LABORATORIO 4

PROGRAMACION ORIENTADA A OBJETOS

ELABORADO POR:

DIEGO FERNANDO RUIZ ROJAS

JUAN FELIPE AGUAS PULIDO

PRESENTADO A: MARIA IRMA DIAZ ROZO

ESCUELA COLOMBIANA DE INGENIERÍA JULIO GARAVITO

PROGRAMACIÓN ORIENTADA A OBJETOS

PROGRAMA DE INGENIERÍA DE SISTEMAS

BOGOTÁ D. C.

26 DE MARZO DE 2021

**PRACTICANDO MDD y BDD con EXCEPCIONES**

1. En su directorio descarguen los archivos contenidos en accion.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.

-No compila porque las clases hijas de la clase abstracta Acción no tienen agregado los métodos de la clase padre.

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

Resuelto en el archivo asta y java

1. Dada la documentación, diseñen, codifiquen y prueben el método diasEstimados() .
2. Documenten, diseñen, codifiquen y prueben el método dias(String nombre). En este método vamos a incluir dos nuevos casos especiales:
3. no existe una acción con ese nombre

2) existe más de una acción con el mismo nombre

**SINAP. Sistema Nacional de Áreas Protegidas**

**EN CONSOLA**

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

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

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

Se tienen 5 archivos, los cuales no tienen ningún orden en específico, deberían estar organizados en un proyecto con 2 paquetes, uno de dominio y otro de presentación.

1. Estudien el diseño del programa: diagramas de paquetes y de clases. ¿cuántos paquetes tenemos? ¿cuántas clases tiene el sistema? ¿cómo están organizadas? ¿cuál es la clase ejecutiva?

Hay 2 paquetes, con 4 clases en total, las cuales están organizadas por presentación y las que hacen parte de la aplicación, la clase ejecutiva es la clase SinapGUI.

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

El archivo SinapGUI en el paquete de presentación, y las otras 3 clases en el paquete presentación, para ejecutarlo hay que compilar los archivos y ejecutar el método main de SinapGUI

1. Ejecute el proyecto, ¿qué funcionalidades ofrece? ¿cuáles funcionan? Realicen el diagrama de casos de uso correspondiente.

Ofrece listar, añadir, buscar y limpiar, todas a excepción de buscar funcionan.

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

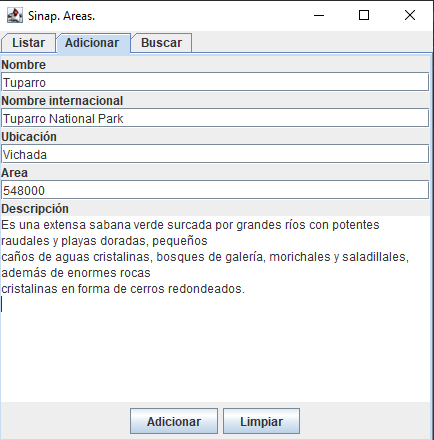
Las áreas iniciales se crean en el método adicioneCinco de la clase Sinap, esta clase también se encarga de pedir y adicionar nuevas áreas.

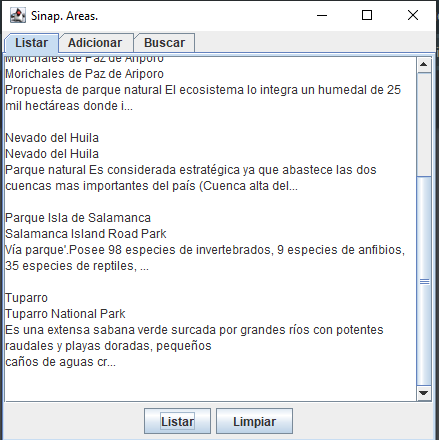
**Adicionar y listar. Todo OK. [En lab04.doc, SINAP.asta y \*.java] (NO OLVIDEN BDD - MDD)**

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

1. Adicionen una nueva área Tuparro Tuparro National Park Vichada 548.000 Es una extensa sabana verde surcada por grandes ríos con potentes raudales y playas doradas, pequeños caños de aguas cristalinas, bosques de galería, morichales y saladillales, además de enormes rocas cristalinas en forma de cerros redondeados. ¿Qué ocurre? ¿Cómo lo comprueban? Capturen la pantalla. ¿Es adecuado este comportamiento?

