Práctica AIS

Ingeniería Informática

**David Sánchez Dueñas**

**Sergio Pérez Casquero**

**Jaime Muñoz Aparicio**

**Sergio Lázaro Matesanz**

Contenido

[ **Planificación** 1](#_Toc448605831)

[ **Plan de proyecto** 1](#_Toc448605832)

[**Resumen del proyecto** 1](#_Toc448605833)

[**Productos entregables** 1](#_Toc448605834)

[**Hitos alcanzables** 1](#_Toc448605835)

[**Método de trabajo** 1](#_Toc448605836)

[**Revisión de planificación** 1](#_Toc448605837)

[**Personal del proyecto** 2](#_Toc448605838)

[**Tareas del proyecto** 2](#_Toc448605839)

[**Calendario** 5](#_Toc448605840)

[ **Plan de gestión de configuración de software** 6](#_Toc448605841)

[**Introducción** 6](#_Toc448605842)

[**Gestión y recursos de la GCS** 6](#_Toc448605843)

[ **Requisitos** 9](#_Toc448605844)

[ **Especificación de requisitos** 9](#_Toc448605845)

[**Requisitos de interfaz** 9](#_Toc448605846)

[**Requisitos de jerarquía de operaciones** 9](#_Toc448605847)

[**Requisitos generales** 9](#_Toc448605848)

[ **Diseño** 10](#_Toc448605849)

[ **Diseño de interfaz** 10](#_Toc448605850)

[**Botones numéricos** 10](#_Toc448605851)

[**Botón “=”** 10](#_Toc448605852)

[**Botón “.”** 10](#_Toc448605853)

[**Botón “CE”** 10](#_Toc448605854)

[**Botón “C”** 11](#_Toc448605855)

[**Botón “(“** 11](#_Toc448605856)

[**Botón “)”** 11](#_Toc448605857)

[**Botón “X” y “/”** 11](#_Toc448605858)

[**Botón “+”** 11](#_Toc448605859)

[**Botón “-“** 11](#_Toc448605860)

[ **Diseño de método suma** 11](#_Toc448605861)

[ **Diseño de método resta** 11](#_Toc448605862)

[ **Diseño de método “multiplicación”** 11](#_Toc448605863)

[ **Diseño de método “división”** 12](#_Toc448605864)

[ **Diseño de método “tramoParentesis”** 12](#_Toc448605865)

[ **Diseño de método “dividirOperacionesSimples”** 12](#_Toc448605866)

[ **Código fuente** 12](#_Toc448605867)

[ **Pruebas y resultados** 12](#_Toc448605868)

[** Caso de prueba A** 13](#_Toc448605869)

[** Caso de prueba B** 13](#_Toc448605870)

[** Caso de prueba C** 14](#_Toc448605871)

[** Caso de prueba D** 14](#_Toc448605872)

[** Caso de prueba E** 14](#_Toc448605873)

[** Caso de prueba F** 14](#_Toc448605874)

[** Caso de prueba G** 15](#_Toc448605875)

[** Caso de prueba H** 15](#_Toc448605876)

[** Caso de prueba I** 15](#_Toc448605877)

[** Caso de prueba J** 16](#_Toc448605878)

[** Caso de prueba K** 16](#_Toc448605879)

[** Caso de prueba L** 16](#_Toc448605880)

[ **Manual de usuario** 17](#_Toc448605881)

[ **Cambios realizados en la documentación** 18](#_Toc448605882)

# **Planificación**

La práctica de la asignatura de Ampliación de Ingeniería del Software pretende poner en práctica el conjunto de conocimientos implicados en dicha asignatura. En la práctica propuesta para esta asignatura, hemos partido de un código Java que implementa una calculadora de funcionalidad muy básica.

La meta era diseñar una calculadora, que fuera capaz de evaluar una expresión matemática teniendo en cuenta la jerarquía de operaciones, y su interfaz, pero realizando todos los pasos previos y posteriores a la codificación (captura de requisitos, asignación de tareas, pruebas de caja negra, etc.)

La planificación es el paso previo a la codificación. Una buena planificación suele dar buenos resultados. En los siguientes apartados se especificarán estos pasos de forma detallada.

## **Plan de proyecto**

### **Resumen del proyecto**

Como ya hemos dicho anteriormente, la meta a conseguir es una calculadora que sea capaz de evaluar una expresión matemática teniendo en cuenta la jerarquía de operaciones. Las operaciones que hará la calculadora se limitan a sumar, restar, multiplicar y dividir.

### **Productos entregables**

Los productos que hay que entregar son:

* Una presentación del proyecto realizado
* Código y documentación del proyecto

### **Hitos alcanzables**

Los hitos que han de ser alcanzados son:

* Se ha de realizar operaciones sin problema
* Se ha de respetar la jerarquía de operaciones
* Se debe poder corregir y prevenir errores
* Se debe poder poner números decimales
* Se debe de enlazar interfaz y código correctamente

### **Método de trabajo**

El método de trabajo que se ha llevado a cabo ha sido el trabajo en equipo. El organigrama de equipo que hemos adoptado es el descentralizado democrático.

Aunque principalmente hemos trabajado en equipo, en ocasiones hemos optado por un reparto de tareas, siempre y cuando estas fuesen sencillas.

### **Revisión de planificación**

La planificación del proyecto se ha tenido en cuenta durante el trabajo. Se han seguido los pasos que marcados, salvo en los casos que nos hemos visto obligados a hacer cambios, los cuales están reflejados en un apartado de este documento.  
Como, principalmente, hemos trabajado en equipo, la revisión del cumplimiento de la planificación ha sido realizada por todos los integrantes del equipo.

