**PROGRAMACIÓN ORIENTADA A OBJETOS**   
**Excepciones**   
**2019-02**   
**Laboratorio 4/6**

**EQUIPOS**   
**EN BLUEJ**  
**PRACTICANDO MDD y BDD con EXCEPCIONES**   
**[En lab04.doc, actividad.asta y BlueJ actividads]**   
En este punto vamos a aprender a diseñar, codificar y probar usando excepciones. Para esto se van a trabajar algunos métodos de la clase Actividad y la excepción ActividadExcepcion

**1. En su directorio descarguen los archivos contenidos en actividades.zip, revisen el contenido y estudien el diseño estructural de la aplicación.**

**2. Expliquen por qué el proyecto no compila. Realicen las adiciones necesarias para lograrlo.**  
Las clases ActividadSimple y ActividadCompuesta están sobre-escribiendo un solo método abstracto y son 3 métodos abstractos de la superclase, sobre-escribimos todos los métodos e hicimos que retornaran 0 por el momento.

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

**4. Dada la documentación, diseñen, codifiquen y prueben el método creditosDefinidos().**

Decimos que una actividad bien definida ademas de las condiciones que generan

**5. Documenten, diseñen, codifiquen y prueben el método creditos(String). En este método vamos a incluir dos nuevos casos especiales: 1) no existe una actividad con esa descripción y 2) existen dos actividades con la misma descripción**

**Plan Colombia**   
**EN CONSOLA**  
El objetivo de plan es gestionar la información de las materias del nuevo plan de estudios. Conociendo el proyecto [En lab04.doc]   
No olviden respetar los directorios bin docs src

**1. En su directorio descarguen los archivos contenidos en plan.zip, revisen el contenido. ¿Cuántos archivos se tienen? ¿Cómo están organizados? ¿Cómo deberían estar organizados?**

Tiene 3 archivos .java, los cuales no están organizados en sus respectivas carpetas y deberían estar en la carpeta src.

**2. Preparen los directorios necesarios para ejecutar el proyecto. ¿qué estructura debe tener? ¿qué instrucciones debe dar para ejecutarlo?**

Debe tener la estructura:

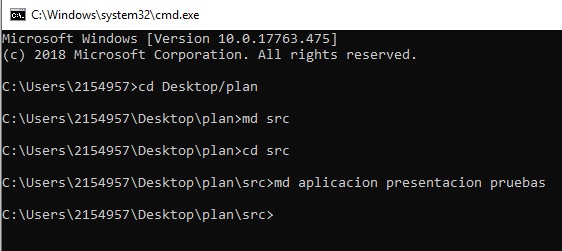
plan

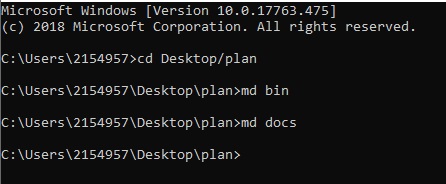
src

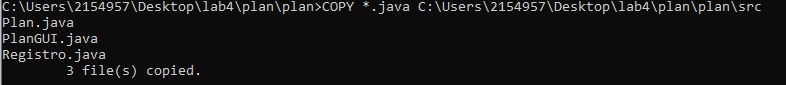
bin

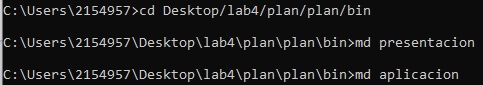
docs

para ejecutarlo desde consola usamos:







