**Laboratorio 04**

**Integrantes:**

**Sebastián Cardona Parra**

**Diego Alexander Cárdenas Beltrán**

**Universidad Escuela Colombiana de Ingeniería Julio Garavito**  
**Programación Orientada a Objetos**  
 **María Irma Rozo**

**13/04/2024**

Texto

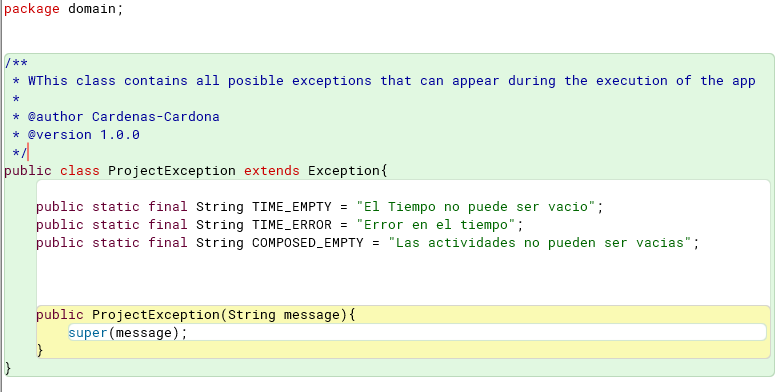
Descripción generada automáticamente



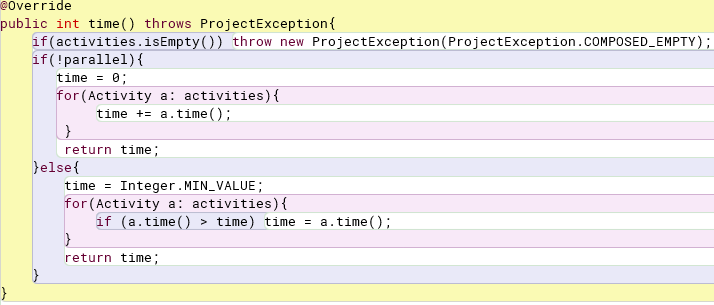
Es un árbol no binario, cuya raíz es una actividad compuestas y hojas actividades simples o compuestas.



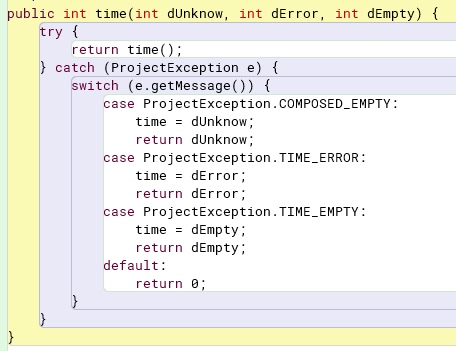
Debido a que el proyecto no cuenta con la clase ProjectException que es el que controla los posibles errores, por lo que hace falta crear las clases exception.