### **Personal del proyecto**

Los integrantes del equipo son:

* David Sánchez Dueñas
* Sergio Pérez Casquero
* Jaime Muñoz Aparicio
* Sergio Lázaro Matesanz

### **Tareas del proyecto**

Las tareas que hemos llevado a cabo son:

1. Identificación de requisitos
2. Elaboración de Plan de proyecto
3. Elaboración de Plan de Gestión de Configuración
4. Documentar requisitos
5. Estudio de las reglas de precedencia y el lenguaje Java
6. Análisis previo de funcionalidades
7. Documentar funcionamiento de clases
8. Documentar funcionalidad interfaz
9. Diseño método “suma”
10. Diseño método” resta”
11. Diseño método “multiplicación”
12. Diseño método “división”
13. Diseño método “tramoParentesis”
14. Diseño método “dividir en operaciones simples”
15. Diseño de Interfaz
16. Implementar método “suma”
17. Implementar método “resta”
18. Implementar método “multiplicación”
19. Implementar método “división”
20. Implementar método “comprobar operadores”
21. Implementar método “ dividir en operaciones simples”
22. Implementación de interfaz
23. Revisar si código de funcionalidad es comprensible ( comentarios, tabulaciones, modulado)
24. Revisar si código de interfaz es comprensible ( comentarios, tabulaciones, modulado)
25. Realización de pruebas y corrección de errores en la funcionalidad
26. Realización de pruebas y corrección de errores en la interfaz
27. Asociar interfaz con funcionalidad
28. Realización de pruebas y corrección de errores en general
29. Recopilar documentación existente y estructurarla
30. Entrega de proyecto y documentación

La relación entre las tareas viene dada por una tabla (que refleja la precedencia de tareas, al duración de las mimas y los recursos asociados a ellas) y por un diagrama de Pert (que refleja la precedencia de tareas de forma gráfica y sirve para identificar el camino crítico

***Tabla de precedencia de tareas***

|  |  |  |  |
| --- | --- | --- | --- |
| Tareas | Precedencia de tareas | Tiempo (días) | Recursos humanos |
| A |  | 3 | Todos |
| B | A | 3 | Todos |
| C | B | 1 | Todos |
| D | C | 1 | Sergio L. |
| E | D | 1 | Todos |
| F | E | 1 | Todos |
| G | F | 2 | Sergio L. |
| H | F | 1 | Sergio L. |
| I | G | 1 | Jaime |
| J | G | 1 | David |
| K | G | 1 | Sergio P. |
| L | G | 1 | Jaime |
| M | G | 2 | Sergio L.; Sergio P. |
| N | G | 3 | David, Jaime |
| O | H | 1 | Sergio P.; Jaime |
| P | I | 1 | Jaime |
| Q | J | 1 | Sergio L. |
| R | K | 1 | David |
| S | L | 1 | Sergio P. |
| T | M | 1 | Jaime; Sergio P. |
| U | N | 4 | Sergio L.; David |
| V | O | 1 | David; Sergio L. |
| W | P,Q,R,S,T,U | 1 | Sergio P. |
| X | V | 1 | Jaime |
| Y | W | 3 | Sergio L. |
| Z | X | 1 | David |
| AA | Y | 2 | Sergio P. |
| BB | AA | 2 | Jaime |
| CC | BB | 2 | Sergio L. |
| DD | CC | 1 | Todos |

Hemos establecido que en un día se trabajan 2 horas.

***Tabla de holguras***

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Tarea | tij | TEi | TLi | TEj | TLj | Holgura Total | Holgura Libre | Holgura Independiente |
| **II** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **A** | 3 | 0 | 0 | 3 | 3 | 0 | 0 | 0 |
| **B** | 3 | 3 | 3 | 6 | 6 | 0 | 0 | 0 |
| **C** | 1 | 6 | 6 | 7 | 7 | 0 | 0 | 0 |
| **D** | 1 | 7 | 7 | 8 | 8 | 0 | 0 | 0 |
| **E** | 1 | 8 | 8 | 9 | 9 | 0 | 0 | 0 |
| **F** | 1 | 9 | 9 | 10 | 10 | 0 | 0 | 0 |
| **G** | 2 | 10 | 10 | 12 | 12 | 0 | 0 | 0 |
| **H** | 1 | 10 | 10 | 11 | 19 | 8 | 0 | 0 |
| **I** | 1 | 12 | 12 | 13 | 18 | 5 | 0 | 0 |
| **J** | 1 | 12 | 12 | 13 | 18 | 5 | 0 | 0 |
| **K** | 1 | 12 | 12 | 13 | 18 | 5 | 0 | 0 |
| **L** | 1 | 12 | 12 | 13 | 18 | 5 | 0 | 0 |
| **M** | 2 | 12 | 12 | 14 | 16 | 2 | 0 | 0 |
| **N** | 3 | 12 | 12 | 15 | 15 | 0 | 0 | 0 |
| **O** | 1 | 11 | 19 | 12 | 20 | 8 | 0 | -8 |
| **P** | 1 | 13 | 18 | 19 | 19 | 5 | 5 | 0 |
| **Q** | 1 | 13 | 18 | 19 | 19 | 5 | 5 | 0 |
| **R** | 1 | 13 | 18 | 19 | 19 | 5 | 5 | 0 |
| **S** | 1 | 13 | 18 | 19 | 19 | 5 | 5 | 0 |
| **T** | 3 | 14 | 16 | 19 | 19 | 2 | 2 | 0 |
| **U** | 4 | 15 | 15 | 19 | 19 | 0 | 0 | 0 |
| **V** | 1 | 12 | 20 | 13 | 21 | 8 | 0 | -8 |
| **W** | 1 | 19 | 19 | 20 | 20 | 0 | 0 | 0 |
| **X** | 1 | 13 | 21 | 14 | 22 | 8 | 0 | -8 |
| **Y** | 3 | 20 | 20 | 23 | 23 | 0 | 0 | 0 |
| **Z** | 1 | 14 | 22 | 23 | 23 | 8 | 8 | 0 |
| **AA** | 2 | 23 | 23 | 25 | 25 | 0 | 0 | 0 |
| **BB** | 2 | 25 | 25 | 27 | 27 | 0 | 0 | 0 |
| **CC** | 2 | 27 | 27 | 29 | 29 | 0 | 0 | 0 |
| **DD** | 1 | 29 | 29 | 30 | 30 | 0 | 0 | 0 |

Holgura total de una actividad HTij = TLj – TEi – tij.

Holgura libre de una actividad HLij = TEj – TEi – tij.

Holgura independiente de una actividad HIij = TEj – TLi – tij.

***Diagrama de Pert***

C:\Users\sergio\Desktop\Nueva carpeta\PertAmarillo (1).png

***Diagrama de Pert – Camino crítico***

C:\Users\sergio\Desktop\Nueva carpeta\PertCritico (1).png

### **Calendario**

La evolución del proyecto se puede ver en el calendario que se adjunta al final de este párrafo (si se mueve el Excel de la carpeta en la que se encuentra, se tendrá que acceder a él manualmente).

