# EQUIPOS

## PRACTICANDO MDD y BDD con EXCEPCIONES

1. --
2. *¿De qué clase no pueden consultar fuentes? ¿En qué formato está? ¿Por qué?*

No se pueden consultar las fuentes de la clase Persona, la cual está en formato .class. Esto se debe a que no se encuentra el archivo .java, únicamente se encuentra la clase compilada, de la cual no se pueden obtener las fuentes.

1. --
2. --

# SINAP

## Conociendo el proyecto SINAP

1. *¿Cuántos archivos se tienen? ¿Cómo están organizados? ¿Cómo deberían estar organizados?*

Se tienen 4 archivos fuente. Estos archivos se encuentran en el directorio raíz. Los archivos deberían estar ubicados en el directorio src.

1. *¿Cuántos paquetes tenemos? ¿Cuántas clases tiene el sistema? ¿Cómo están organizadas? ¿Cuál es la clase ejecutiva?*

Se tienen dos paquetes: aplicación y presentación. El sistema cuenta con 4 clases, 3 de aplicación y 1 de presentación, todas ubicadas en un mismo directorio. La clase ejecutiva es **SinapGUI**,ya que es la clase que contiene el método main.

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

Es directorio debe estar compuesto por 3 carpetas: src, bin y docs, las cuales se usan para los archivos fuente, los archivos compilados y la documentación respectivamente.

Para ejecutarlo primero se debe compilar utilizando el comando javac de la siguiente forma:

*Javac -d <directory> <.java files>*

Luego lo podemos ejecutar utilizando el comando java así:

*Java -cp \path\to\bin <packageName>.<MainClassName>*

1. *Ejecute el proyecto, ¿Qué funcionalidades ofrece? ¿Cuáles funcionan? Realizar diagrama de casos de uso correspondiente.*

En el programa se pueden listar, agregar y buscar áreas. Únicamente funcionan las partes de agregar y listar áreas.

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 encuentran en la clase Sinap, en el método adicioneCinco. La clase SinapGUI pide que se adicionen las áreas, mientras que la clase Sinap las adiciona.

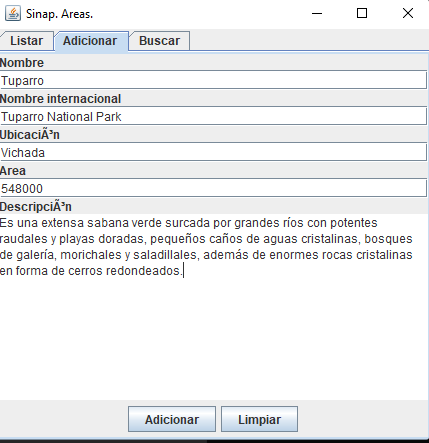
## Adicionar y listar. Todo OK.

1. *Adicionen una nueva área*

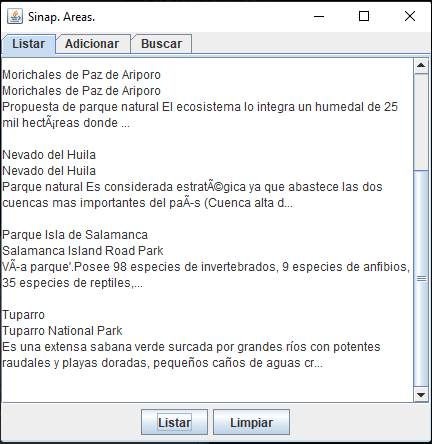
*¿Qué ocurre? ¿Cómo lo comprueban? Capturen la pantalla. ¿Es adecuado este*

*comportamiento?*

El área se agrega correctamente a la lista de áreas*.*



Esto se comprueba por medio de la pestaña listar.



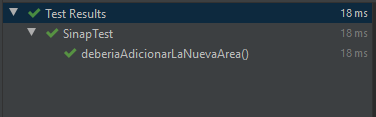
El comportamiento obtenido es el esperado.