Al agragarlo no ocurrenada pero para ver que se guardo volvemos a listar las áreas y ya aparece la nueva área.

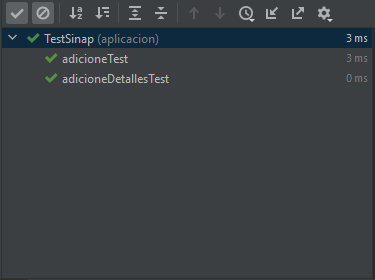




1. Revisen el código asociado a adicionar en la capa de presentación y la capa de aplicación. ¿Qué método es responsable en la capa de presentación? ¿Qué método en la capa de aplicación?

En la capa de presentación el método encargado es accionAdicionar() y en la capa de aplicación el método encargado es el método adicione()

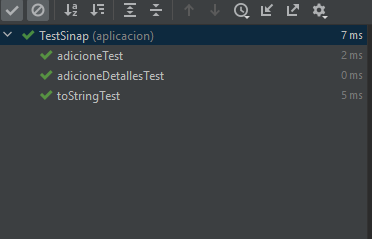
1. Realicen ingeniería reversa para la capa de aplicación para adicionar. Capturen los resultados de las pruebas de unidad.



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 la capa de presentación el método encargado es accionListar() y en la capa de aplicación el método encargado es el método toString()

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



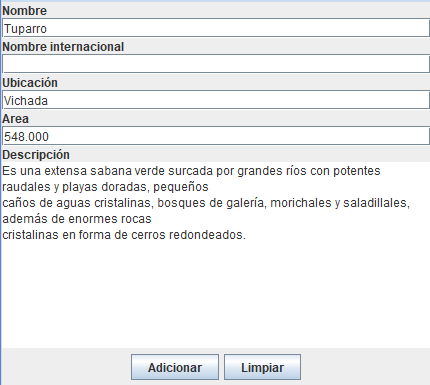
1. Propongan y ejecuten una prueba de aceptación.

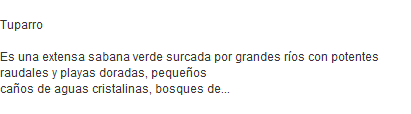
**Adicionar un área. ¿Y si no da un nombre? [En lab04.doc, SINAP.asta y \*.java] (NO OLVIDEN BDD – MDD)**

El objetivo es perfeccionar la funcionalidad de adicionar un área.

1. Adicionen el área Tuparro sin nombre internacional. ¿Qué ocurre? ¿Cómo lo comprueban? Capturen la pantalla. ¿Es adecuado este comportamiento?

Es posible adicionar el área, pero en la pantalla hay un espacio en blanco donde debería estar el nombre internacional



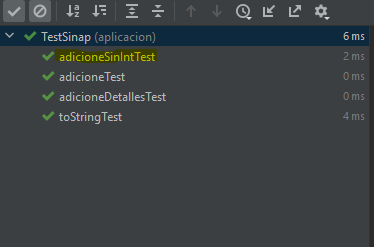


No es adecuado pues debería informar que no tiene nombre internacional

1. Vamos a evitar la creación de áreas con un nombre internacional vacío manejando una excepción SINAPExcepcion. Si el área no tiene nombre internacional, no la creamos y se lo comunicamos al usuario. Para esto lo primero que debemos hacer es crear la nueva clase SINAPExcepcion considerando este primer mensaje.
2. Analicen el diseño realizado. ¿Qué método debería lanzar la excepción? ¿Qué métodos deberían propagarla? ¿Qué método debería atenderla? Explique claramente.