<Calendario.xlsx>

## **Plan de gestión de configuración de software**

### **Introducción**

Los elementos de configuración que hemos identificado son:

* **Calendario**
* **Plan de proyecto**
* **Especificación de requisitos**
* **Diseños**
* **Implementaciones**
* **Casos de prueba**
* **Documentación**
* **Manual de usuario**

Las líneas base (LB) que hemos establecido son:

* **LB Planificación:** al finalizar el plan de proyecto. En esta línea base se encuentran la especificación de tareas, el calendario, el personal del proyecto, el método de trabajo, los hitos alcanzables, los productos entregables y la revisión de la planificación.
* **LB Asignación:** al finalizar el diseño preliminar del proyecto. En esta línea base se encuentran los bocetos realizados y las primeras ideas de cómo iba a funcionar el proyecto.
* **LB Diseño:** al finalizar el estudio del diseño preliminar y de los problemas intrínsecos del proyecto que van surgiendo. En esta línea base se encuentran todo el estudio de Java y de las restricciones propias del proyecto, además de un diseño más detallado.
* **LB Producto:** al finalizar la codificación. En esta línea base se encuentra todo el código y las pruebas que se han realizado con el mismo, junto con su resultado.
* **LB Explotación:** al finalizar la implantación. En esta línea base se encuentra el manual de usuario del proyecto.

### **Gestión y recursos de la GCS**

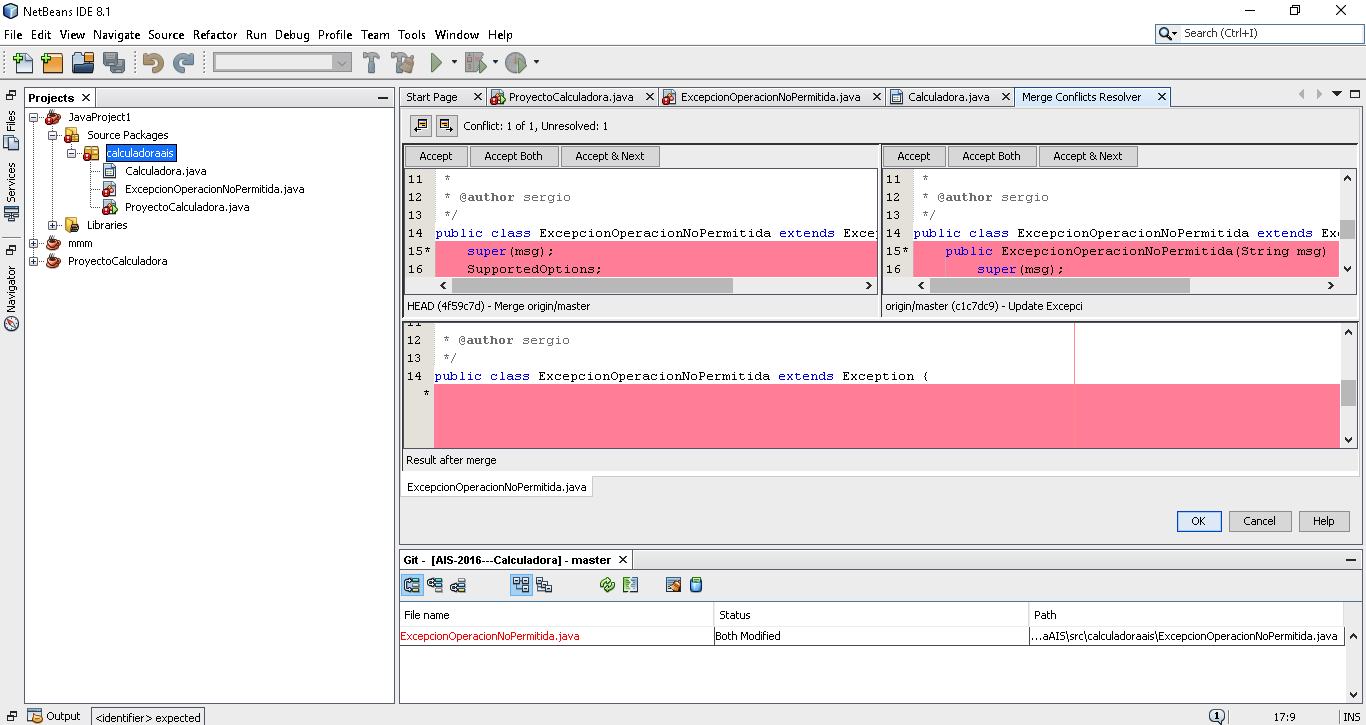
Los miembros responsables de la gestión de configuración son todos los integrantes del equipo.

Puesto que hemos utilizado una herramienta de control de versiones, llamada “GitHub”, basada en “Copiar-Modificar-Mezclar” sobre un repositorio centralizado, las actividades de “control de cambios” y “almacenamiento de copias de seguridad” han sido realizadas por todos.

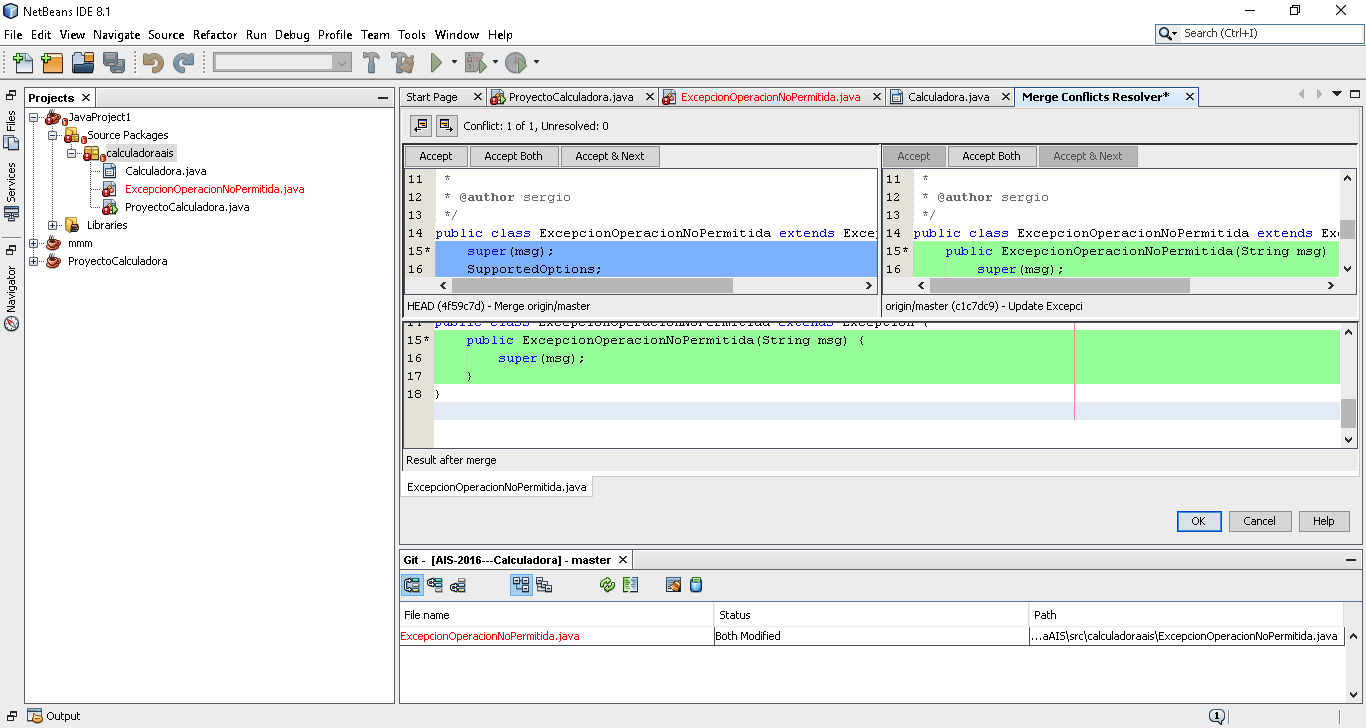
La dirección del repositorio es: <https://github.com/Sherlam/AIS-2016---Calculadora.git>

