**UNIVERSIDAD ESCUELA COLOMBIANA DE INGENIERÍA JULIO GARAVITO**

LABORATORIO 4

PROGRAMACIÓN ORIENTADA A OBJETOS

NOMBRES:

CHRISTIAN ALFONSO ROMERO MARTÍNEZ

ANDERSON FABIAN GARCIA NIETO

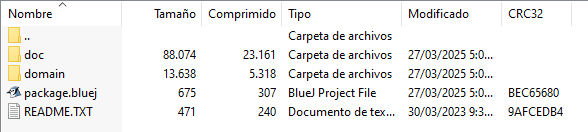
PROFESORA:

MARIA IRMA DIAZ ROZO

28/03/25

**PRACTICANDO MBB Y BDD CON EXCEPCIONES**

En este punto vamos a aprender a diseñar, codificar y probar usando excepciones. Para esto se van a trabajar algunos métodos de la clase Core.

1. **En su directorio descarguen los archivos contenidos en u****nits.zip revisen el contenido y estudien el diseño estructural de la aplicación (únicamente la zona en azul).**

Dentro del directorio raíz, en este caso siendo Units, se encuentran los siguientes archivos y subdirectorios:

Donde:

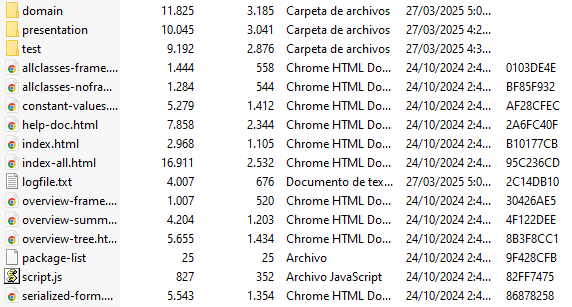
-El primer subdirectorio es una forma en la que Winrar representa volver atrás.



-El segundo subdirectorio, donde se encuentran los archivos .html que conforman la documentación de las respectivas clases.



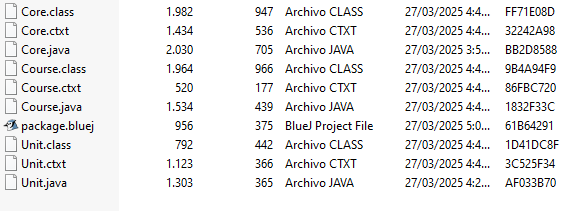
Dentro de esta se encuentra:



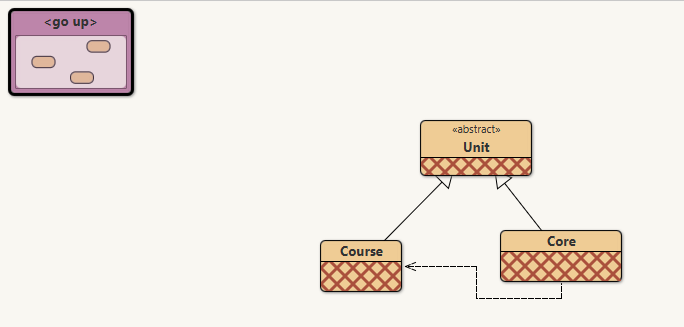
Efectivamente se encuentran los archivos .html, que conforman la documentación inicial, también con los archivos que le dan al diseño.

Dentro de los subdirectorios que se ven, estan los archivos que corresponden a la documentación de cada paquete

-Regresando al subdirectorio raíz encontramos, la carpeta domain, dentro de esta se encuentran **3 clases:** Core, Course y Unit, esto se observa gracias a sus respectivos archivos .class. .java y .ctxt, donde son los archivos compilados generados a partir de los archivos .java., archivos de código fuente escritos en el lenguaje de programación Java y los archivos de contexto específicos de BlueJ, respectivamente



Ya entrando en bluej analizado lo que está ahí, entonces:



Se observa, lo mismo que en el astah, la misma jerarquía, donde Course y Core heredan de la clase Unit, y Core usa a Course.

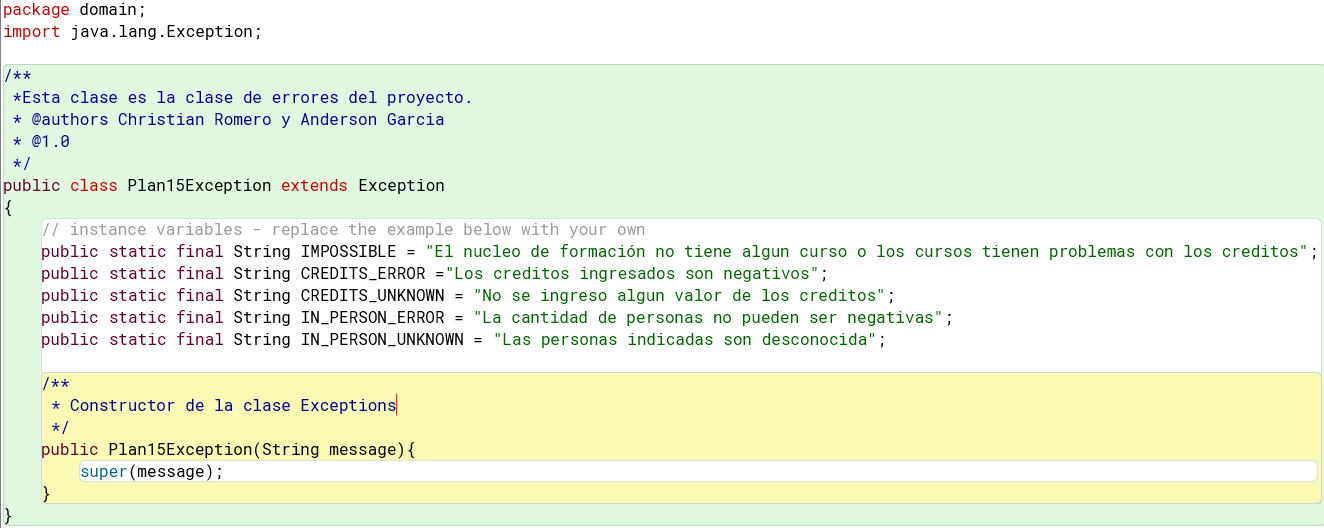
En resumidas palabras, con lo anterior sustentado y mirando el diseño (astah) se puede decir lo siguiente:

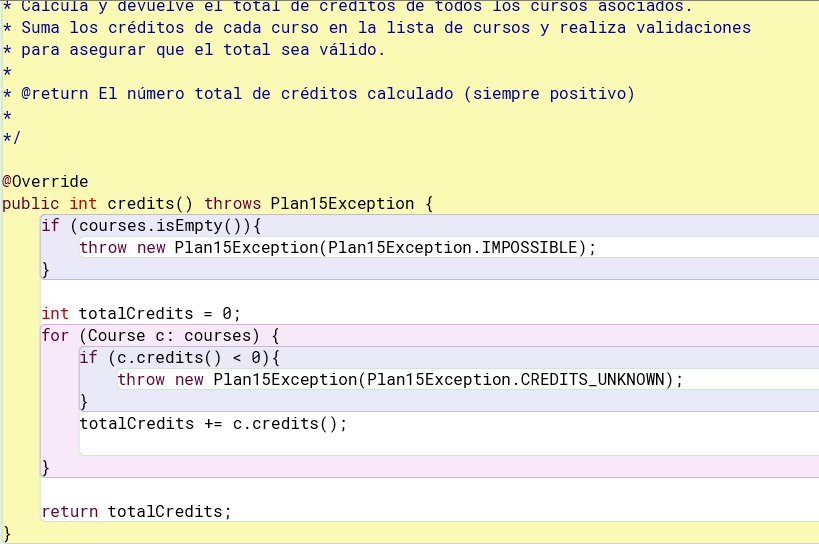
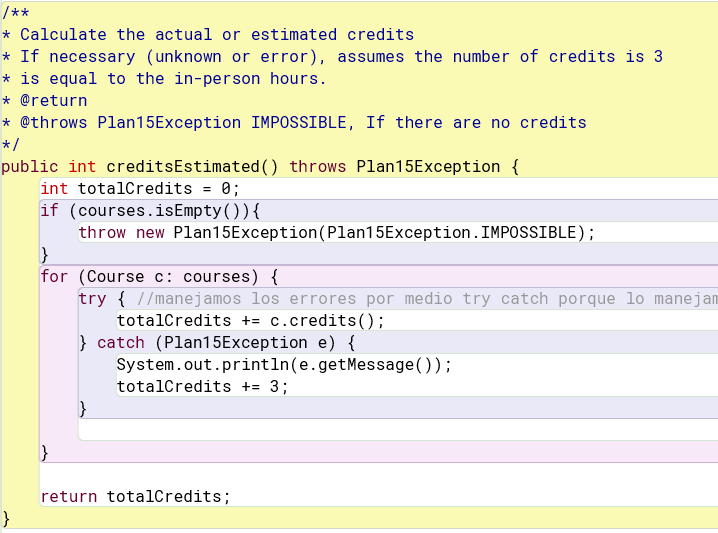
* Existen 3 clases diferentes Core, Course y Unit
* Las clases anteriormente mencionadas hacen parte del paquete domain
* Las clases Course y Core heredan de la clase Unit
* La clase Course puede ser usada por la clase Core, pero en sentido contrario esto ya no aplíca
* Existen dos objetos que pueden ser instanciados, Core y Course, Unit no debido a que es una clase abstracta

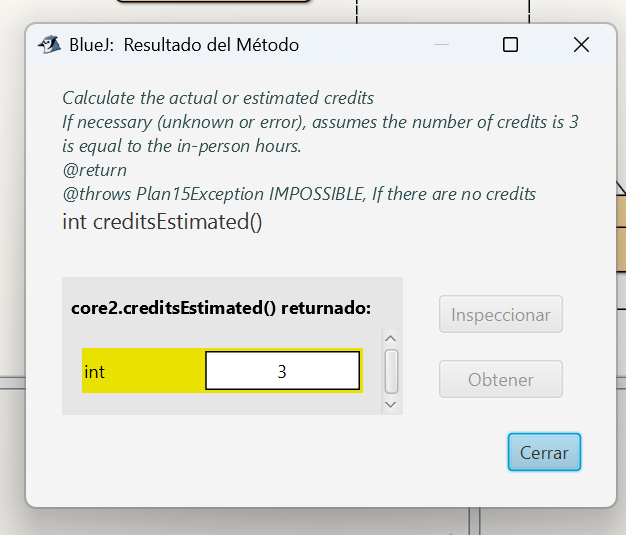
1. **Expliquen por qué el proyecto no compila. Realicen las adiciones necesarias para lograrlo.**

El proyecto no compila debido a que en la codificación respectiva de la clase se hace uso de otra clase de nombre Plan15, quien actuará como nuestra clase de Excepciones, para que el proyecto compile, se necesita adicionar esta clase con las respectivas excepciones.

