## **PROGRAMACIÓN ORIENTADA A OBJETOS**

**Excepciones 2024-1**

**Laboratorio 4/6**

## **OBJETIVOS**

1. Perfeccionar el diseño y código de un proyecto considerando casos especiales y errores.

2. Construir clases de excepción encapsulando mensajes.

3. Manejar excepciones considerando los diferentes tipos.

4. Registrar la información de errores que debe conocer el equipo de desarrollo de una aplicación en producción.

5. Vivenciar las prácticas ***Designing*** [*Simplicity*](http://www.extremeprogramming.org/rules/simple.html)*.*

[Refactor](http://www.extremeprogramming.org/rules/refactor.html) whenever and wherever possible.

## **ENTREGA**

1. Incluyan en un archivo .zip los archivos correspondientes al laboratorio. El nombre debe ser los dos apellidos de los miembros del equipo ordenados alfabéticamente.
2. Deben publicar el avance al final de la sesión y la versión definitiva en la fecha indicada, en los espacios preparados para tal fin.

# **Proyectos EN BLUEJ**

### **PRACTICANDO MDD y BDD con EXCEPCIONES**

**[En lab04.doc, activities.asta y BlueJ activities]**

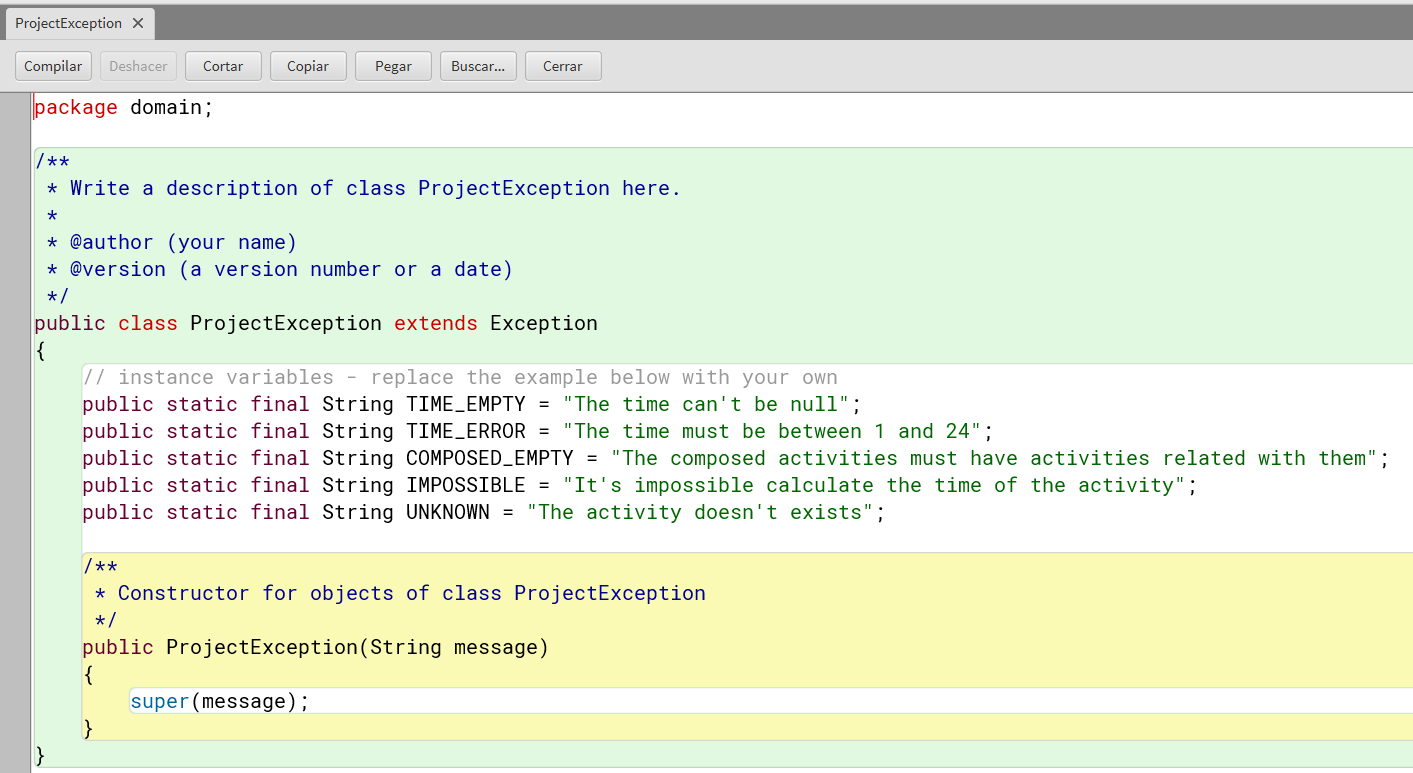
En este punto vamos a aprender a diseñar, codificar y probar usando excepciones. Para esto se van a trabajar algunos métodos de la clase Activity

1. En su directorio descarguen los archivos contenidos en activities.zip revisen el contenido y estudien el diseño parcial que está en el diagrama de clases. ¿Qué estructura de datos es Actividad? Justifique la respuesta.

Activity es una estructura de datos de árbol n-ario, porque las actividades compuestas tienen activiades dentro.

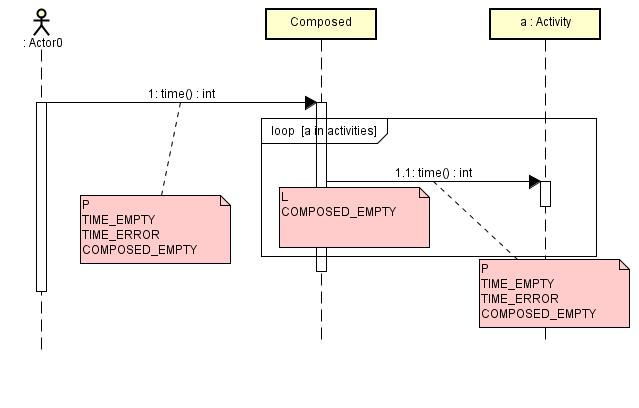
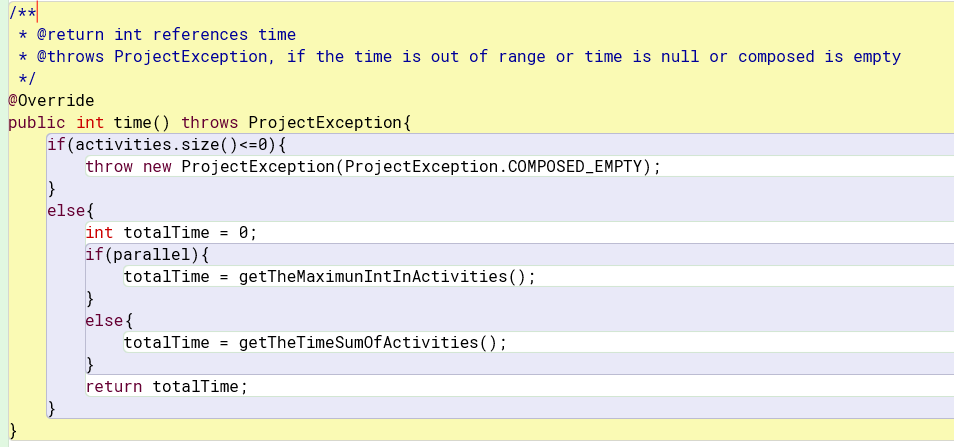
1. Expliquen porque el proyecto no compila. Realicen las adiciones necesarias para lograrlo.

No compila porque no hay una clase ProjectException que hereda de Exception. Para corregir ello, hay que crear una clase ProjectException y allí añadir todas las variables constantes que contendrán los mensajes de las excepciones que ya están siendo implementadas en el código.

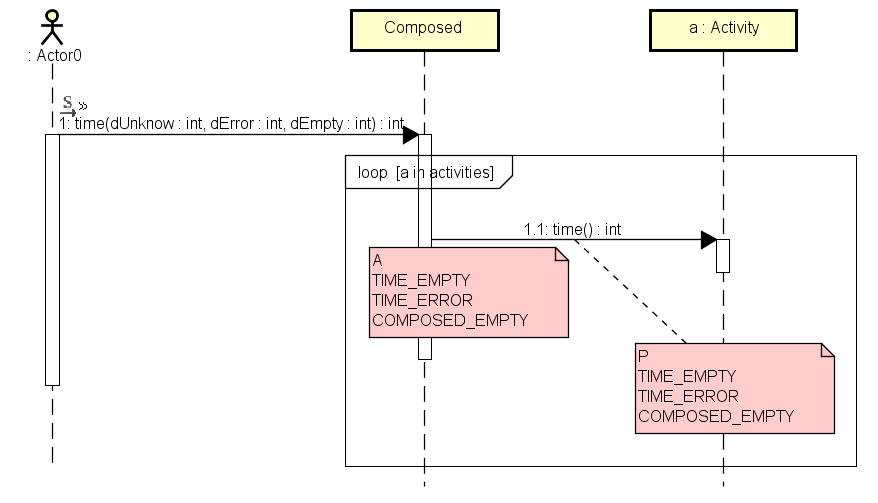
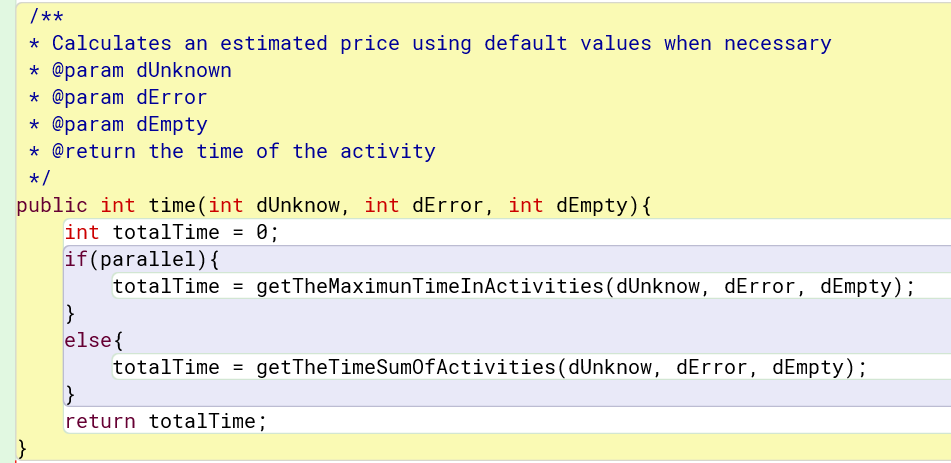