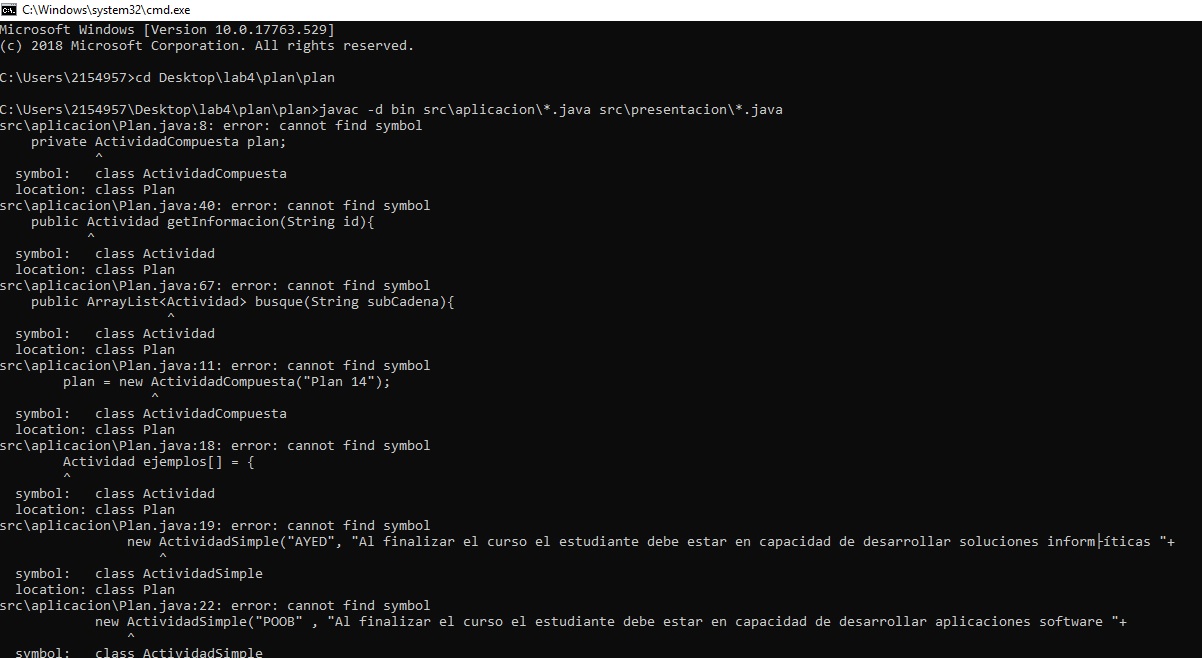


**3. Adicionen las clases correspondiente a las actividades.**

**4. Compilen el proyecto, ¿qué métodos tienen que adicionar para lograrlo? (No olviden adicionarlos en el diseño)**

Para compilar todo el proyecto usamos:

C:\Users\2154957\Desktop\lab4\plan\plan >javac -d bin src\aplicacion\\*.java src\presentacion\\*.java



Faltan 2 métodos:

getActividades()

getId()

**5. Ejecuten el proyecto, ¿qué funcionalidades ofrece? ¿cuáles funcionan?**

Ofrece: Listar, Adicionar y Buscar.

Funcionan: Listar y Adicionar.

**6. Revisen el código del proyecto. ¿De dónde salen las asignaturas iniciales? ¿Qué clase pide que se adicionen? ¿Qué clase los adiciona?**

Las asignaturas iniciales salen de Plan.

La clase PlanGUI pide que se adicionen.

La clase ActividadCompuesta los adiciona.

**Arquitectura**   
**[En lab04.doc, seleccion.asta y \*.java]**

**1. Realicen el diseño arquitectónico con un diagrama de paquetes en el que se presente los componentes y las relaciones entre ellos.**

**2. Completen el diseño actual de la capa de aplicación.**

**3. Considerando las funcionalidades del sistema, realicen el diagrama de casos de uso correspondiente. Organice todos los elementos en un modelo llamado useCases0**

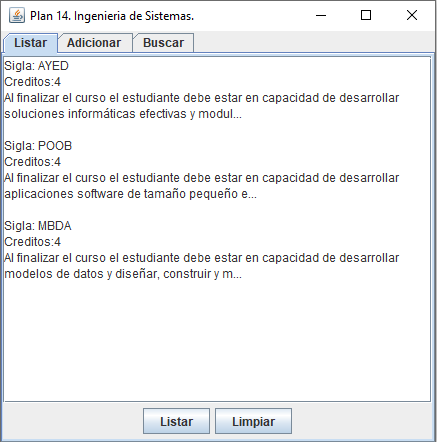
**Adicionar y listar. Todo OK.**   
**[En lab04.doc, seleccion.asta y \*.java] (NO OLVIDEN BDD - MDD)**   
El objetivo es realizar ingeniería reversa a las funciones de adicionar y listar.

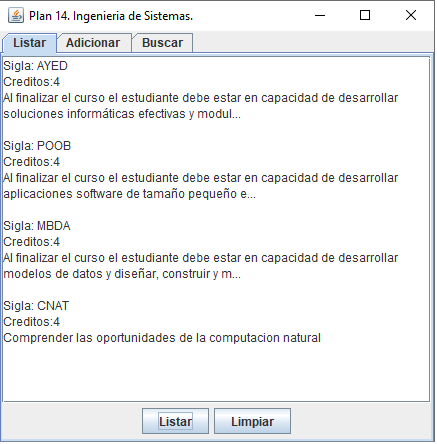
**1. Adicionen una nueva asignatura: CNAT, 4 créditos. Comprender las oportunidades de la computación natural**

**¿Qué ocurre?** Se adiciona a listar.

**¿Cómo lo comprueban?** Colocando listar antes y después de adicionar la nueva asignatura.

**Capturen la pantalla:**





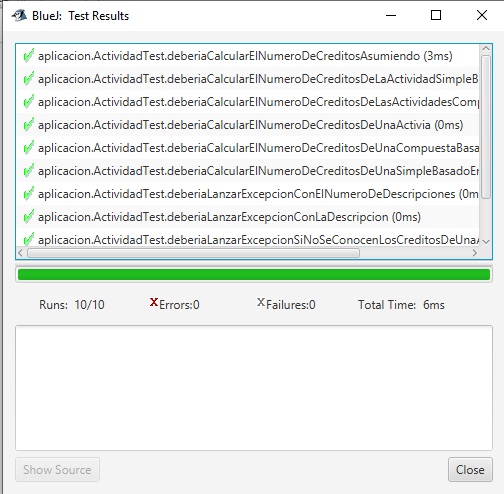
**¿Es adecuado este comportamiento?**  es adecuado.

