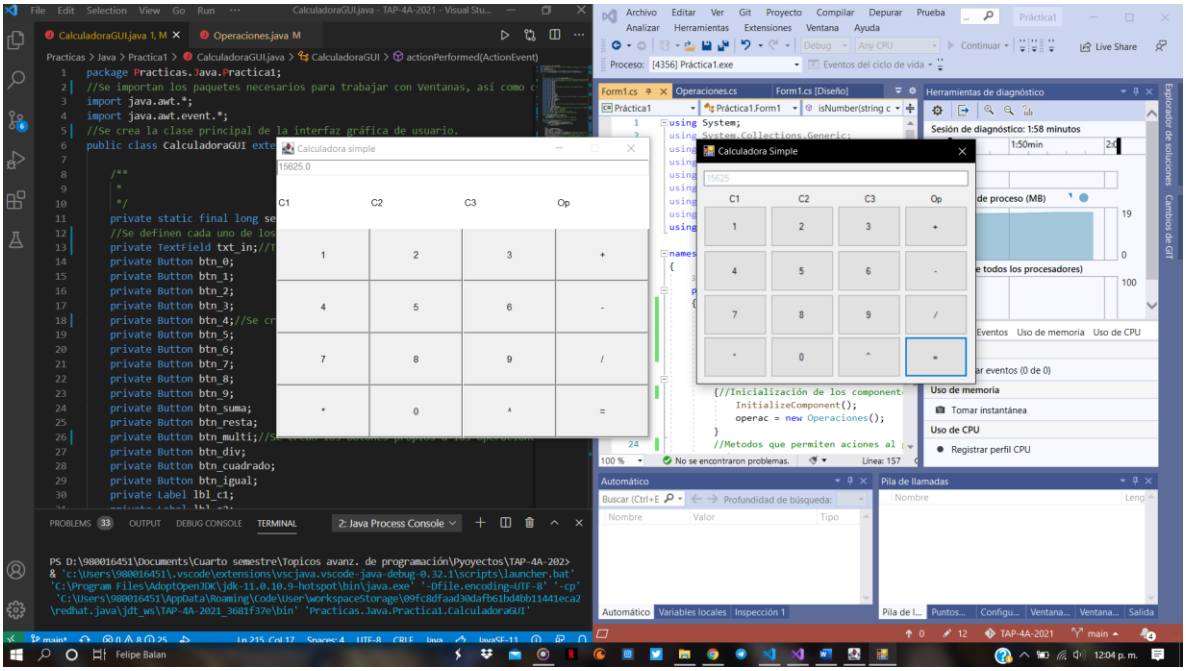
Primera prueba.



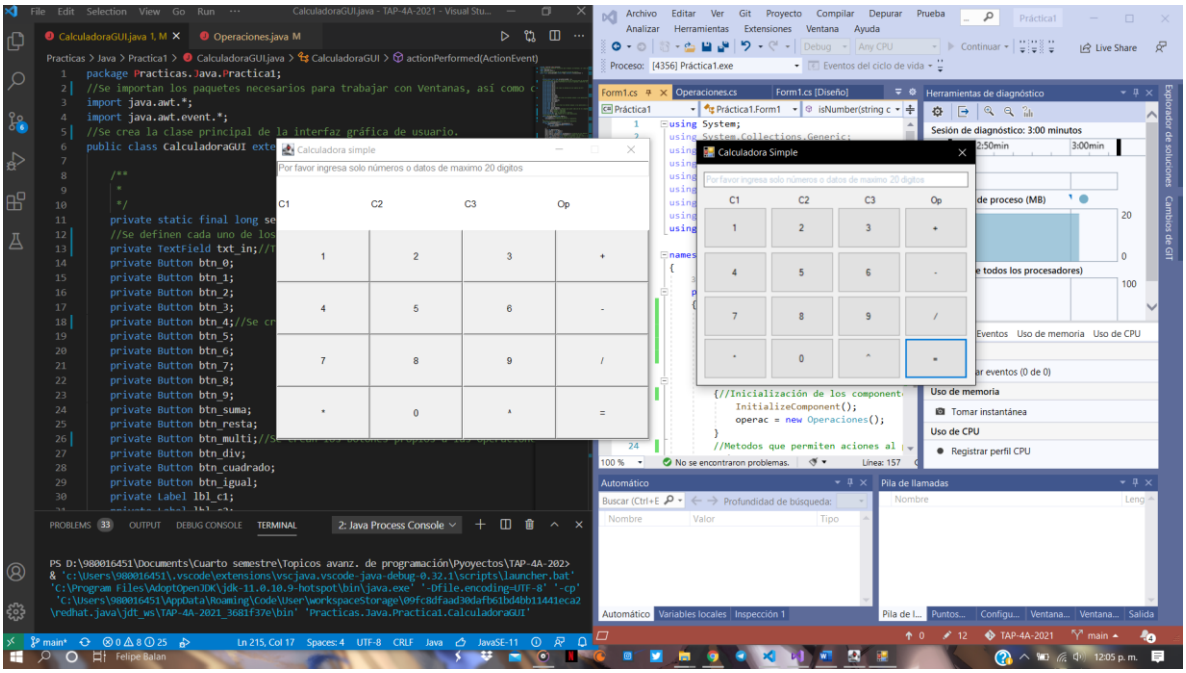
Segunda prueba.