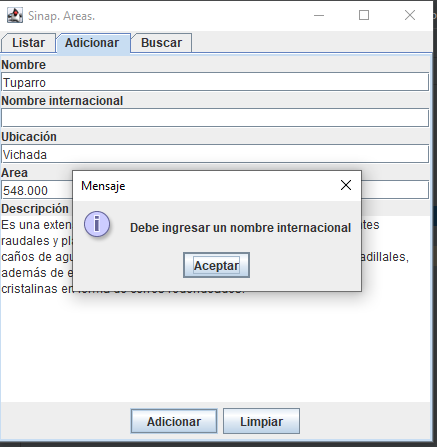
La excepción debe ser lanzada por el método adicioneDetalles(), propagada por adicione() y atendida por accionAdicionar() que es el último nivel antes de que se le muestre algo al usuario.

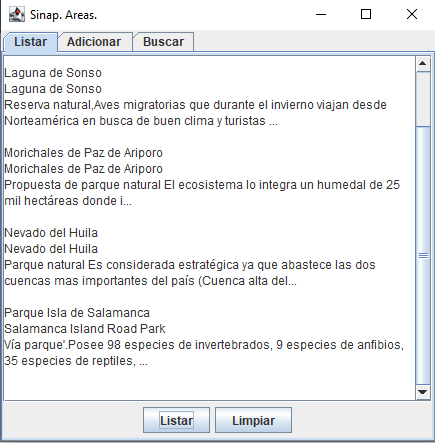
1. Construya la solución propuesta. Capturen los resultados de las pruebas.



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

Envía un mensaje y no agrega el área



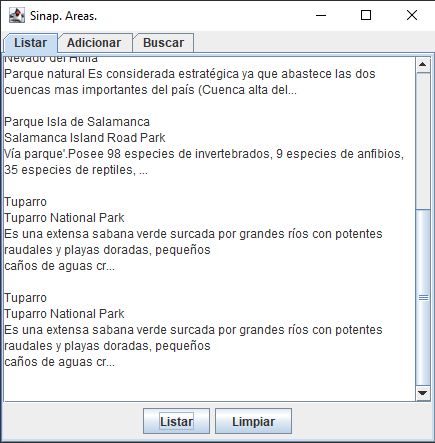


**Adicionar un área. ¿Y si ya se encuentra? [EnSINAP.asta, lab04.java y \*.java] (NO OLVIDEN BDD - MDD)**

El objetivo es perfeccionar la funcionalidad de adicionar un área.

1. Adicionen dos veces la nueva área Tuparro ¿Qué ocurre? ¿Cómo lo comprueban? Capturen la pantalla. ¿Es adecuado este comportamiento?

El área queda guardada dos veces lo notamos al listar las áreas en pantalla.

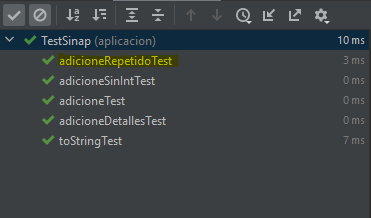


No es adecuado pues al ya estar guardado se usa más memoria y puede generar confusión al usuario.

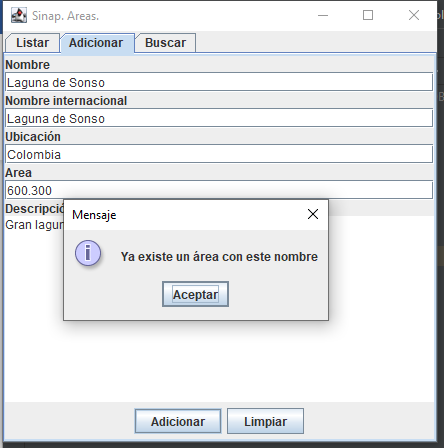
1. Analicen el diseño realizado. ¿Qué método debería lanzar la excepción? ¿Qué métodos deberían propagarla? ¿Qué método debería atenderla? Explique claramente.