Los errores que se han producido por el solapamiento de versiones, se han resuelto editando manualmente las líneas conflictivas:

* **Error en el código:**



Las líneas que originan el error están en rojo

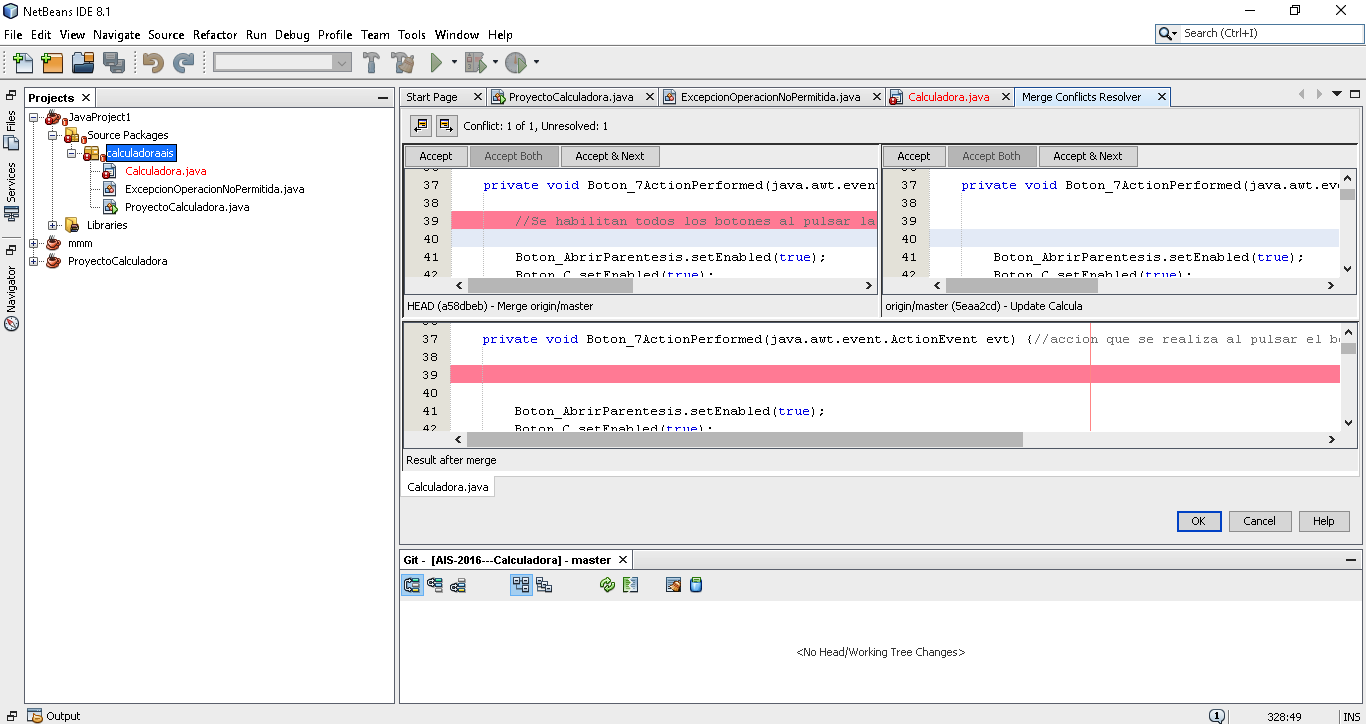


Hemos resuelto el conflicto quedándonos con la versión que había en el repositorio

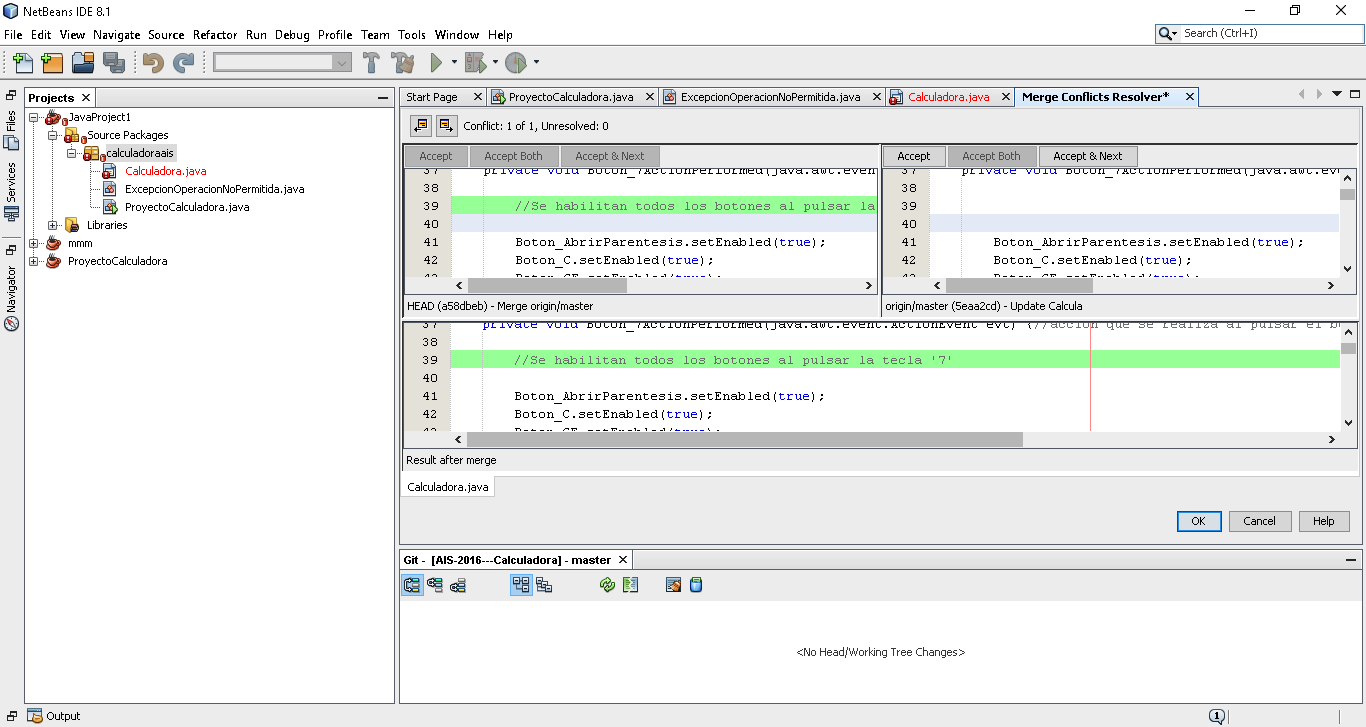
La estrategia de cambios más adecuada para esta clase es la del control informal puesto que el conflicto ha ocurrido en una fase importante del desarrollo del software, concretamente en la fase de la gestión del error “Operación no valida”.

