**PROGRAMACIÓN ORIENTADA A OBJETOS**

**Excepciones**

**2019-02**

**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 actividad de desarrollo de una aplicación en producción.**
5. **Vivenciar la prácticas *Designing***  [***Simplicity***](http://www.extremeprogramming.org/rules/simple.html)***.***

**Coding Code must be written to agreed** [**standards**](http://www.extremeprogramming.org/rules/standards.html)

**ENTREGA**

** Incluyan en un archivo .zip los archivos correspondientes al laboratorio. El nombre debe ser los dos apellidos de los miembros del actividad ordenados alfabéticamente.**

** Deben publicar el avance al final de la sesión y la versión definitiva en la fecha indicada, en los espacios preparados para tal fin.**

# EQUIPOS

**EN BLUEJ**

**PRACTICANDO MDD y BDD con EXCEPCIONES**

**[En lab04.doc, actividadad.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**

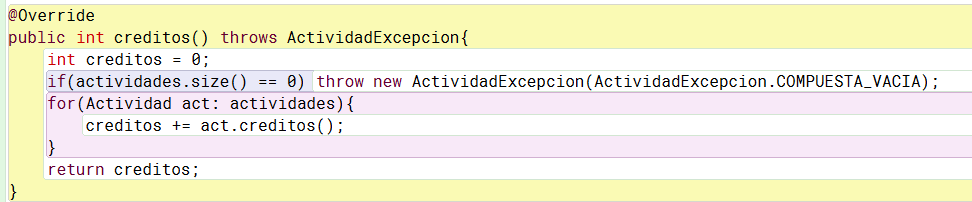
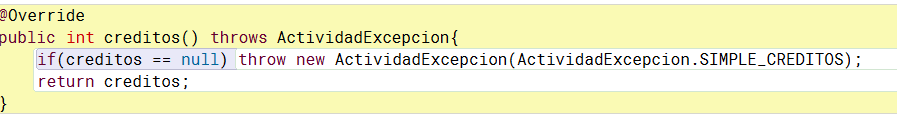
1. **En su directorio descarguen los archivos contenidos en actividades.zip, revisen el contenido y estudien el diseño estructural de la aplicación.**

Este archivo tiene 4 clases de objetos, en las cuales se encuentra una clase abstracta y también hay una clase de pruebas de unidad, las clases actividadSimple y AcividadCompuesta heredan de la clase abstracta actvidad

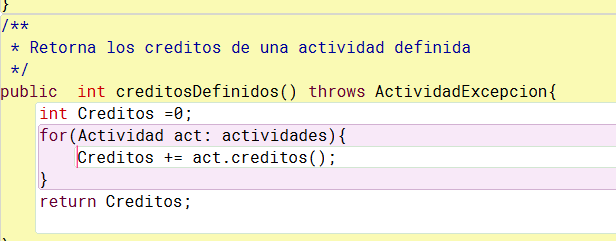
1. **Expliquen por qué el proyecto no compila. Realicen las adiciones necesarias para lograrlo.**

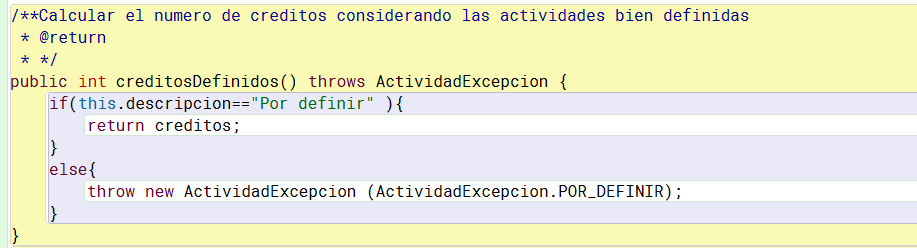
No compila porqué están mal implementada la herencia. Toca añadirle los métodos que faltan

