**Laboratorio 4**

**Santiago Gualdron- Sofia Gil**

**Proyectos**

**PRACTICANDO MDD y BDD con EXCEPCIONES**

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.

Actividad es una estructura de árbol n-ario, puesto que una actividad compuesta puede tener 1,2 o más simples y también puede tener 1, 2 o más actividades compuestas; por esto se puede considerar un árbol n-ario, tiene diferentes ramas de actividades compuestas hasta las hojas actividades simples.

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

El proyecto encuentra varios errores al no encontrar un método de una clase de una excepción que se llama ProjectException, donde tenemos que implementarlo junto a los atributos correspondientes que hacen referencia a los mensajes que puede llegar a lanzar.

Hemos elaborado una clase llamada ProjectException que extiende la clase Exception. Adicionalmente, hemos establecido tres constantes de tipo cadena que indican los distintos tipos de excepciones que pueden ser generadas en el laboratorio.

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

Adicionen pruebas análogas a cada una de las dadas para una actividad más profunda (nivel3)

Diagrama

Descripción generada automáticamente

Texto

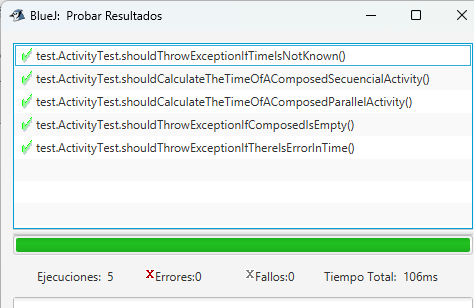
Descripción generada automáticamente

Diagrama

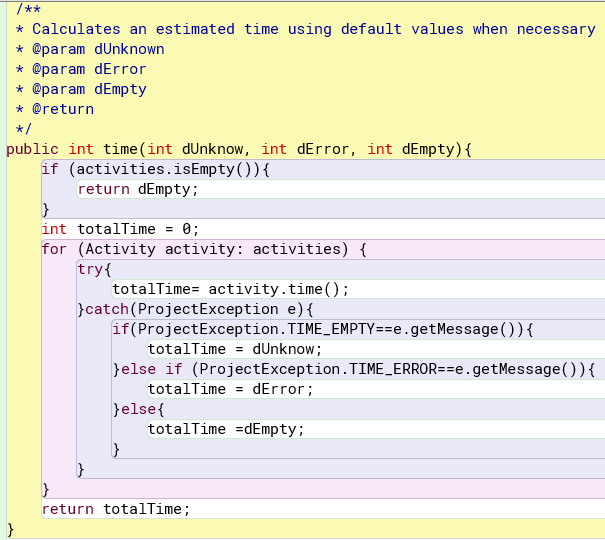
Descripción generada automáticamente