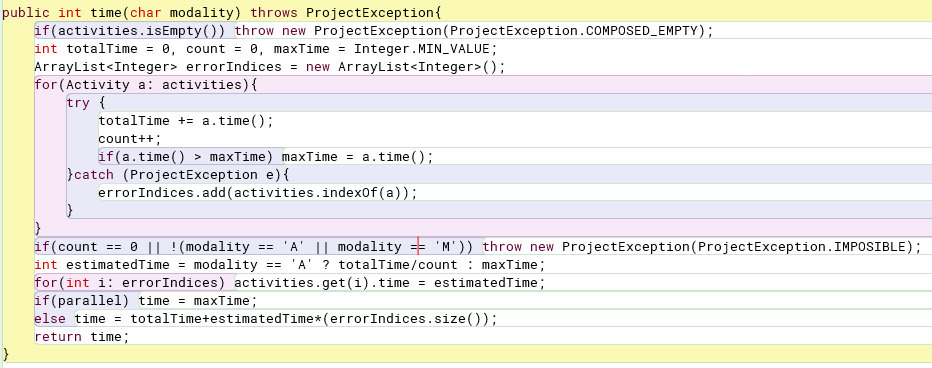




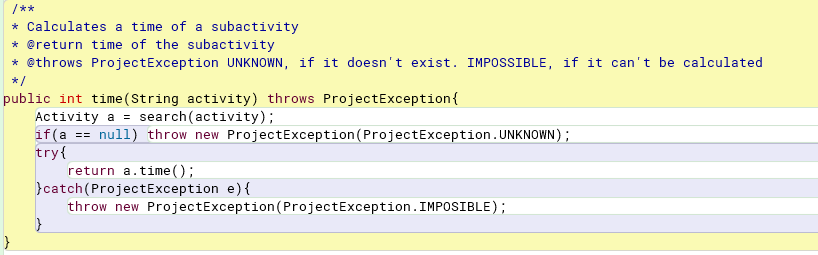


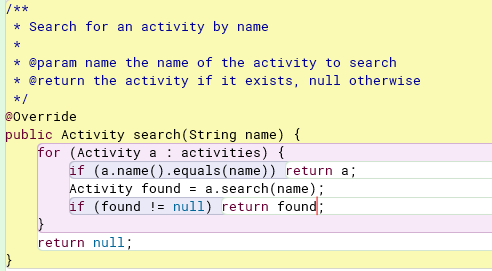


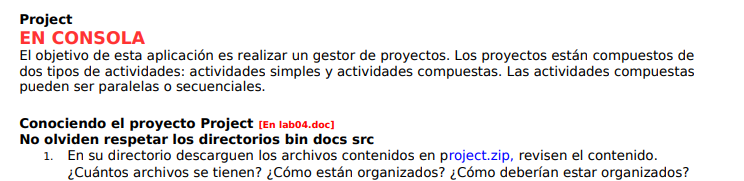






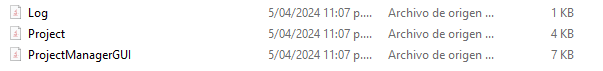






El contenedor tiene 3 archivos de tipo .java, en ellos está contenido la clase Project, que es quien maneja la lógica del gestor de proyectos, está ProjectManajerGUI, en él está la interfaz de usuario del gestor de proyectos y finalmente está la clase Log, y en esta están los registros para el mantenimiento y depuración del programa.

Deberían estar organizados de tal manera que estén en el directorio src.





Tenemos un total de 2 paquetes **domain** y **presentation**, además si queremos podemos añadir el paquete **test**

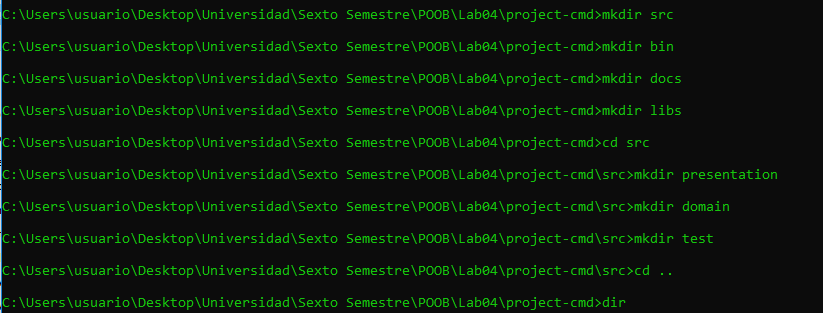
Para el proyecto total tenemos un total de 6 clases sin contar la clase de testeos y de excepciones

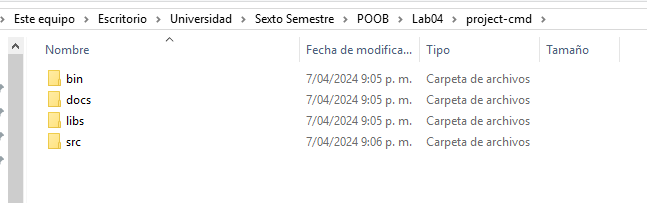
La clase ejecutiva será ProjectManajerGUI

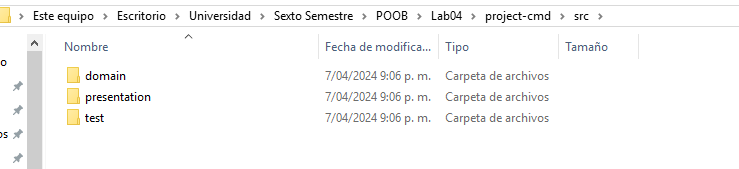


Debemos crear los directorios bin, src y docs, dentro de src estarán los paquetes presentation, domain y test, por lo cual en general en src van todas las clases .java

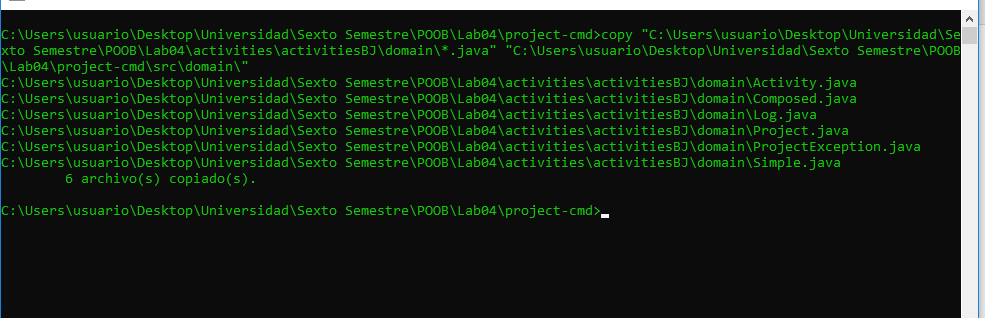
Estas clases originalmente se encuentran en el archivo de blueJ, por lo que hay que moverlas a la carpeta que creamos llamada project-cmd

Inicialmente crearemos los directorios con el comando de mkdir  






Procedemos a copiar las clases a src:  
copy "C:\Users\usuario\Desktop\Universidad\Sexto Semestre\POOB\Lab04\activities\activitiesBJ\domain\\*.java" "C:\Users\usuario\Desktop\Universidad\Sexto Semestre\POOB\Lab04\project-cmd\src\domain\"

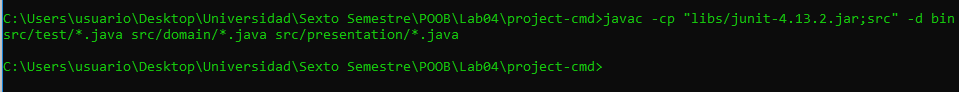


Repetimos con los demás paquetes

Y así ya tenemos los archivos .java en cada directorio correspondiente

Para compilar los archivos tengamos en cuenta que en libs, ya están los .jar del frameWork de Junit y hamcrest que funcionan para compilar y ejecutar las pruebas

javac -cp "libs/junit-4.13.2.jar;src" -d bin src/test/\*.java src/domain/\*.java src/presentation/\*.java