En base al diseño proporcionado en astah, se realiza la codificación:

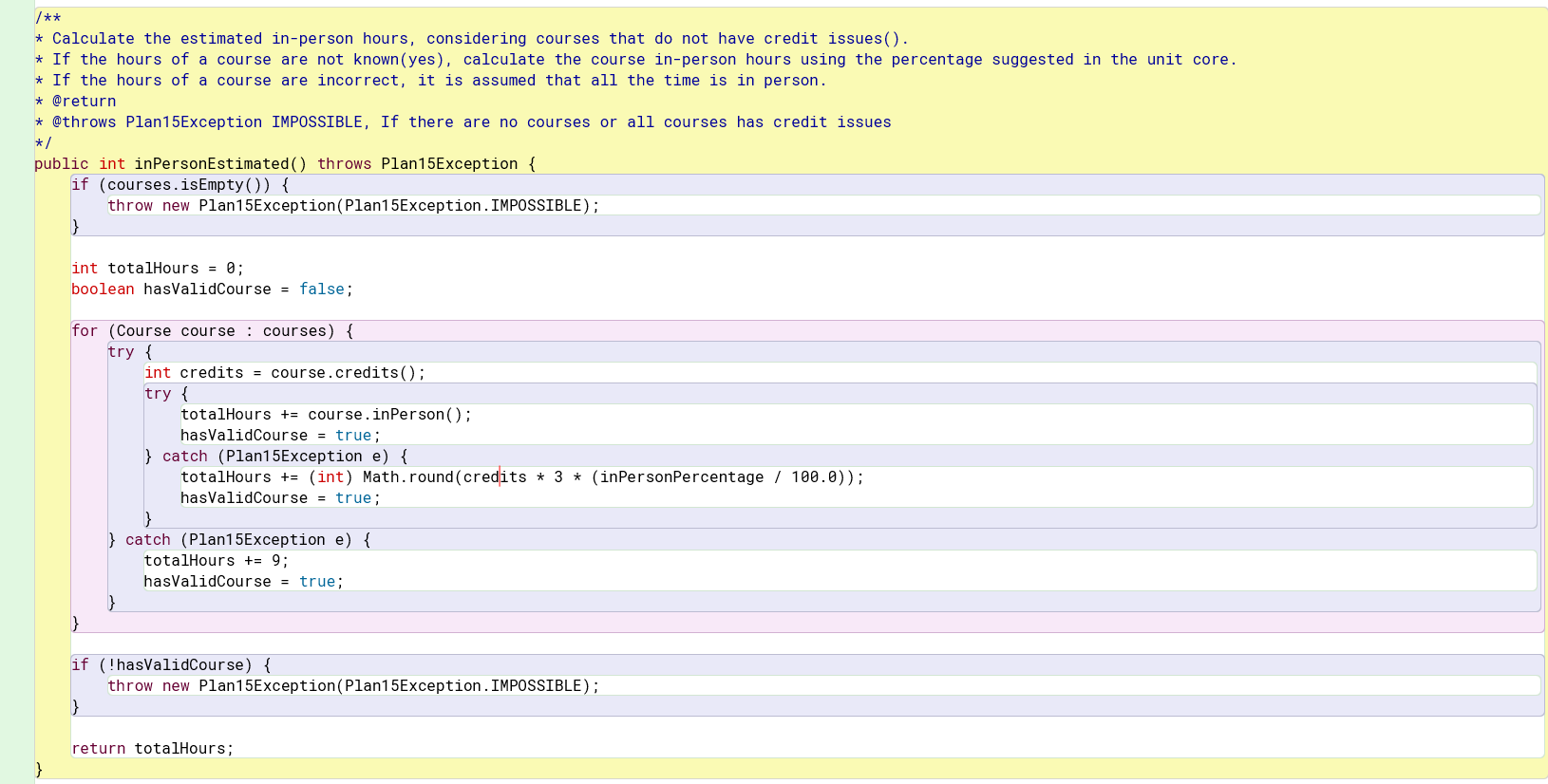


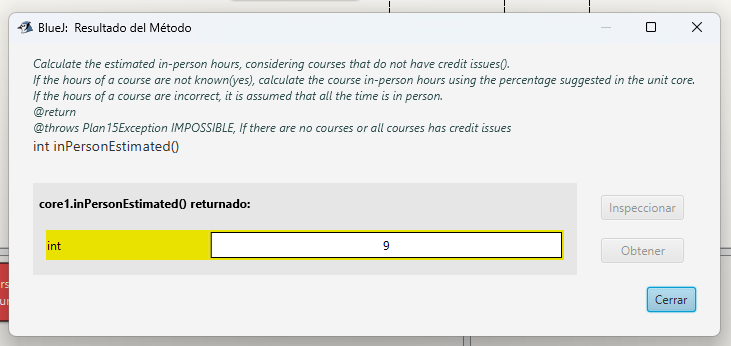
1. **Dado el diseño y las pruebas, documenten y codifiquen el método credits().**
2. **Dada la documentación y el diseño, codifiquen y prueben el método creditsEstimated().**

****

Probamos credits Estimated() con un departamento que tenía un solo curso con 0 créditos, y tuvimos como retorno 3, correspondiente a 3 créditos y al correcto funcionamiento.

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

****

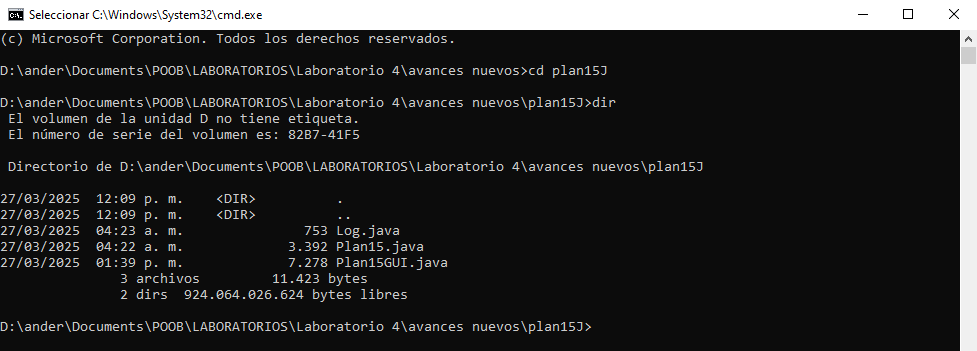
****

Probamos el método con una clase de 3 créditos, al un crédito valer 3 horas, deberíamos tener como tiempo 9 horas, tal como se muestra en pantalla

**Plan15**

**EN CONSOLA**

**1. En su directorio descarguen los archivos contenidos en plan15.zip, revisen el contenido.**

****

****

**¿Cuántos archivos se tienen?**

Cuenta con 3 archivos.

**¿Cómo están organizados?**

No están organizados, solo se encuentran todos los archivos en un único directorio

**¿Cómo deberían estar organizados?**

Analizando el código de los archivos .java se observa que los archivos Log y Plan15 deben estar en la carpeta (paquete) domain, en cambio Plan15GUI debe estar en la carpeta presentation(paquete). Obviamente también con los respectivos directorios bin,src y docs

**2. Estudien el diseño del programa: diagramas de paquetes y de clases.**

**¿Cuántos paquetes tenemos?**

Tenemos 3 paquetes

Presentation

Domain

Test

**¿Cuántas clases tiene el sistema?**

El sistema tiene un total de 8 clases

**¿Cómo están organizadas?**

Presentation = 1, qué es la clase Plan15GUI

Domain = 6, que son las clases Log, Core, Courses, Plan15, Unit, Plan15Exception

Test = 1, siendo la clase de este: CoreTeste

**¿Cuál es la clase ejecutiva?**

La clase ejecutiva es Plan15GUI puesto que contiene el método main.

**3. Prepare los directorios necesarios para ejecutar el proyecto.**

**¿Qué estructura debe tener?**

La estructura que sigue el proyecto en la siguiente bin, src y docs, donde en el directorio src están los archivos .java, en el bin están los .class y en el docs los .html.

**¿Qué clases deben tener?**

Debe tener las clases de domain puesto que son las clases funcionales.