1. Dadas las pruebas, documenten, diseñen y codifiquen el método time().

Adicionen pruebas análogas a cada una de las dadas para una actividad más profunda (nivel 3)



1. Diseñen, codifiquen y prueben el método time(unknown, error, empty).



1. Diseñen, codifiquen y prueben el método time(estimate)

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Diagrama

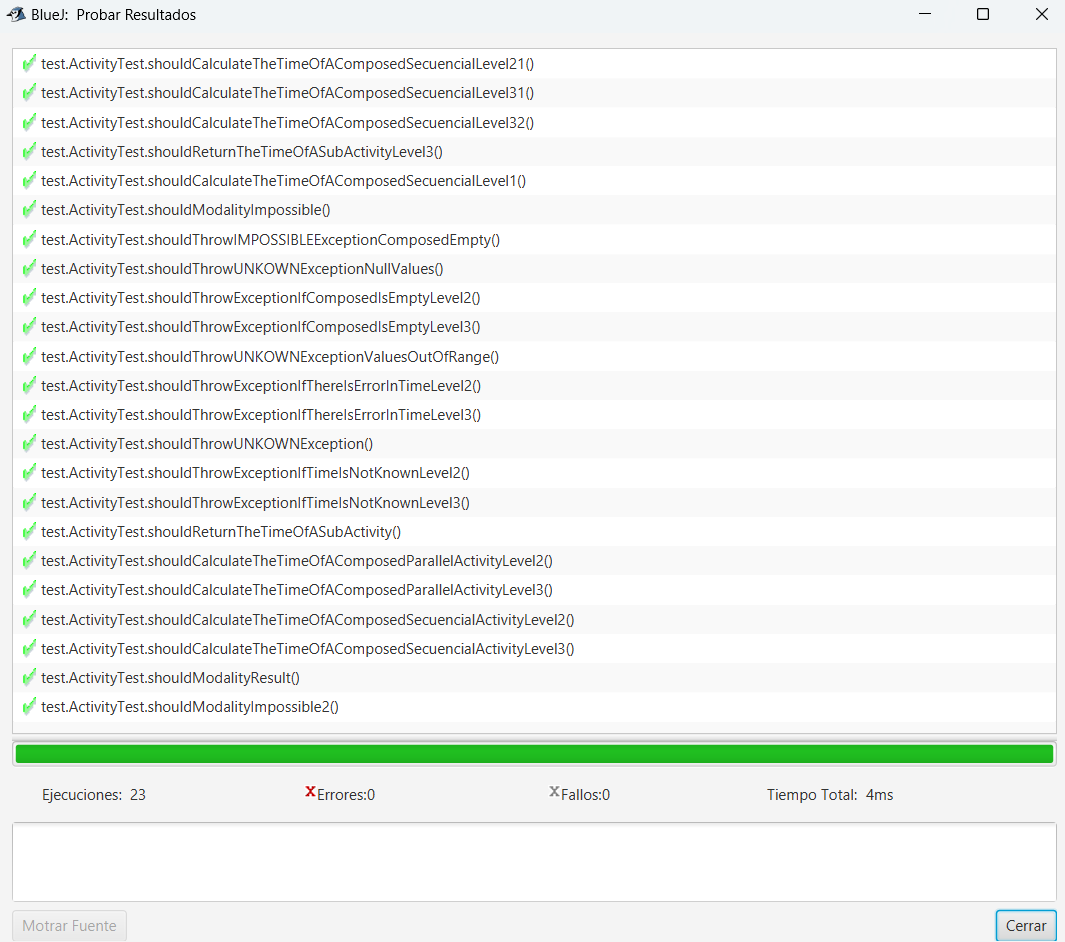
Descripción generada automáticamente

1. Diseñen, codifiquen y prueben el método time(activity).

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

LAS PRUEBAS DE TODOS LOS METODOS TIME() SE ENCUENTRAN EN ActivityTest



**Project**

# **EN CONSOLA**

El objetivo de esta aplicación es realizar un gestor de proyectos. Los proyectos están compuestos de dos tipos de actividades: actividades simples y actividades compuestas. Las actividades compuestas pueden ser paralelas o secuenciales.

**Conociendo el proyecto Project [En lab04.doc]**

**No olviden respetar los directorios bin docs src**

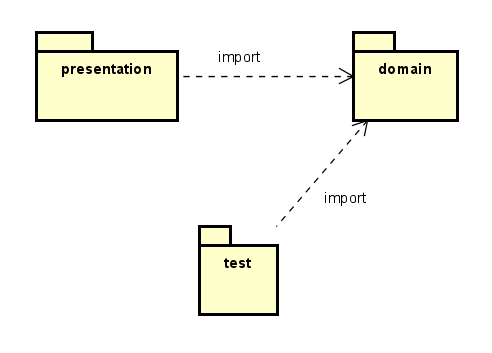
1. En su directorio descarguen los archivos contenidos en project.zip, revisen el contenido.

¿Cuántos archivos se tienen? ¿Cómo están organizados? ¿Cómo deberían estar organizados?

Hay 3 archivos .java, los cuales son: log, Project, ProjectManagerGUI.

Estos archivos deberían de encontrarse respectivamente en las carpetas de src.

1. Realicen el diseño del programa: diagramas de paquetes y de clases (presentación y dominio). ¿cuántos paquetes tenemos? ¿cuántas clases tiene el sistema? ¿cuál es la clase ejecutiva?



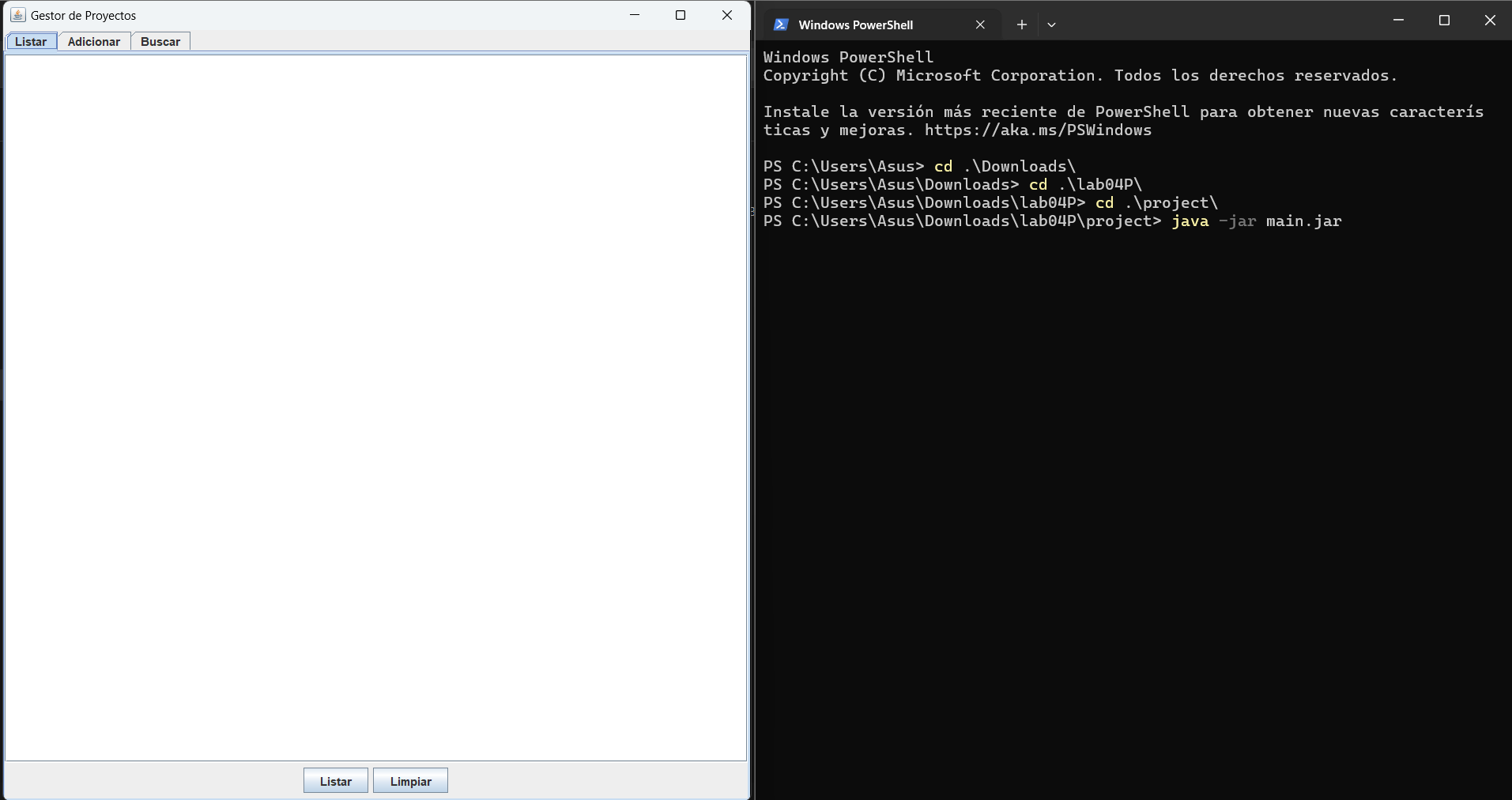
La clase ejecutiva es ProjectManagerGUI. Tenemos 3 paquetes. En total tenemos 5 clases.