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

****

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

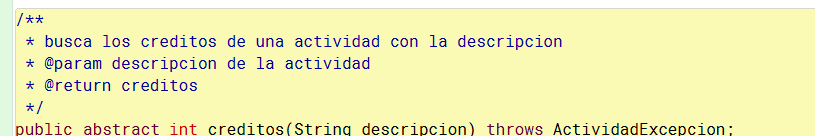
****

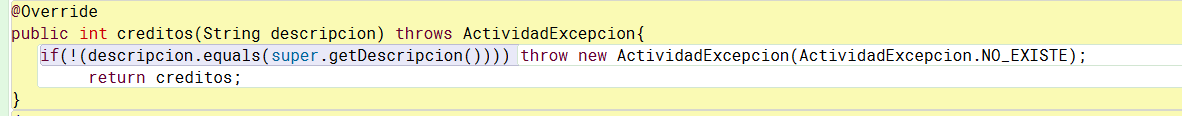
****

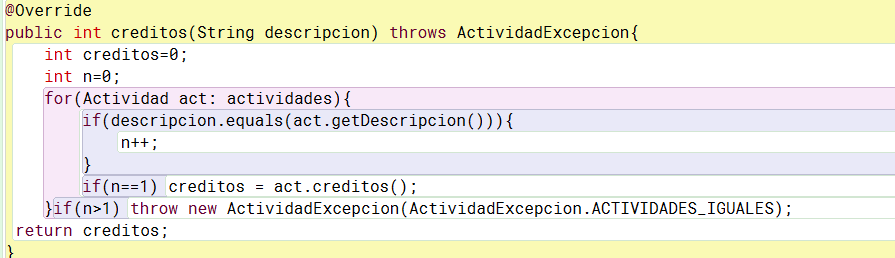
1. **Documenten, diseñen, codifiquen y prueben el método creditos(String).**

**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**

****

****

****

**Selección Colombia EN CONSOLA**

**El objetivo de Selección es contar con la información básica de los asignaturaes de la selección Colombia.**

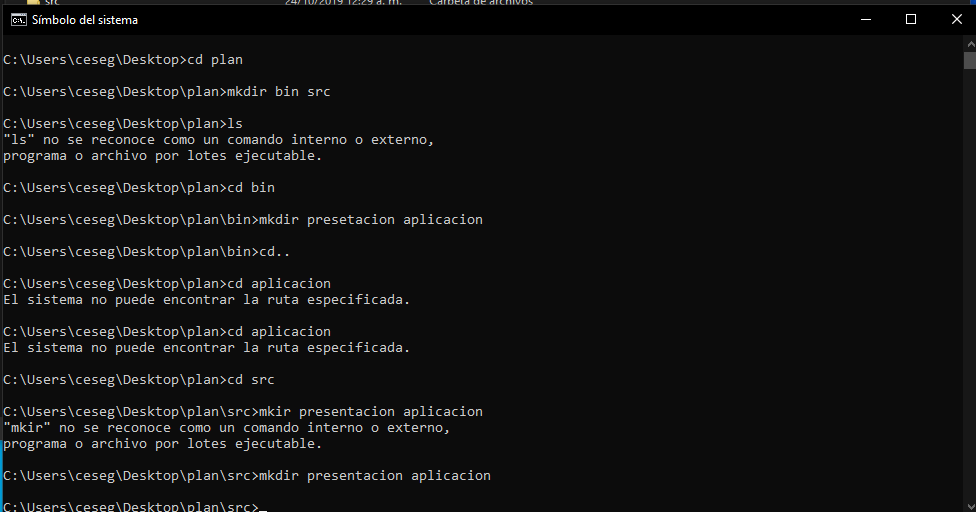
## Conociendo el proyecto [En lab04.doc]

**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?**

3 archivos, están todos en la misma carpeta, deberían estar distribuidos en aplicación y presentacion

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

Debe tener:

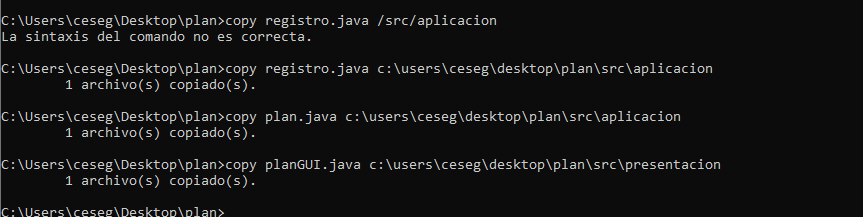
Plan

Bin

Src

Aplicación

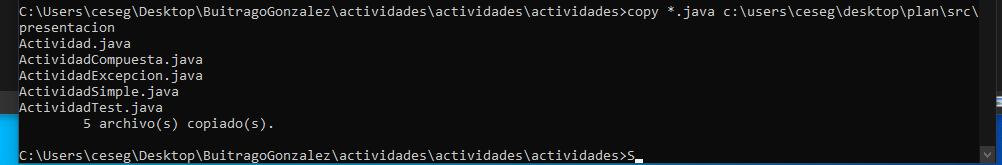
Registro.java

 Plan.java

Presentación

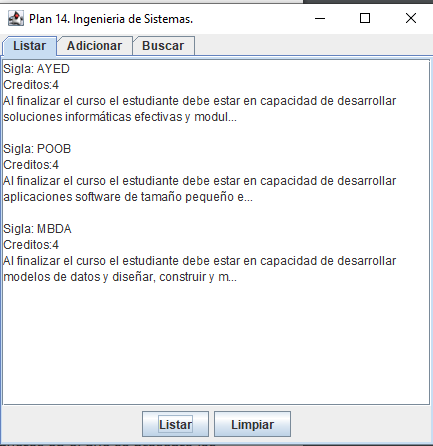
PlanGUI.java

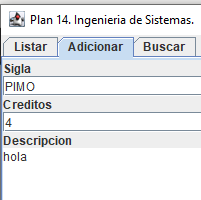
1. **Adicione las clases correspondiente a las actividades.**

****

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

Adicionar, buscar, listar limpiar funcionan todas menos buscar

****

****

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

**La clase plan es la que las pide adicionar**

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

1. **Realice el el diseño arquitectonico con un diagrama de paquetes en el que se presente los componentes y las relaciones entre ellos.**
2. **Complete 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.**