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

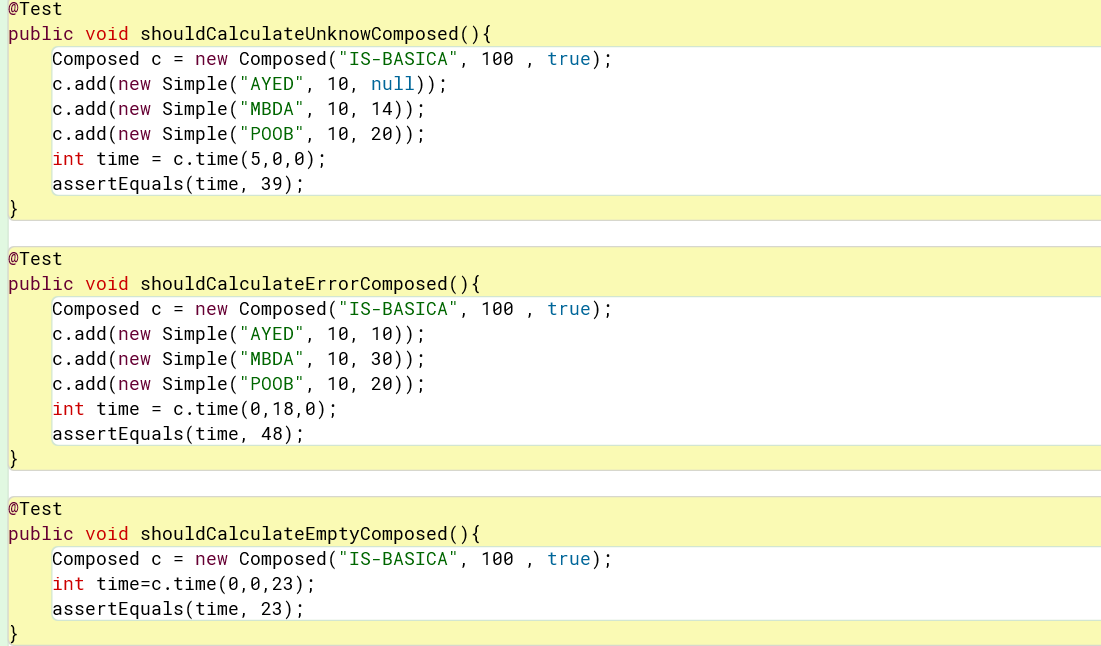
Descripción generada automáticamente



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



Pruebas de unidad:



Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

5. Diseñen, codifiquen y prueben el método time(modality).

Diagrama

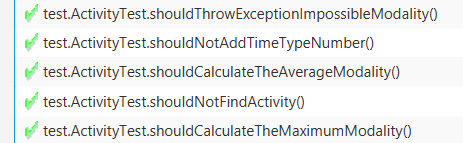
Descripción generada automáticamente

Tabla

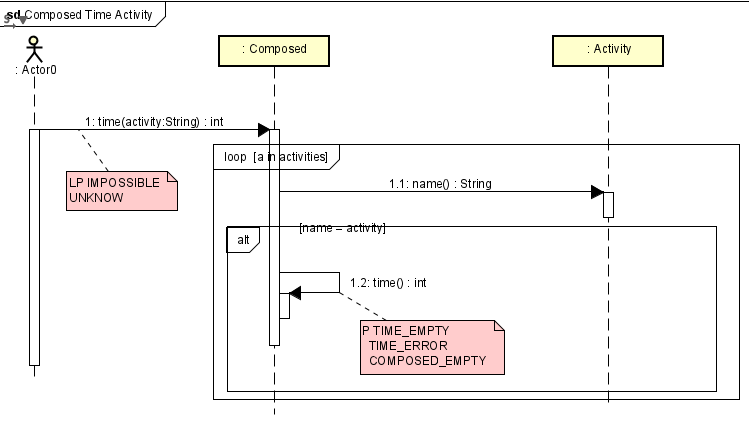
Descripción generada automáticamente con confianza baja

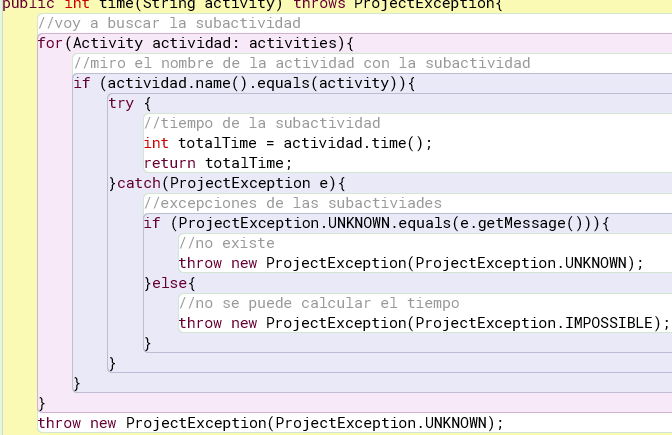
Pruebas de unidad:

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

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





Pruebas de unidad:

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamenteTexto

Descripción generada automáticamenteInterfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

**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?

Tiene 3 archivos los cuales son Project, ProjectManagerGUI, y Log, están organizados los 3 archivos .Java en el directorio llamado Project, deberían estar organizados mediante un directorio principal, directorio src donde tener los otros subdirectorios bin, doc y test lo cual ayuda a la organización del laboratorio.

2. 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?

Tenemos dos packages presentation y domain cada uno respectivamente ProjectManagerGui.java, Project.java y Log.java.