1. Prepare los directorios necesarios para ejecutar el proyecto. ¿qué estructura debe tener?

¿qué clases debe tener cada directorio? ¿dónde están esas clases? ¿qué instrucciones debe dar para ejecutarlo?

Para ejecutar el proyecto nos debemos dirigir a la clase ProjectManagerGui, y ejecutar su metodo main. Una vez se tiene el jar de la clase principal, se usa java –jar main.jar   
para ejecutar el archivo.

El directorio debería tener todas las clases necesarias para que el archivo pueda ser ejecutado de manera correcta.



1. Ejecute el proyecto, ¿qué funcionalidades ofrece? ¿cuáles funcionan?

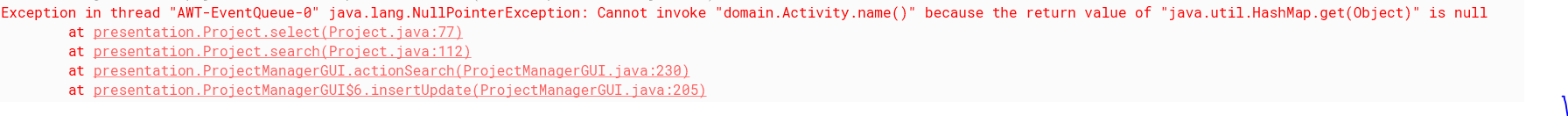
Ofrece 4 funcionalidades listar, adicionar, limpiar y buscar.

Listar: Al listar se nos indica el total de actividades y de cuales se componen estas mismas con su costo y tiempo, además, de que tipo son.

Adicionar: Nos permite adicionar una nueva actividad, podemos tanto decirle el tipo o indicarle un tiempo.

Limpiar: Permite borrar los campos de entrada en Adicionar y también borrar los datos de cuando se usa la funcionalidad de listar.

Buscar: Al intentar escribir en el espacio indicado sale una excepción en la terminal.



1. Revisen el código y la documentación del proyecto. ¿De dónde salen las actividades iniciales?

¿Qué clase pide que se adicionen? ¿Qué clase los adiciona?

ProjectManagerGUI en su constructor instancia un objeto de Project, a su vez, en el constructor de Project, se hace un llamado al metodo addSome(), este adiciona las actividades.

**Adicionar y listar. Todo OK.** [En lab04.doc, Project.astay \*.java]

(NO OLVIDEN BDD - MDD)

El objetivo es realizar ingeniería reversa a las funciones de adicionar y listar.

1. Adicionen una nueva actividad simple y una nueva actividad compuesta

Simple

Iterar

tiempo 500 , Costo 10000

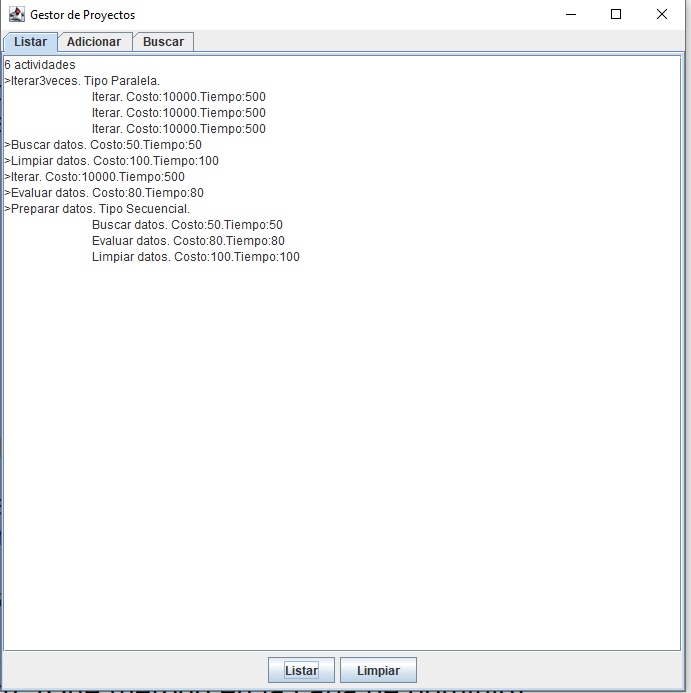
*Completo*

Iterar 3 veces

100, Pararela

Iterar Iterar Iterar

¿Qué ocurre? ¿Cómo lo comprueban? Capturen la pantalla. ¿Es adecuado este comportamiento?



Se adicionan ambas actividades. Se comprueba usando el botón listar. No es adecuado, ya que no cumplen las condiciones de los tiempos de las actividades. Es decir, que no está implementado el paquete domain de manera correcta.

1. Revisen el código asociado a **adicionar** en la capa de presentación y la capa de dominio.

¿Qué método es responsable en la capa de presentación? ¿Qué método en la capa de dominio?

Para la capa de presentación:

Tenemos un método prepareElements(), quien se encarga de alistar toda la ventana que se

despliega, además, crea una pestaña “adicionar” que llama al método prepareAreaAdd(), que

se encarga de preparar la interfaz gráfica de la ventana adicionar. Hay otro método que se

encarga de ejecutar lo indicado, según el botón que presione el usuario. La acción indicada

para el botón de listar es, llamar al método actionAdd que a su vez llama al metodo add de

project, pasándole los atributos insertados en cada una de las casillas en la ventana adicionar.

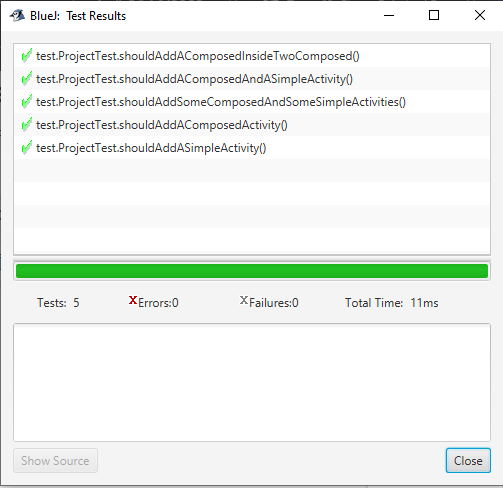
Para la capa de dominio:

El método encargado de adicionar las actividades es add(String name, String cost, String timeType, String theActivities). Quien a partir de unas verificaciones que realiza, determina si es compuesta o simple. Posterior a eso se encarga de limpiar los datos, para llamar a los constructores de la actividad correspondiente y finalmente añadir la actividad al HashMap de activities.

1. Realicen ingeniería reversa para la capa de dominio para **adicionar**. Capturen los resultados de las pruebas de unidad.

Diagrama

Descripción generada automáticamente



1. Revisen el código asociado a **listar** en la capa de presentación y la capa de dominio. ¿Qué método es responsable en la capa de presentación? ¿Qué método en la capa de dominio?

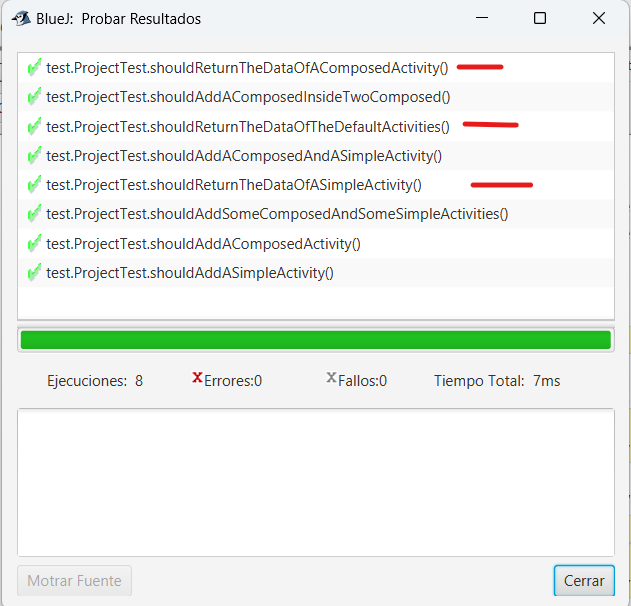
Para la capa de presentación:

Tenemos un método prepareElements(), quien se encarga de alistar toda la ventana que se despliega, además, crea una pestaña “Listar” que llama al método prepareAreaList(), que se encarga de preparar la interfaz gráfica de la ventana listar. Hay otro método que se encarga de ejecutar lo indicado, según el botón que presione el usuario. La acción indicada para el botón de listar es, llamar al método actionList que a su vez llama al metodo toString de project, para actualizar así lo mostrado en pantalla.