Tercera prueba.



Cuarta prueba.



Quinta y última prueba.

Como se puede observar, en cada uno de los casos, al realizar la pruebas correspondientes, se pude observar que los resultados esperados fueron conseguidos, quedando un programa funcional, aunque si analizamos detenidamente cada una de las características que este posee, así como la manera en la que este fue estructurado, podemos observar que aún existe un gran margen de mejora, pero considerando los aprendizajes, la comprensión de las instrucciones, así como los temas vistos en clase, no queda más que claro que este es un programa bastante completo y como se observó, funcional.

Conclusión.

Luego de haber realizado esta práctica no me quedan más que aprendizajes claramente estupendos, debido a que como es costumbre al momento de la realización de las prácticas, estas cuentan con objetivos establecidos, las cuales de manera general es ayudar a practicar y reforzar los temas de manera acelerada, debido a que se aplican cosas vistas en clase.

En este caso me ayudó a comprender o mejor dicho, a seguir aprendiendo más aplicaciones que se le pueden dar o programas en los que pueden ser

implementados los eventos, los cuales sin lugar a duda presentan una gran mejora a la hora de programar, hablando de la parte visual, ya que con estos se pueden lograr un sinfín de programas con una interfaz gráfica de usuario llamativa y como ya menciono, sobre todo funcional que al fin y al cabo viene siendo lo importante.

Cabe mencionar que como se dijo anteriormente debido a la manera en la que se entendieron las instrucciones es la manera en la que se realizó la estructura del programa, esta es completamente funcional, aunque puede mejorarse, pero debido al tiempo y a la complejidad de los demás trabajos pendientes, se dejó de esta manera.

Entonces, la realización de esta práctica trajo sus problemas y sus complicaciones, pero aun así se presenta según lo visto y entendido en clases.

- EVALUACIÓN Y RESULTADOS

Se describe la forma de evaluar la práctica desarrollada mediante la entrega del Informe Técnico solicitado por el Profesor, el cual puede contener tablas, planos, prototipos, gráficas, diagramas o dibujos, observaciones, conclusiones, cuestionario y referencias.

-REFERENCIAS

Se presenta el listado de bibliografías utilizadas en el fundamento teórico y en el desarrollo de la práctica en sistema de referencia APA.

-ANEXOS

Diagramas UML.

Diagrama de Clases.