**¿Dónde están esas clases?**

Se encuentran en units.zip

**¿Qué instrucciones debe dar para ejecutarlo?**

Debemos unificar los archivos de Domain y Presentation, consiguientemente

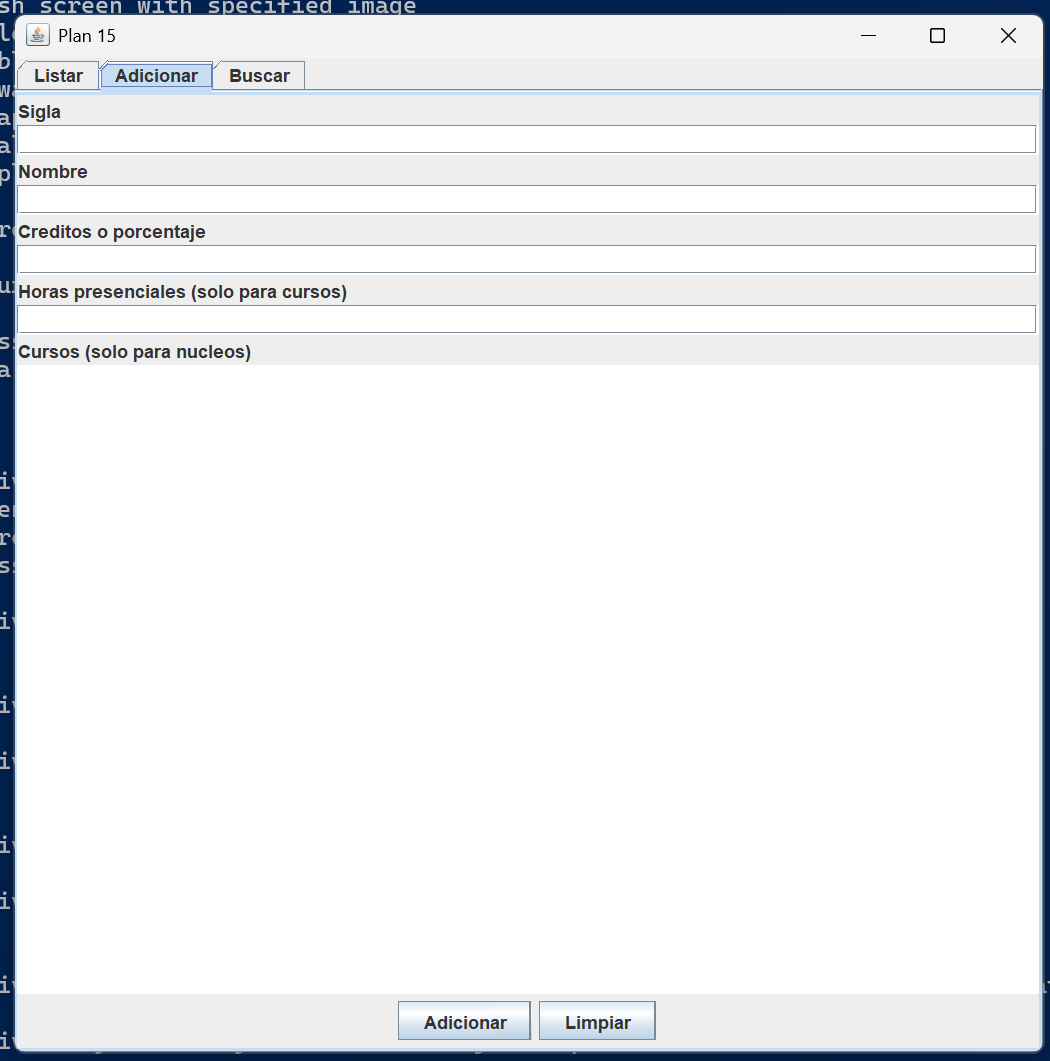
compilamos

javac -d bin -sourcepath src src/presentation/\*.java src/domain/\*.java

Ejecutamos

java -cp bin presentation/Plan15GUI

**4. Ejecute el proyecto**,

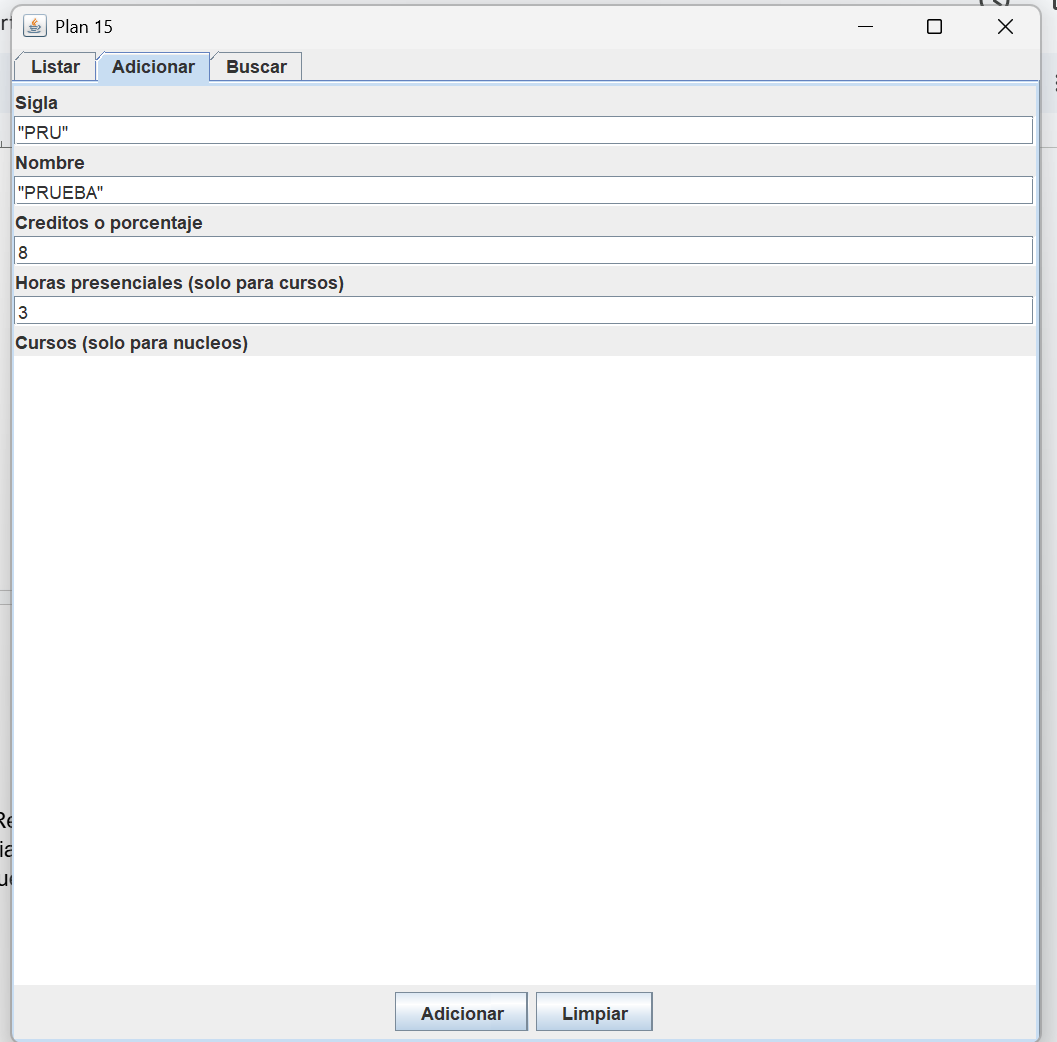


**¿Qué funcionalidades ofrece?**

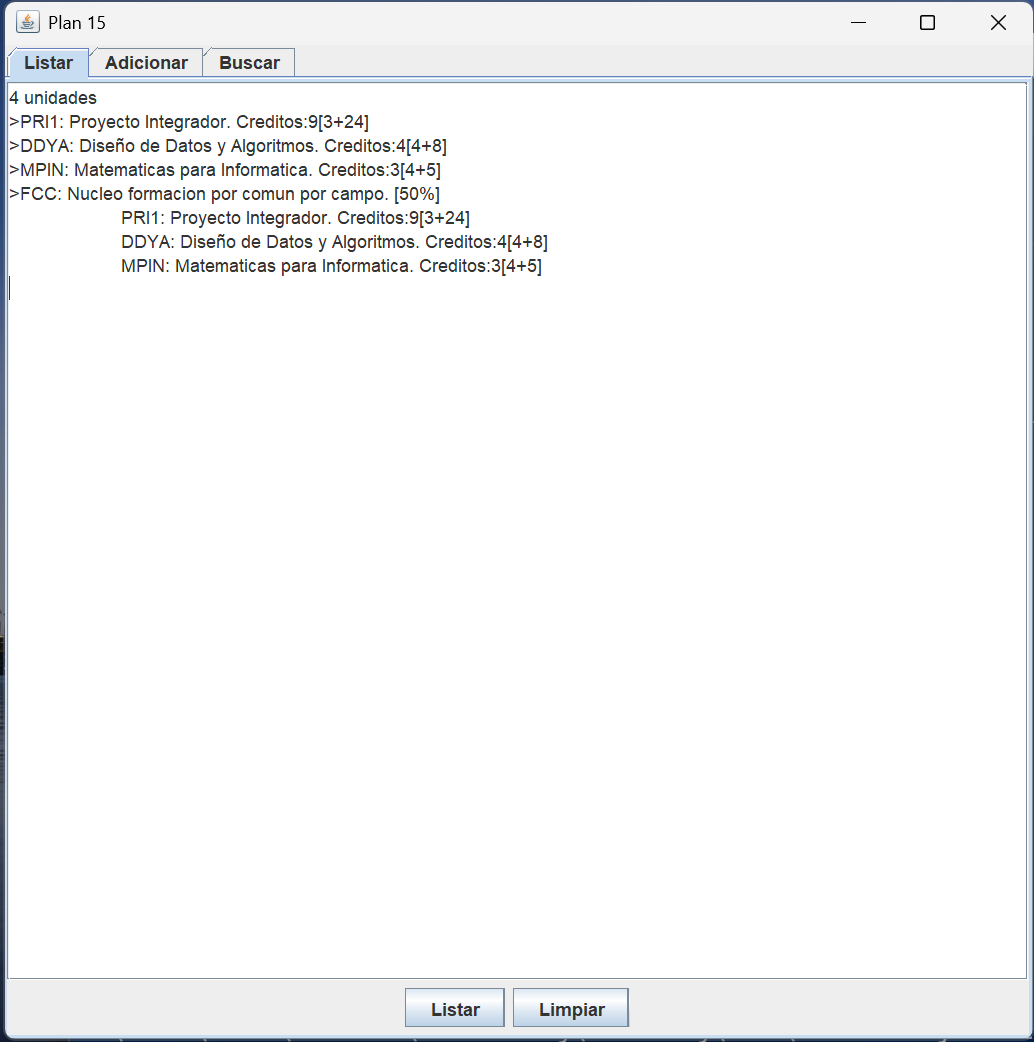
Las funcionalidades que nos ofrece son, enlistar, adicionar y buscar.

**¿Cuáles funcionan?**

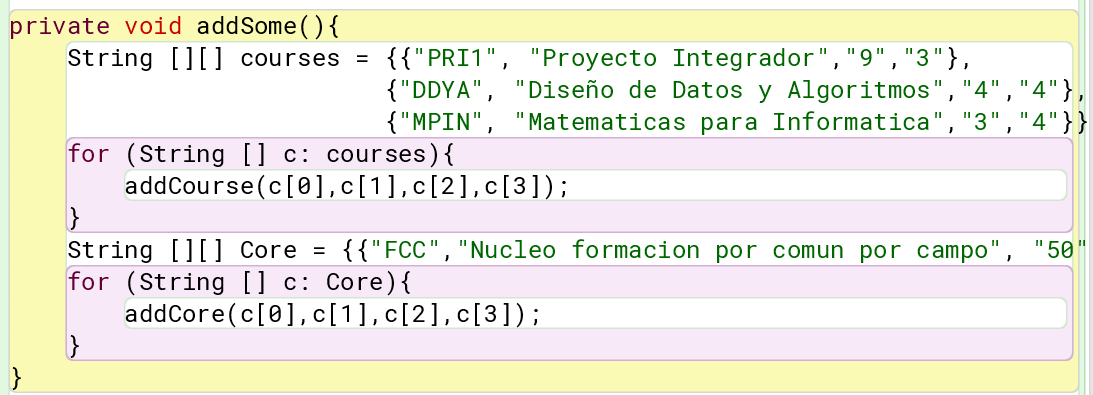
Funciona enlistar, y adicionar.



**5. Revisen el código y la documentación del proyecto. ¿De dónde salen las unidades iniciales?**



Las unidades iniciales se encuentran en el metodo addsome() en la clase plan15



**¿Qué clase pide que se adicionen?**

La clase que pide que se adicionen es plan15, pues es a la que pertenece addsome()

**¿Qué clase los adiciona?**

La clase que hace las adiciones es course y core respectivamente, pues por medio de addcourse() y addCore() es que se adicionan.

**ADICIONAR Y LISTAR. TODO OK.**

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

**1. Adicionen un nuevo curso y un nuevo nucleo**

Curso

DOSW Desarrollo y Operaciones Software 4 y 4

Core

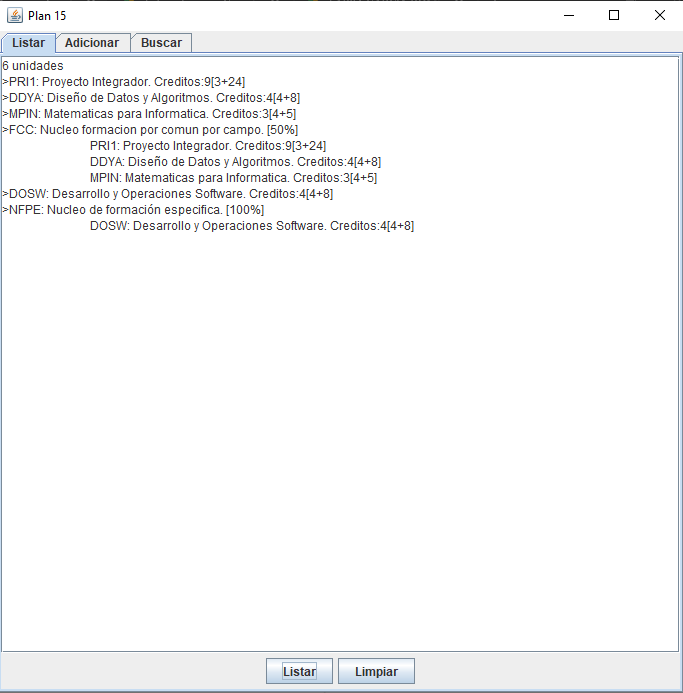
NFPE Nucleo de formación específica 100

DOSW

**¿Qué ocurre?**

Principalmente se añade la clase y después se añade el núcleo de formación, donde se indicaba que a este pertenecía la clase inicialmente creada. A la hora de listar, se mostró primero la clase, después el núcleo de formación y se la clase que pertenece a este.

**¿Cómo lo comprueban? Capturen la pantalla.**



**¿Es adecuado este comportamiento?**

Si, nos encontramos de acuerdo con el comportamiento, pues se agregó, y además recibió cursos core

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

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

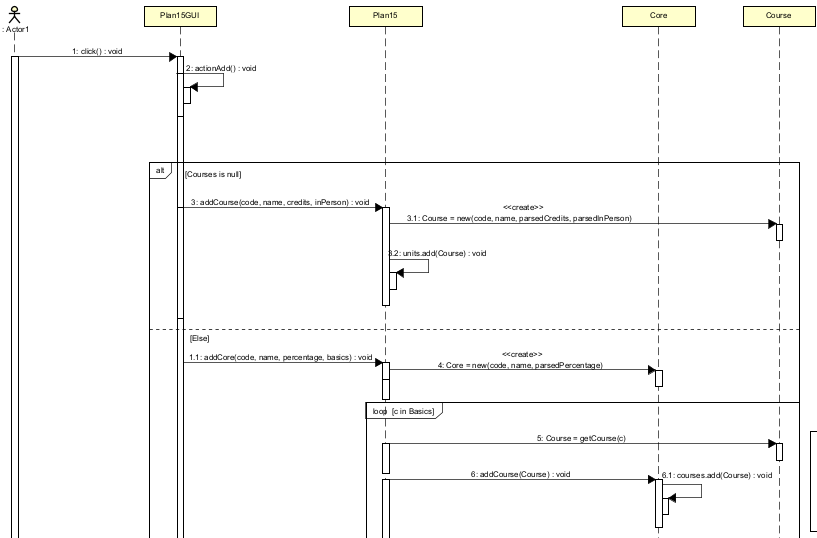
**dominio?**

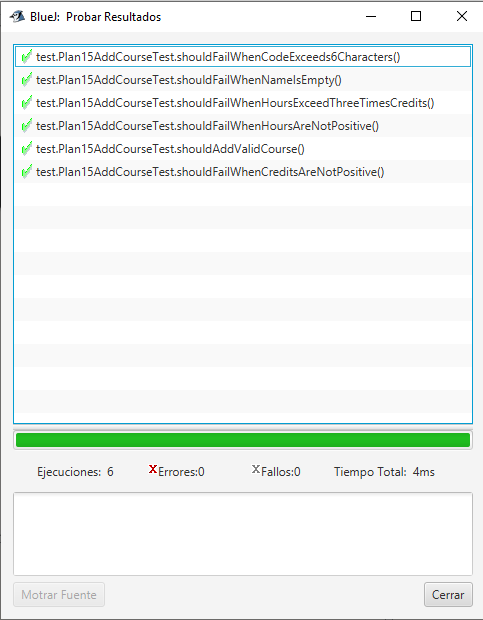
-El responsable en la capa de presentación es el método **action.add()** de la clase PLAN15GUI, porque este lo que hace es que cuando se le da click al botón de Adicionar, se ejecuta y la información que se le haya proporcionado se envía a la clase Plan15, especificamente a los metodos addCourse() y addCore(), siendo los encargados de crear el curso o el núcleo respectivo en la capa de dominio.