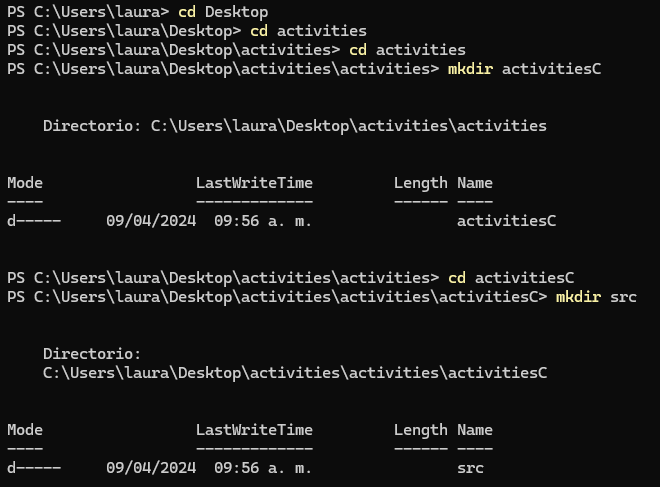
- El sistema tiene 3 clases : Project, ProjectManagerGUI, y Log

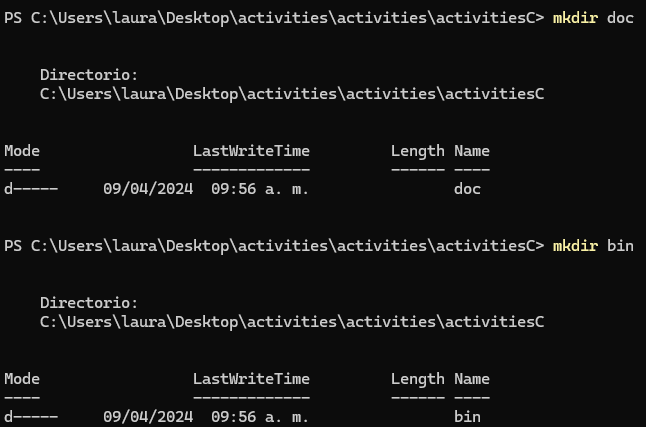
- Están organizadas en directorio Project

- La clase ejecutiva es ProjectManagerGUI

3. 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?

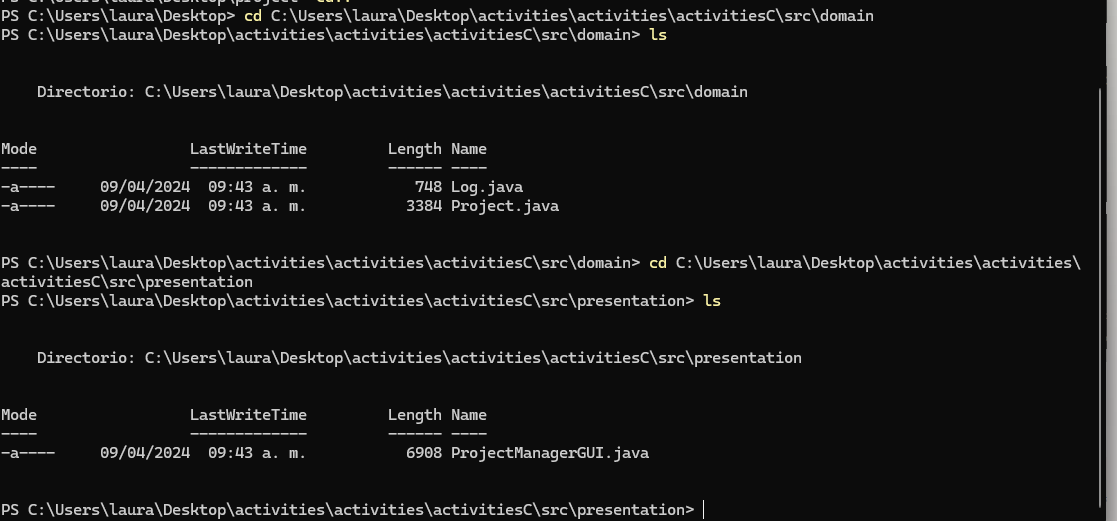
Debemos tener la estructura bin src, doc, los cuales tienen los paquetes presentation y domain, los archivos. Java se den copiar en su respectivo directorio.