De esta manera ya se compile el Proyecto y está guardado como .class en el directorio bin

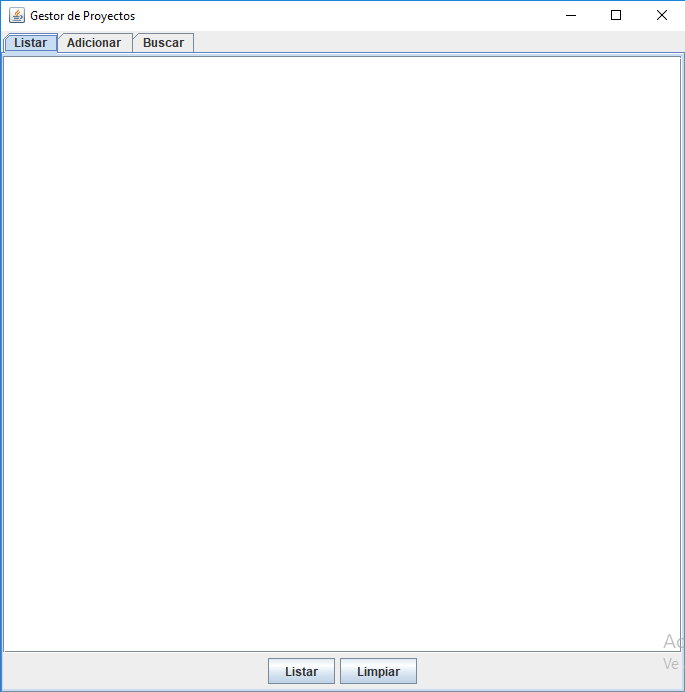
Ahora bien, vamos a crear los archivos para construir la documentación

javadoc -d docs -sourcepath src -subpackages domain:presentation -exclude test

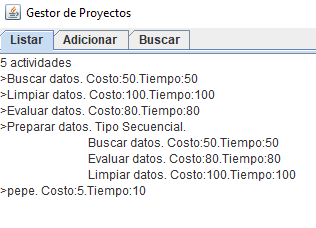
De esta manera el Proyecto queda listo para ejecutar usando la instrucción

java -cp bin presentation.ProjectManagerGUI





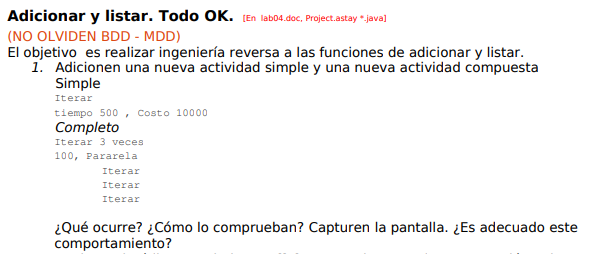
El proyecto permite editar, adicionar y buscar una actividad



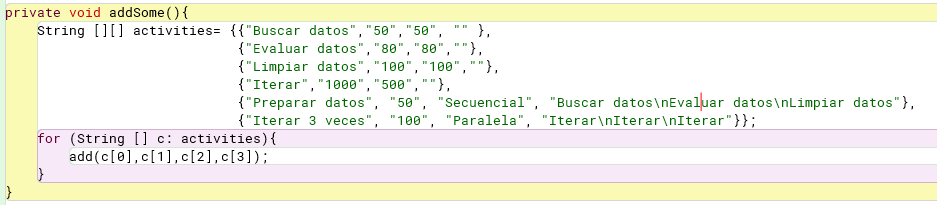
Funciona correctamente adicionar, listar, limpiar la pantalla, pero buscar no funciona bien.



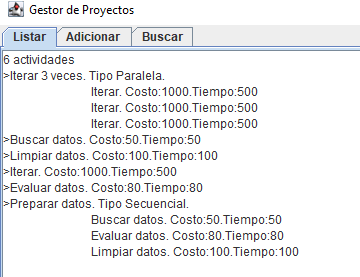
Las actividades iniciales nacen de la clase Activity, la clase Project pide que se adicionen usando el método addSome, esta que a su vez llama al método add para que añada cada actividad inicial.



Lo haremos usando el método de addSome



Al compilar y usar la funcionabilidad de listar aparecen todas las nuevas actividades creadas:



No es adecuado este comportamiento, ya que estamos usando tiempos para las actividades simples inválidos y mayores a 24, por lo que, si quiero calcular los tiempos de las actividades compuestas, nos va a lanzar un error.

Otro error es que nada me impide que haya dos actividades con el mismo nombre y esto no se debe poder pasar.

Que sucede cuando dije que una actividad era paralela o secuencial (compuesta) pero no le pongo sub-actividades, el código lo va a reconocer como una actividad simple, pero van a existir errores respecto a tiempo y la palabra paralelo o secuencial.

Lo mismo sucede si diseño una actividad simple (con tiempo) pero le pongo actividades, el código lo reconocerá como una actividad compuesta y habrá conflictos con el entero del tiempo y la palabra paralelo o secuencial.