**2. Revisen el código asociado a adicionar en la capa de presentación y la capa de aplicación.**

**¿Qué método es responsable en la capa de presentación?**  La clase PlanGUI, en su constructor hace el llamado a la función adicione() de plan, y tiene el método accionAdicionar().

**¿Qué método en la capa de aplicación?** La clase Plan con su método adicione() y adicione(id,descripcion,ceditos).

**3. Realicen ingeniería reversa para la capa de aplicación 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 aplicación.**

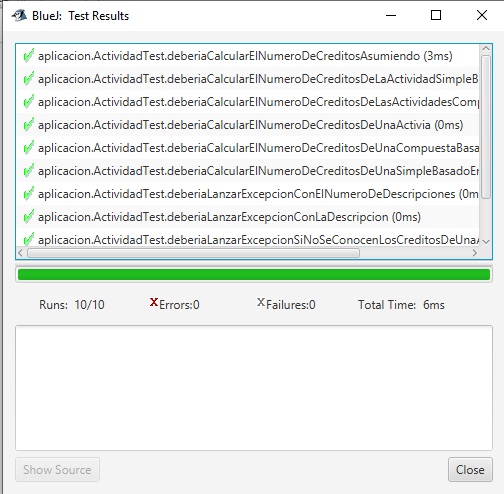
**¿Qué método es responsable en la capa de presentación?**

PrepareElementos().

**¿Qué método en la capa de aplicación?**

El metodo de plan de toString()

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



**6. Propongan y ejecuten una prueba de aceptación.**

Vamos a adicionar la materia LCAL que tiene 4 créditos y su descripción es: es una materia diseñada para que los estudiantes de sistemas inicien en su mentalidad lógica

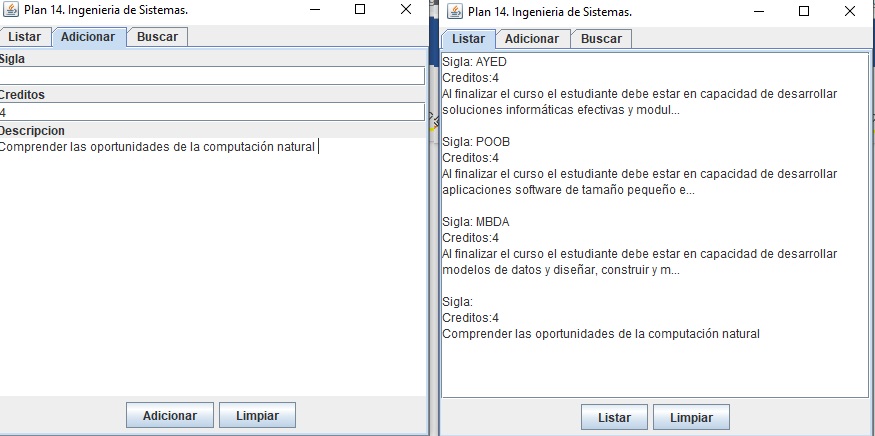
**Adicionar una asignatura. ¿Y si no da la sigla?**   
**[En lab04.doc, seleccion.asta y \*.java] (NO OLVIDEN BDD – MDD)**   
El objetivo es perfeccionar la funcionalidad de adicionar una asignatura.

**1. Adicionen a CNAT sin sigla**

**¿Qué ocurre?** Lo permite.

**¿Cómo lo comprueban?** Colocando listar antes y después de adicionar la nueva asignatura. Y debido a que no nos informa el programa que algo malo ocurrió.

**Capturen la pantalla:**



**¿Es adecuado este comportamiento?** No es adecuado.

**2. Vamos a evitar la creación de asignaturas con siglas vacías manejando la excepción ActividadExcepcion. Si la asignatura no tiene sigla no la creamos y se lo comunicamos al usuario1. Para esto lo primero que debemos hacer es crear el mensaje en la clase ActividadExcepcion.**

**3. Analicen el diseño realizado.**

**¿Qué método debería lanzar la excepción?**

adicione(id,descripcion,ceditos) de ActividadCompuesta

**¿Qué métodos deberían propagarla?**

adicione(id,descripcion,ceditos) de Plan

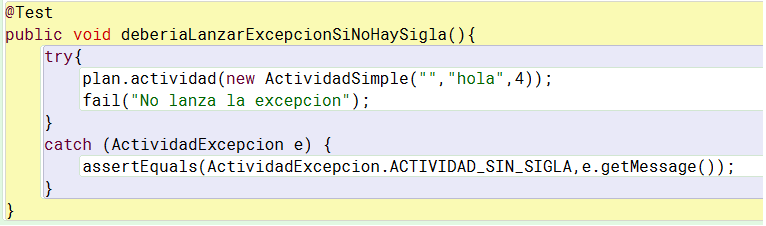
**¿Qué método debería atenderla?**

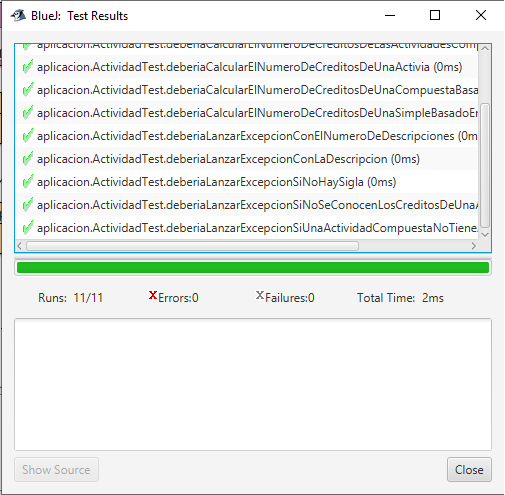
accionAdicionar() de PlanGUI dado que es el responsable de llamar al metodo en Plan.