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 existen dos métodos: para interfaz prepareAreaAdicionar() y para la parte de funcionalidad accionAdiconar(), ambos en la clase SinapGUI. En la capa de aplicación los métodos responsables son adicione() Y adicioneDetalles() de la clase Sinap.

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

Se realizó una prueba verificando que la nueva área se agregue correctamente, se verificaron cada uno de sus campos en la prueba, y efectivamente, se agrega de manera correcta.

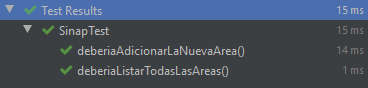


1. *Revisen el código asociado a* ***listar*** *en la capa de presentación y la capa de aplicación. ¿Qué método es responsable en la capa de presentación? ¿Qué método en la capa de aplicación?*

En la capa de presentación existen dos métodos: para interfaz prepareAreaListar() y para la parte de funcionalidad accionLitar(), ambos en la clase SinapGUI. En la capa de aplicación se utiliza el método toString() de la clase Sinap.

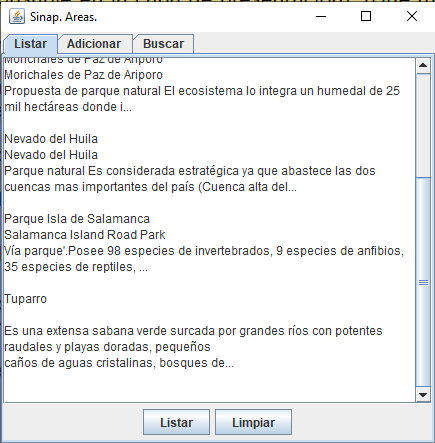
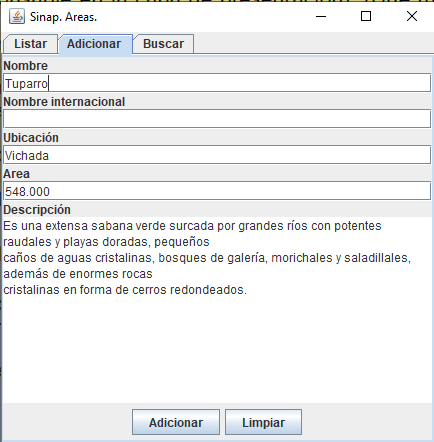
1. *Realicen ingeniería reversa para la capa de aplicación para* ***listar****. Capturen los resultados de las pruebas de unidad.*

Se realizó la prueba adicionando una nueva área y verificando que se liste correctamente.



## Adicionar un área. ¿Y si no da un nombre?

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

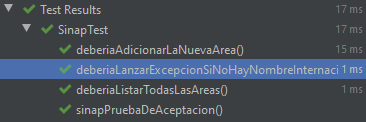
El área se agrega de manera normal, el espacio de nombre internacional queda vacío, esto se comprueba en la pestaña de **Listar**.

No se considera que este comportamiento sea adecuado, ya que al adicionar un área con falta de información estamos dejando posibles huecos para problemas más adelante.

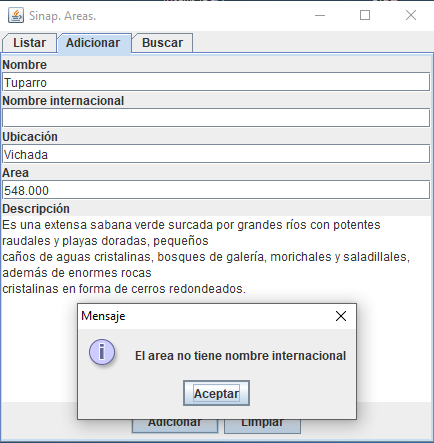
1. *--*
2. *Analicen el diseño realizado. ¿Qué método debería lanzar la excepción? ¿Qué métodos deberían propagarla? ¿Qué método debería atenderla? Explique claramente.*

* Lanzar: El constructor de la clase Area.
* Propagar: adicione y adicioneCinco de la clase Sinap.
* Atrapar: acción AdcciAdicionar de la clase SinapGUI.