Para una actividad compuesta, nada me impide que ponga otra palabra distinta a secuencial y paralelo, si pongo ejemplo “pepito” lo reconoce como un paralelo y es incoherente, lo mismo si pongo “Carlos”, lo reconocerá como secuencial

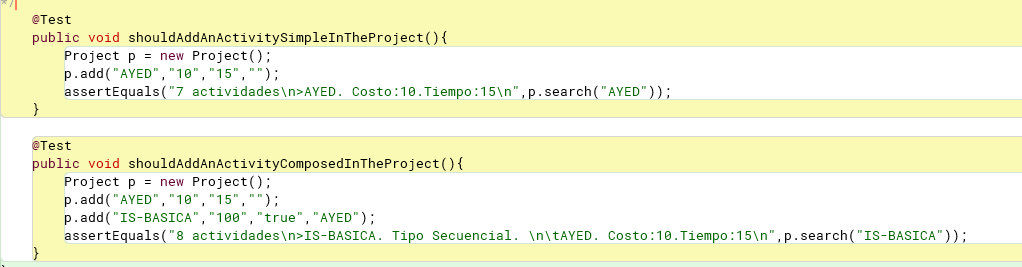
Esos párrafos resaltados creo que me equivoqué, van más adelante

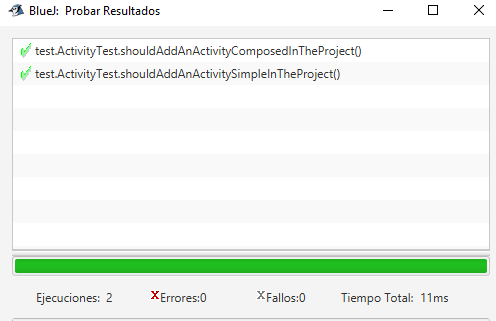


En la capa de dominio, el encargado de añadir actividades es el método add

En la capa de presentación, el método encargado de añadir actividades es ActionAdd



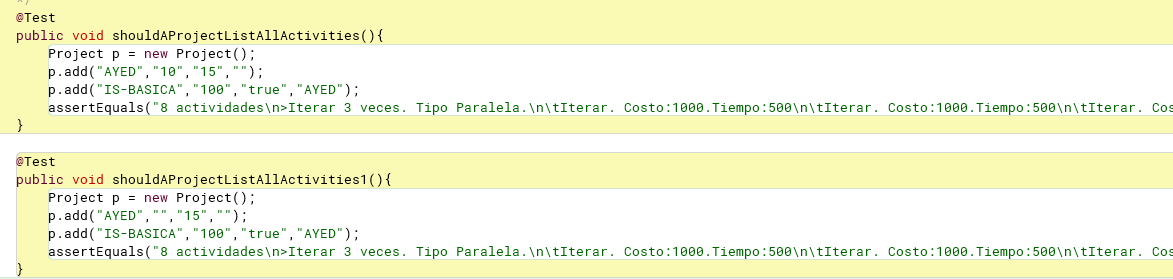


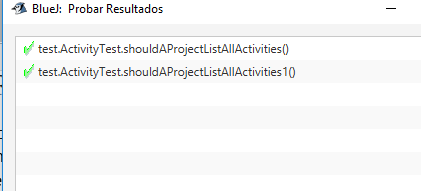


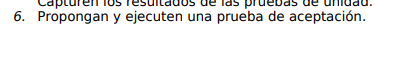


En la capa de domain el método responsable de listar todas las actividades es toString en la clase de Project, este a su vez usa el método de data que toma un linkedlist con las actividades que quiere listar. En la capa de presentación, el método encargado es actionList

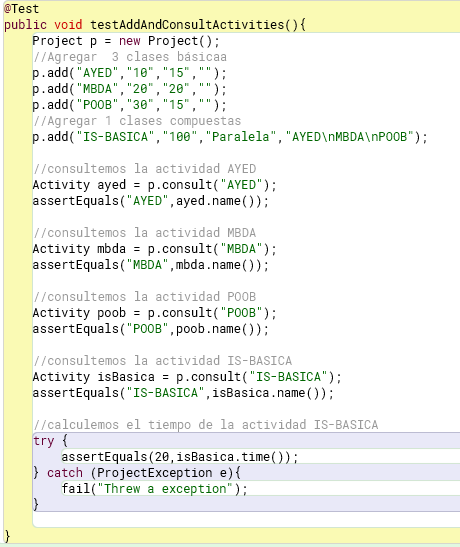


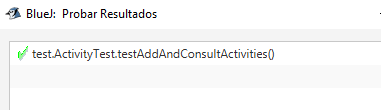


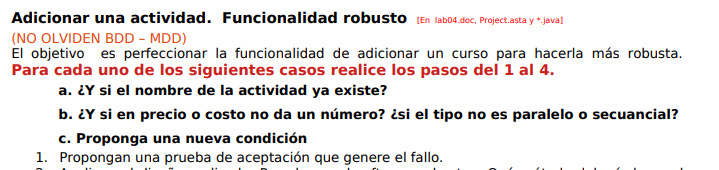




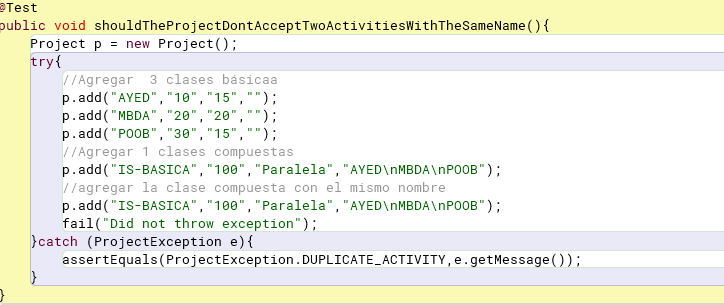
Crearemos una nueva actividad compuesta por tres simple, después de ello verificaremos que, al consultar, verdaderamente existan y calculamos que el tiempo de la actividad compuesta sea correcto