-Los responsables en la capa de dominio son addCore() y addCourse() en la clase Plan15, estos usan a los respectivos constructores (Core y Course), y también un método cuando se le añade un curso a un core.

**3. Realicen ingeniería reversa para la capa de dominio para adicionar.**

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

****



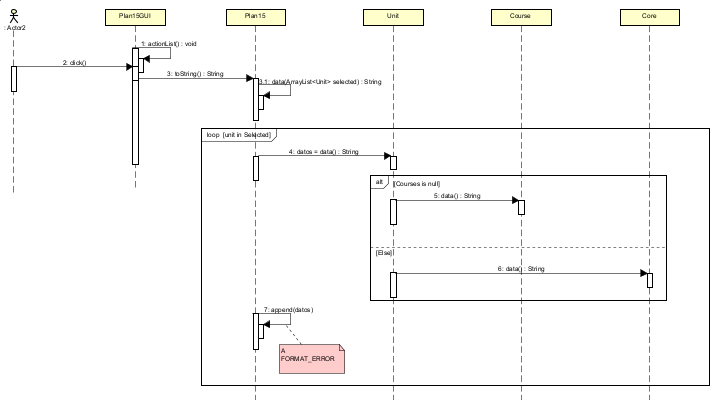
**4. Revisen el código asociado a listar en la capa de presentación y la capa de dominio. ¿Qué método es responsable en la capa de presentación? ¿Qué método en la capa de dominio?**

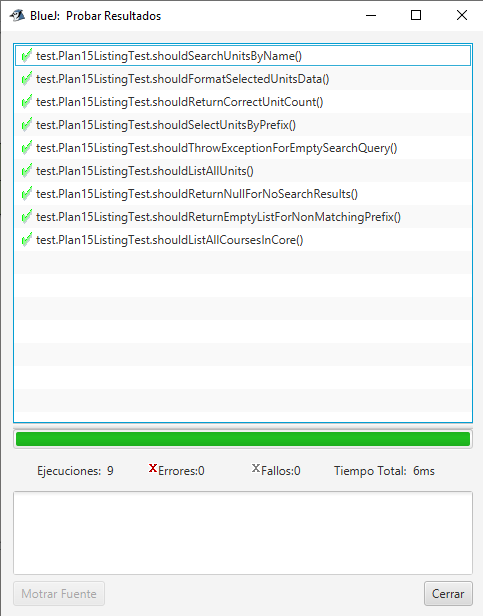
El método responsable de listar (Mostrar la información ) en la capa de presentación es **ActionList()**, debido a que este le envía la orden a la clase Plan15, para que este trabaje sobre la capa de dominio.

El método responsable de listar en la capa de dominio es **data()**, siendo este método usado en las clases Plan15, Core y Course, está en general permite obtener la información de todos los cursos, y se imprimen en la interfaz del usuario.

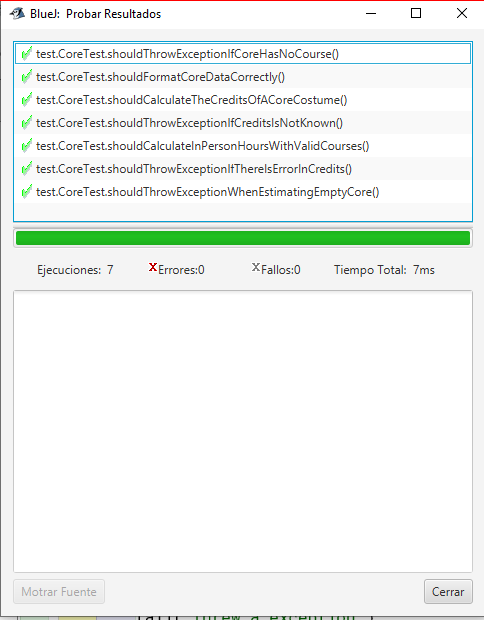
**5. Realicen ingeniería reversa para la capa de dominio para listar.**

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

****

****

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

****

Las nuevas pruebas que se añadieron fueron:

-shouldFormatCoreDataCorrectly()

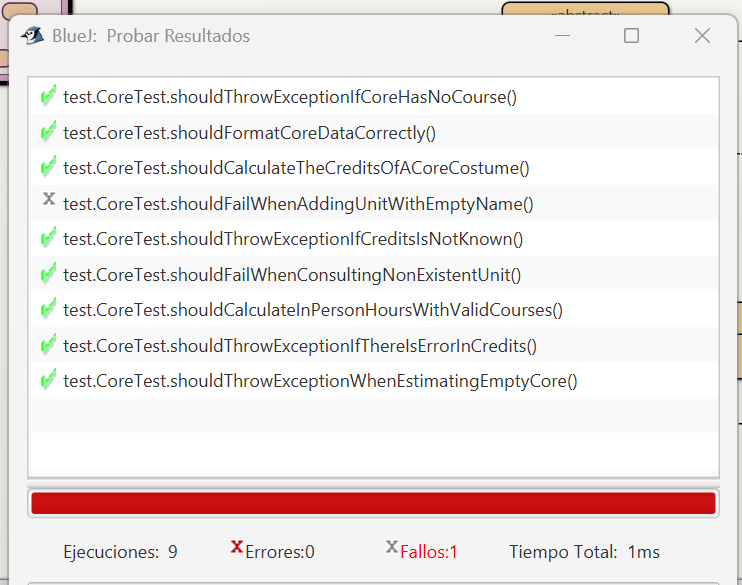
-shouldThrowExceptionWhenEstimatingEmptyCore()

**ADICIONAR UNA UNIDAD. FUNCIONALIDAD ROBUSTO**

**Para cada uno de los siguientes casos realice los pasos del 1 al 4.**

**a. ¿Y si el nombre de la unidad no existe?**

**1. Propongan una prueba de aceptación que genere el fallo.**



Generamos la prueba donde se verifica si al ingresar algún dato vacío lo adiciona.

**2. Analicen el diseño realizado. Para hacer el software robusto:**

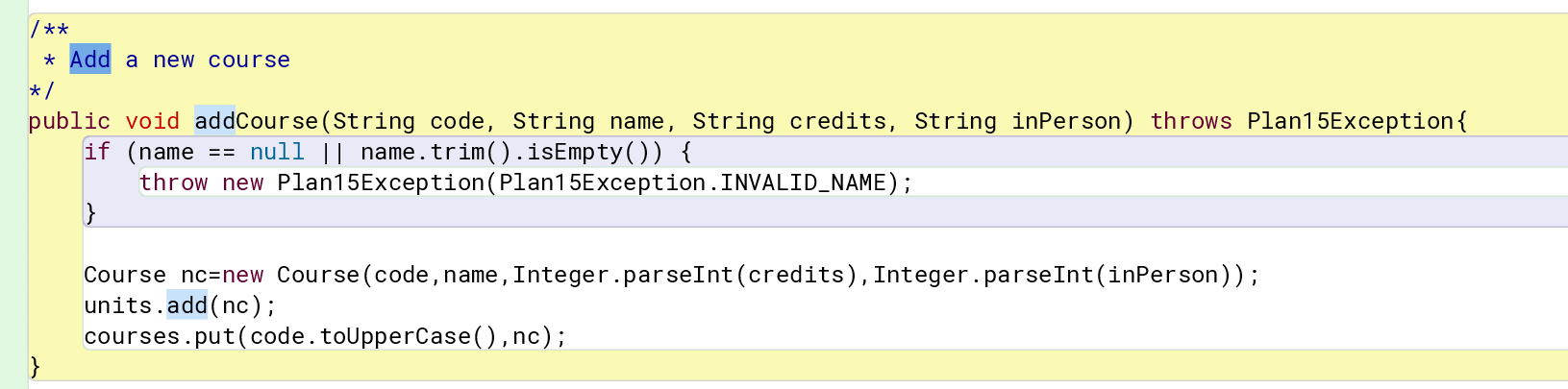
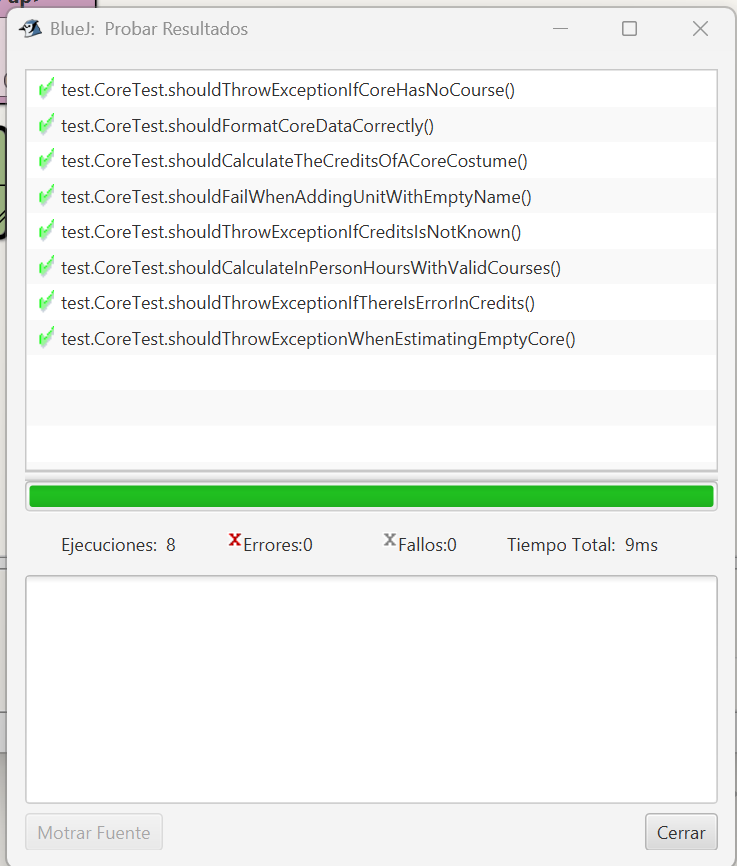
**¿Qué método debería lanzar la excepción?**

El método que debe lanzar la exception es addCourse()