Copiar archivos <





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

Para compilar domain

Texto

Descripción generada automáticamente

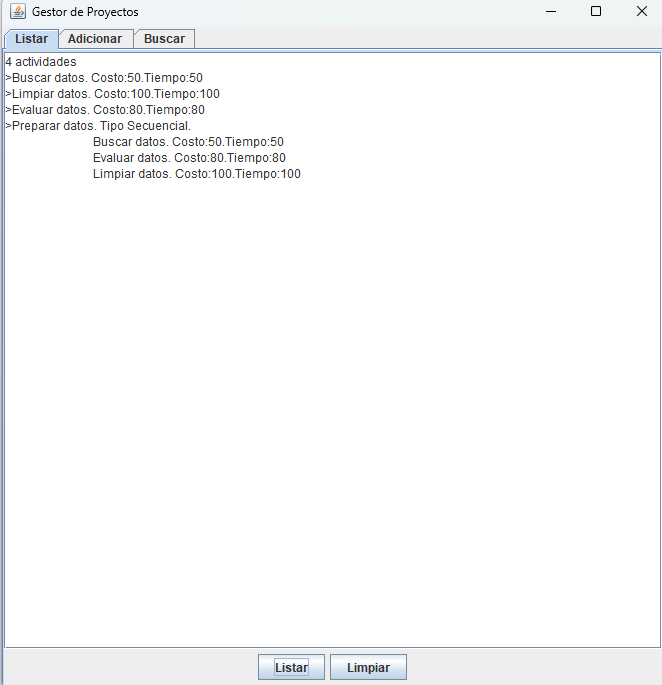
Para compilar presentation

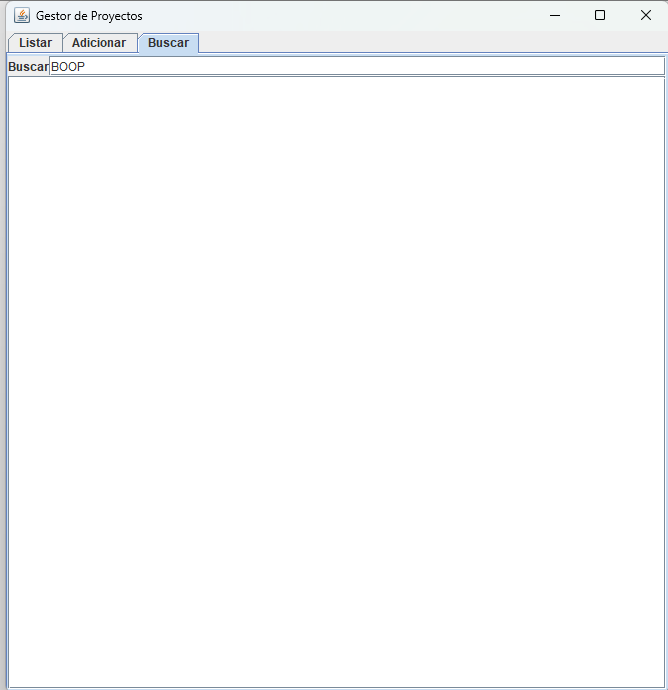


Para ejecutar



Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente



Ofrece Listar, Adicionar, Buscar y limpiar

Las funcionalidades que funcionan Listar, Adicionar y limpiar. La de buscar presenta un error.

5. Revisen el código y la documentación del proyecto. ¿De dónde salen las actividades iniciales? ¿Qué clase pide que se adicionen? ¿Qué clase los adiciona?

Se originan en la clase Project, la cual solicita su inclusión mediante el método addSome(). Posteriormente, este método invoca a los métodos add() para llevar a cabo esta tarea. Dentro de add invoca a las clases simple y composed.

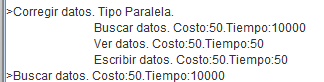
**Adicionar y listar. Todo OK.**

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

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

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente



Aplicación

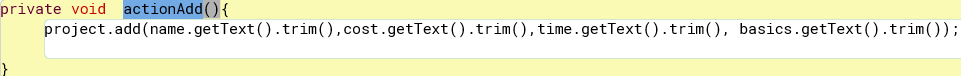
Descripción generada automáticamente

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

Se despliega la lista, donde con esta prueba vemos la lista de actividades, para nosotros no es adecuado ese comportamiento debido a que no es nada eficiente.

2. 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?

Capa presentación:



Capa dominio:

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

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





4. 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?

