**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, “/” para la división, “X” para la multiplicación, “=” para dar el resultado, “CE” para borrar el último número y “C” para borrar la operación entera.

**Requisitos de jerarquía de operaciones:**

* **RF01: Solicitar al usuario que meta la expresión matemática entera**
* **RF02: 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)
* **RF03: Asociar funciones a teclas de interfaz** (enlazar la interfaz con el programa): Asociar la tecla “+” a la operación suma, la tecla “-“ a la operación resta, la tecla “/” a la operación división, la tecla “X” a la función multiplicación, las teclas de los números para que muestren en el display los números que se han pulsado, así como los paréntesis y el punto decimal; la tecla “CE” para borrar el último número , la tecla “C” para borrar la expresión completa y la tecla “=” para mostrar el resultado

**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
* **RF04: 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)

**Tareas:**

1. Identificación de requisitos
2. Identificación de tareas
3. Asignación de tareas a miembros del grupo
4. Documentar requisitos y tareas
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 “comprobar operadores”
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

|  |  |  |  |
| --- | --- | --- | --- |
| 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. |
| N | G | 3 | David |
| O | H | 1 | Sergio P. |
| P | I | 1 | Jaime |
| Q | J | 1 | Sergio L. |
| R | K | 1 | David |
| S | L | 1 | Sergio P. |
| T | M | 1 | Jaime |
| U | N | 4 | Sergio L. |
| V | O | 1 | David |
| 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 |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 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.

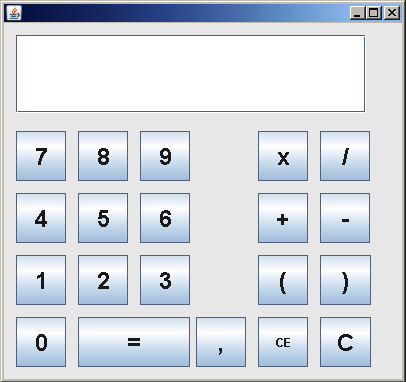
Holgura independiente de una actividad HIij = TEj – TLi – tij.

**Herramienta de control de versiones**

La herramienta de control de versiones que hemos utilizado es GitHub, que es una plataforma de desarrollo colaborativo de software para alojar proyectos utilizando el sistema de control de versiones de Git.

El funcionamiento se basa en torno a un repositorio centralizado y la estrategia de versionado que sigue es la realización de forks para clonar un repositorio ajeno) y así eliminar algún bug o modificar cosas de él. Una vez que se haya modificado, se puede enviar un pull al dueño del proyecto. Este, en última instancia, considerará los cambios y si los acepta, lo adjuntará al repositorio original

**Documentación de funcionalidad de interfaz**



Como se ha mencionado antes en los requisitos, la distribución de los elementos de la interfaz sigue el esquema de un estereotipo de calculadora. El tamaño, la fuente, el color y la elección de los símbolos de la calculadora también están especificados en el apartado requisitos.

El funcionamiento es el siguiente:

* 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.
* Si la expresión matemática termina con un símbolo de operación, dicho símbolo se eliminará del display al pulsar el botón “=” y no se tendrá en cuenta en la evaluación de la expresión
* Si en la expresión matemática aparecen seguidos los símbolos “+” y “-“o viceversa, solo se representará el último símbolo de operación

**Documentar funcionamiento de métodos**

* *Método “Suma”*: El método recibirá 2 valores, los sumará y devolverá dicho resultado al programa.
* *Método “Resta”*: El método recibirá 2 valores, los restará y devolverá dicho resultado al programa.
* *Método “Multiplicación”:* El método recibirá 2 valores, los multiplicará y devolverá dicho resultado al programa.
* *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.
* *Método” Comprobar operandos”*: El método contará el número de operandos y paréntesis que hay en la expresión matemática para saber cuántas iteraciones y paréntesis hay.

Si el número de paréntesis abiertos no es igual al de cerrados o se han introducido de forma errónea, se mostrará en el display el error “Paréntesis mal introducido” o “Numero de paréntesis erróneo”.

Si la expresión matemática termina con un símbolo de operación, se mostrará en el display el error “Expresión errónea”.

Posteriormente llamará al método “Dividir en operaciones simples” para evaluar la expresión matemática

* *Método “Dividir en operaciones simples”*: 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”) y mostrará el resultado por el display.