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

**1. Adicionen un nuevo asignatura**

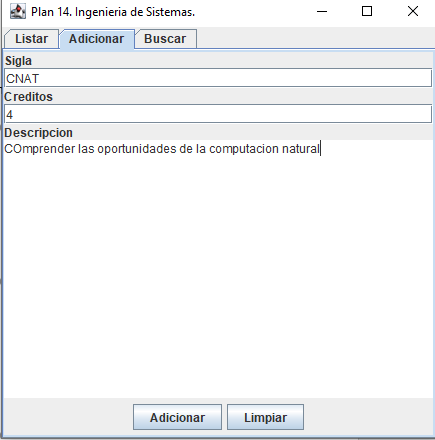
**CNAT**

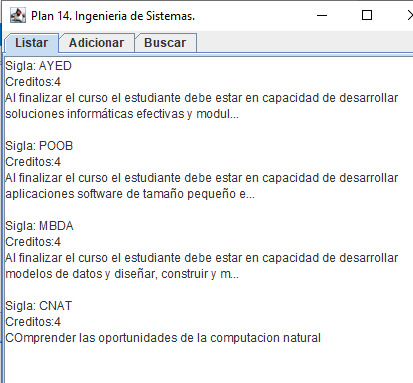
**4 créditos**

**Comprender las oportunidades de la computación natural**

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

**Es adecuado porque la agrega**

****

****

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

En aplicación el método responsable es adicione en plan

Y en presentación es accionAdicionar en la clase planGUI

1. **Realicen ingeniería reversa para la capa de aplicación para adicionar.**

**Capturen los resultados de las pruebas de unidad.**

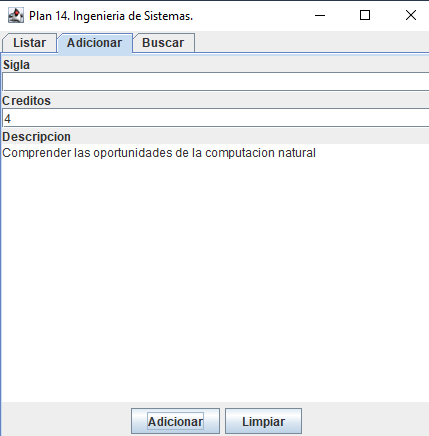
1. **Revisen el código asociado a listar en la capa de presentación y la capa de aplicación. ¿Qué método es responsable en la capa de presentación? ¿Qué método en la capa de aplicación?**

En presentación el responsable accionListar y en aplicación el responsable es la clase plan porque esta muestra un arreglo de actividades

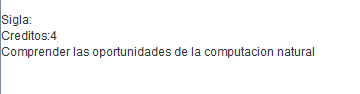
1. **Realicen ingeniería reversa de la capa de aplicación para listar.**

**Capturen los resultados de las pruebas de unidad.**

1. **Propongan y ejecuten una prueba de aceptación. Adicionar una asignatura. ¿Y si no da la sigla?**
2. **Adicionen a CNAT sin sigla ¿Qué ocurre? ¿Cómo lo comprueban? Capturen la pantalla. ¿Es adecuado este comportamiento?**

****

**La adiciona, no debería ser adecuado este comportamiento.**

****

1. **Vamos a evitar la creación de asignaturas con siglas vacías manejando una excepción ActividadExcepcion. Si la asignatura no tiene sigla no la creamos y se lo comunicamos al usuario[[1]](#footnote-1). Para esto lo primero que debemos hacer es crear el mensaje en la clase ActividadExcepcion.**

****

1. **Analicen el diseño realizado. ¿Qué método debería lanzar la excepción? ¿Qué métodos deberían propagarla? ¿Qué método debería atenderla? Explique claramente.**

¿Qué método debería lanzar la excepción?

Constructor de la actividad o el plan

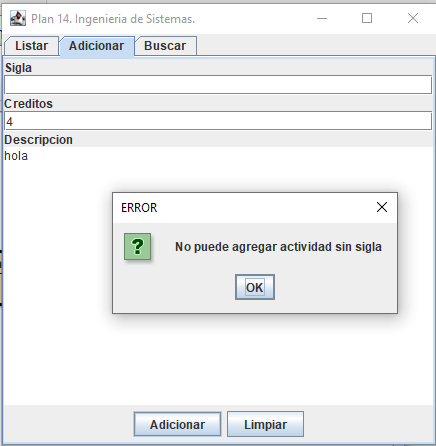
¿Qué métodos deberían propagarla?

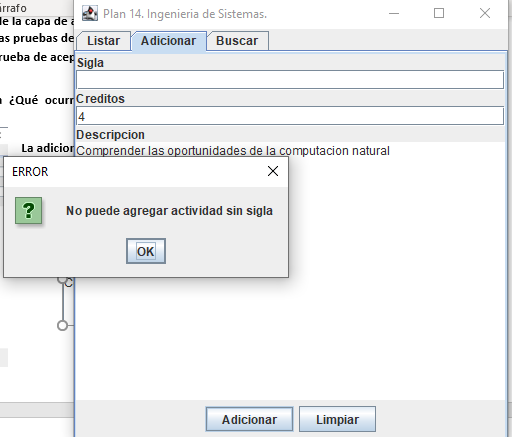
Constructpr de la actividad o el plan y adiciones en plan

¿Qué método debería atenderla?