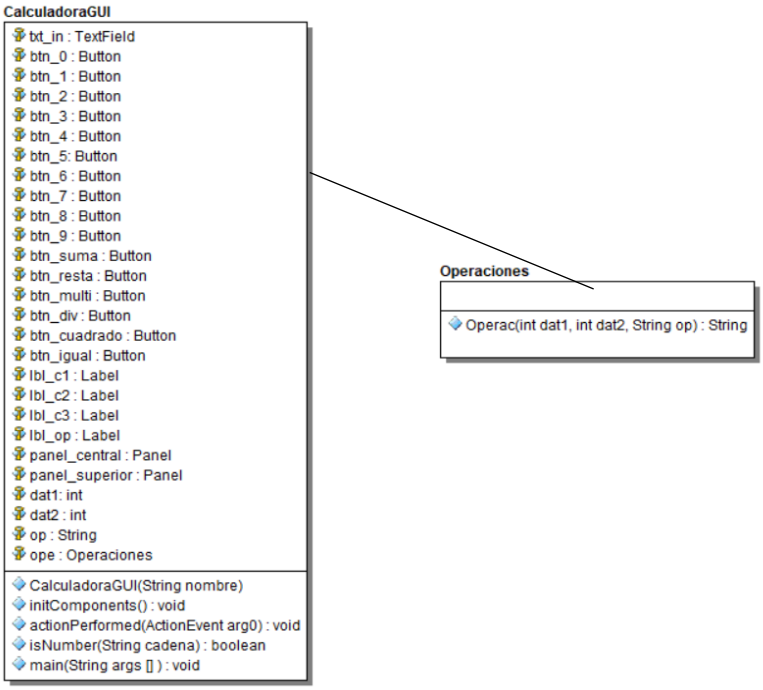


Diagrama de secuencia.