Es un elemento de configuración que forma parte de la línea base producto.

* **Error en el código**



Las líneas que originan el error están en rojo.



Hemos resuelto el conflicto quedándonos con nuestra versión.

La estrategia de cambios más adecuada para esta clase es la del control informal puesto que el conflicto ha ocurrido en una fase importante del desarrollo del software, concretamente en la implementación de la funcionalidad del botón 7.

Es un elemento de configuración que forma parte de la línea base producto.

# **Requisitos**

## **Especificación de requisitos**

### **Requisitos de interfaz**

* **RNF01:Distribuir elementos de la interfaz** ( colocación de las teclas y el display): Siguiendo el estereotipo de una calculadora estándar
* **RNF02:Tamaño de elementos de la interfaz:** 24 para teclas ( a excepción de la tecla “CE”, que tendrá tamaño 12) y 36 para el display
* **RNF03:Color de la interfaz:** Look and feel por defecto de Java
* **RNF04:Fuente de elementos de interfaz:** Tahoma Plain para display y Dialog Bold para teclas
* **RNF05: Elección de símbolos de interfaz** (elección de símbolos que no den lugar a error, basándonos en características como la ubicación geográfica): Números del 0 al 9 para representar los números,” +” para la suma,”-“para la resta y signo negativo, “/” para la división, “X” para la multiplicación, “=” para dar el resultado, “CE” para borrar el último número , “C” para borrar la operación entera, “(“ para el paréntesis abierto, “)” para el paréntesis cerrado y “.” para el carácter decimal.
* **RF01: Implementar interfaz: Hacer la interfaz funcional, teniendo en cuenta las restricciones** (no comenzar con signos, no permitir terminar con signos, etc.)

### **Requisitos de jerarquía de operaciones**

* **RF02: Solicitar al usuario que meta la expresión matemática entera**
* **RF03: Implementar jerarquía de operaciones:** Hacer que se resuelva la expresión matemática siguiendo los conocimientos generales (1º paréntesis, 2º multiplicaciones,3º divisiones, 4º sumas, 5º restas)
* **RF04: Asociar funciones a teclas de interfaz** (enlazar la interfaz con el código): Puesto que la calculadora va a evaluar una expresión, al introducir la expresión y pulsar el botón “=”, la calculadora deberá evaluar dicha expresión teniendo en cuenta los operandos y paréntesis introducidos.

### **Requisitos generales**

* **RNF06:Hacer código modular:** Dividir el código en clases para mejorar mantenimiento
* **RNF07:Comentar el código:** Para una mejor comprensión del mismo
* **RNF08:Estructurar el código:** Para una mejor comprensión del mismo
* **RF05: Manejo de excepciones:** Para prevenir errores en tiempo de ejecución
* **RNF09: Corrección de funcionalidad anterior:** Revisar errores en el código que se nos ha dado.
* **RNF10: Usar lenguaje Java**.
* **RNF11:Implementar funciones extras** ( añadir operaciones necesarias para su completo funcionamiento)

# **Diseño**

## **Diseño de interfaz**

La acción que resulta de pulsar un botón se puede ver condicionada por los valores que hay en la expresión y puede conllevar a una serie de restricciones para otros botones

### **Botones numéricos:**

* Si en el display aparece un mensaje de información de la calculadora, al pulsar el botón se quitará el mensaje de la calculadora y se escribirá el valor del botón.
* Si en el display aparece una expresión matemática y dicha expresión no es resultado de haber dado previamente al botón “**=**”, al pulsar el botón numérico se añadirá a la expresión matemática el valor del botón.
* Si en el display aparece una expresión matemática y dicha expresión es resultado de haber dado previamente al botón “**=**”, al pulsar el botón numérico se quita la expresión matemática y se pondrá el valor del botón

### **Botón “=”:**

* Si la expresión matemática no termina en un número o en paréntesis cerrado, al pulsar el botón aparecerá en el display “**Expresión errónea**”.
* Si la expresión termina en un número o en paréntesis cerrado, al pulsar el botón se evaluará la expresión matemática. Esta evaluación es realizada por el método “tramoParentesis”, que devolverá un valor que se mostrará en el display
* Si la expresión contiene una operación invalida, como “5/0”, al pulsar el botón se mostrará el mensaje “Operación invalida”.
* Si el ultimo botón pulsado ha sido el botón “**=**”, no se podrá pulsar otra vez el mismo botón ni tampoco el botón “)” para evitar errores.

### **Botón “.”:**

* Si en el display no aparece nada, aparece un mensaje de información de la calculadora, al pulsar el botón se quitará lo que hay y se pondrá “**0.**”.
* Si la expresión acaba en un signo de operación o en un paréntesis, al pulsar el botón se añadirá a la expresión “**0.**”.
* Si la expresión acaba en un número, al pulsar el botón se añadirá a la expresión el carácter “**.**”.
* Al pulsar el botón”**.**” se deshabilitará los botones “**(**“ , “**)**” , “**.**” , “**/**” , ”**=**” , “**X**” , “**-**“ , “**+**” para evitar dar errores.