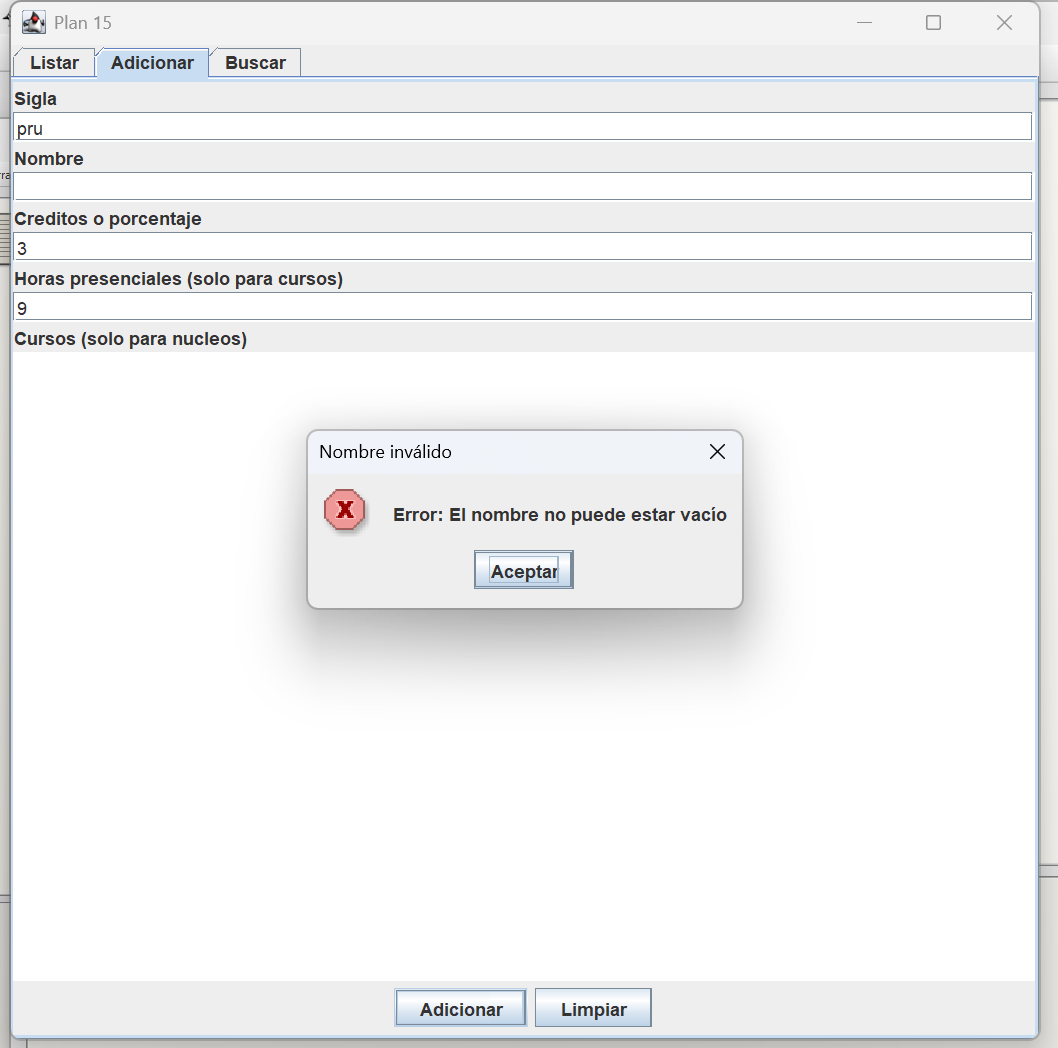
**¿Qué método debería atenderla? Explique claramente.**

El método que la atiende es actionadd, puesto en este maneja todo lo que tiene que ver con add, lo que nos permite tener un código mejor estructurado, así mismo el error se radica en los valores ingresados por el usuario, por qué se debe atender en presentation y se pueden correr allí mismo

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



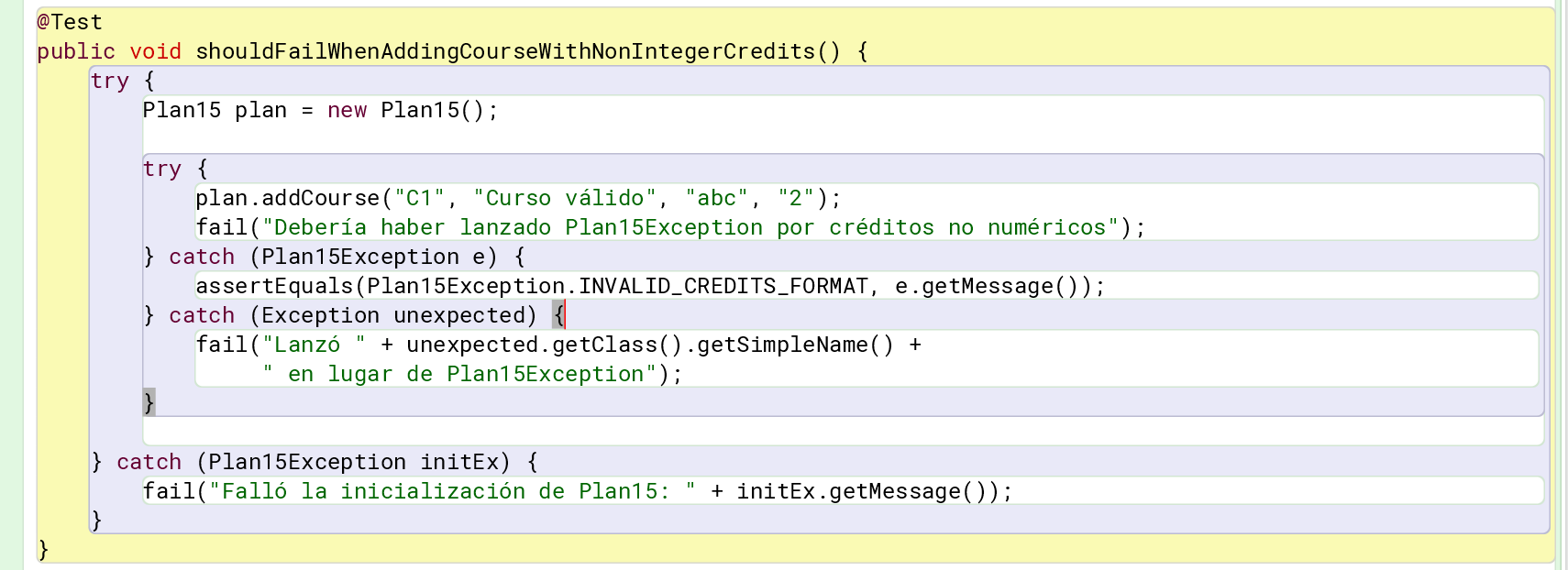
**4. Ejecuten nuevamente la aplicación con el caso de aceptación propuesto en 1. ¿Qué sucede ahora? Captura Pantalla**

****

Se muestra el error, para este caso se especifica que el curso debe tener nombre

**b. ¿Y si los valores enteros no son enteros?**

1. **Propongan una prueba de aceptación que genere el fallo.**

****

1. **Analicen el diseño realizado. Para hacer el software robusto:**

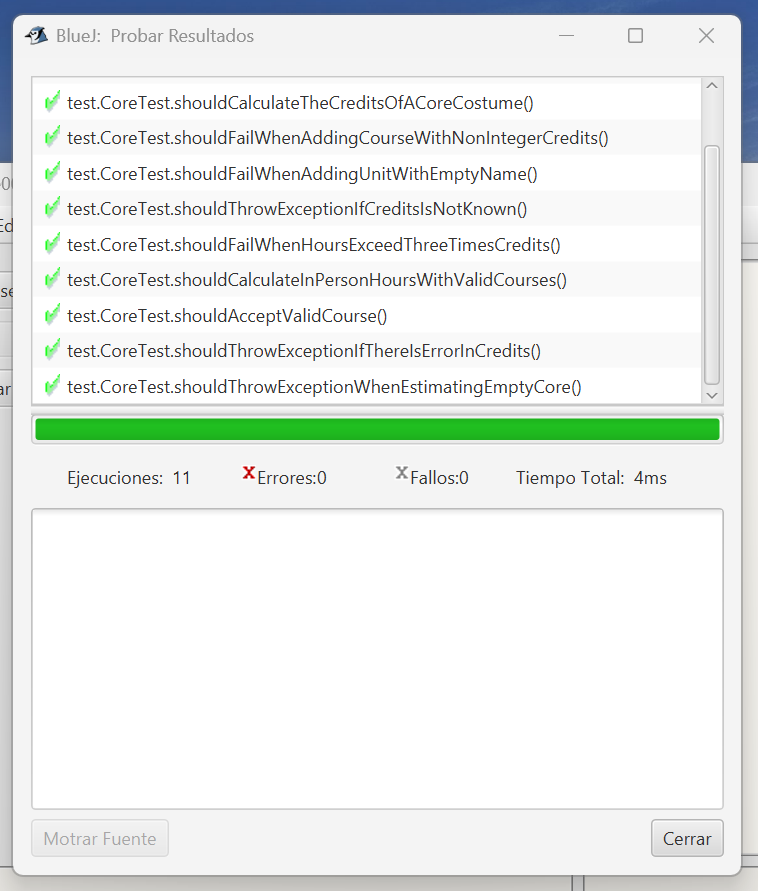
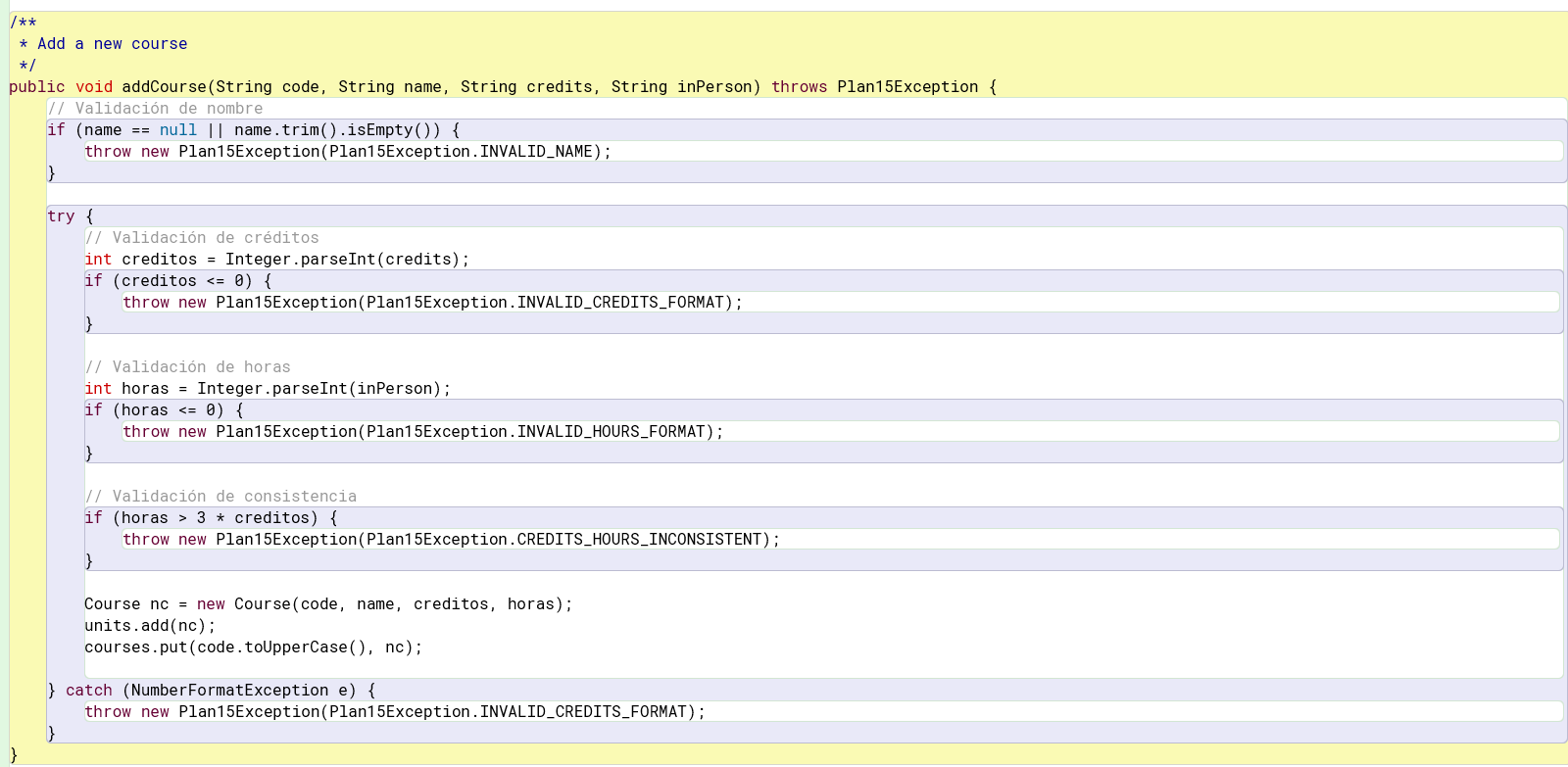
**¿Qué método debería lanzar la excepción?**

El método que debe lanzar la exception es addCourse()