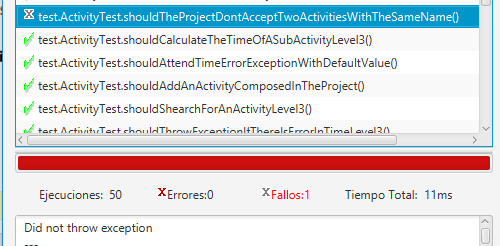






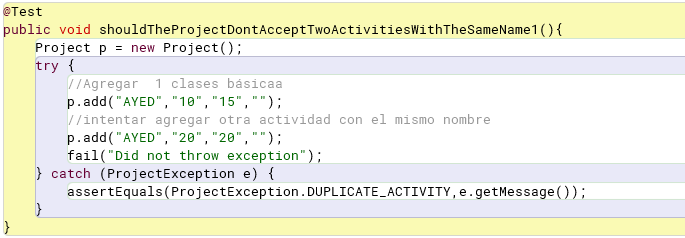
1. **Nombre duplicado**

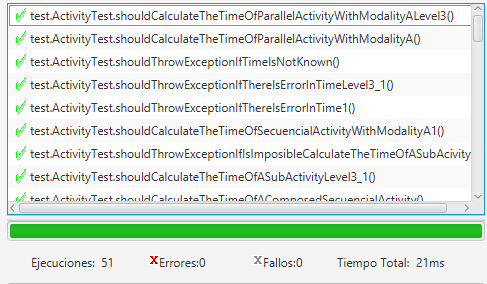






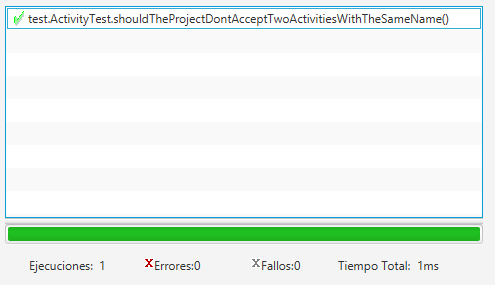
El método que debería lanzar la excepción es el método de add en Project, así mismo este método debe lanzarla, quien la atiende son las pruebas y aquello métodos que vayan relacionados a adicionar en la capa de presentación que es actionAdd.



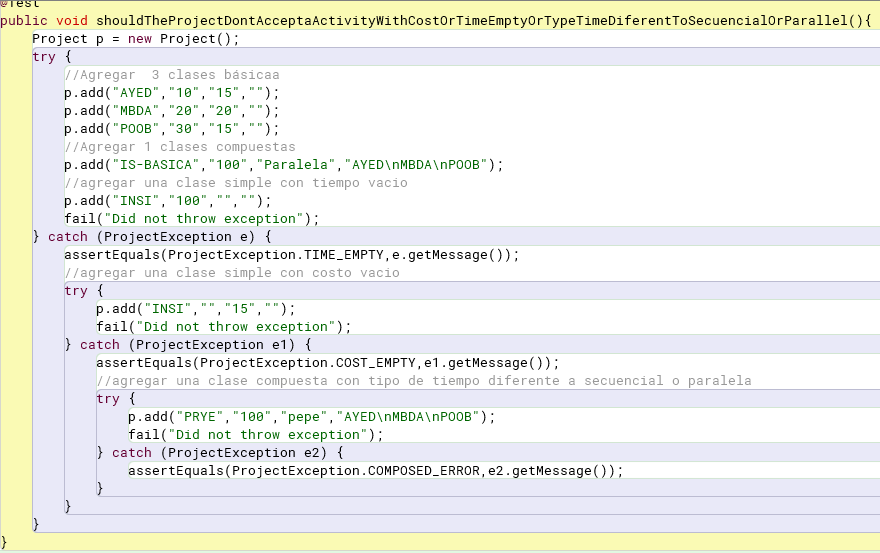


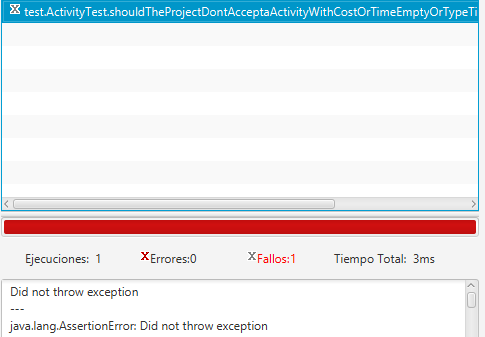
Ahora si se impide que se cree la nueva actividad repetida y se lanza el error.