**Explique claramente.**

Dado que Plan es una actividad compuesta y esta es la que tiene el método para adicionar actividades, es allí en donde empieza lanzando la excepción, en Plan se propaga dado que allí se usa el adicione de AC y el que debería manejarlo sería el método en el GUI

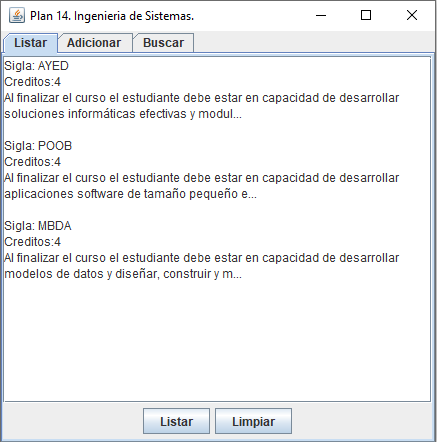
**4. Construya la solución propuesta. Capturen los resultados de las pruebas.**

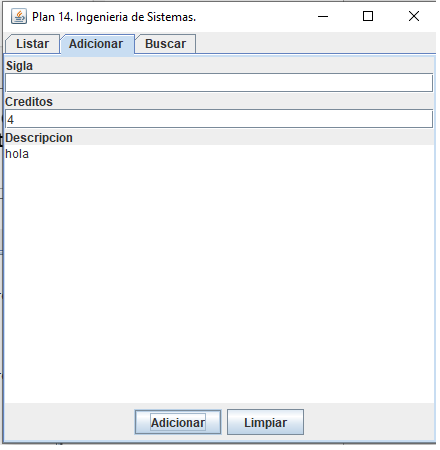




**5. Ejecuten nuevamente la aplicación con el caso de prueba propuesto en 1., ¿Qué sucede ahora? Capture la pantalla.**

Ahora no se adiciona si no tiene la sigla





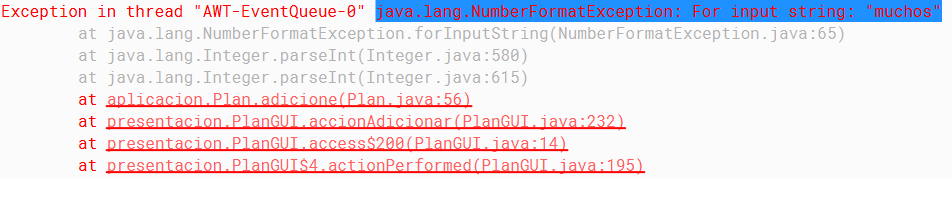
**Adicionar una asignatura. ¿Y si da créditos como un texto?**   
**[En lab04.doc, seleccion.asta y \*.java] (NO OLVIDEN BDD - MDD)**   
El objetivo es perfeccionar la funcionalidad de adicionar una asignatura.

**1. Adicionen la nueva asignatura CNAT, pero en lugar de 4 coloquen muchos.**

**¿Qué ocurre? Capturen la pantalla.**

La terminal lanza el error

java.lang.NumberFormatException: For input string: "muchos"



**¿Es adecuado este comportamiento?**

Es adecuado en el sentido que no deja hacer algo invalido, sin embargo, no es adecuado que el usuario vea eso.

**2. Analicen el diseño realizado.**

**¿Qué método lanzan la excepción?**

adicione(id,descripcion,ceditos) de Plan

**¿Qué métodos la propagan?**

accionAdicionar() de PlanGUI dado que es el responsable de llamar al método en Plan.

**¿Por qué se propaga?**

Porque es una excepción no chequeada

**Explique claramente.**

Es una excepción que extiende la clase runtime error, por lo que no la chequeamos.

