MIEMBROS

* Laura Moreno París GII
* Borja Rodríguez González GII
* Rubén López Fraile GII
* Alejandro Parrilla Monrocle GII
* Jesús Álvarez Tenorio GII

|  |  |
| --- | --- |
| práctica  Ampliación de Ingeniería de Software | Descripción breve  La práctica de la asignatura de Ampliación de Ingeniería del Software pretende poner en práctica el conjunto de conocimientos implicados en dicha asignatura. Por ello, aunque la práctica es única, será llevada a cabo por partes y en base a los temas que abarca la asignatura. |

Contenido

[1. INTRODUCCIÓN 1](#_Toc447872560)

[2. PLANIFICACIÓN 1](#_Toc447872561)

[2.1. Identificar errores 3](#_Toc447872562)

[2.2. Establecer requisitos 3](#_Toc447872563)

[2.3. Diagrama de clases 4](#_Toc447872564)

# 1. INTRODUCCIÓN

En esta práctica tenemos que gestionar el mantenimiento de un software, en este caso una calculadora muy básica la cual tenemos que corregir y añadir funcionalidad nueva (mantenimiento evolutivo). Para ello lo primero que tenemos que hacer es realizar planificar las tareas y asignarlas a cada miembro del grupo.

# 2. PLANIFICACIÓN

En primer lugar tenemos que identificar las tareas que encontramos y posteriormente realizar la planificación en cuanto a tiempo estimado mediante un diagrama Pert.

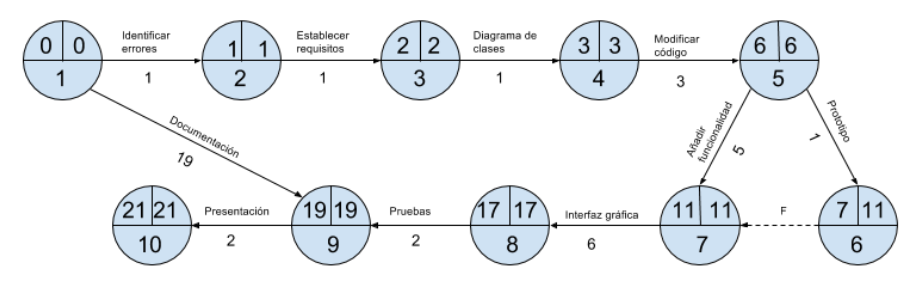
Las tareas que hemos encontrado son las siguientes:

1. Identificar errores
2. Establecer requisitos
3. Diagrama de clases
4. Corregir código según los errores identificados
5. Realizar prototipo
6. Añadir nueva funcionalidad
7. Realizar interfaz
8. Pruebas
9. Documentar todo el proceso
10. Presentación

En cuanto al reparto de dichas actividades tenemos lo siguiente:

|  |  |  |  |
| --- | --- | --- | --- |
| Tarea | Participantes | Tarea | Participantes |
| 1 | Borja Rodríguez | **6** | Alejandro Parrilla y Rubén López |
| 2 | Rubén López | **7** | Alejandro Parrilla |
| 3 | Laura Moreno | **8** | Rubén López |
| 4 | Jesús Á., Laura M. y Borja R. | **9** | Laura, Borja, Rubén y Alejandro |
| 5 | Alejandro Parrilla | **10** | Rubén López |

El diagrama de Pert, llevado a cabo por Laura, resultante de las anteriores actividades según su realización hasta la fecha fijada por el grupo para finalizar la primera parte de la práctica es el siguiente:



En este diagrama hemos calculado los tiempos en función de días, puesto que hemos presupuesto que nuestro producto debería estar realizado antes del día 16 de Marzo, por tanto contamos con 21 días.

A los sucesos los hemos asignado números desde el 1 hasta el 10 y las actividades tienen el nombre completo de la actividad que se va a llevar a cabo.

Nos encontramos con que, hasta el suceso número 5 tenemos una precedencia lineal de las actividades. En este suceso tiene lugar una precedencia divergente puesto que para la iniciación tanto del prototipo como de la nueva funcionalidad es necesario que se haya completado la modificación del código. En este mismo punto vemos que ha sido necesario el añadir una actividad ficticia, la cual no consume recursos, puesto que teníamos que añadir funcionalidad y prototipo podían realizarse al mismo tiempo pero ambas eran precedentes de la tarea consistente en la realización de la interfaz gráfica. Esto motivó a que añadiésemos esa actividad ficticia para que ambas fuesen precedentes de esa actividad pero no surgiese un conflicto entre ambas.