### **Botón “CE”:**

* Si la expresión solo consta de un carácter o es un mensaje de información de la calculadora, al pulsar el botón borrará dicho carácter y mostrará en el display el mensaje “Inserte exp”. En este caso se deshabilitarían los botones “**)**” , ”**=**” , “**/**” , “**X**” , “**+**” para evitar dar errores.
* Si en el display aparece una expresión matemática, borrará el ultimo carácter de la expresión y, dependiendo cual sea el nuevo ultimo carácter de la cadena, deshabilitará unos botones u otros.
  + En el caso de que el ultimo carácter fuera un signo “**+**” o “**-**“, se deshabilitarían los botones “**)**” , “**/**” , ”**=**” , “**X**” , “**-**“ , “**+**” para evitar errores.
  + En el caso de que el ultimo carácter fuera un signo “**X**”, “**/**” o un paréntesis abierto, se deshabilitarían los botones “**)**” , “**/**” , ”**=**” , “**X**” , “**+**” para evitar errores.
  + En el caso de que el ultimo carácter fuera un paréntesis cerrado y todos los paréntesis abiertos ya estuviesen cerrados, se deshabilitaría el botón “**)**” para evitar errores.
  + En el caso de que el ultimo carácter fuera el carácter decimal, se deshabilitarían los botones“**(**” , “**)**” , “**.**” , “**/**” , ”**=**” , “**X**” , “**-**“ , “**+**” para evitar errores.

### **Botón “C”:**

* Al pulsar el botón “**C**” borrará lo que hay en el display, mostrará el mensaje “Inserte exp” y se deshabilitarían los botones “**)**” , “**/**” , ”**=**” , “**X**” , “**+**” para evitar dar errores.

### **Botón “(“:**

* Si la expresión matemática no acaba en el carácter “**.**”, se añadiría el carácter “**(**“al final de la expresión y se deshabilitaría los botones “**)**” , “**/**” , ”**=**” , “**X**” , “**+**” para evitar errores.

### **Botón “)”:**

* Si la expresión matemática acaba en número o en paréntesis cerrado y haya paréntesis abiertos sin sus correspondientes cierres, al pulsar el botón se añadirá el carácter “**)**” y si este último carácter cierra el ultimo paréntesis abierto, entonces se deshabilitaría el botón “**)**” para evitar errores.

### **Botón “X” y “/”:**

* Si la expresión matemática termina en número o en paréntesis cerrado, al pulsar el botón se añadirá a la expresión el carácter “**X**” o “**/**” y se deshabilitarían los botones “**)**” , “**/**” , ”**=**” , “**X**” , “**+**” para evitar errores.

### **Botón “+”:**

* Si el ultimo carácter de la expresión es un numero o un paréntesis cerrado, al pulsar el botón se añadirá el carácter “**+**” y se deshabilitarían los botones “**)**” , “**/**” , ”**=**” , “**X**” , “**+**” para evitar errores

### **Botón “-“:**

* Si el ultimo carácter de la expresión no es un signo “+” ni un signo”-“, al pulsar el botón se añadirá a la cadena el carácter “-“y se deshabilitarían los botones “**)**” , “**/**” , ”**=**” , “**X**”, “**+**” para evitar errores.

## **Diseño de método suma**

El método recibirá 2 valores, los sumará y devolverá dicho resultado al programa.

## **Diseño de método resta**

El método recibirá 2 valores, los restará y devolverá dicho resultado al programa.

## **Diseño de método “multiplicación”**

El método recibirá 2 valores, los multiplicará y devolverá dicho resultado al programa.

## **Diseño de método “división”**

El método recibirá 2 valores (dividendo y divisor), realizará la operación de división (salvo que el divisor sea 0, en cuyo caso mostrará en el display el error “Operación invalida”) y devolverá dicho resultado al programa.

## **Diseño de método “tramoParentesis”**

El método buscará el tramo de paréntesis más prioritario y enviará lo que contiene ese tramo al método “dividirOperacionesSimples”, el cual retornará un valor y dicho valor ocupará el espacio del tramo de paréntesis que ha sido calculado.

De nuevo buscará el tramo de paréntesis más prioritario con la nueva expresión resultante y hará lo mismo que antes. En caso de que no haya más paréntesis, mandará la expresión al método “dividirOperacionesSimples” que calculará un valor y mandará dicho valor a este método, que mostrará el valor por el display.

## **Diseño de método “dividirOperacionesSimples”**

El método hará una iteración por cada operando respetando la jerarquía de operaciones. Dependiendo del operando, hará una llamada a un método distinto (“Suma”, “Resta”, “Multiplicación” o “División”), mandará el valor al método “tramoParentesis” y dicho método mostrará el resultado por el display.

# **Código fuente**

Los siguientes hipervínculos mostrarán las distintas clases de Java (si se mueven los archivos de la carpeta, se deberán acceder a ellos manualmente).

* Clase Calculadora (interfaz):

<Calculadora.txt>

* Clase ProyectoCalculadora:

<ProyectoCalculadora.txt>

* Clase ExcepcionOperacionNoPermitida:

<ExcepcionOperacionNoPermitida.txt>

# **Pruebas y resultados**

La técnica que hemos utilizado es “Clases de equivalencia”. Es un método de prueba de caja negra que divide el campo de entrada de un programa en clases de datos de los que se pueden derivar casos de prueba.

El procedimiento para la identificación de los casos de prueba que hemos seguido ha sido mirar las partes del código que pudieran dar error. Los errores pueden darse en:

* Expresión escrita,
* Combinación de signos
* Operaciones
* Paréntesis.
* Teclas de interfaz

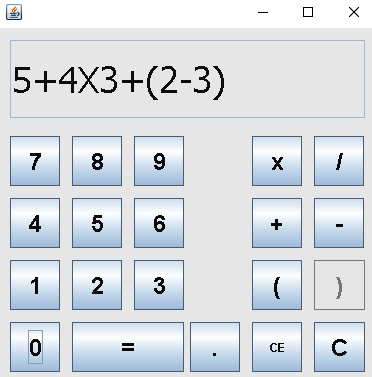
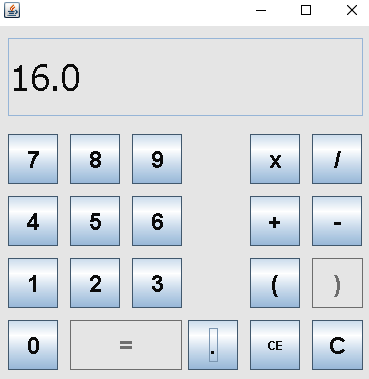
Dado que la técnica de clases de equivalencia propone dar un ejemplo válido y dos no válidos para cada una de las condiciones de entrada, habrá que proponer ejemplos válidos y no válidos para cada uno de los errores anteriores.