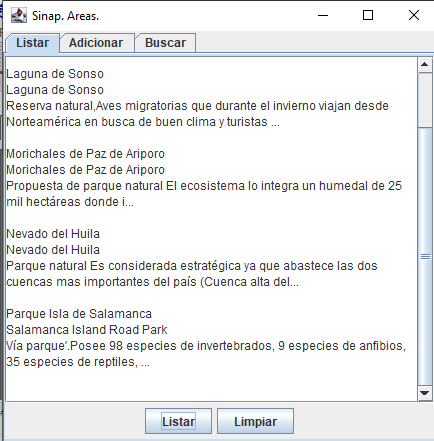
La excepción debe ser lanzada por el método adicioneDetalles(), propagada por adicione() y atendida por accionAdicionar() que es el último nivel antes de que se le muestre algo al usuario.

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.





Envía un mensaje adecuado y no se repiten áreas en la aplicación.

**Adicionar un área. ¿Otras condiciones? [En lab04.doc, SINAP.asta y \*.java] (NO OLVIDEN BDD – MDD)**

El objetivo es perfeccionar la funcionalidad de adicionar un área.

1. Propongan nuevas condiciones para que la adición de un área sea más robusta.

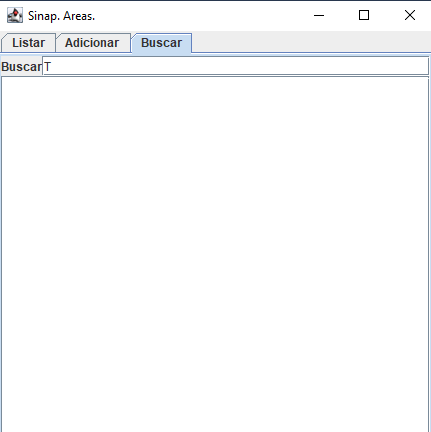
Si el número ingresado en el área tiene letras y si la ubicación no es completada.

1. Construya la solución propuesta. (diseño, prueba de unidad, código) Capturen los resultados de las pruebas.

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

1. Consulten un área especial que inicie con T. ¿Qué sucede? ¿Qué creen que pasó? Capturen el resultado. ¿Quién debe conocer y quien NO debe conocer esta información?

No sucede nada y en ocasiones no se ve lo escrito, probablemente en el código de buscar haya un error.



Esta información debe ser conocida por el programador más no por el usuario puesto que no fue un error al usar la aplicación.

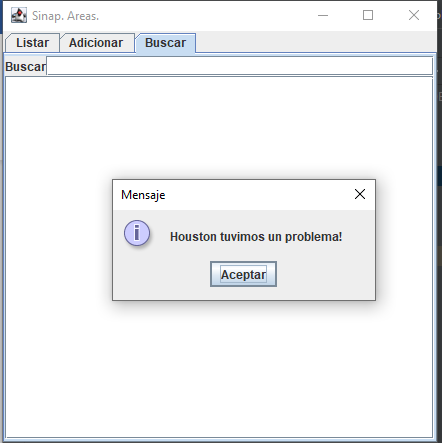
1. Exploren el método registre de la clase Registro ¿Qué servicio presta?

En un archivo “.log” escribe las excepciones que son pasadas por este método.

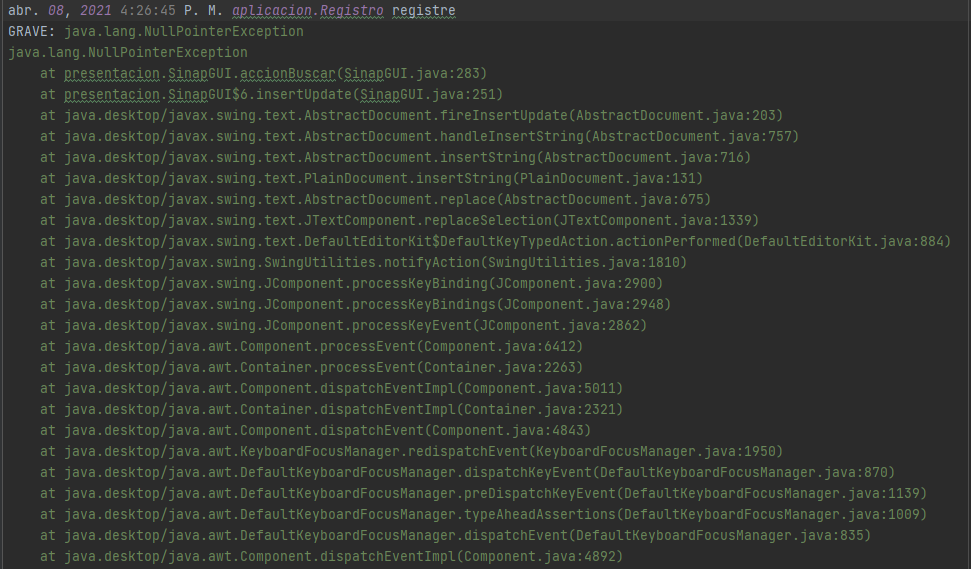
1. Analicen el punto adecuado para que SIEMPRE, al sufrir en cualquier punto el sistema un incidente como este, se presente un mensaje especial de alerta al usuario, se guarde la información del error en el registro de error y termine la ejecución. Expliquen y construyan la solución.