Para la capa de dominio:

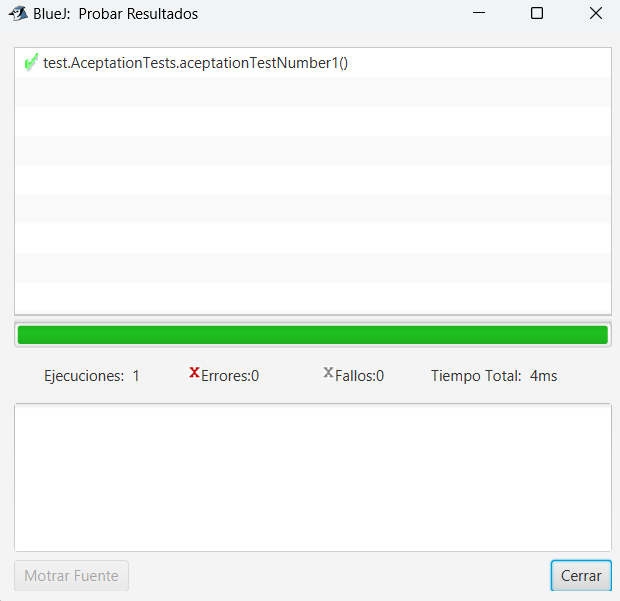
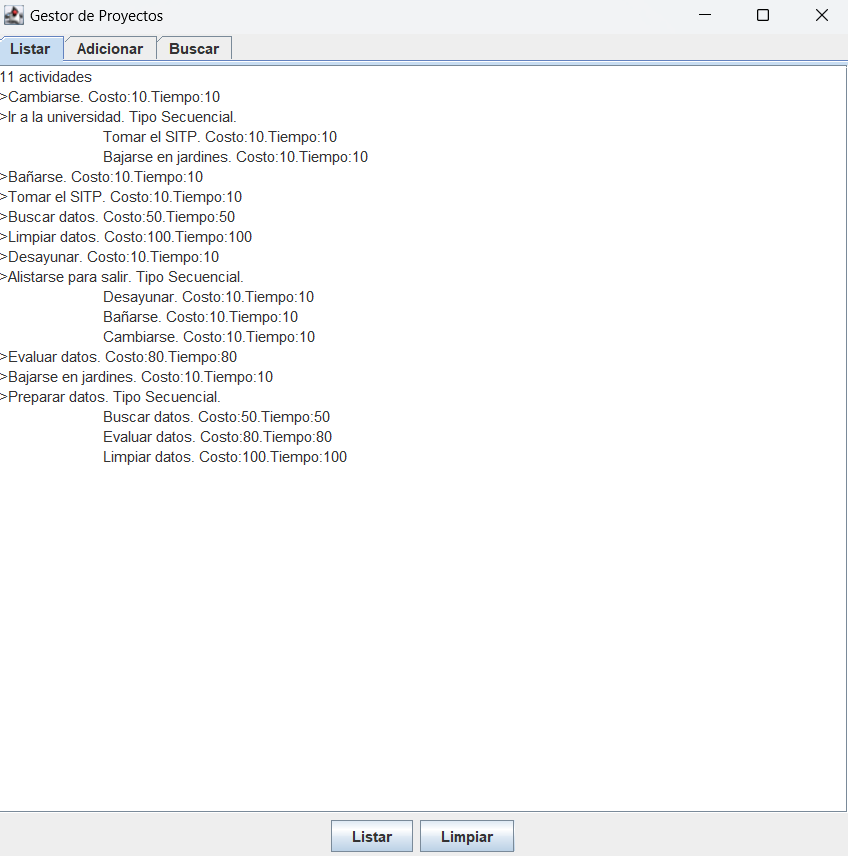
Tenemos el método data(), presente en Project y que a su vez llama a data() en las actividades que tiene almacenada proyectto. Donde va concatenando la información de las diferentes actividades.

1. Realicen ingeniería reversa para la capa de dominio para **listar**. Capturen los resultados de las pruebas de unidad.



1. Propongan y ejecuten una prueba de aceptación.

La descripción de la prueba se encuentra en el paquete test, en la clase AcceptanceTest



**Adicionar una actividad. Funcionalidad robusto** [En lab04.doc, Project.asta y \*.java]

(NO OLVIDEN BDD – MDD)

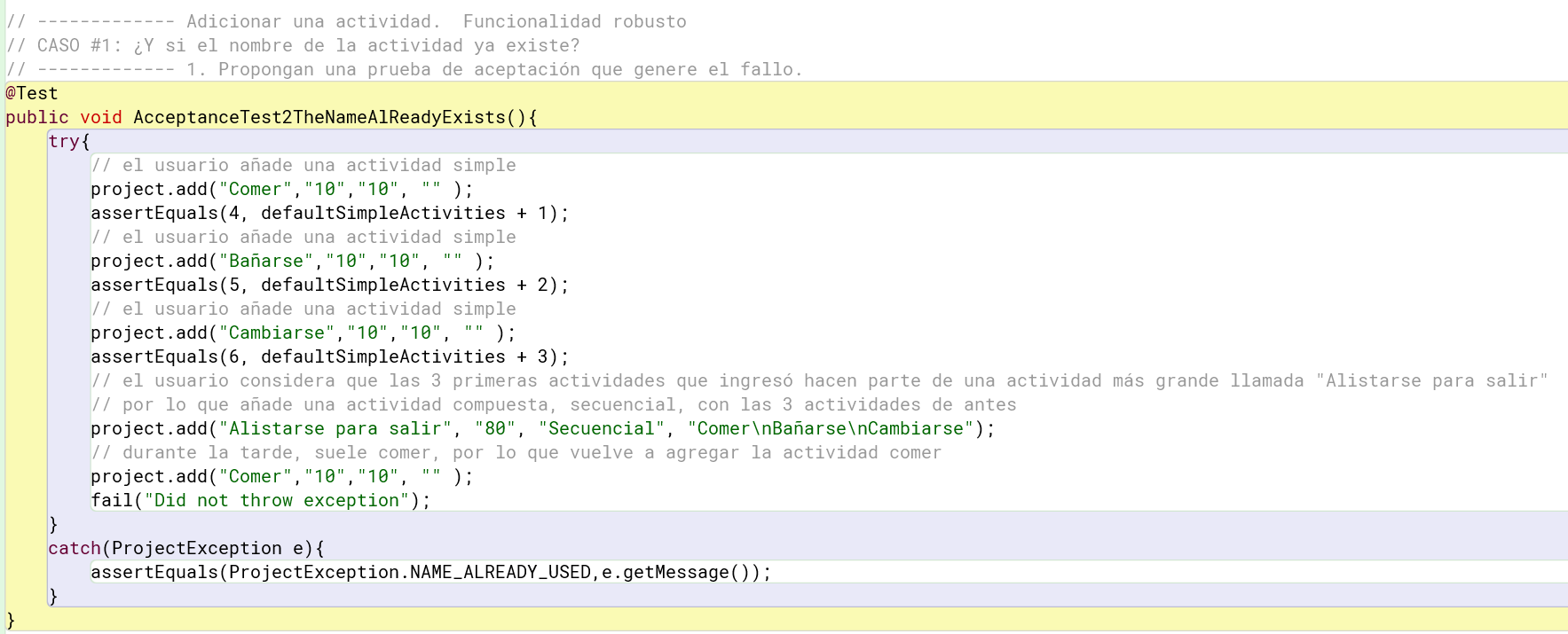
El objetivo es perfeccionar la funcionalidad de adicionar un curso para hacerla más robusta.