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | Casos de prueba | Resultado |
| Expresión escrita | CEV | A : "5+4\*3+(2-3)" | 16 |
| CEI | B : "(X)+3" | No es posible |
| CEI | C : "3/(/2+2)" | No es posible |
| Combinación de signos | CEV | D : "5X-3+(2/-1)" | -17 |
| CEI | E : "5X/3+2" | No es posible |
| CEI | F : "5++2-+4" | No es posible |
| Operaciones | CEV | G : "5/4+2+3-15" | -8,75 |
| CEI | H : "5/0+2+9-20" | No es posible |
| CEI | I : "5/(2-2)" | No es posible |
| Paréntesis | CEV | J : "5+(2-3)-(3+2)" | -1 |
| CEI | K : ")3+2X-2+3)" | No es posible |
| CEI | L : "((2+3)-4)))" | No es posible |

Los casos de prueba para mostrar el buen funcionamiento de las teclas de la interfaz son todos los anteriores

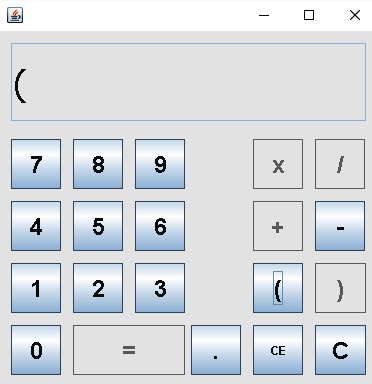
A continuación, la demostración de que la calculadora es capaz de cumplir los CEV y de que no es capaz de hacer los CEI de la tabla anterior.

## Caso de prueba A

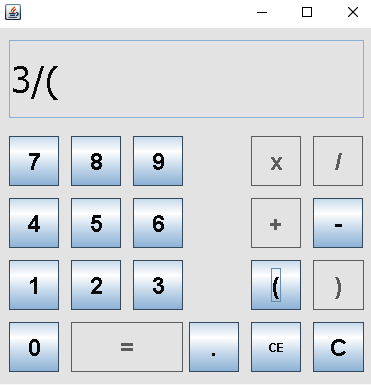
Como se puede ver, la calculadora muestra el resultado esperado

## Caso de prueba B



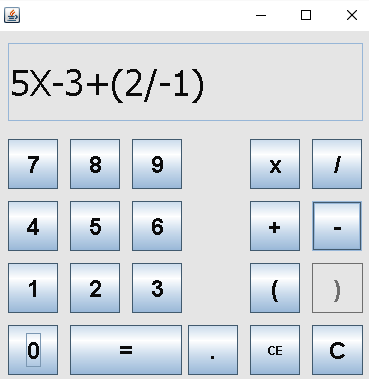
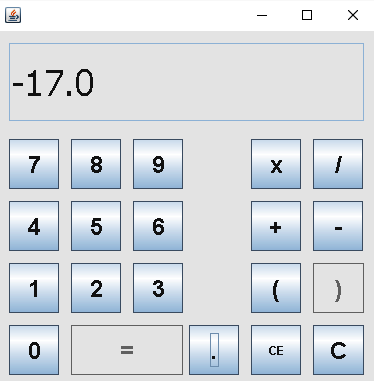
Como se puede ver, la calculadora no permite pulsar el signo “X” después del paréntesis abierto.

## Caso de prueba C



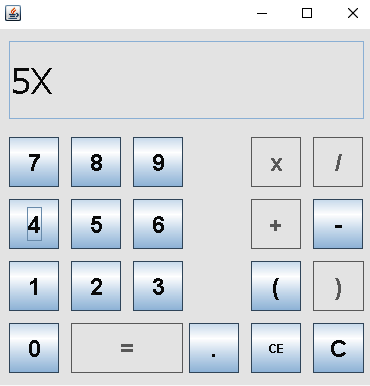
Como se puede ver, la calculadora no permite pulsar el signo “/” después del paréntesis abierto.

## Caso de prueba D

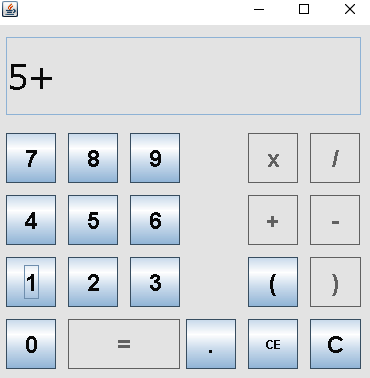
Como se puede ver, la calculadora muestra el resultado esperado

## Caso de prueba E



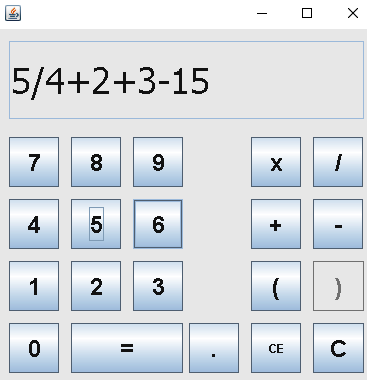
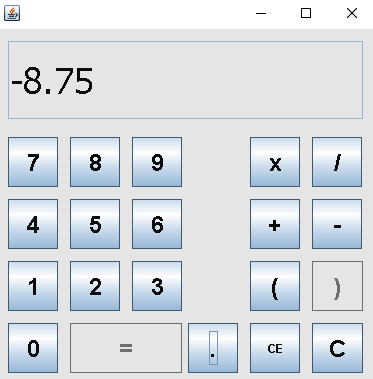
Como se puede ver, la calculadora no permite pulsar el signo “/” después del signo “X”

## Caso de prueba F



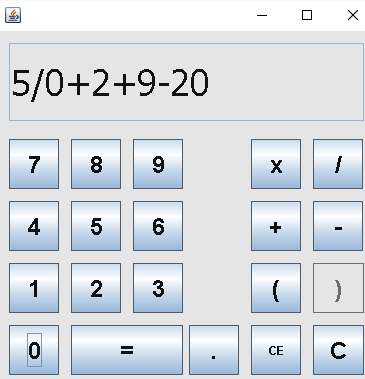
Como se puede ver, la calculadora no permite poner el signo “+” después de otro signo “+”