Para manejar este error lo que haremos es atrapar la excepción en la función encargada de la búsqueda en presentación, y enviaremos un mensaje de error para luego terminar la ejecución de programa.

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?



Si termina y el archivo contiene lo siguiente:



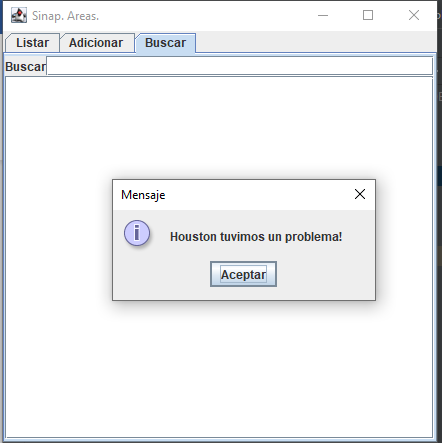
1. ¿Es adecuado que la aplicación continúe su ejecución después de sufrir un incidente como este? ¿de qué dependería continuar o parar?

No ya que no es un error fatal el cual impide continuar con la ejecución del resto de usos, continuar o para depende de que si el error daña todas las funcionalidades de la aplicación.

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. No eliminen la solución de 3.

Se usará la misma solución que dimos al punto 3 con la diferencia de que en registre2 no se terminará la ejecución del programa.

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?



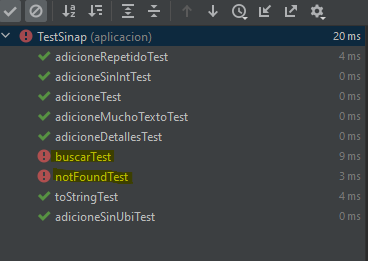
La aplicación no termina y el archivo contiene el registro completo del error generado.

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

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?

En la capa de presentación el método encargado es accionBuscar() y en la capa de aplicación el método encargado es el método busque()

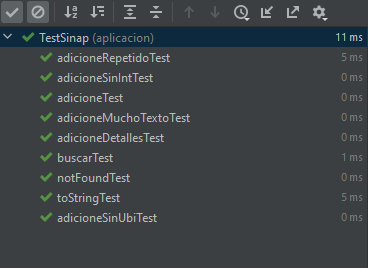
1. Realicen ingeniería reversa para la capa de aplicación para buscar. Capturen los resultados de las pruebas. Deben fallar.



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

No se esta inicializando el arreglo de resultados en el método busque() de manera correcta.

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



No tiene información nueva se mantiene la información de los pasos anteriores.

**RETROSPECTIVA**

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

-Aguas: 8 horas

-Ruiz: 6 horas

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

completado

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

Simplicity porque por medio de la clase SINAPException se pudieron trabajar todos los errores que se querían arreglar

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

El aprender el manejo de excepciones en diferentes clases

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

Generar los errores en pantalla con Joptionpane lo cual resolvimos investigando sobre esta herramienta.

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

Desarrollamos los problemas propuestos en el laboratorio con facilidad. Nos comprometemos a mejorar la documentación.