1. **Y si el precio o costo no se da un número o un tipo desconocido**





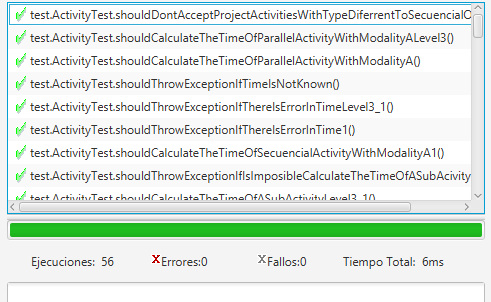




El método que debería lanzar y propagar la excepción de agregar un costo o precio nulo en una actividad simple o compuesta es el método de add de Project y se atiende son los testeos o en la capa de presentación.

Para la excepción de verificar si el tipo es paralelo o secuencial, se lanzará y propagará desde el método de add en Project, para atenderse en los testeos y capa de presentación





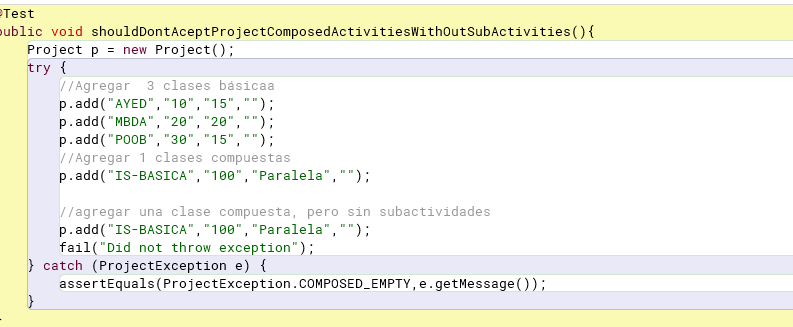


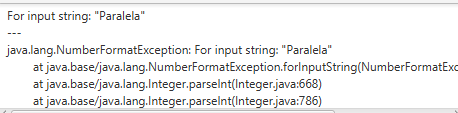
Ahora si se pasa la prueba de aceptación ya que no acepta los errores que inducimos



1. **Que sucede cuando dije que una actividad era paralela o secuencial (compuesta) pero no le pongo sub-actividades, el código lo va a reconocer como una actividad simple, pero van a existir errores respecto a tiempo y la palabra paralelo o secuencial.**





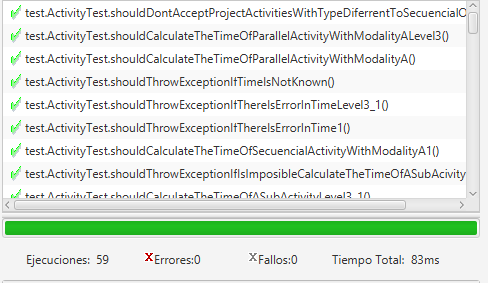


Se genera el NumberFormatException y es como hablábamos al inicio cuando decíamos que había conflicto entre la palabra y el número



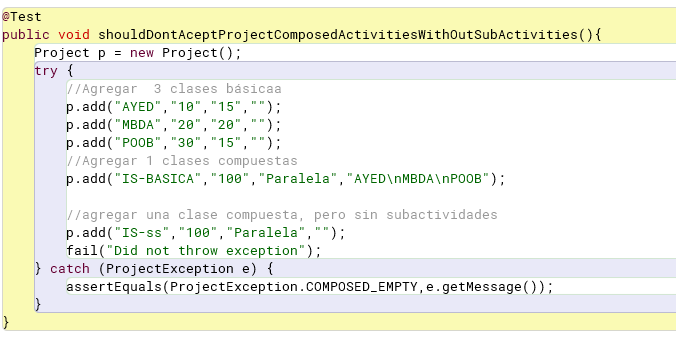
Quien debe lanzar y propagar es el método de add, y la atiende los testeos y la capa de presentación

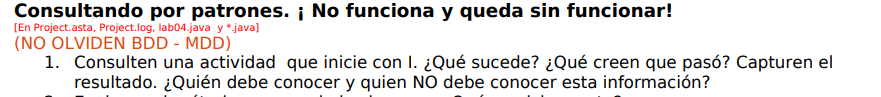


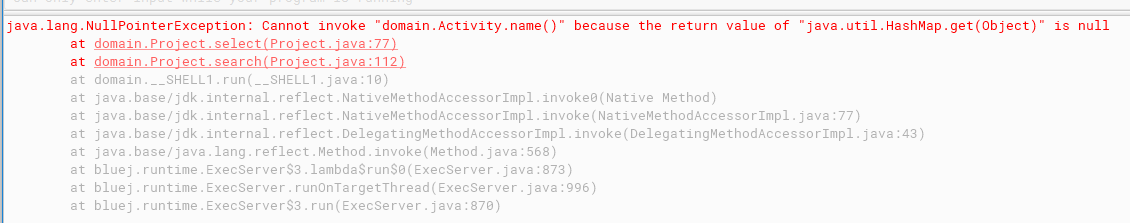




Ahora la prueba de aceptación corre sin problemas







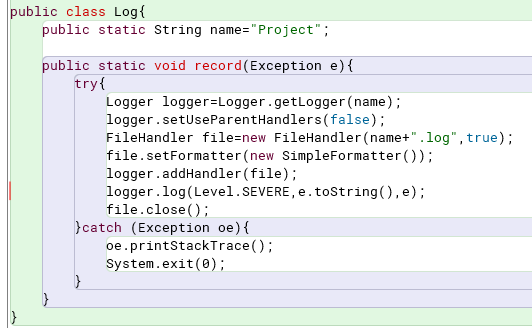
Se lanza un error que dice que el resultado es nulo, pues no se encuentra dentro del hashMap de actividades