### **Para cada uno de los siguientes casos realice los pasos del 1 al 4.**

* 1. **¿Y si el nombre de la actividad ya existe?**
  2. **¿Y si en precio o costo no da un número? ¿si el tipo no es paralelo o secuancial?**
  3. **Proponga una nueva condición**

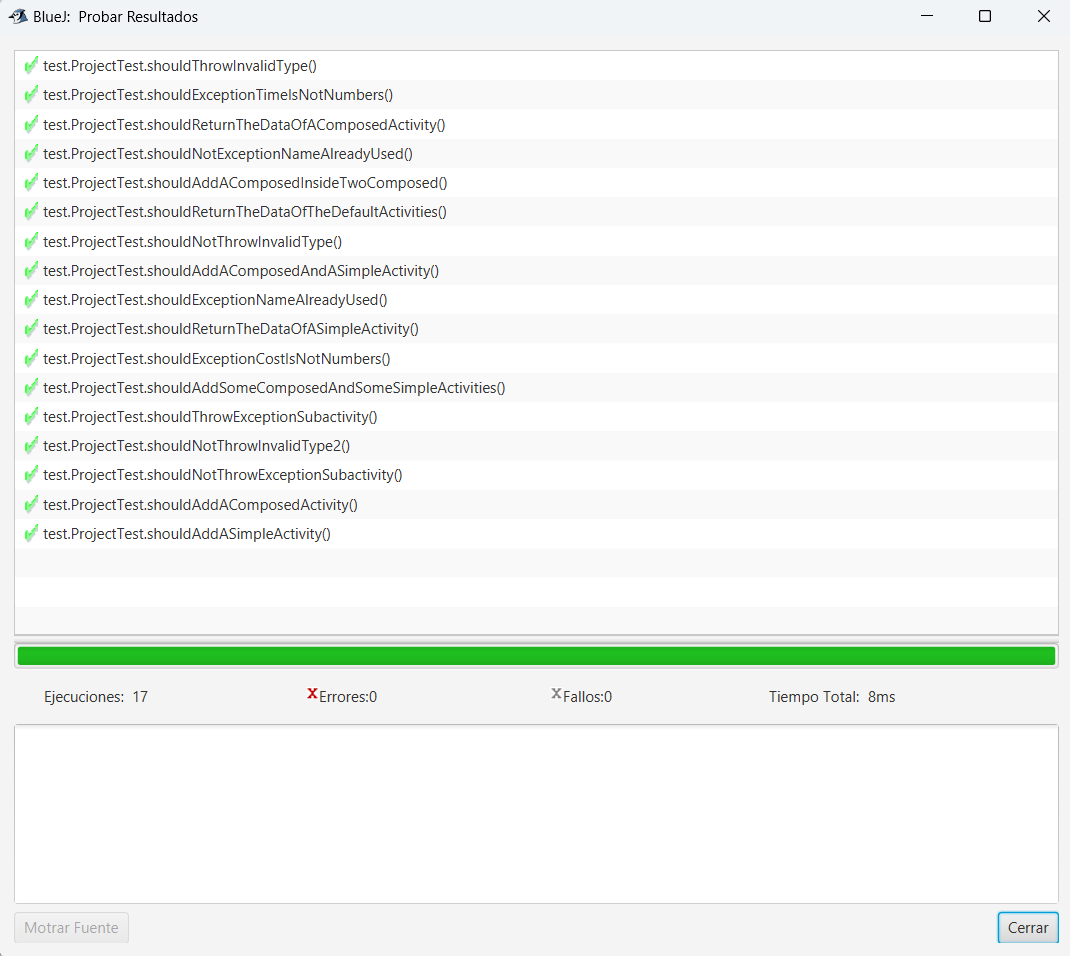
1. Propongan una prueba de aceptación que genere el fallo.
2. Analicen el diseño realizado. Para hacer el software robusto: ¿Qué método debería lanzar la excepción? ¿Qué métodos deberían propagarla? ¿Qué método debería atenderla? Explique claramente.
3. Construya la solución propuesta. Capture los resultados de las pruebas de unidad.
4. Ejecuten nuevamente la aplicación con el caso de aceptación propuesto en 1. ¿Qué sucede ahora? Capture la pantalla.

PARA A:

1. 

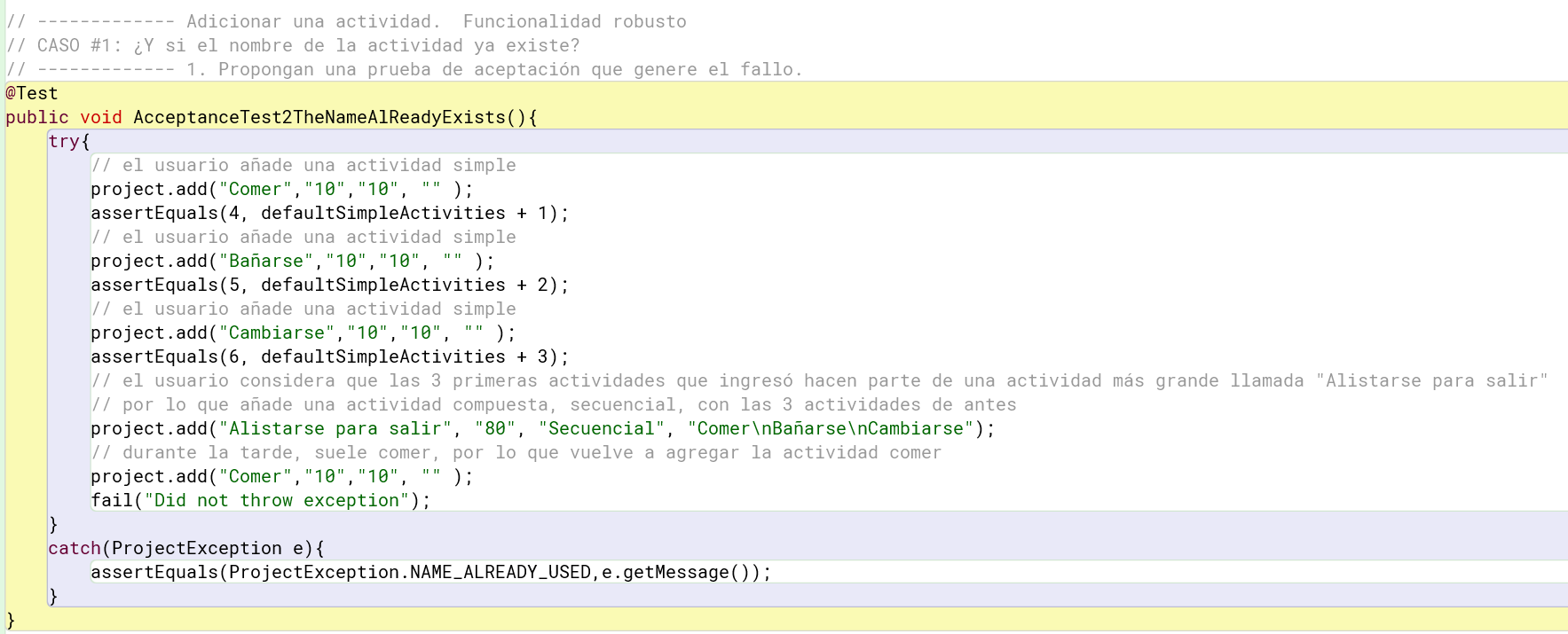
La prueba entra por fail("Did not throw exception"). Da error.

1. El método encargado de lanzar la excepción sería el add() de la clase Project. Los métodos que llamen a add deberían ser capaces de propagarla. El método que debería atenderlo debe ser actionAdd() de la clase ProjectManagerGui, en la cual se debe mandar un mensaje para que el usuario entienda lo sucedido. Para el primer caso la condición para lanzar la excepción seria si en el HashMap, ya contiene de llave ese nombre, lanzar una excepción
2. Pruebas unitarias



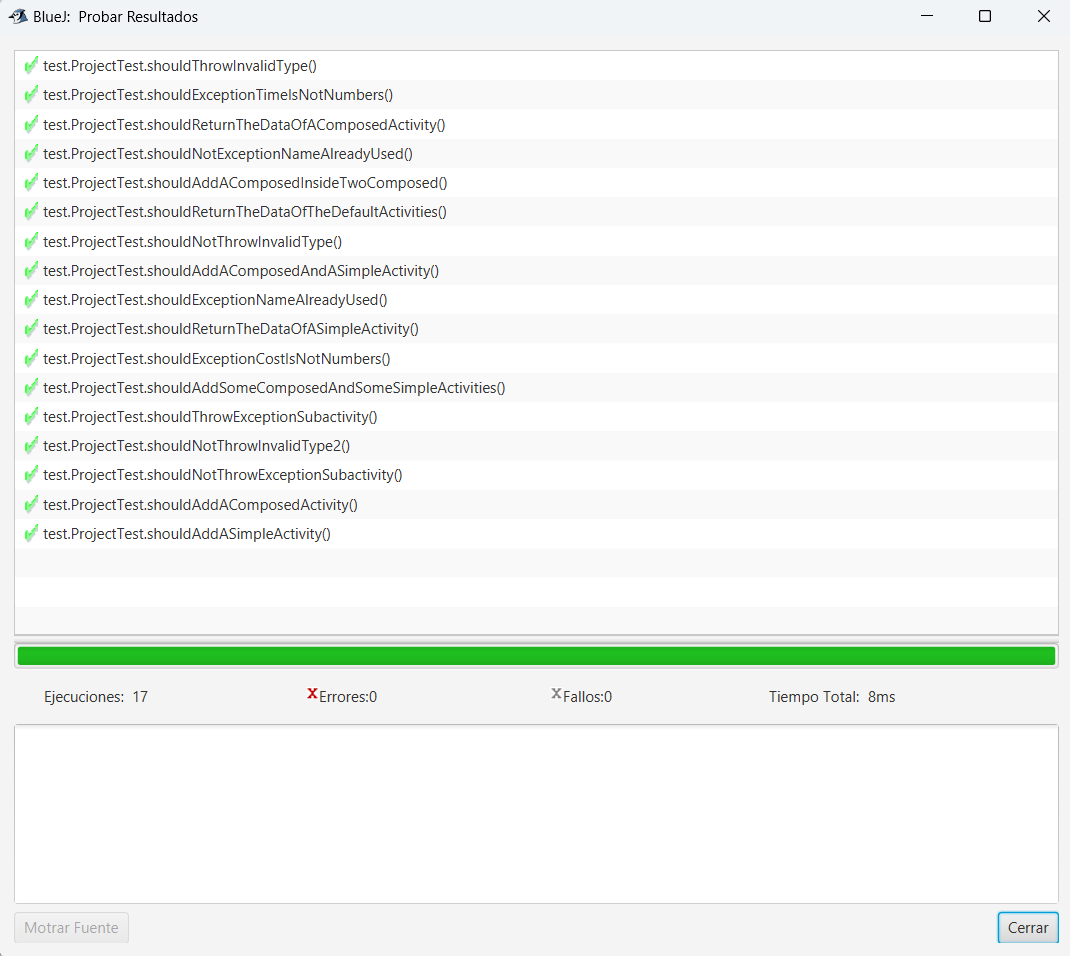
1. Ahora pasa el test

PARA B:

1. 