En él vemos que el camino crítico es el que implica a los sucesos 1, 2, 3, 4, 5, 7, 8, 9 y 10, lo que quiere decir que las actividades implicadas en esos sucesos tienen una holgura de 0 días. Sin embargo, la actividad consistente en realizar el prototipo de nuestra futura calculadora tiene una holgura de 4 días.

## 2.1. Identificar errores

En un primer lugar, se llevará a cabo la identificación de los errores que presenta el código aportado. Para ello, enumeraremos los siguientes errores encontrados:

1. Nos encontramos que tras cada línea donde se ha querido insertar un salto de línea, en el código, sólo aparece “n” en lugar de “\n”.
2. En el menú inicial, si se inserta un carácter distinto a 1-5 (distintas opciones que nos permite la aplicación), falla y se cierra. En su lugar debería notificar el error y permitir recuperarse.
3. Una vez seleccionada una de las primeras opciones, en el apartado de introducir un operando, tanto el primero como el segundo, no se comprueban los valores de entrada, lo que permite la inserción de caracteres no numéricos como operandos de la calculadora.
4. En los apartados de introducir operando, al seleccionar cancelar se finaliza la aplicación con error. Sin embargo, debería mostrar de nuevo el menú inicial con las operaciones.
5. En el menú principal, ocurre lo mismo que en el error anterior, pero además este botón de cancelar carece de funcionalidad, ya que para salir sería con introducir el valor 5. Por lo que habría que quitar dicho botón o este sustituirlo por “Salir” y quitar esta opción, la 5, de las opciones dadas.
6. De nuevo en el apartado de introducir un apartado, si pulsas sin haber insertado nada, la aplicación se cierra con un error en lugar de esperar a que se inserte algún valor para el operando.
7. Por último, al pulsar sobre  se notifica un error y esto no debería ser así, sino que se debería cerrar la aplicación sin error alguno.

## 2.2. Establecer requisitos

**1. Requisitos Funcionales (**funciones que realizará el sistema, dando soporte al usuario**)**

* **RF01** El cliente requiere de una aplicación con interfaz gráfica que permita llevar a cabo operaciones aritméticas básicas, donde se respete en todo momento un jerarquía de operaciones con y sin paréntesis.
* **RF02** Como requisito básico e inicial, la calculadora debe realizar las funcionalidades aritméticas más básicas (suma, resta, multiplicación y división).
* **RF03** El cliente desea poder ampliar la funcionalidad básica de la aplicación permitiendo poder cumplir una jerarquía de operaciones a través de paréntesis.
* **RF04** La aplicación evaluará las expresiones tras esperar a que el usuario las introduzca de manera completa.
* **RF05** La aplicación debe impedir que se introduzcan letras que puedan dar lugar a errores en la aplicación.
* **RF06** Por su parte, en caso de error, la aplicación debe mostrar por pantalla el error cometido y permitir al usuario recuperarse del mismo, introduciendo otro carácter que sí sea válido o retrocediendo al estado anterior que dio lugar al error.
* **RF07** La aplicación debe tener consistencia, evitando que el usuario cometa errores tales como salir de la aplicación cuando no lo desee u operar con caracteres inválidos.

**2. Requisitos No Funcionales (**restricciones de diseño y de implementación**)**

* **RNF01** El lenguaje de programación que se utilizará en el desarrollo de la aplicación será el lenguaje Java.
* **RNF02** La distribución de los botones de la calculadora en la interfaz gráfica estará basada en la misma distribución que podemos encontrar en la calculadora de Windows o en los teclados de los ordenadores.
* **RNF03** Los colores utilizados en la interfaz deberán ser claros y sobrios.
* **RNF04** El cliente tiene como requisito el añadido del icono de la Universidad Rey Juan Carlos en la propia interfaz gráfica de la aplicación.

## 2.3. Diagrama de clases

Usamos el diagrama de clases antes de la implementación de nuestra calculadora  para visualizar las relaciones establecidas entre las clases. En este caso tendremos una clase llamada calculadora y otra interfaz, que será una interfaz gráfica.

En la clase calculadora tendremos dos atributos, que serán dos operadores y cuatro métodos que serán las operaciones aritméticas de sumar, restar, dividir y multiplicar.