**¿Qué problema tenemos ahí?**

Como es runtime error es una excepción no chequeada

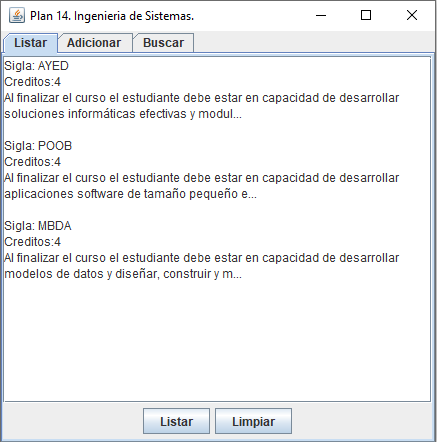
**3. Analicen el diseño realizado.**

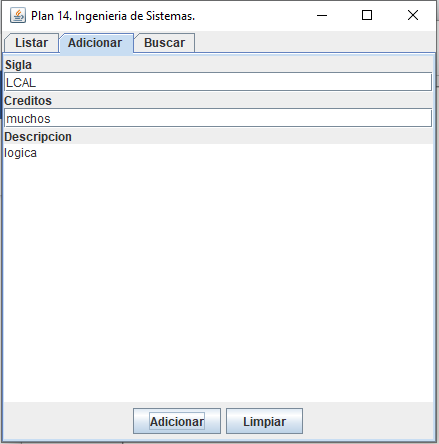
**¿Cuál es la propuesta de solución?**

Atender esa excepción en especifico

**4. Ejecuten nuevamente la aplicación con el caso de prueba propuesto en 1., ¿Qué sucede ahora? Capture la pantalla.**

Ahora no se adiciona si presenta problemas con sus créditos





**Adicionar una asignatura. ¿Y si ya se encuentra?**   
**[En lab04.doc, seleccion.asta y \*.java] (NO OLVIDEN BDD - MDD)**   
El objetivo es perfeccionar la funcionalidad de adicionar una asignatura.

**1. Adicionen dos veces la nueva asignatura**

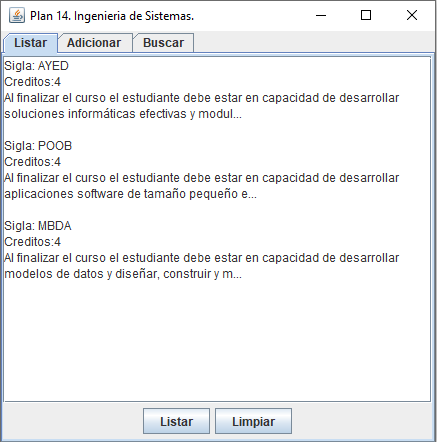
**¿Qué ocurre?**

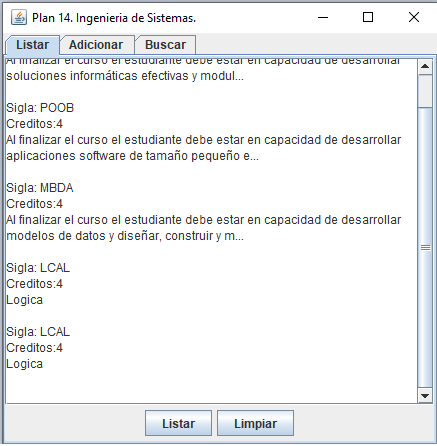
Se adiciona 2 veces la misma Actividad

**¿Cómo lo comprueban?**

Listamos inicialmente, limpiamos y volvemos a listar una vez agregadas las Actividades

**Capturen la pantalla.**





**¿Es adecuado este comportamiento?**

No es adecuado este comportamiento

**2. Analicen el diseño realizado.**

**¿Qué método debería lanzar la excepción?**

adicione(id,descripcion,ceditos) de ActividadCompuesta

**¿Qué métodos deberían propagarla?**

adicione(id,descripcion,ceditos) de Plan

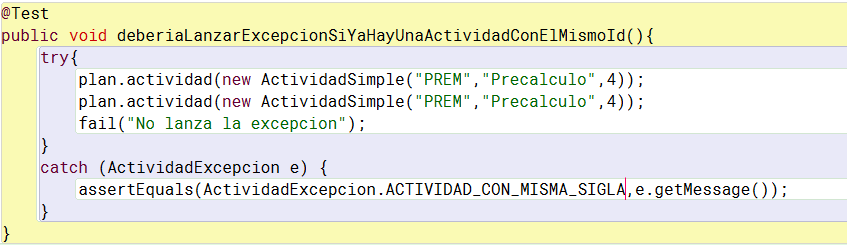
**¿Qué método debería atenderla?**

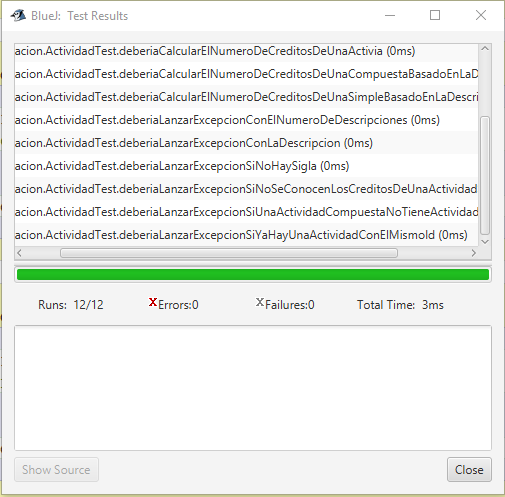
accionAdicionar() de PlanGUI dado que es el responsable de llamar al método en Plan.

**Explique claramente.**

Dado que Plan es una actividad compuesta y esta es la que tiene el método para adicionar actividades, es allí en donde empieza lanzando la excepción, en Plan se propaga dado que allí se usa el adicione de AC y el que debería manejarlo sería el método en el GUI.