## Caso de prueba G

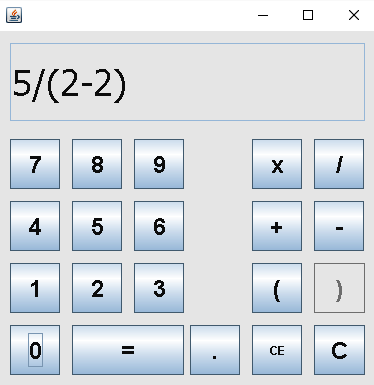
Como se puede ver, la calculadora muestra el resultado esperado

## Caso de prueba H



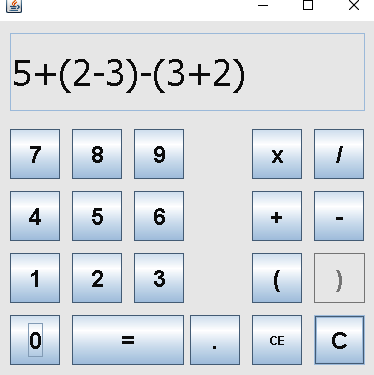
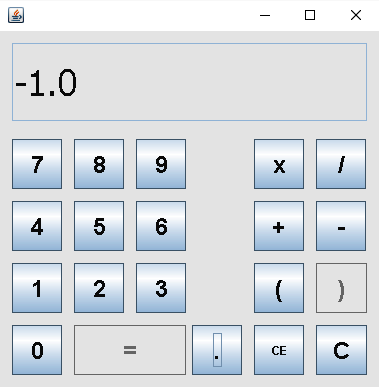
Como se puede ver, la calculadora muestra el mensaje “Operación invalida” si encuentra en la expresión una división entre 0.

## Caso de prueba I



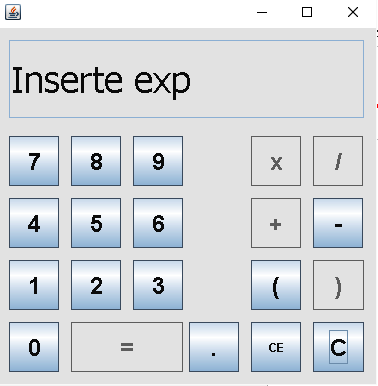
Como se puede ver, la calculadora muestra el mensaje “Operación invalida” si encuentra en la expresión una división entre 0.

## Caso de prueba J

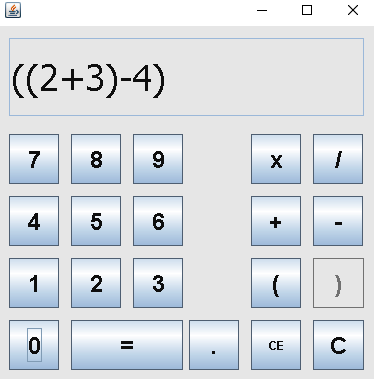
Como se puede ver, la calculadora muestra el resultado esperado

## Caso de prueba K



Como se puede ver, la calculadora no permite empezar una expresión con el carácter “)”.

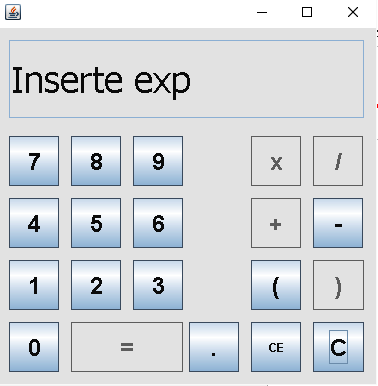
## Caso de prueba L



Como se puede ver, la calculadora no permite poner más paréntesis cerrados porque no hay más paréntesis abiertos.

# **Manual de usuario**

Las restricciones del funcionamiento de la calculadora se especificarán en un apartado de este mismo documento llamado “Diseño”.



* Al pulsar un número, ese número aparecerá en el display de la calculadora.
* Al pulsar una tecla de operación, esa operación aparecerá en el display.
* Al pulsar el botón de apertura o cierre de paréntesis, ese paréntesis aparecerá en el display.
* Al pulsar el botón “C”, se borrará toda la expresión matemática del display.
* Al pulsar el botón “CE”, se borrará el último número introducido.
* Al pulsar el botón “=”, se evaluará la expresión matemática y se devolverá el resultado (o un error) en el display.
* Al pulsar el botón “.”, aparecerá el carácter “.” en el display
* Al pulsar el botón de cierre de pestaña, la calculadora se cerrará
* Al pulsar el botón de minimizar, la calculadora se minimizara
* Al pulsar el botón de maximizar, la calculadora se hará más grande

# 

# **Cambios realizados en la documentación**

Las modificaciones que hemos tenido que hacer son:

* **Cambiar al carácter decimal “.”**: En un principio elegimos “,” pero Java trabaja con “.”, por lo que estuvimos obligados a cambiar para evitar errores
* **Cambiar al método “tramoParentesis”**: En un principio pensamos en un método que contará el número de operandos y el número de paréntesis para poder saber cuál era el paréntesis más profundo. Pero había que hacer muchas comprobaciones y optamos por una nueva forma de pensar
* **Cambiamos al diseño actual de la interfaz**: En un principio no pensamos en que se pudiera poner el carácter decimal detrás de un signo. Al mirar en una calculadora real, vimos que esto sí se podía hacer y decidimos cambiarlo.
* **Cambiamos la definición del *RF04***: En un principio definimos mal el requisito (se podía confundir con el ***RNF05***) y optamos por redefinirle.
* **Cambiamos la forma de prevenir algunos errores**: En un principio todos los errores iban a ser gestionados por el código (mostrarían un mensaje de error por el display), pero al ver que se llevaba mucho tiempo decidimos que algunos errores se previnieran en la interfaz mediante el bloqueo de los botones que creasen conflicto con la expresión matemática.
* **Cambiamos los recursos humanos asociados a las tareas:** Aunque en un principio las tareas se repartieron de forma aleatoria, después decidimos que las tareas más complejas las llevasen a cabo 2 personas, para lograr un reparto de trabajo equitativo.
* **Cambiamos al actual calendario:** En un principio solo representamos el calendario con un único color, pero después decidimos poner 3 colores distintos para que se viese reflejado la bifurcación del proyecto en 2 ramas: Interfaz y código.
* **Cambiamos a los actuales mensajes de información de la calculadora:** Como hemos dicho antes, habíamos pensado en gestionar los errores por el código y, por tanto, para distinguir un error de otro, eran necesarios más mensaje de información. Al cambiar a la prevención de algunos errores mediante la interfaz, los mensajes de información los redujimos a 3.