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



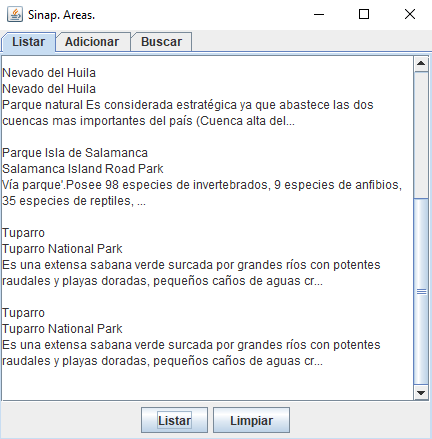
1. *Ejecuten nuevamente la aplicación con el caso de prueba propuesto en 1., ¿Qué sucede ahora? Capture la pantalla.*



## Adicionar un área. ¿Y si ya se encuentra?

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

El área tuparro se agrega dos veces. Esto se comprueba en la pestaña Listar.

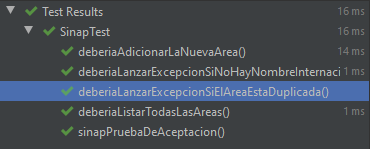


No es un comportamiento correcto debido a que se puede generar un duplicidad de los datos, lo cual ocupa un espacio de memoria inadecuadamente y reduce la calidad del Software.

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

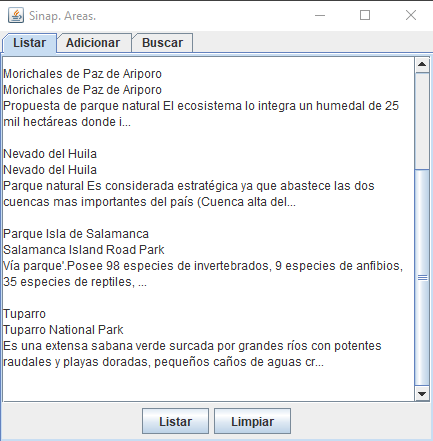
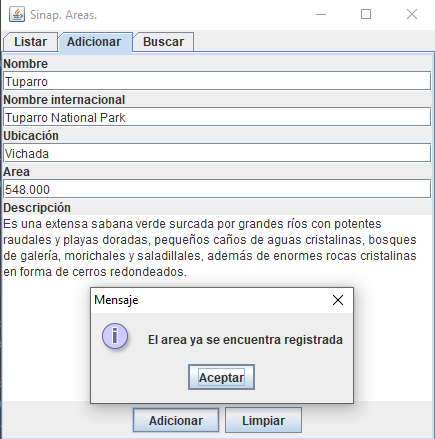
* Lanzar: adicioneDetalles en la clase Sinap
* Propagar: adicione en la clase Sinap
* Atender: accionAdicionar en la clase SinapGUI

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.*

La primera vez que se agrega el área, el procedimiento funciona de forma normal, la segunda vez que se realiza, se lanza un error que nos dice que el área ya existe (nombre internacional).



(solo se agregó una vez)

## Adicionar un área. ¿Otras condiciones?

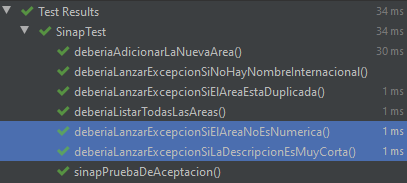
1. *Propongan nuevas condiciones para que la adición de un área sea más robusta.*

* El área debe ser numérica.
* La descripción debe tener al menos 20 caracteres.