Capa presentación:

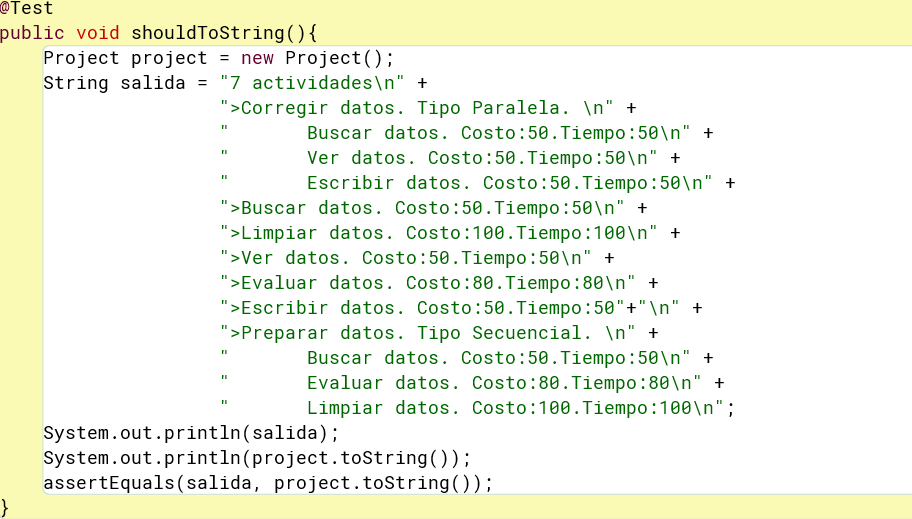


Capa dominio:



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

Pruebas para toString

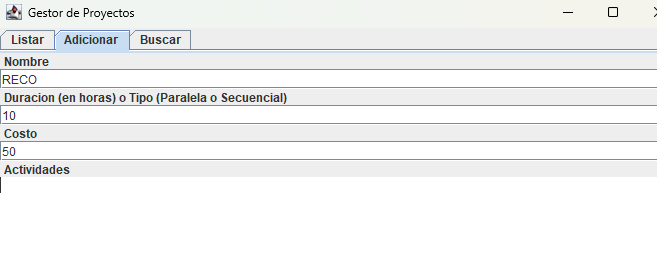


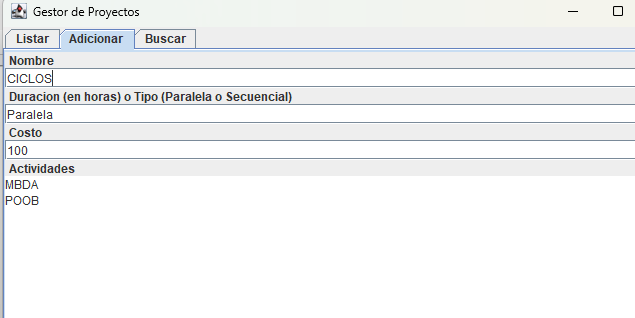
Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

6. Propongan y ejecuten una prueba de aceptación.

Prueba de aceptación: El usuario quiere agregar una actividad simple llamada RECO con un costo de 50 y un tiempo de 10. Luego de esto, el usuario agrega una actividad compuesta llamada CICLOS con un costo de 100, donde en esta actividad el usuario agrega 2 actividades simples MBDA con un costo de 30 y tiempo 5 ; y POOB con un costo de 40 y un tiempo de 7. El usuario solicita ver todas las actividades del proyecto.



  
Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

**Adicionar una actividad. Funcionalidad robusta**

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?

Si el nombre de la actividad ya existe, este mandara un error llamado “REPLICATE\_NAME”

Texto

Descripción generada automáticamente

1. ¿Y si en precio o costo no da un número? ¿si el tipo no es paralelo o secuencial?

Si el costo no es un numero y tampoco esta vacio, implica a que será un string, por lo que mandara la excepción “COST\_IS\_NOT\_NUMBER”

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

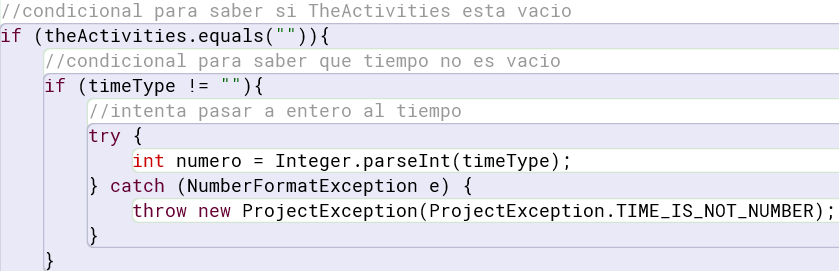
Si el tipo de tiempo para una actividad compuesta es un numero, este mandara el error “TIMETYPE\_IS\_NOT\_STRING”, pero si esta vacio o es una cadena string normal entonces no mandara nada.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