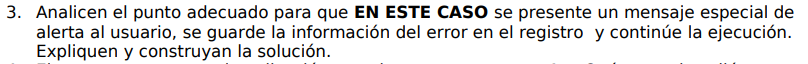
Este error se debe a que estoy usando un get(i), donde i es un entero, no obstante, las llaves del hashMap de actividades es un string, entonces debería ser get(nombre)

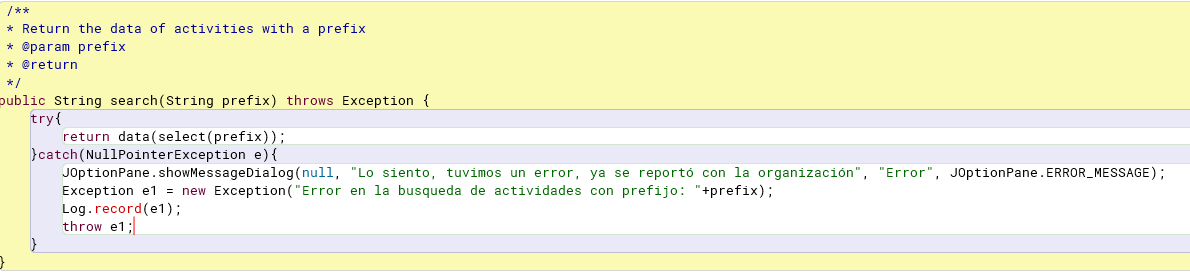
El desarrollador debe conocer dicha información ya que es un error dentro de la lógica y del código del proyecto, algo que no entiende el usuario por tanto no debe tener acceso a esta información



Este método se usa para guardar en un archivo .log todas las excepciones posibles que emerjan de Project

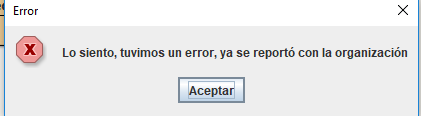




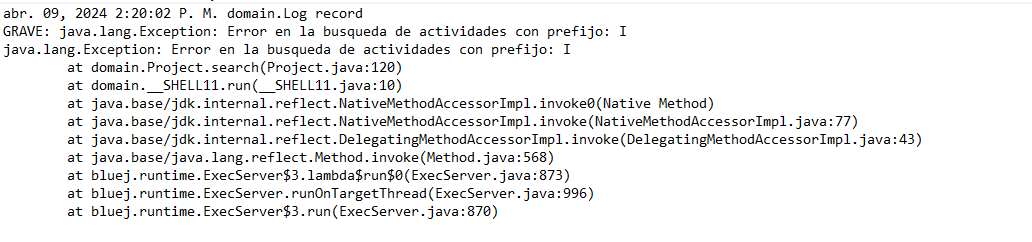


De esta manera siempre se va a generar error, intenta ejecutar data, como va a lanzar error, de tipo NullPointerException, vamos a lanzar el mensaje de error y guardamos el error en el arhivo.log







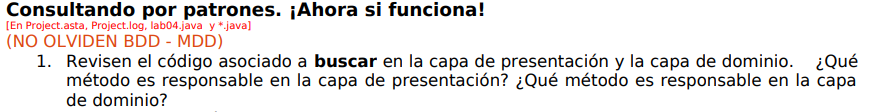


Se re factorizó el código como se ve en pantalla, simplemente lanzamos el mensaje y guardamos el error en el archivo .log



Depende, si el error, fue como este de tipo lógico y de código, es decir, si el error, fue por culpa del desarrollador, la ejecución no debería seguir, si el error, fue por una mala entrada del usuario, simplemente debemos mostrarle al usuario su error y seguir la ejecución esperando que el usuario corrija su entrada  

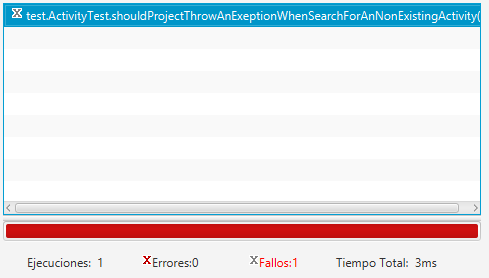
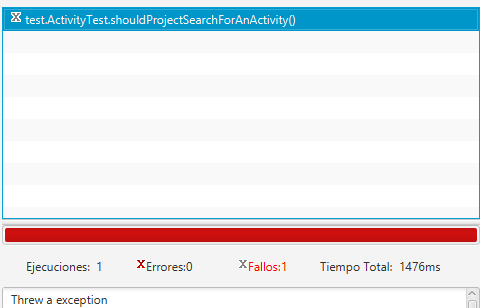

Cuando se lanza una excepción, antes de lanzarla, vamos a guardarla en el log, se esa manera se crea el registro, después de ello lanzamos ahora si la excepción. Todas las excepciones de add y search, se atenderán en la capa de presentación, donde se lanzará un mensaje al usuario



En la capa de presentación la responsable de buscar una actividad es actionSearch