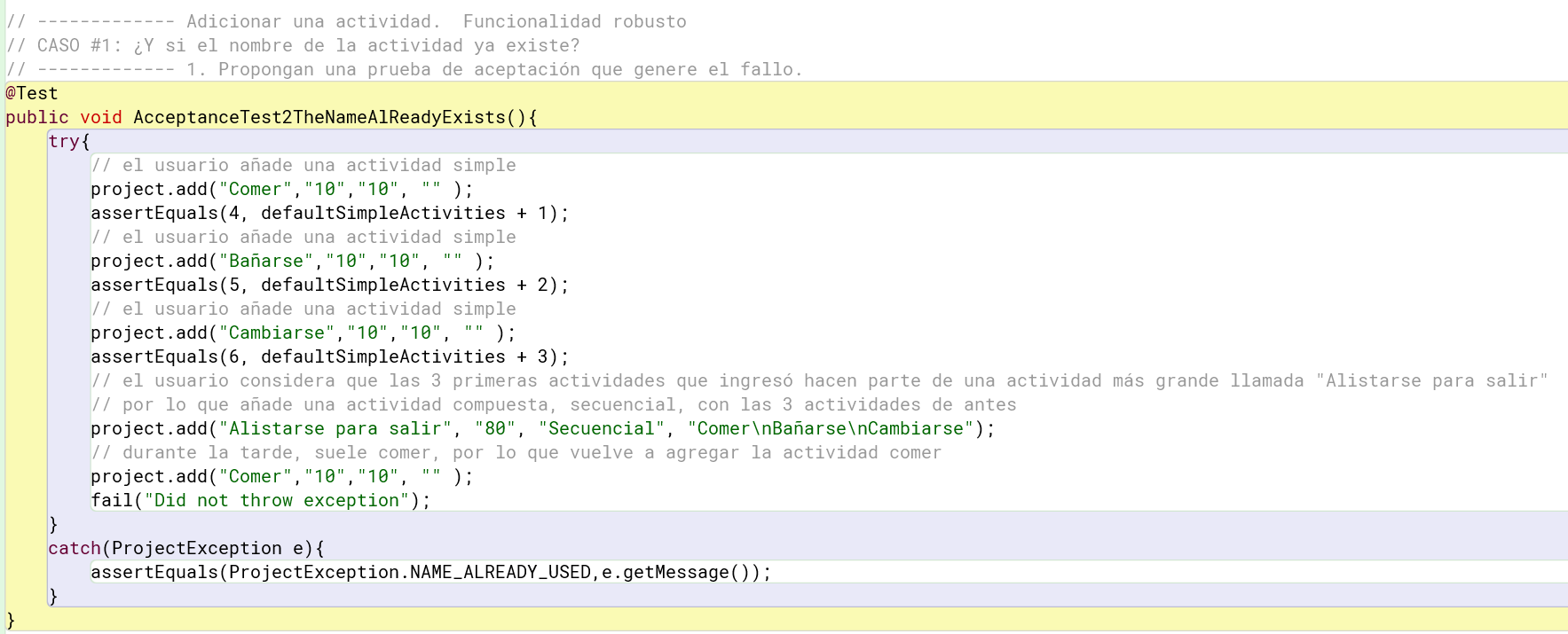
La prueba entra por fail("Did not throw exception"). Da error.

1. El método encargado de lanzar la excepción sería el add() de la clase Project. Los métodos que llamen a add deberían ser capaces de propagarla. El método que debería atenderlo debe ser actionAdd() de la clase ProjectManagerGui, en la cual se debe mandar un mensaje para que el usuario entienda lo sucedido. Para el primer caso la condición para lanzar la excepción seria si en el HashMap, ya contiene de llave ese nombre, lanzar una excepción
2. Pruebas unitarias



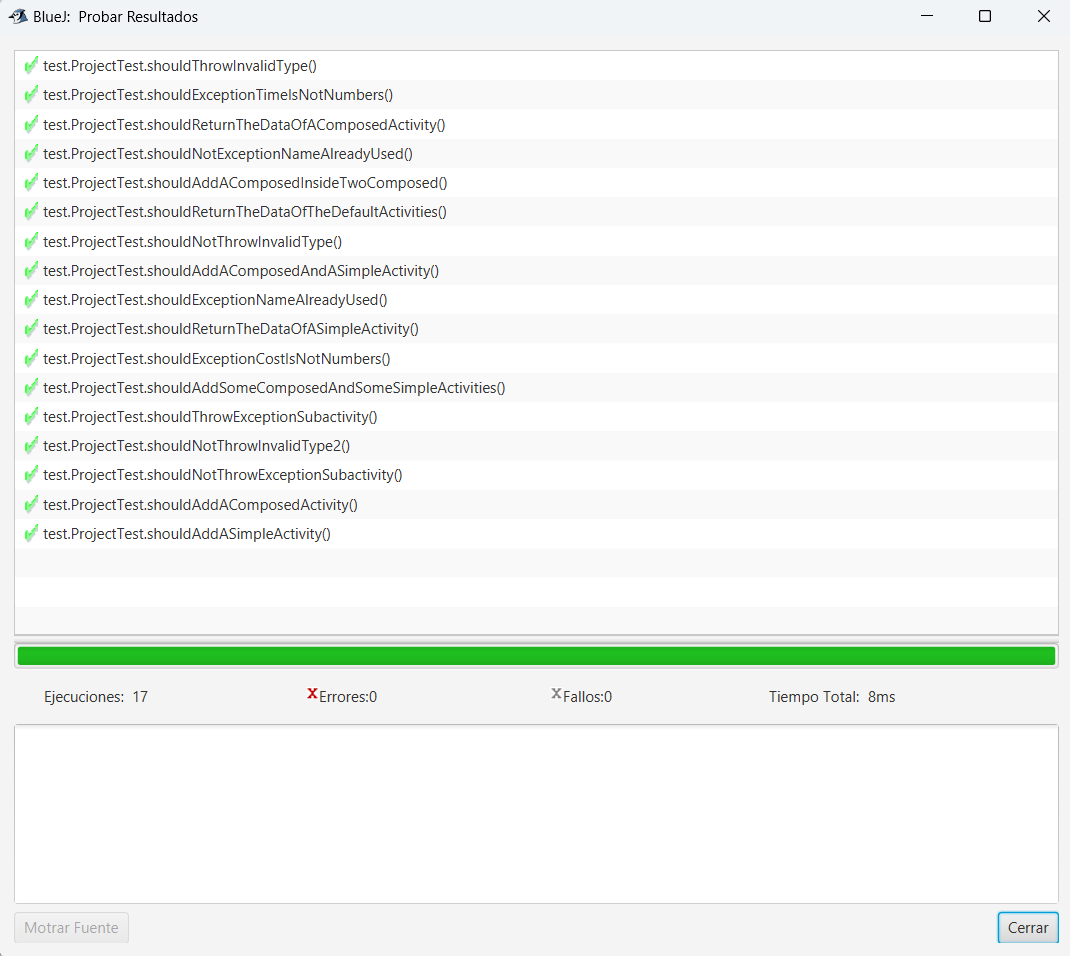
1. Ahora pasa el test

PARA C:

1. 

La prueba entra por fail("Did not throw exception"). Da error.

1. El método encargado de lanzar la excepción sería el add() de la clase Project. Los métodos que llamen a add deberían ser capaces de propagarla. El método que debería atenderlo debe ser actionAdd() de la clase ProjectManagerGui, en la cual se debe mandar un mensaje para que el usuario entienda lo sucedido. Para el primer caso la condición para lanzar la excepción seria si en el HashMap, ya contiene de llave ese nombre, lanzar una excepción
2. Pruebas unitarias



1. Ahora pasa el test

### **Consultando por patrones. ¡ No funciona y queda sin funcionar!**

[En Project.asta, Project.log, lab04.java y \*.java]

(NO OLVIDEN BDD - MDD)

1. Consulten una actividad que inicie con I. ¿Qué sucede? ¿Qué creen que pasó? Capturen el resultado. ¿Quién debe conocer y quien NO debe conocer esta información?

Se lanza una excepción propia de java. El usuario final no debería de ver este tipo de excepciones / errores que se presentan durante la ejecución del programa. Mientras que los desarrolladores si deberían de conocer qué errores se le presentaron al usuario, para así, poder determinar en dónde se presentó el error y poder solucionarlo.