Se atiende en acciónAdicionar de planGUI

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

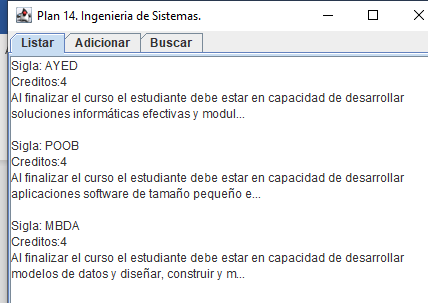
****

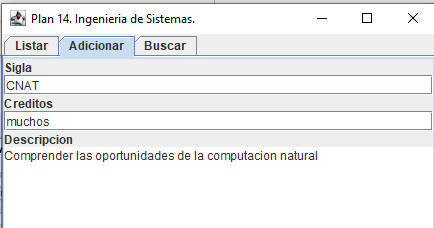
1. **Ejecuten nuevamente la aplicación con el caso de prueba propuesto en 1., ¿Qué sucede ahora? Capture la pantalla.**

**Adicionar una asignatura. ¿Y si da créditos como un texto?**

**El objetivo es perfeccionar la funcionalidad de adicionar un asignatura.**

1. **Adicionen el nuevo asignatura *CNAT* pero en lugar de 4 colquen muchos. ¿Qué ocurre? Capturen la pantalla. ¿Es adecuado este comportamiento?**

****

****

No lo agrega. Es adecuado el comportamiento

1. **Analicen el diseño realizado. ¿Qué método lanzan la excepción? ¿Qué métodos la propagan?**

**¿Por qué se propaga? Explique claramente. ¿Qué problema tenemos ahí?**

La lanza el constructor de la actividad, nadie la propaga, s Se está tratando de convertir un string a entero por lo que se genera la excepción NumberFormatException.

1. **Analicen el diseño realizado. ¿Cuál es la propuesta de solución?**

Que la lance la actividad en su constructor y que ahí también la propague junto a adicione de plan y debería atenderla acciónAdicionar de planGUI

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

**Adicionar un asignatura. ¿Y si ya se encuentra?**

**El objetivo es perfeccionar la funcionalidad de adicionar un asignatura.**

1. **Adicionen dos veces el nuevo asignatura ¿Qué ocurre? ¿Cómo lo comprueban? Capturen la pantalla. ¿Es adecuado este comportamiento?**
2. **Analicen el diseño realizado. ¿Qué método debería lanzar la excepción? ¿Qué métodos deberían propagarla? ¿Qué método debería atenderla? Explique claramente.**
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.**

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

1. **Consulten un asignatura especial que contenga solución. ¿Qué sucede? ¿Qué creen que pasó?**

**Capturen el resultado. ¿Quién debe conocer y quien NO debe conocer esta información?**

1. **Exploren el método registre de la clase Registro ¿Qué servicio presta?**
2. **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.**
3. **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?**
4. **¿Es adecuado que la aplicación continúe su ejecución después de sufrir un incidente como este? ¿de qué dependería continuar o parar?**
5. **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.**
6. **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?**

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

1. **Realicen ingeniería reversa de la capa de aplicación para buscar. Capturen los resultados de las pruebas. Deben fallar.**
2. **¿Cuál es el error? Soluciónenlo. Capturen los resultados de las pruebas.**
3. **Ejecuten la aplicación nuevamente con el caso propuesto. ¿Qué tenemos en pantalla? ¿Qué información tiene el archivo de errores?**

## RETROSPECTIVA

1. **¿Cuál fue el tiempo total invertido en el laboratorio por cada uno de ustedes? (Horas/Hombre) 2. ¿Cuál es el estado actual del laboratorio? ¿Por qué?**

**8 por hombre, incompleto por mala distribución del tiempo**

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

**Programación a pares**

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

**Entender mejor el manejo de excepciones y su importancia, porque es necesario para presentar un trabajo decente y bajo los entandares**

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

**Entender bien las excepciones cuando lanzan, propagan y atienden**

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

**Lo que realizamos esta bien, nos comprometemos a terminar el próximo lab**

1. [↑](#footnote-ref-1)