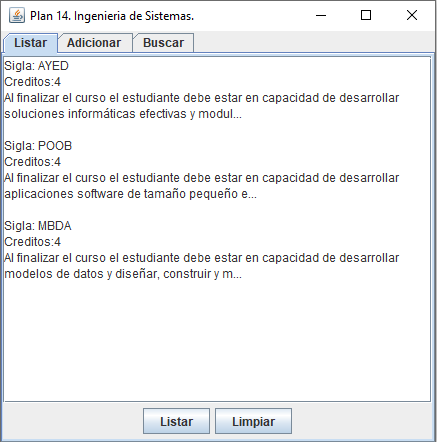
**3. Construya la solución propuesta. Capturen los resultados de las pruebas.**

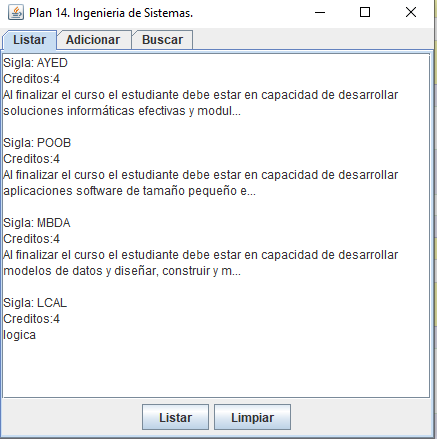




**4. Ejecuten nuevamente la aplicación con el caso de prueba propuesto en 1., ¿Qué sucede ahora? Capture la pantalla.**

Ahora solo se registra una vez cada actividad





**Consultando por patrones. ¡No funciona y queda sin funcionar!**   
**[En lab04.doc, seleccion.asta y \*.java] (NO OLVIDEN BDD - MDD)**

**1. Consulten una asignatura especial que contenga finalizar.**

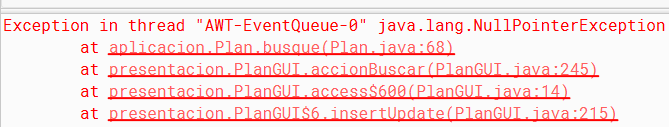
**¿Qué sucede?**

El terminal muestra un error

**¿Qué creen que pasó?**

Durante la ejecución del programa surgió un error

**Capturen el resultado.**



**¿Quién debe conocer y quien NO debe conocer esta información?**

La persona que debe conocer esta información es el desarrollador y no el usuario

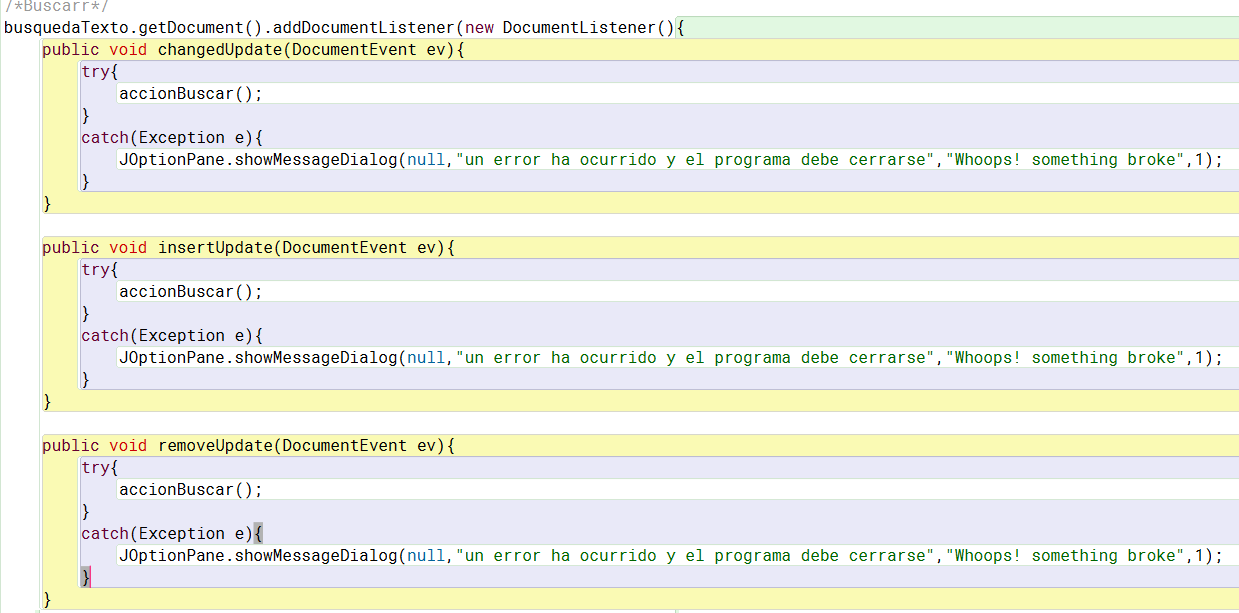
**2. Exploren el método registre de la clase Registro ¿Qué servicio presta?**

Registra las excepciones que se presentan

**3. Analicen el punto adecuado para que SIEMPRE, al sufrir en cualquier punto el sistema un incidente como este, se presente un mensaje especial de alerta al usuario, se guarde la información del error en el registro de error y termine la ejecución.**