1. Proponga una nueva condición

La condición propuesta es si el tiempo para una actividad simple diferente a un numero, lo cual si es vacio o un numero no hace nada pero si es un string entonces arrojara el error “TIME\_IS\_NOT\_NUMBER”



1. Propongan una prueba de aceptación que genere el fallo.

-Como usuario del sistema quiero que el sistema impida agregar actividades con nombres duplicados para mantener la integridad de los datos del proyecto entonces agregamos actividad con nombre duplicad, entonces el sistema debe mostrar un mensaje de error indicando que el nombre ya está en uso y la nueva actividad no debe ser agregada al proyecto. Agregando ARTE con duración y costo 30.

- Como usuario del sistema quiero que el sistema valide los datos al agregar una nueva actividad para asegurar la integridad de la información del proyecto mediante la agregación de actividades con datos inválidos. DANZA con costo idhia y tiempo fosdh

-Como usuario del sistema quiero que el sistema valide los datos ,al agregar una nueva actividad para asegurar la integridad de la información del proyecto donde agrega actividad con datos válidos dado que estoy en el sistema de gestión de proyectos cuando intento agregar una nueva actividad con datos válidos

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.

Diagrama

Descripción generada automáticamente

El método que debería lanzar las excepciones debería ser add, al utilizar la practica xp de refactorización, los 2 metodos privados “noRepetido” y “comprobate” ayudaran a mirar todas las excepciones que pueden aparecer en la entrada del mismo.

Los métodos que deben propagarlo y atenderlo deberían ser los add de compuesto y simple, estos ayudaran a colocar valores predeterminados para los errores que pueden llegar a surgir.

1. Construya la solución propuesta. Capture los resultados de las pruebas de unidad.

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Interfaz de usuario gráfica, Aplicación

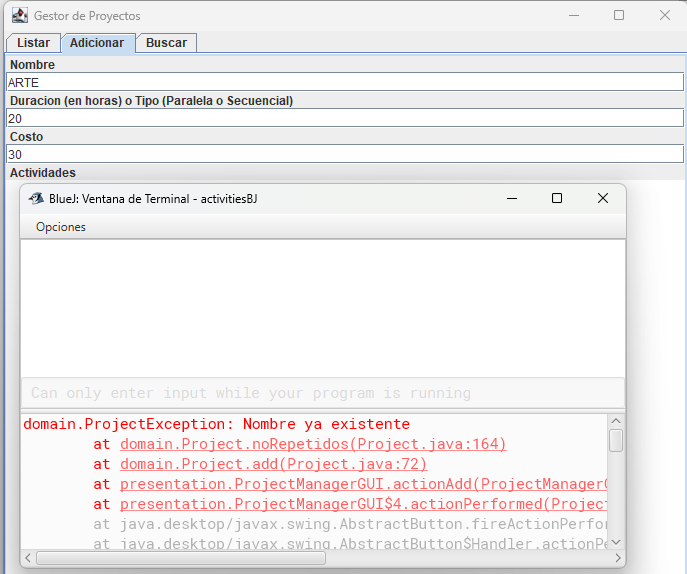
Descripción generada automáticamente

Texto

Descripción generada automáticamente

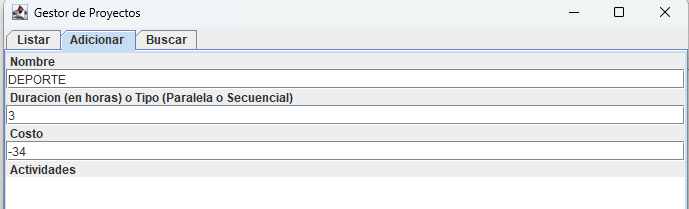
1. Ejecuten nuevamente la aplicación con el caso de aceptación propuesto en 1. ¿Qué sucede ahora? Capture la pantalla.

Ahora genera la excepción dada donde da la información al usuario y al desarrollador.



Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente



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

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?