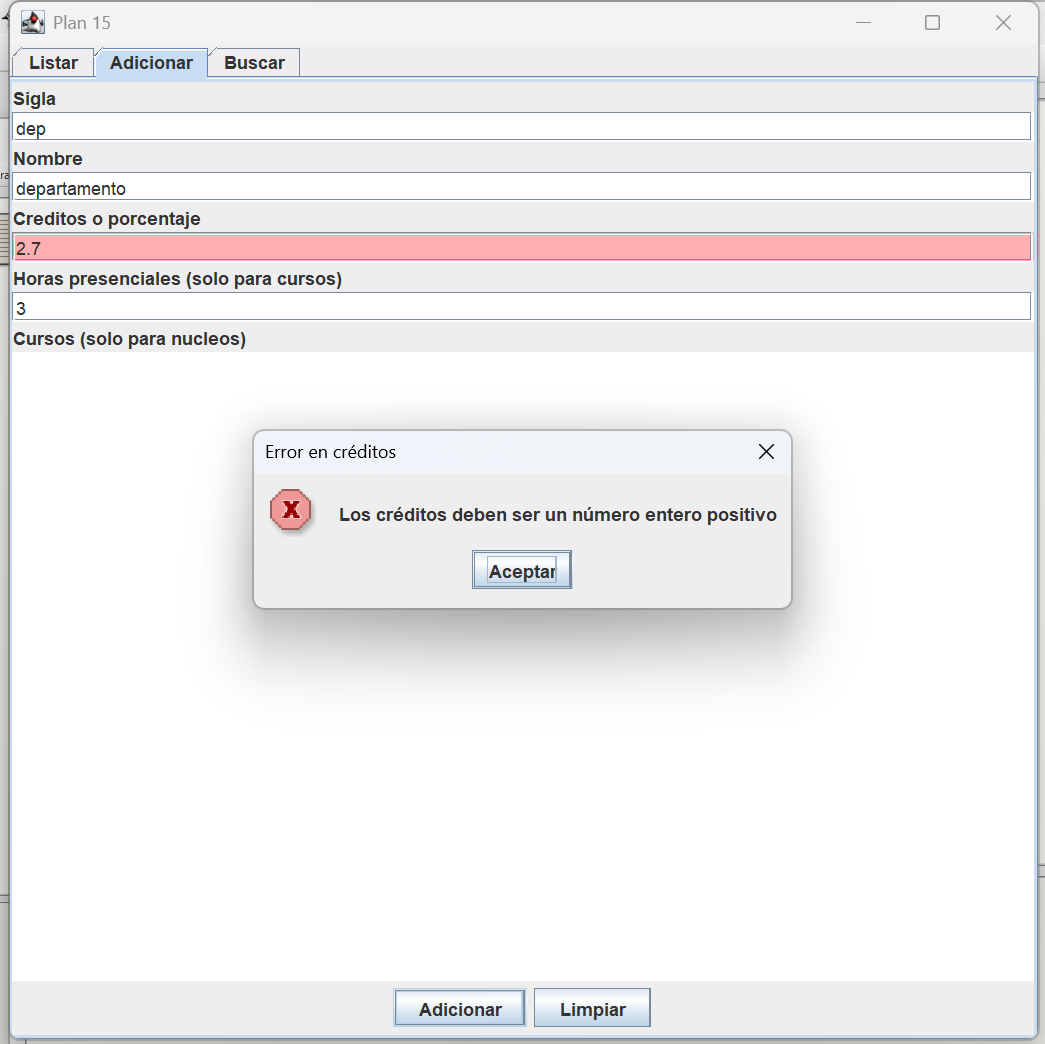
**¿Qué método debería atenderla? Explique claramente.**

El método que la atiende es actionadd, puesto en este maneja todo lo que tiene que ver con add, lo que nos permite tener un código mejor estructurado, así mismo el error se radica en los valores ingresados por el usuario, por qué se debe atender en presentation y se pueden correr allí mismo

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

****

1. **Ejecuten nuevamente la aplicación con el caso de aceptación propuesto en 1. ¿Qué sucede ahora? Captura pantalla**

****

Se muestra en pantalla un aviso donde se especifica el error, en este caso que los números deben ser positivos

**c. ¿Y si el portage no está entre 0 y 100?**

1. **Propongan una prueba de aceptación que genere el fallo.**

****

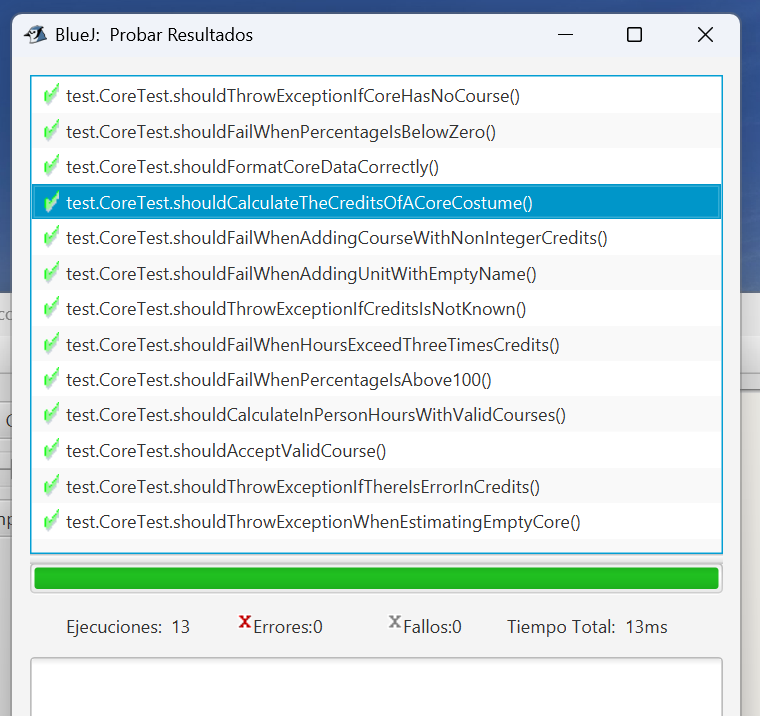
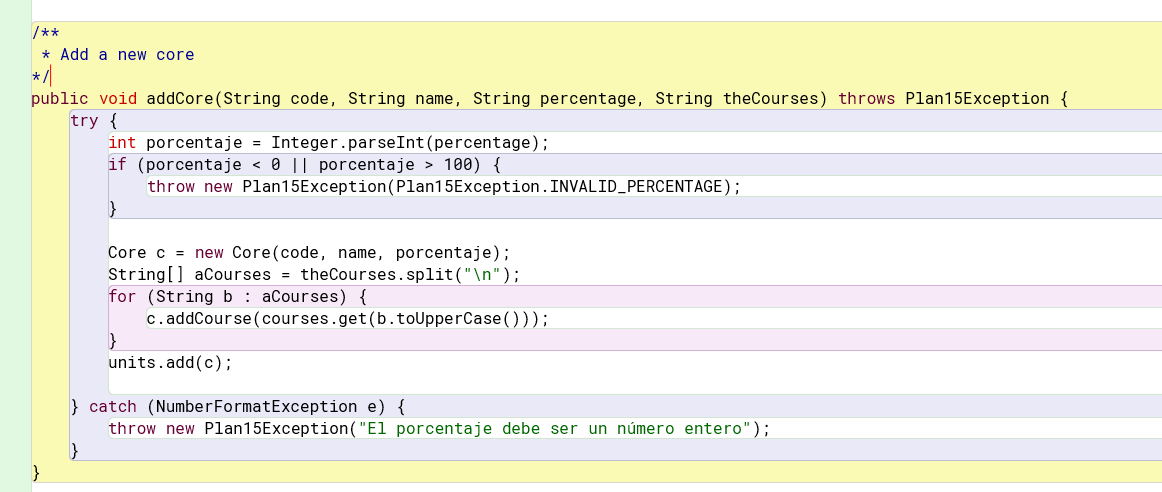
1. **¿Qué método debería lanzar la excepción?**

El método que debe lanzar la exception es addCore()

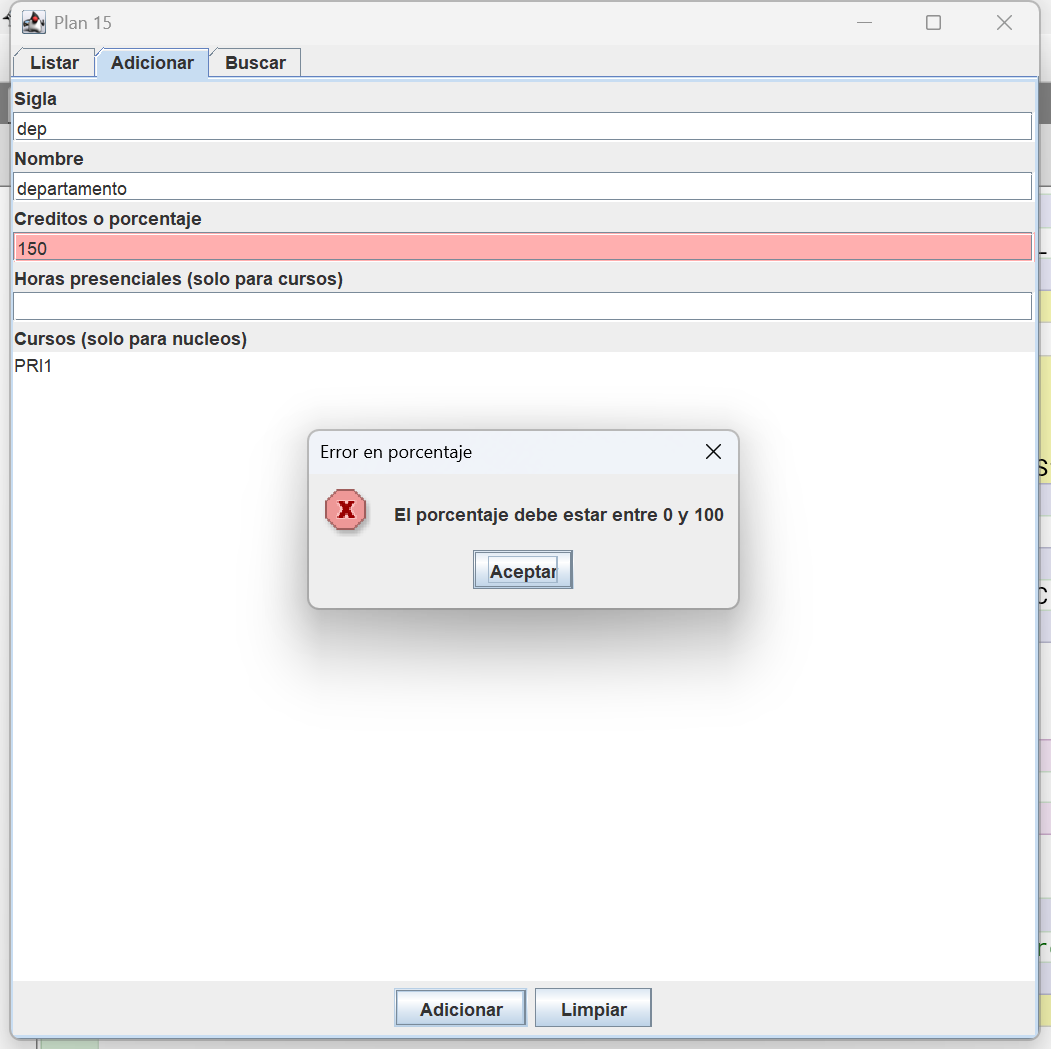
**¿Qué método debería atenderla? Explique claramente.**

El método que la atiende es actionAdd(), puesto en este maneja todo lo que tiene que ver con add(), lo que nos permite tener un código mejor estructurado, así mismo el error se radica en los valores ingresados por el usuario, por qué se debe atender en presentation y se pueden correr allí mismo

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

****

1. **Ejecuten nuevamente la aplicación con el caso de aceptación propuesto en 1. ¿Qué sucede ahora? Captura pantalla**