**Expliquen y construyan la solución.**

El punto en el que se manejaria este tipo de excepciones seria en PlanGUI dado que este es el que recibe la informacion erronea



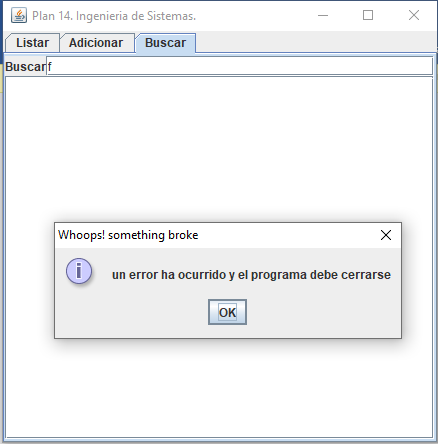
**4. Ejecuten nuevamente la aplicación con el caso propuesto en 1. ¿Qué mensaje salió en pantalla?**

**¿La aplicación termina?**

No termina

**¿Qué información tiene el archivo de errores?**

No se ha creado ningun archivo



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

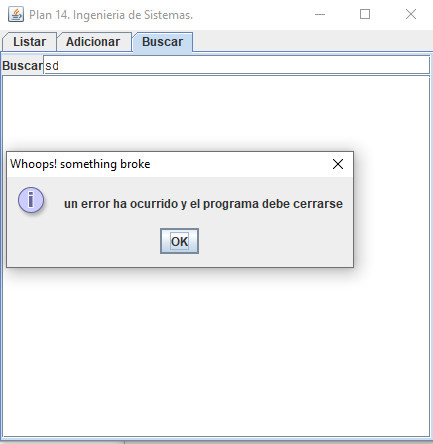
No es adecuado, ya que la aplicación debe terminar.

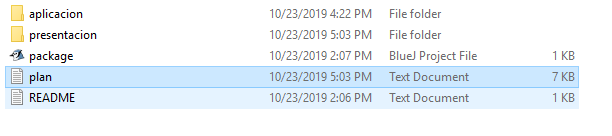
Depende si es un error esperado o inesperado. Si es un error inesperado, la aplicación debería terminar su ejecución.

**6. 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. No eliminen la solución de 3.**

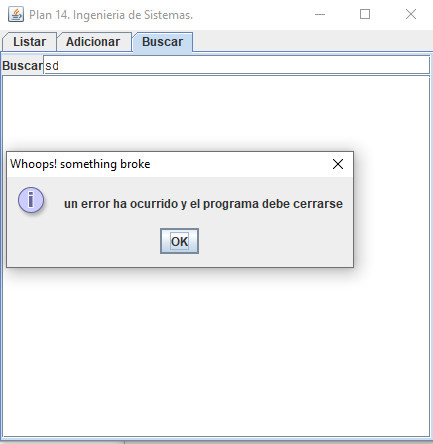
En este caso el punto adecuado sería el mismo, sin embargo, se modificó la solución







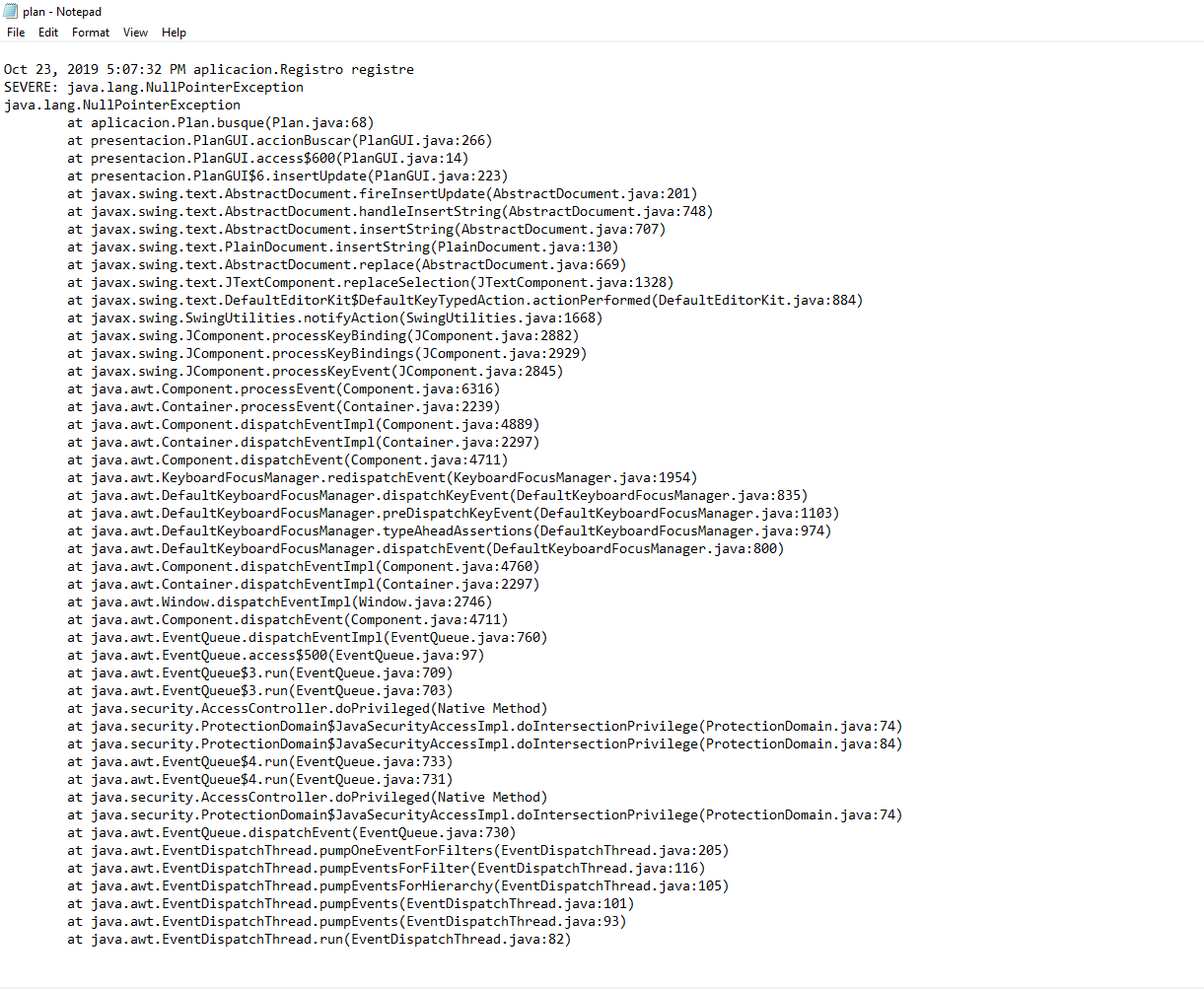
**7. Ejecuten nuevamente la aplicación con el caso propuesto en 1. ¿Qué mensaje salió en pantalla?**