Interfaz de usuario gráfica, Aplicación, Word

Descripción generada automáticamente

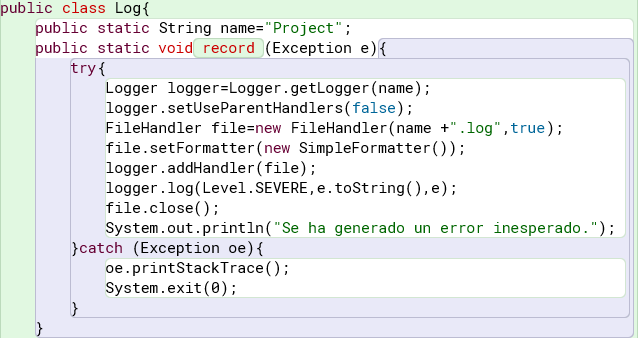
Se busco la actividad de inicio índice, pero en el buscador sale el orden inverso, además se genera un error a causa de que no se puede buscar la actividad por la letra

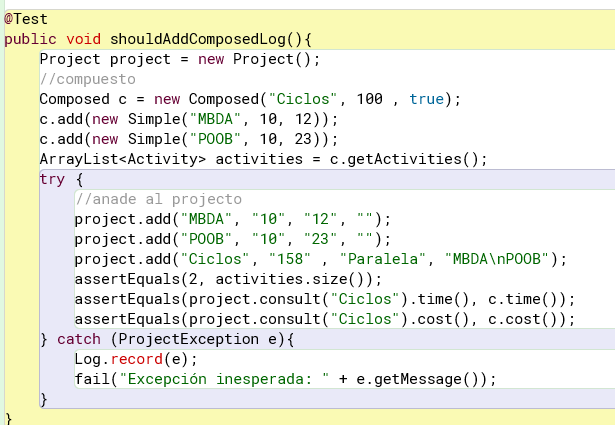
Los que deben conocer esta información son los colaboradores de las actividades y desarrolladores que ayudan con el sistema, y los que no deben mirar el error deben ser los usuarios.

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

* Verifica si la variable logger no es nula. La variable logger debe ser una instancia de una clase que proporciona funcionalidad de registro, como un java.util.logging.Logger. Si logger es nulo, el método no hace nada y termina.
* Utiliza el objeto logger para registrar la excepción e con un nivel de severidad (Level) de SEVERE. El nivel SEVERE generalmente se utiliza para registrar errores graves que pueden causar un fallo en la aplicación. La cadena e.toString() se utiliza como mensaje en el registro, y el objeto e se pasa como un parámetro adicional. Esto permite registrar información adicional sobre la excepción, como su pila de llamadas (stack trace).

3. 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.





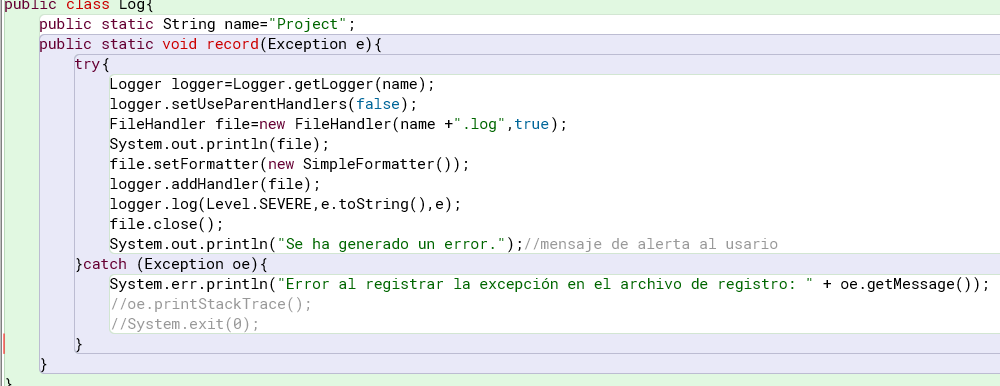
4. Ejecuten nuevamente la aplicación con el caso propuesto en 1. ¿Qué mensaje salió en pantalla? ¿La aplicación termina? ¿Qué información tiene el archivo de errores?

El mensaje que salió en pantalla es: "Se ha generado un error." Y la aplicación se termina después de imprimir el mensaje de error en la pantalla y la información que tiene el archivo de errores debe tener la información sobre la excepción que se produjo, con los detalles estipulados.