Ambas las genera la clase Area:

* Las propaga el método adicione de la clase Sinap
* Las maneja el método accionAdicionar de la clase SinapGUI

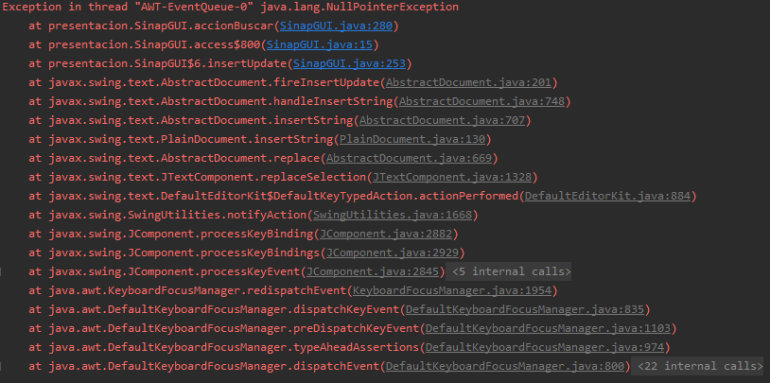
1. *Construya la solución propuesta. (diseño, prueba de unidad, código Capturen los resultados de las pruebas.*



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

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?*

En la aplicación no sucede nada, mientras que en consola aparece un error de tipo NullPointerException. Esto se debe a que se está intentando acceder a una variable que no se encuentra inicializada.



Esta información debe ser conocida por las clases implicadas, en este caso Sinap y SinapGUI, y no debe ser conocida por otras clases como por ejemplo Area.

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

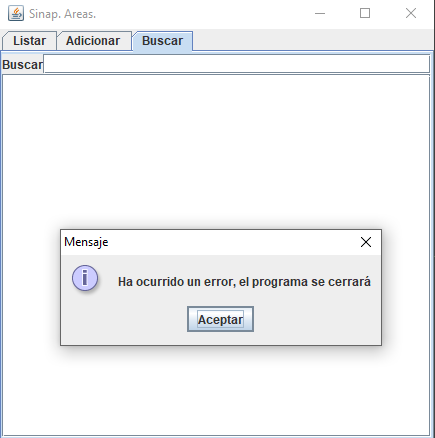
El servicio prestado por registre de la clase Registro, se trata de tomar una excepción, o un stack de excepciones y escribirlas en un log, para conocer que puede estar saliendo mal con la aplicación

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.*

En el método accionBuscar se declarará un bloque try-catch con una excepción genérica, y cada vez que ocurra una excepción de cualquier tipo, se observará el mismo mensaje y se cerrará la aplicación, además de guardar el log.

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?*

Salió en la pantalla un mensaje de error que dice: "Ha ocurrido un error, el programa se cerrará". La aplicación termina.



En el archivo SINAP.log se encuentra la pila de errores de cada excepción que ha ocurrido, junto con la fecha en la que ocurrió

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

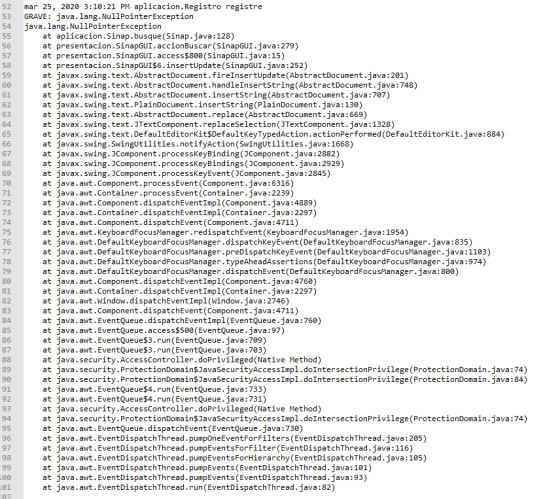
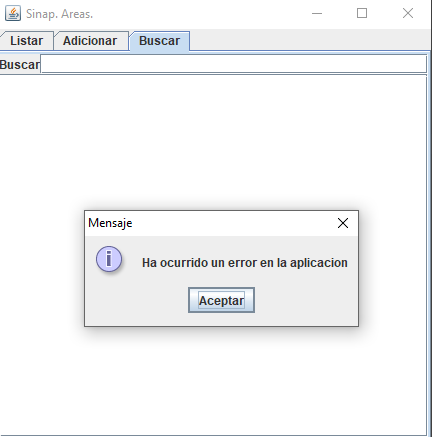
La aplicación idealmente jamás debería detenerse sin que el usuario se lo diga, sin embargo, es adecuado que la aplicación le muestre un mensaje amigable al usuario si debe detenerse por un error irrecuperable