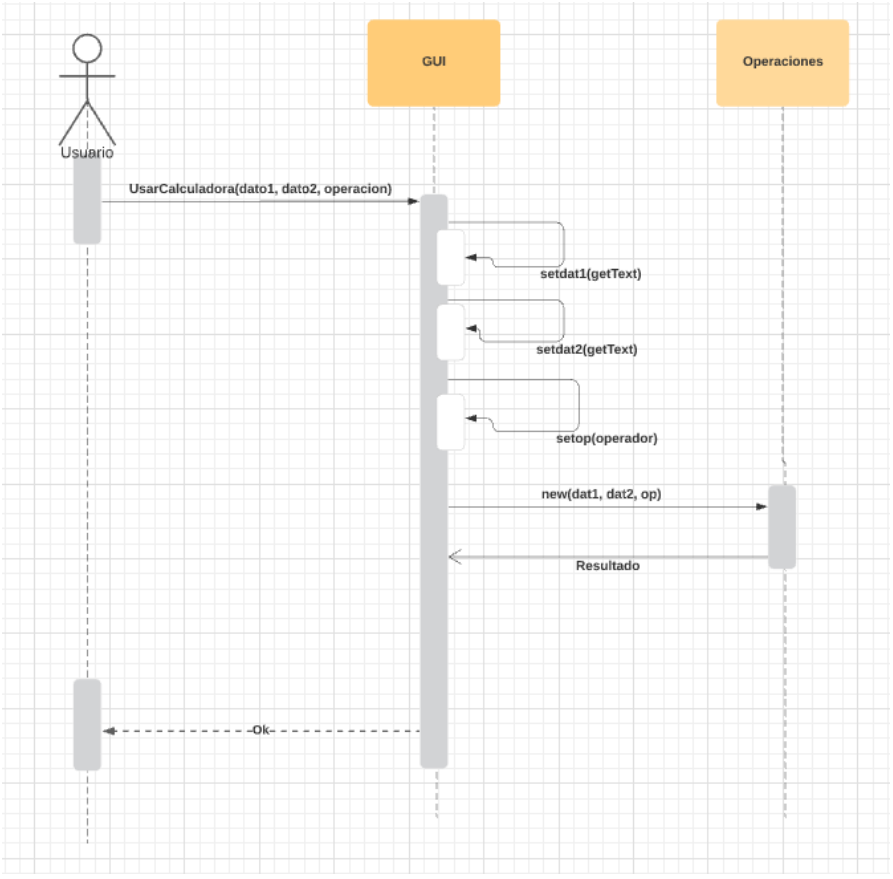
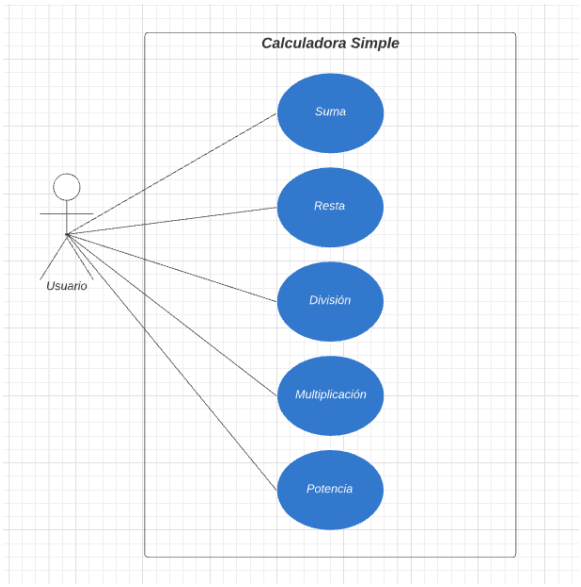
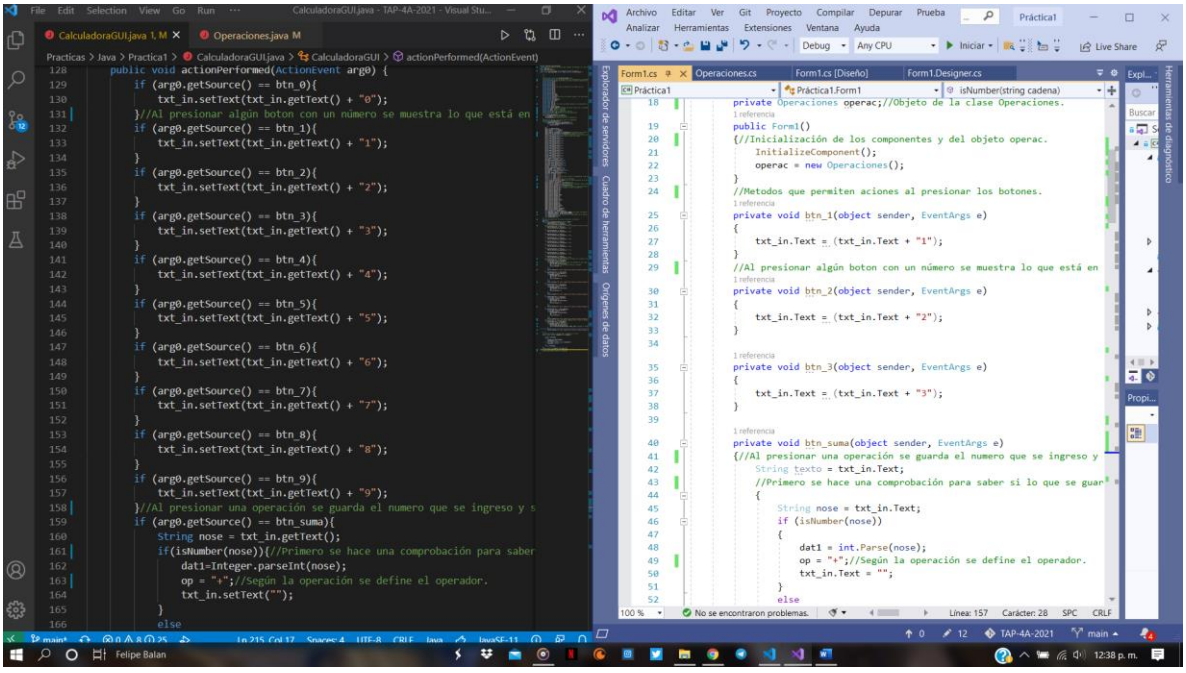
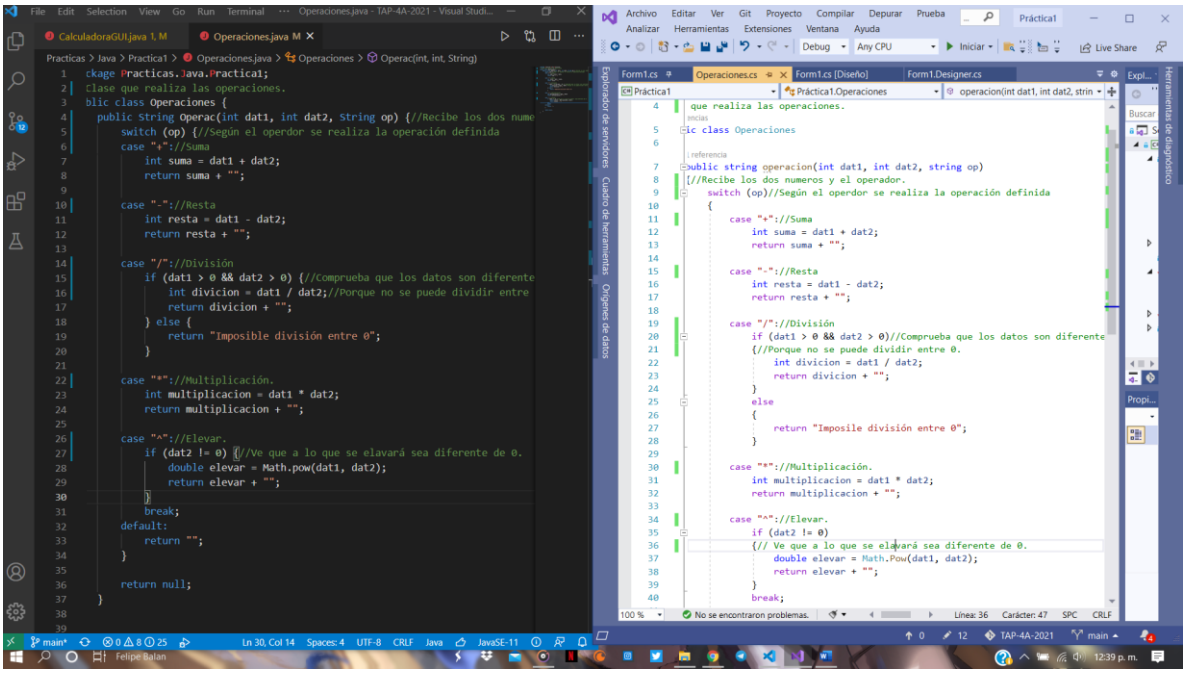