En la capa de dominio la responsable de buscar una actividad es search

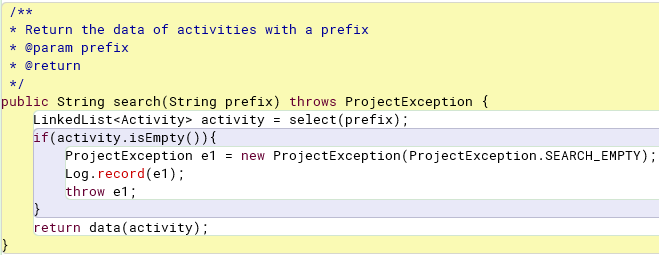


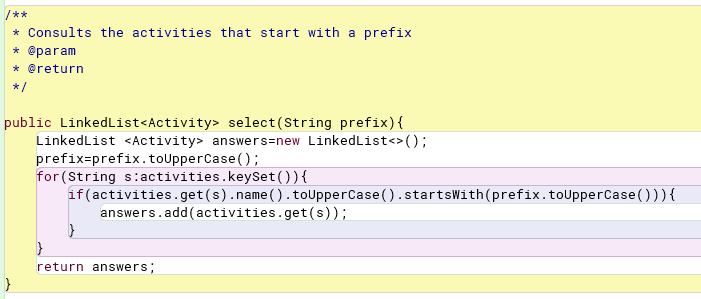


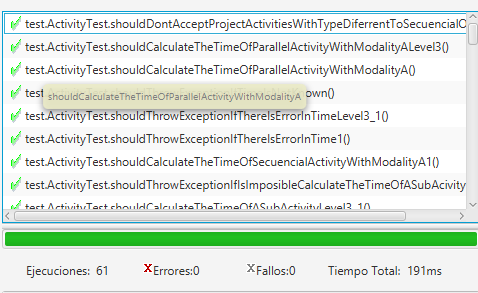




Respecto al error lógico del método search que era sobre que se iteraba sobre un entero y no sobre un nombre String que es la clave del hashMap, lo arreglamos para que no diera el error de NullPointerException sino de SEARCH\_EMPTY, que se lanza cuando no hay ninguna actividad con dicho prefijo.

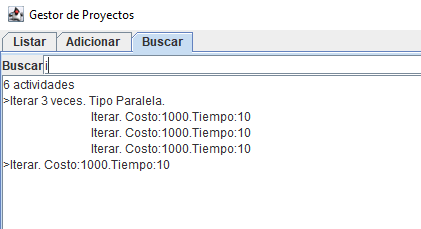






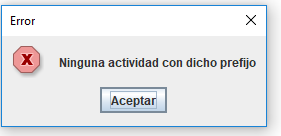


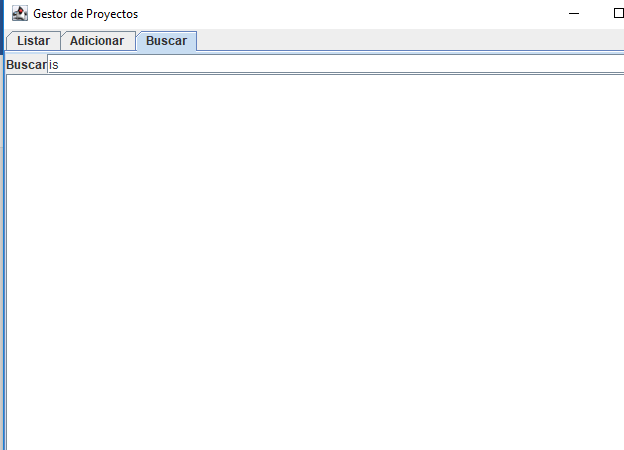
Si ejecutamos el código nuevamente, es decir buscar con actividades que inicien con I tenemos lo siguiente:



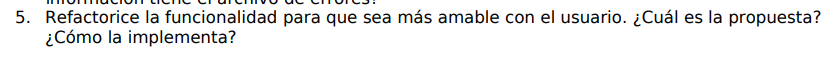
No obstante, cuando agregamos un “is” por ejemplo ya se manda el mensaje de error, pues dicha actividad con dicho prefijo no existe, por lo cual se mandará el mensaje de error al usuario y se guardará en el archivo de Log











Decidimos añadir los mensajes de error cada vez que el usuario comete una equivocación, entre ello más limitaciones y más filtros respecto a equivocaciones al poner el costo y tiempo, ya que respeta los límites de estos.

## **RETROSPECTIVA**

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

El total de tiempo invertido por hombre fue de 16 horas

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

El estado del laboratorio es terminado, ya que trabajamos en conjunto y pudimos terminar el laboratorio como fue esperado

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

La práctica xp más útil a la hora de realizar el laboratorio fue la programación a pares, puesto que así trabajamos de una forma coordinada y eficiente.

También aplicamos la simplicidad del código para entenderlo mejor y usamos las pruebas unitarias para que cuando re factorizáramos el código verificar que siguiera funcionando bien

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

El mayor logro fue poder hacer el laboratorio en su totalidad

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

El mayor problema técnico fue manejar y pensar bien las excepciones, ya que había algunas que se escapaban y otras que no sabíamos dónde lanzarlas atenderlas o propagarlas

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

Hicimos bien las pruebas de unidad, ya que tuvimos bastantes pruebas de unidad y todas compilaron, también nos coordinamos bien y trabajamos en equipo