**¿La aplicación termina?**

Ahora si termina

**¿Qué información tiene el archivo de errores?**



**Consultando por patrones. ¡Ahora si funciona!**   
**[En lab04.doc, seleccion.asta y \*.java] (NO OLVIDEN BDD - MDD)**

**1. Revisen el código asociado a buscar en la capa de presentación y la capa de aplicación.**

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

prepareBuscar() en la clase PlanGUI.

**¿Qué método es responsable en la capa de aplicación?**

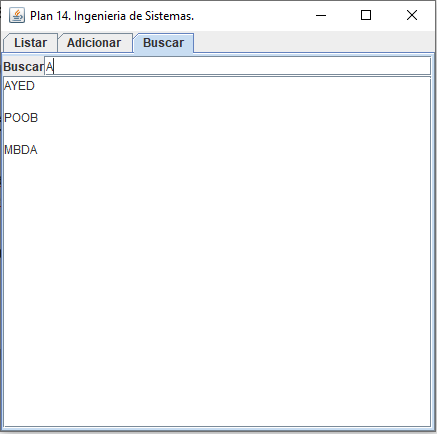
busque() en la clase Plan.

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

Las pruebas fallan debido a que el programa está lanzando excepciones no chequeadas.

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

El error era que nunca creaba un arraylist de actividades en el metodo busque, por lo que el programa lanzaba una excepcion no chequeada cuando se le apuntaba a ese arraylist.



**4. Ejecuten la aplicación nuevamente con el caso propuesto.**

**¿Qué tenemos en pantalla?**

Los resultados de la búsqueda, es decir el nombre de las materias que coinciden con la búsqueda

**¿Qué información tiene el archivo de errores?**

En este caso dado que no se generó un error, no se actualizo el archivo de errores.

**RETROSPECTIVA**

**1. ¿Cuál fue el tiempo total invertido en el laboratorio por cada uno de ustedes?** Fue un total del tiempo del laboratorio juntos y por fuera día y medio juntos, con los respectivos descansos.

**2. ¿Cuál es el estado actual del laboratorio? ¿Por qué?** El laboratorio se encuentra completo, teniendo en cuenta que se presentaron bastantes dificultades a la hora de trabajar por separado.

**3. Considerando las prácticas XP del laboratorio. ¿cuál fue la más útil? ¿por qué?** La mejor practica en este laboratorio fue programar a pares y codificar las pruebas primero, debido a que nos pudimos ayudar con las respectivas dudas y el desarrollo del laboratorio fue mucho mejor.

**4. ¿Cuál consideran fue el mayor logro? ¿Por qué?** Haber logrado terminar el laboratorio, debido a que eran bastantes cosas que había que mirar ya hechas y buscar cual fue el diseño, y la idea detrás.

**5. ¿Cuál consideran que fue el mayor problema técnico? ¿Qué hicieron para resolverlo?** Lidiar con la consola, las pruebas no se lograron ejecutar como en el laboratorio pasado y desconocemos las razones.

**6. ¿Qué hicieron bien como actividad? ¿Qué se comprometen a hacer para mejorar los resultados?** Trabajar ayudando al otro y dedicarle el tiempo juntos necesario al desarrollo de este. Prometemos mantener una buena relación y estar para ayudar al otro cuando lo necesite, para hacer más ameno el espacio de trabajo de POOB.