Diagrama de casos de uso.



Capturas del parte del código comentado.




CalculadoraGUI.java 1, M
Operaciones.java M
Practicas > Java > Practica1 > Operaciones.java > Operaciones > Operac(int, int, String)
ckage Practicas.Java.Practica1;
clase que realiza las operaciones.
blic class Operaciones {
public string Operac(int dat1, int dat2, String op) {//Recibe los dos nume
switch (op) {//Según el operdor se realiza la operación definida
case "+"://Suma
int suma = dat1 + dat2;
return suma + "";
case "-"://Resta
int resta = dat1 - dat2;
return resta + "";
case "/"://División
if (dat1 > 0 && dat2 > 0) {//Comprueba que los datos son diferente
int divicion = dat1 / dat2;//Porque no se puede dividir entre
return divicion + "";
} else {
return "Imposible división entre 0";
}
case "*"://Multiplicación.
int multiplicacion = dat1 * dat2;
return multiplicacion + "";
case "^"://Elevar.
if (dat2 != 0) {//Ve que a lo que se elavará sea diferente de 0.
double elevar = Math.pow(dat1, dat2);
return elevar + "";
}
break;
default:
return "";
}
return null;
}
Ln 30, Col 14
Spaces: 4
UTF-8
CRLF
Java
JavaSE-11
Archivo
Editar
Ver
Git
Proyecto
Compilar
Depurar
Prueba
Analizar
Herramientas
Extensiones
Ventana
Ayuda
Práctica1
Debug
Any CPU
Iniciar
Live Share
Form1.cs
Operaciones.cs
Form1.cs [Diseño]
Form1.Designer.cs
Práctica1
Práctica1.Operaciones
operacion(int dat1, int dat2, strin
que realiza las operaciones.
ic class Operaciones
referencia
ublic string operacion(int dat1, int dat2, string op)
//Recibe los dos numeros y el operador.
switch (op)//Según el operdor se realiza la operación definida
{
case "+"://Suma
int suma = dat1 + dat2;
return suma + "";
case "-"://Resta
int resta = dat1 - dat2;
return resta + "";
case "/"://División
if (dat1 > 0 && dat2 > 0)//Comprueba que los datos son diferente
//Porque no se puede dividir entre 0.
int divicion = dat1 / dat2;
return divicion + "";
}
else
{
return "Imposile división entre 0";
}
case "*"://Multiplicación.
int multiplicacion = dat1 * dat2;
return multiplicacion + "";
case "^"://Elevar.
if (dat2 != 0)
// Ve que a lo que se elavará sea diferente de 0.
double elevar = Math.Pow(dat1, dat2);
return elevar + "";
}
break;
100 %
No se encontraron problemas.
Línea: 36
Carácter: 47
SPC
CRLF
TAP-4A-2021
main
Explorador de servidores
Cuadro de herramientas
Orígenes de datos
Herramientas de diagnóstico
Buscar
Propi...
Felipe Balan
12:39 p. m.