1. Exploren el método record de la clase Log ¿Qué servicio presta?

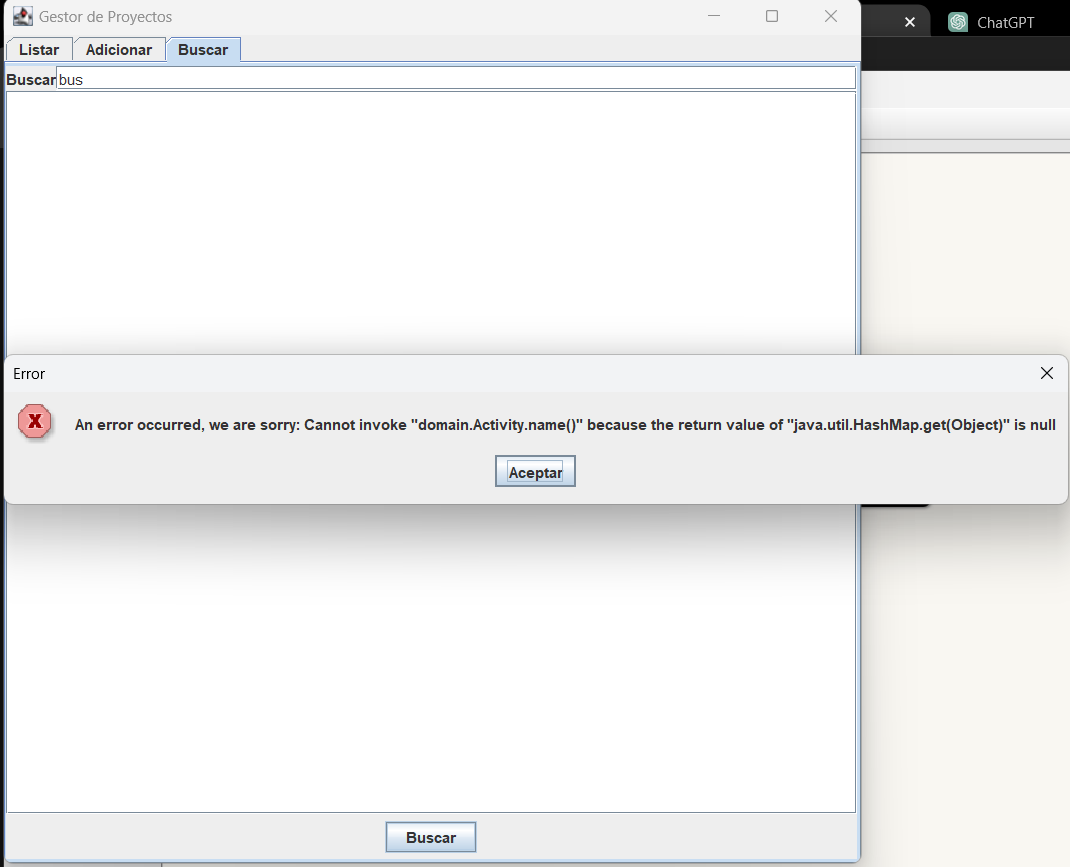
Presta el servicio de poder tener un registro de las excepciones en un archivo (log), donde esa excepción se captura, se guarda y se imprime.

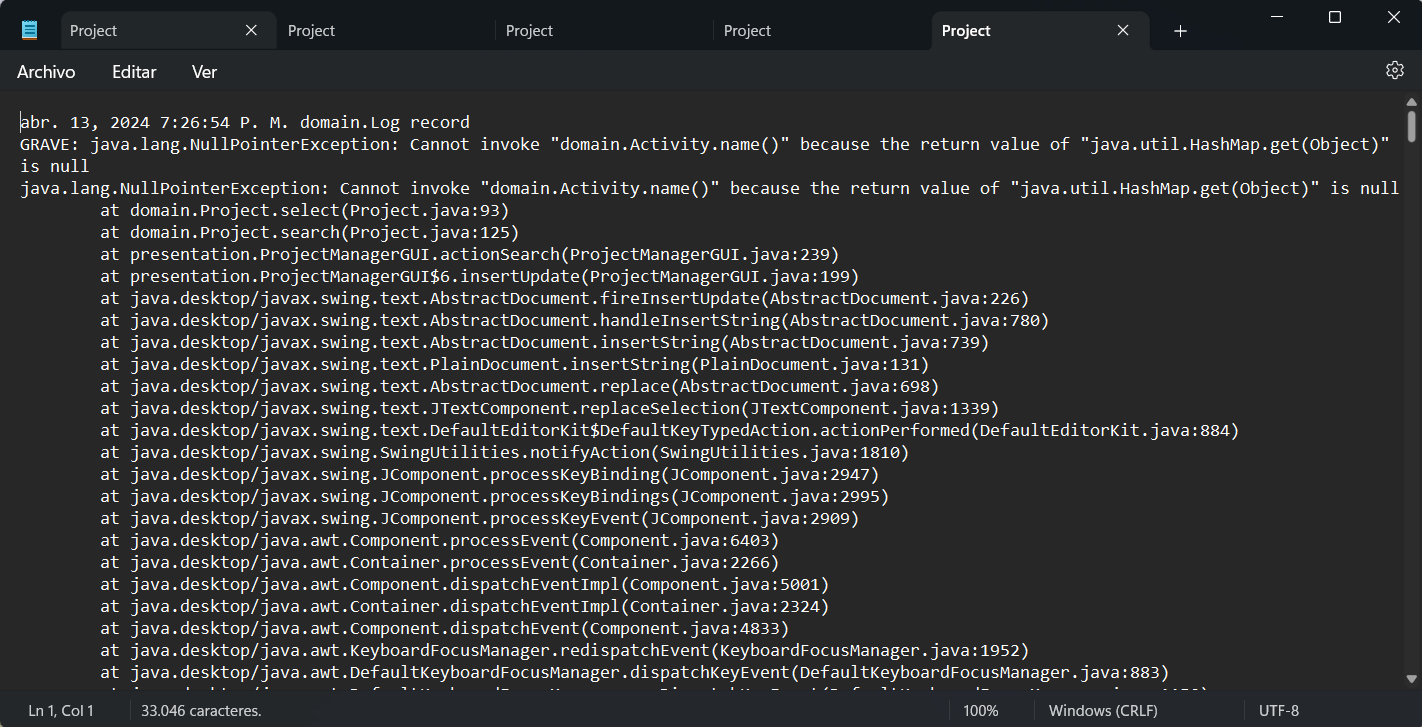
1. Analicen el punto adecuado para que **EN ESTE CASO** se presente un mensaje especial de alerta al usuario, se guarde la información del error en el registro y continúe la ejecución. Expliquen y construyan la solución.

Vamos a hacer, al igual que en los puntos anteriores, la validacion en la capa de dominio, y la capa de presentación, atrapa las excepciones que propaga la capa de dominio, manejandolas haicneod que el usuario vea un mensaje de disculpas y guardando los erroes en el .log.

1. Ejecuten nuevamente la aplicación con el caso propuesto en 1. ¿Qué mensaje salió en pantalla? ¿La aplicación termina? ¿Qué información tiene el archivo de errores?

La aplicación no termina, se sigue ejecutando sin problema.





1. ¿Es adecuado que la aplicación continúe su ejecución después de sufrir un incidente como este? ¿de qué dependería continuar o parar?

Si se refiere a que la palabra no pudo ser encontrada, si, la aplicación debería seguir ejecutándose. Pero si se trata de un error desconocido como el null pointer, lo mejor sería que la aplicación se cierre/pare, para evitar que este posible error que no manejamos, desencadene otra serie de errores.

1. Modifiquen la aplicación para garantizar que **SIEMPRE** que haya un error se maneje de forma adecuada. ¿Cuál fue la solución implementada?

Implementamos un try-catch, para los errores que conocemos, simplemente informarle al usuario qué está haciendo de manera incorrecta. Pero para los errores desconocidos, atraparlos, pedirle disculpas al usuario y posteriormente almacenrlos en el .log para tenerlos en cuenta más adelante.

### **Consultando por patrones. ¡Ahora si funciona!**

[En Project.asta, Project.log, lab04.java y \*.java]

(NO OLVIDEN BDD - MDD)

1. Revisen el código asociado a **buscar** en la capa de presentación y la capa de dominio. ¿Qué método es responsable en la capa de presentación? ¿Qué método es responsable en la capa de dominio?