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.*

En el bloque try-catch del método accionBuscar, se agregará un nuevo caso el cual específicamente buscará la excepción NullPointerException. Allí se pondrá la instrucción para registrar el error y mostrarle el mensaje al usuario.

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?*

En este caso salió un mensaje que dice únicamente "Ha ocurrido un error en la aplicación". La aplicación no termina. El archivo de errores tiene un nuevo log, del nuevo error que ocurrió.

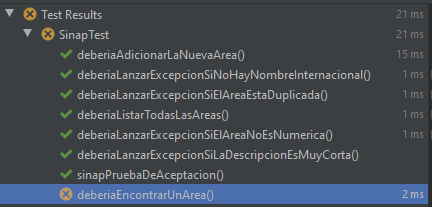


## Consultando por patrones. ¡Ahora si funciona!

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?*

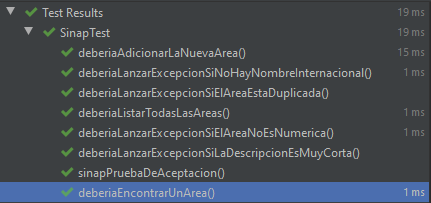
Para la capa de presentación existen dos métodos: En la parte visual se utiliza prepareAreaBuscar() y en la parte de funcionalidad se utiliza accionBuscar(). Para la capa de aplicacion el método responsable es busque().

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

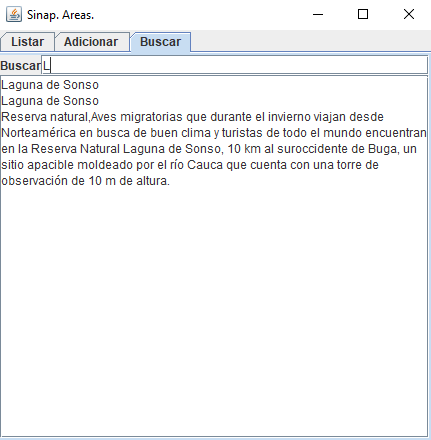


1. *¿Cuál es el error? Soluciónenlo. Capturen los resultados de las pruebas.*

El error está en la clase Sinap en el método busque, ya que el ArrayList de resultados no está inicializado.

f

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

Si volvemos a probar con el caso propuesto ("T"), no ocurre nada, ya que ninguna de las áreas por defecto tiene un nombre que comience por T. El archivo de errores no se modifica en este caso, ya que deja de ocurrir la excepción. Sin embargo, si ponemos una letra que si corresponda a un lugar como la "L", ocurre lo siguiente:

## RETROSPECTIVA

1. *¿Cuál fue el tiempo total invertido en el laboratorio por cada uno de ustedes?*

(6/66)

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

Terminado satisfactoriamente.

1. *Considerando las prácticas XP del laboratorio. ¿cuál fue la más útil? ¿por qué?*
2. *¿Cuál consideran fue el mayor logro? ¿Por qué?*

El comportamiento de la excepción teniendo en cuenta su útil funcionamiento para saber cuando se deben implementar.

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

Adaptarse a otro software de desarrollo, donde se invirtió tiempo de cómo manejar la aplicación y conocer bien los usos y las funcionalidades que ofrece.

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

*resultados?*

Plantear ideas para la correcta finalización del laboratorio, nos comprometemos hacer mejor uso del tiempo.