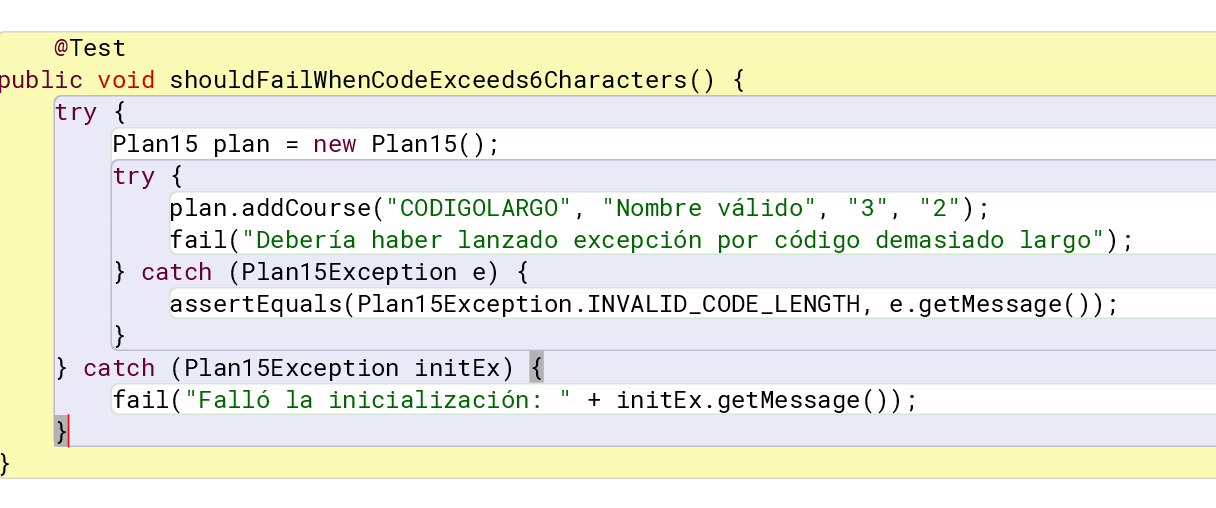
****

Se muestra en pantalla un aviso donde se especifica el error, en este caso que el porcentaje debe estar en el intervalo de 0 a 100

**d. Proponga una nueva condición**

1. **Propongan una prueba de aceptación que genere el fallo.**

Elegimos limitar la longitud en el campo SIGLAS, esto lo pensamos por las reglas de negocio donde las siglas significan una forma fácil de nombrar cada course o core

****

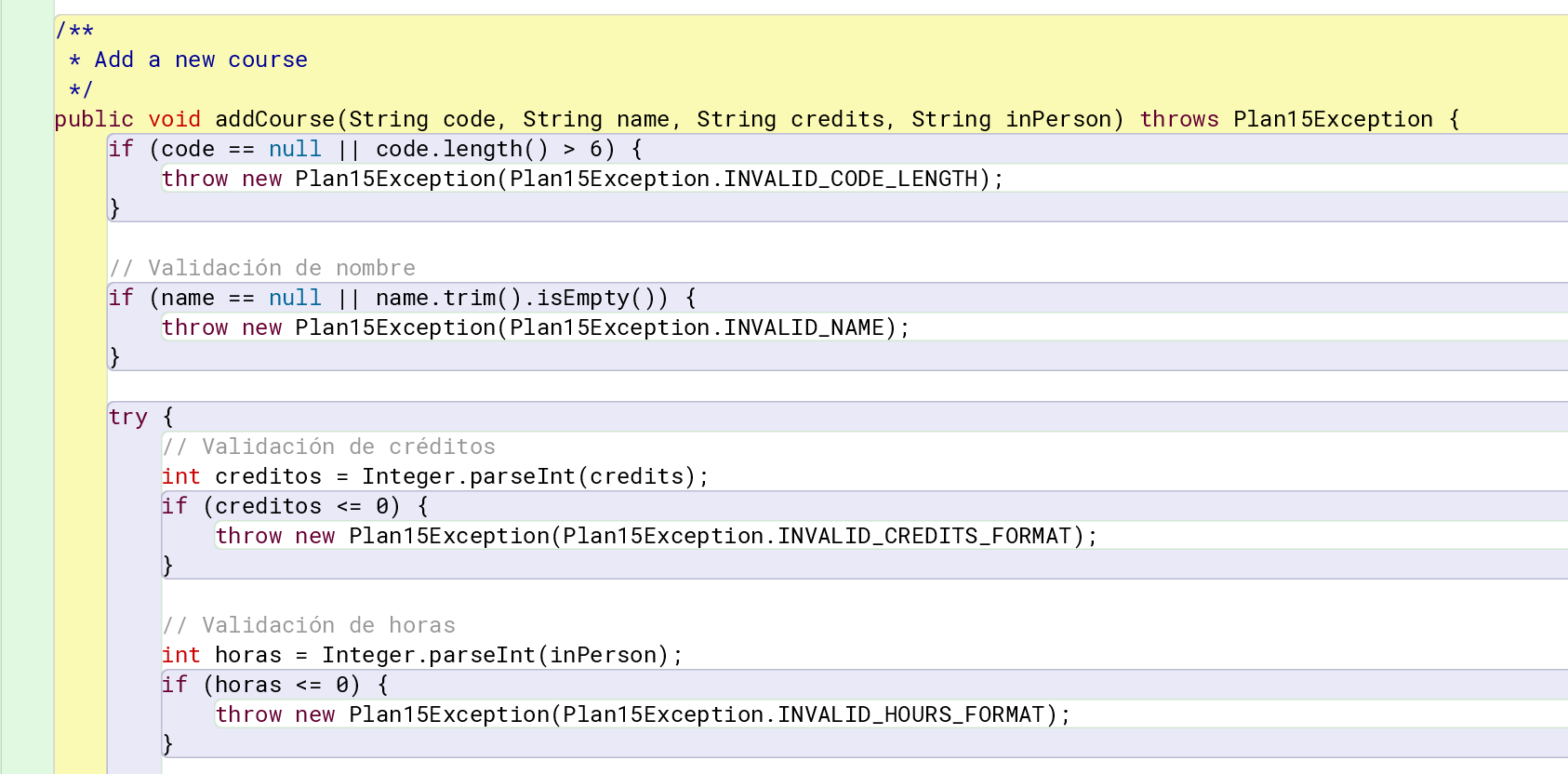
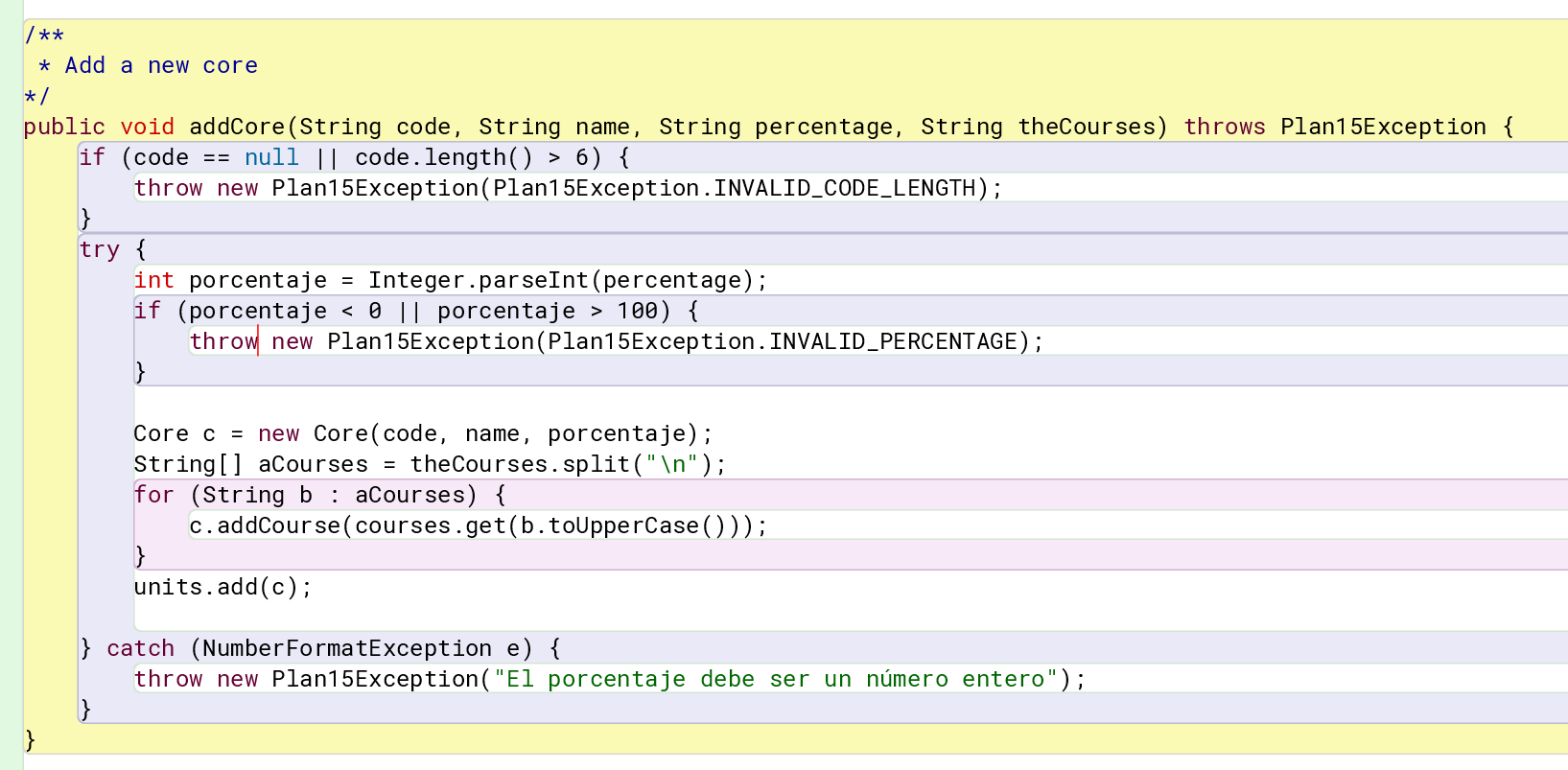
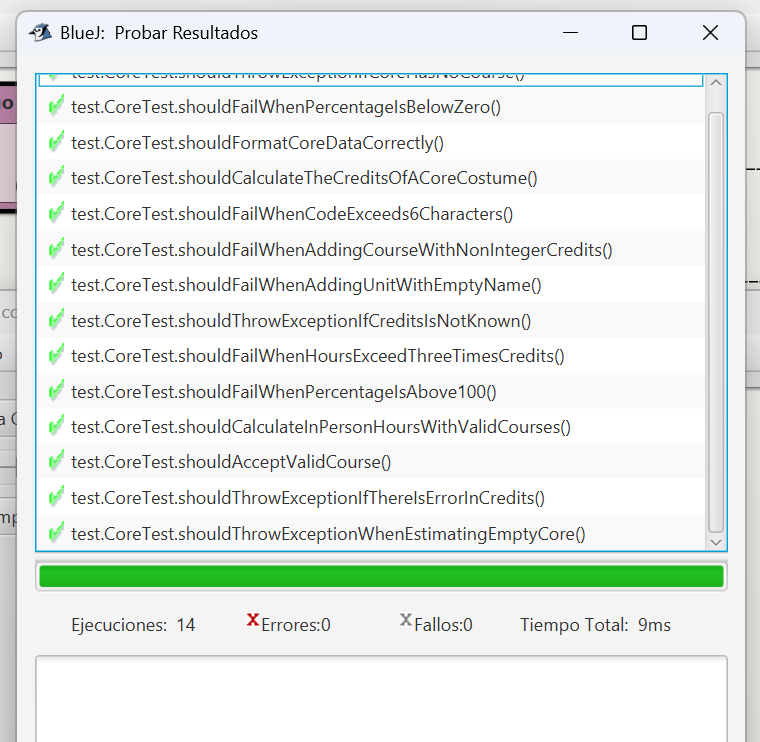
1. **¿Qué método debería lanzar la excepción?**

El método que debe lanzar la exception es addCore() y addcourse()