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

Después de experimentar una excepción no contralada, no es apropiado que la aplicación siga funcionando. La decisión de continuar o detenerse dependerá del tipo de error y como afecta el funcionamiento general de esta. Factores como la gravedad del error, la integridad de los datos y la experiencia del usuario determina si es seguro o no continuar o detener con la ejecución para evitar daños adicionales o sucesos impredecibles.

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



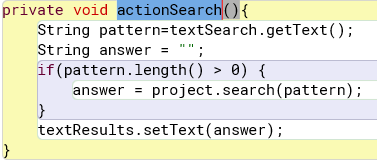
La modificación principal realizada es en el método record. Ahora, este método acepta cualquier excepción (Exception e) y la registra utilizando el objeto logger. Esto garantiza que todas las excepciones sean manejadas adecuadamente y registradas en el archivo de registro. El archivo de registro se ha configurado para guardar los detalles de la excepción, incluyendo su mensaje y su pila de llamadas.

Con esta modificación, cualquier error que ocurra en la aplicación será registrado en el archivo de registro, lo que permite un seguimiento detallado de los errores y facilita su resolución. La aplicación manejará todas las excepciones de esta manera, asegurando un manejo adecuado de los errores en todo momento.

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

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?

Capa de presentación:



Capa de dominio:

Texto

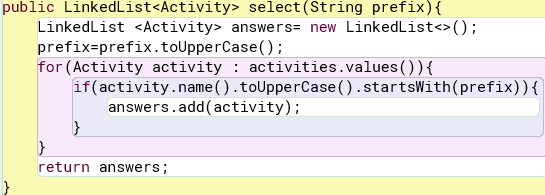
Descripción generada automáticamente con confianza media

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



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

La causa del error radicó en la función "select," que generaba una lista vacía, lo que provocó un fallo de tipo "null" al buscar, ya que la lista requerida no se encontraba disponible.



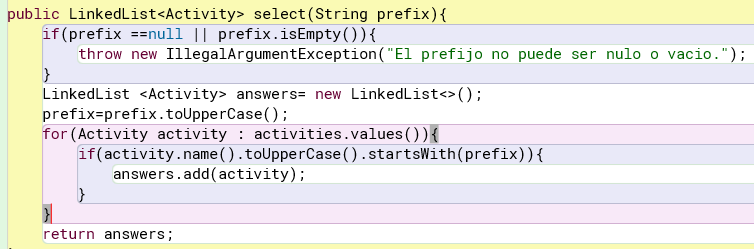


4. Ejecuten la aplicación nuevamente con el caso propuesto. ¿Qué tenemos en pantalla? ¿Qué información tiene el archivo de errores?

Debido a que en el método de búsqueda no tiene ninguna excepción, no se registra nada y solo que da en si la solución.

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

La refactorización del método “select” en la clase Project implicó ajustar la implementación para lanzar IllegalArgumentException cuando se detecta un prefijo nulo o vacío proporcionado por el usuario, asegurando un manejo efectivo de argumentos inválidos mientras se mantiene la funcionalidad principal del método, que es devolver una lista de actividades que coinciden con el prefijo dado. Esto mejora la experiencia del usuario al proporcionar mensajes de error claros y consistentes.



**RETROSPECTIVA**

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

13 horas por persona

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

El laboratorio esta completado en una gran parte, omitiendo algunos detalles de pruebas de aceptación o algún diagrama de secuencia corto

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

Utilizamos las prácticas de programación a pares, utilizadas especialmente para los métodos de time() y sus sobrecargas, además de utilizar la practica de cada vez que encontramos un bug, le creamos una prueba de unidad que confirme que no se volverá a cometer ese mismo error.

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

Compilar todos los casos de throws que mandan excepciones y a parte completar todo el miniciclo de LOG, ya que tuvimos problemas al entender inicialmente como poder hacer los cambios de los archivos por medio de la consola

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

El mayor problema técnico fue obtener el archivo .log donde guardábamos todos los throws lanzados por el programa, además de entender bien cuales eran las ideas y estructurarlas para el miniciclo de funcionalidad robusta

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

Hicimos bien el tema de la comprensión y desarrollo de todos los códigos pedidos por el laboratorio, colocando las practicas para atrapar, lanzar y propagar los throws, además de mejorar los diagramas de secuencia dados en el astah

Nos comprometemos a mejorar el tema de la calidad en el tiempo y la organización de archivos y carpetas.