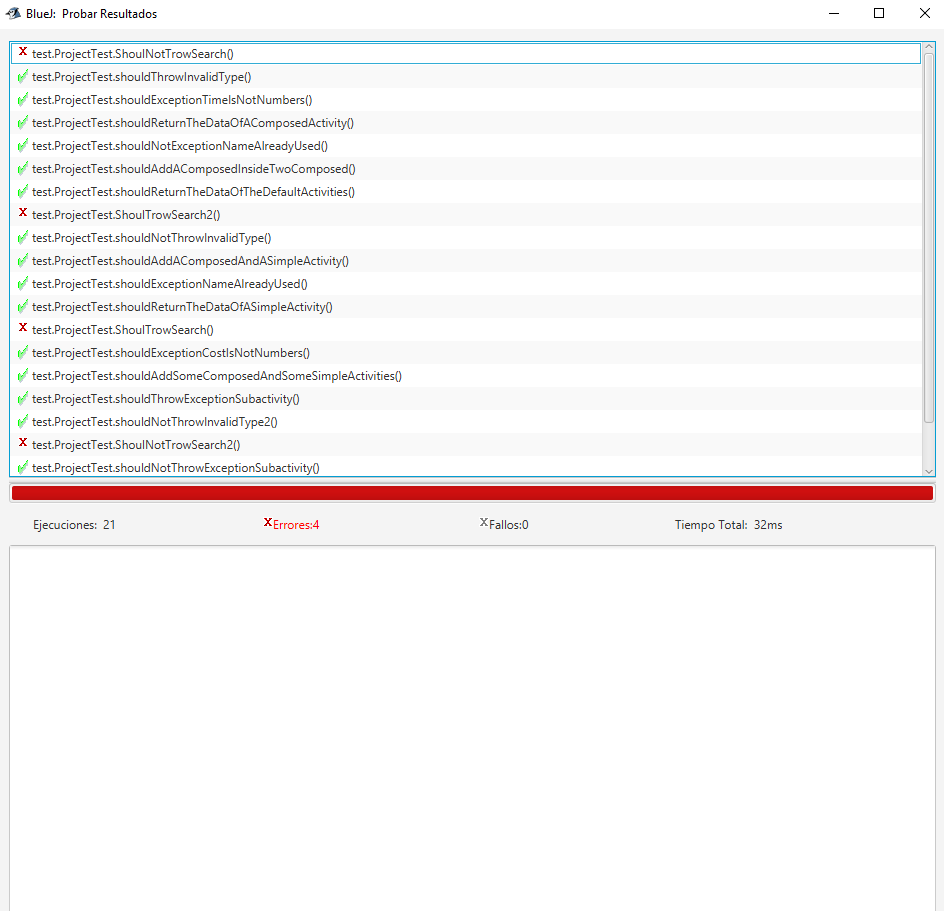
En la capa de presentación: el botón buscar, que llama a la acción de actionSearch().

En la capa de dominio: los responsables son select(), search() y data(). Quienes a su vez filtran y organizan los datos para que el usuario pueda verlos de manera correcta.

1. Realicen ingeniería reversa para la capa de dominio para **buscar**. Capturen los resultados de las pruebas. Deben fallar.

Diagrama

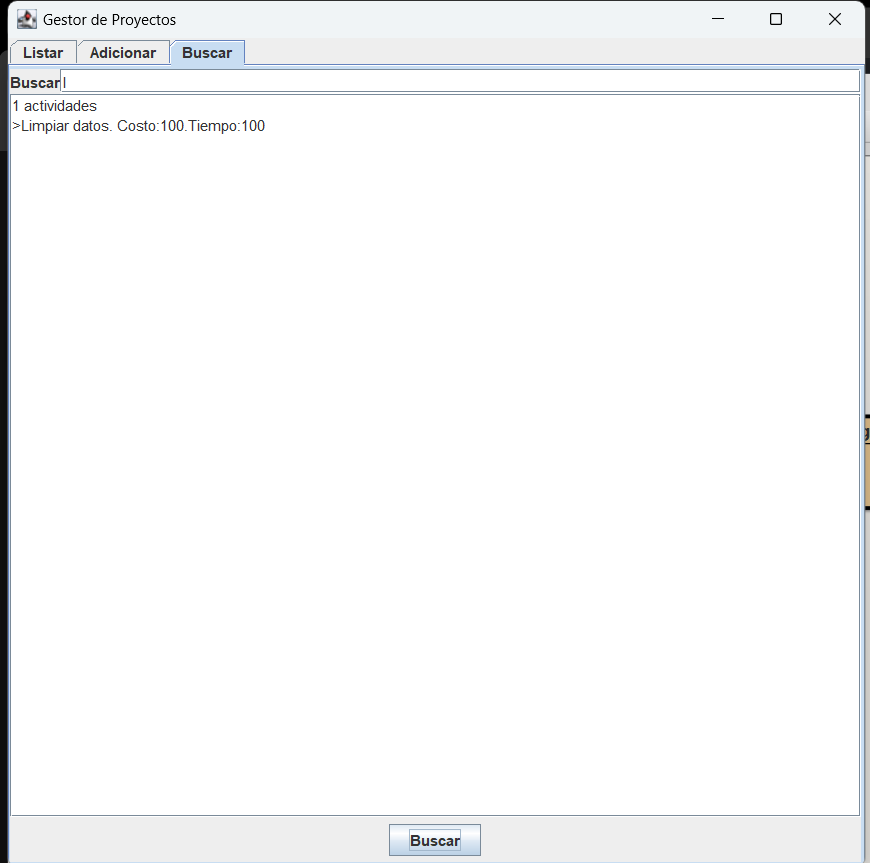
Descripción generada automáticamente

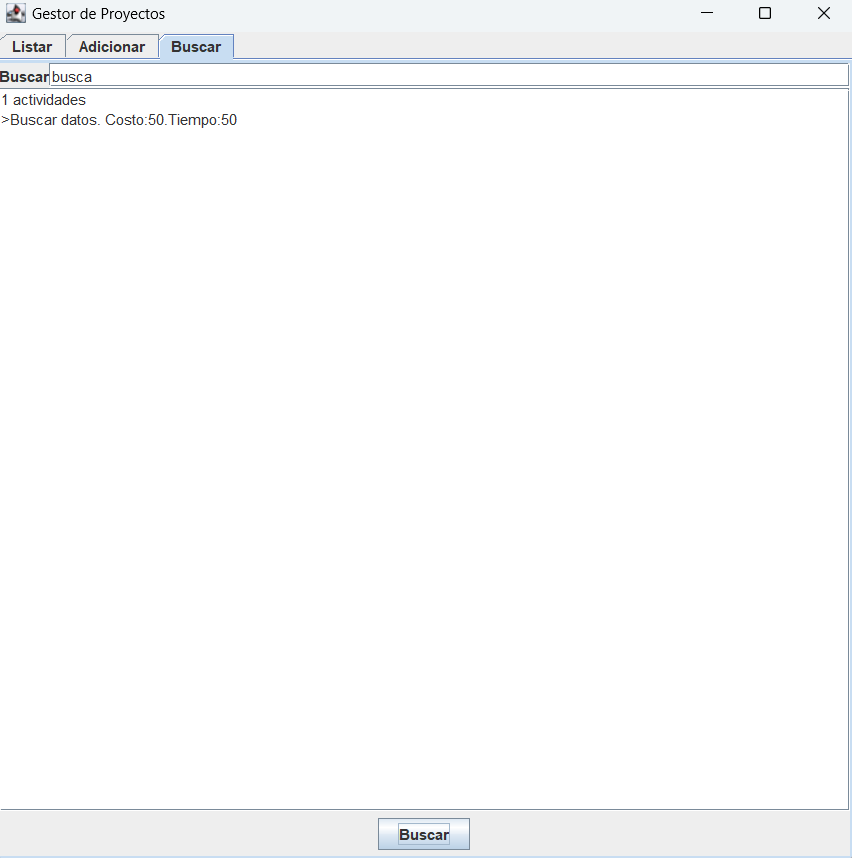


1. ¿Cuál es el error? Soluciónenlo. Capturen los resultados de las pruebas.

Hay varios errores, entre ellos un null pointer exception, generado por varias funciones.

1. Ejecuten la aplicación nuevamente con el caso propuesto. ¿Qué tenemos en pantalla? ¿Qué información tiene el archivo de errores?





El archivo de texto (.log) tiene todos los errores que habían ocurrido hasta antes de implementar la solución.

1. Refactorice la funcionalidad para que sea más amable con el usuario. ¿Cuál es la propuesta?

Hicimos una modificación, ya que el responsable de llamar al método de buscar era el panel de texto, pero con cada mínima modificación que se hiciera en este, posiblemente lanzaba un error, ya que su instrucción era ejecutar buscar cada que hubiera una mínima modificación en el cuadro de texto. Algo que podría llegar a ser molesto para el usuario. Así que, implementamos un botón buscar, que llama a la acción de actionSearch(). Tambien un botón que limpiara el texto escrito por el usuario.

¿Cómo la implementa?

Creando los botones necesarios, añadiendo los botones a la pestaña de buscar, creando los listenner necesarios a cada botón, documentando los oyentes que ya no van a realizar acciones.

**RETROSPECTIVA**

1. ¿Cuál fue el tiempo total invertido en el laboratorio por cada uno de ustedes? (Horas/Hombre)

En total fueron 22 horas.

1. ¿Cuál es el estado actual del laboratorio? ¿Por qué?

Completo.

1. Considerando las prácticas XP del laboratorio. ¿cuál fue la más útil? ¿por qué?

Clean code, testear constanemente, refactorizar

1. ¿Cuál consideran fue el mayor logro? ¿Por qué?

Practica en terminal y manejo de excepciones. Además de la práctica con interfaces gráficas y sus componentes. Manejo y almacenamiento de excepciones para el desarrollador.

1. ¿Cuál consideran que fue el mayor problema técnico? ¿Qué hicieron para resolverlo?

Lo extenso, y hacer ingenieria reversa.

1. ¿Qué hicieron bien como actividades? ¿Qué se comprometen a hacer para mejorar los resultados?

Seguir así