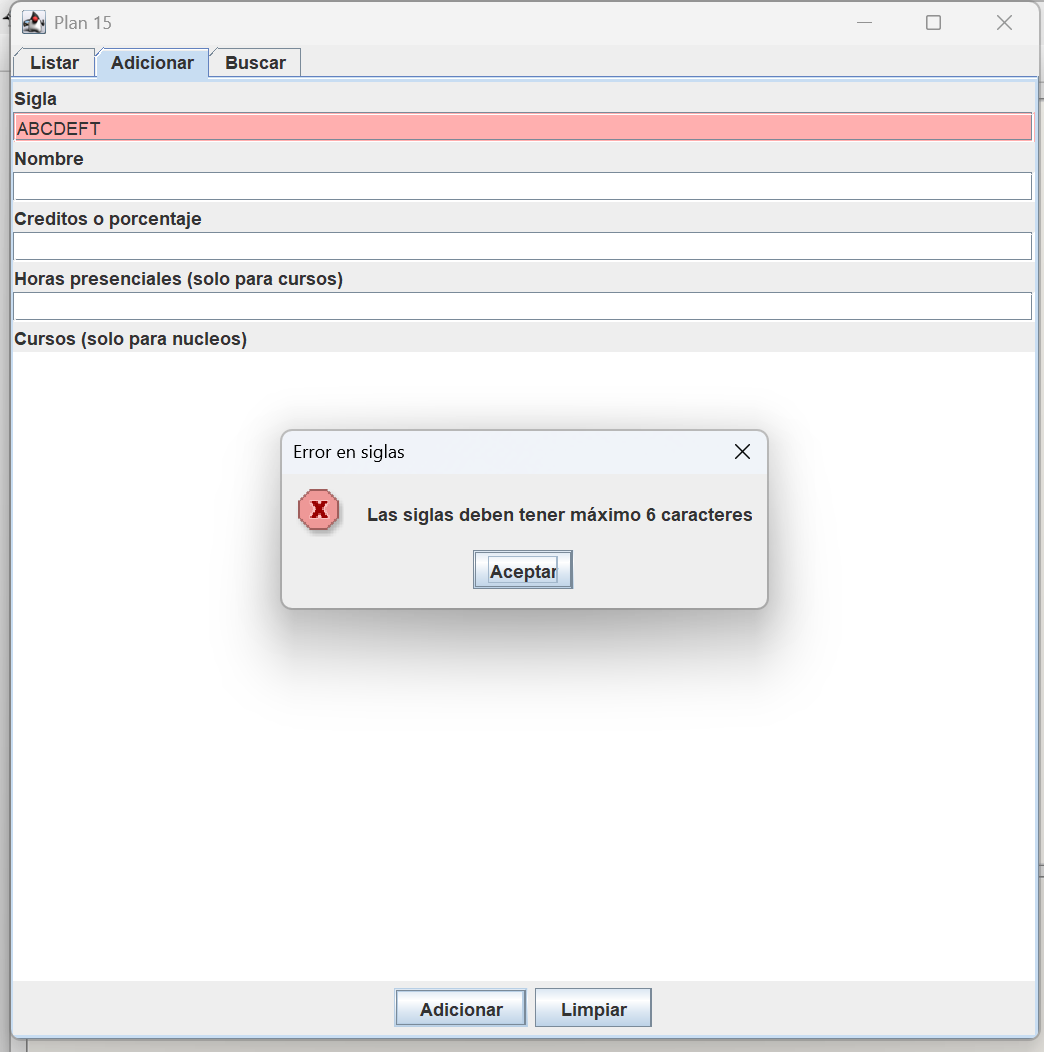
**¿Qué método debería atenderla? Explique claramente.**

El método que la atiende es actionAdd(), puesto en este maneja todo lo que tiene que ver con add(), lo que nos permite tener un código mejor estructurado, así mismo el error se radica en los valores ingresados por el usuario, por qué se debe atender en presentation y se pueden correr allí mismo

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

****

1. **Ejecuten nuevamente la aplicación con el caso de aceptación propuesto en 1. ¿Qué sucede ahora? Captura pantalla**

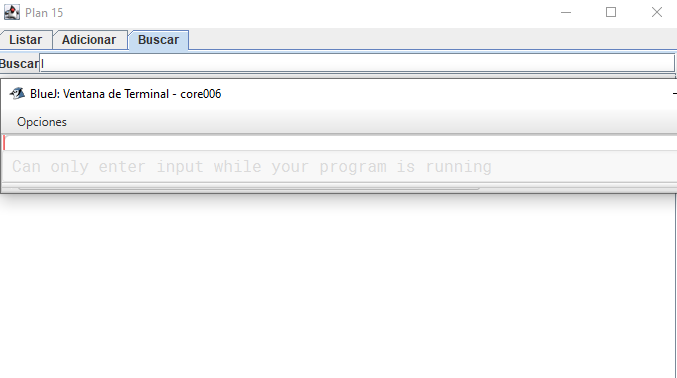
****

En la pantalla se muestra un mensaje de error, donde especifica que se debe a que en el campo se SIGLAS excede el límite de 6 caracteres

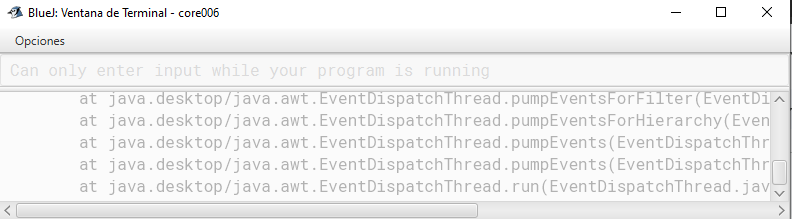
**CONSULTANDO POR PATRONES. !NO FUNCIONA Y QUEDA SIN FUNCIONAR¡**

1. **Consulten una unidad que inicie con I. ¿Qué sucede? ¿Qué creen que pasó? Capturen el resultado. ¿Quién debe conocer y quien NO debe conocer esta información?**

Cuando se agrega una o varias letras “I”, lo que pasa es que sale una especie de error, y no busca nada, sale lo mismo.



En otro caso si se escribe otra cosa, con otros caracteres, sale lo siguiente:



Lo que creemos que está pasando es que se está tratando de buscar una clase que empiece con la letra I, pero esta no existe, entonces se está creando un montón de errores, a través del stack trace.

Las únicas personas que deberían conocer estos errores, son las personas que están codificando/desarrolladores, el usuario no, debido a que se vuelve una falla de seguridad de la aplicación.

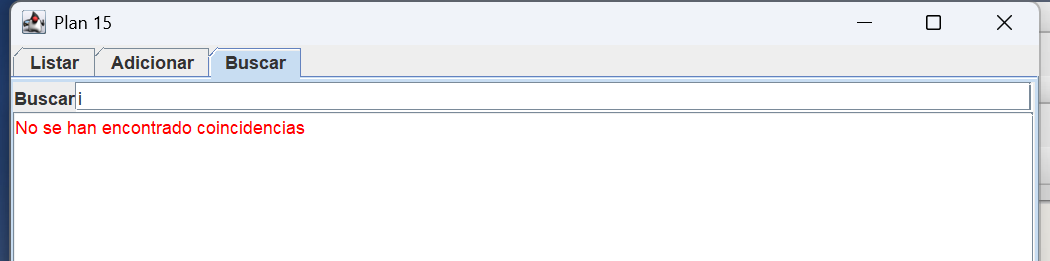
1. **Exploren el método récord() de la clase Log ¿Qué servicio presta?**

El método récord() de la clase Log tiene como función principal registrar excepciones en un archivo de log de manera estructurada y persistente: captura las excepciones que ocurren en la aplicación, las almacena en un archivo con extensión .log utilizando un formato legible (gracias a SimpleFormatter), y garantiza la conservación de logs anteriores mediante el modo append (parámetro true en FileHandler), lo que permite mantener un historial completo de errores para su posterior análisis o depuración.

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.**

Para manejar correctamente la situación en la que se consulta una unidad que no existe, es necesario identificar el punto del programa donde puede ocurrir ese fallo e incorporar un bloque try-catch alrededor de dicho código. Dentro del bloque catch, se debe mostrar un mensaje claro al usuario, y registrar el error utilizando el método **Log.record(e)**. Es importante destacar que, en este caso, no se debe finalizar la ejecución del programa; es decir, no se debe utilizar System.exit() dentro del catch, ya que se espera que el programa continúe funcionando normalmente tras manejar el error.

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

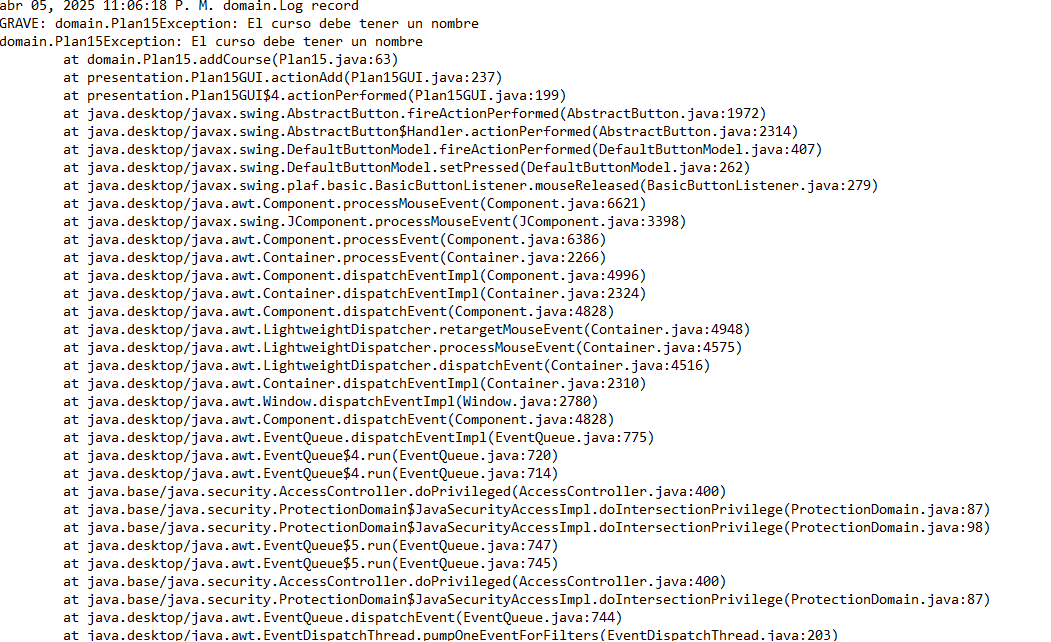
****

**¿La aplicación termina?**

No, la aplicación no termina.

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

-El archivo que contiene los errores se encuentra en la carpeta doc, ya sabiendo esto lo abrimos y contiene todos los errores que nos salian en consola, la imagen es la siguiente:



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

Si, si es adecuado que la aplicación continúe, debido a que el usuario se puede equivocar al escribir el nombre de una clase, haciendo que en el programa se genere un error porque esta no existe.

Para que continúe la aplicación depende de que se generen errores pensados, siendo el caso de que el usuario se equivoque o escriba el nombre de una clase que no existe. Por otro lado para que deba parar, esto depende de que los errores sean críticos como por ejemplo que no se pueda manejar.

1. **Modifiquen la aplicación para garantizar que SIEMPRE que haya un error se maneje de forma adecuada. ¿Cuál fue la solución implementada?**

La solución que se implementó fue agregarle el metodo Log.record(e) a todos los bloques try+catch, haciendo que los errores se guarden en el archivo .log, siendo esto la mejor forma para manejar a los errores

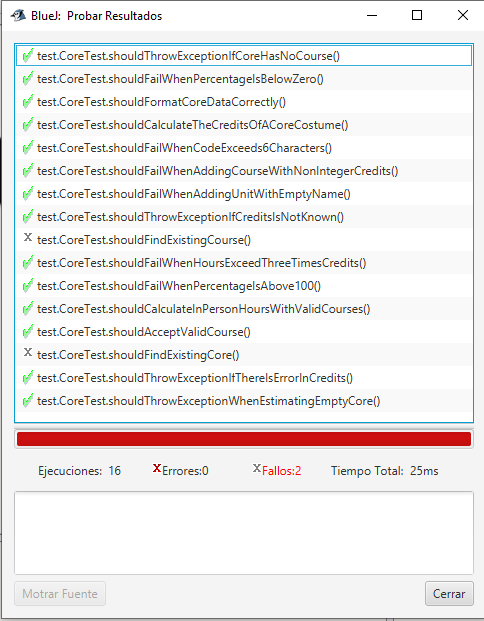
**CONSULTANDO POR PATRONES. ¡AHORA SI FUNCIONA!**

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

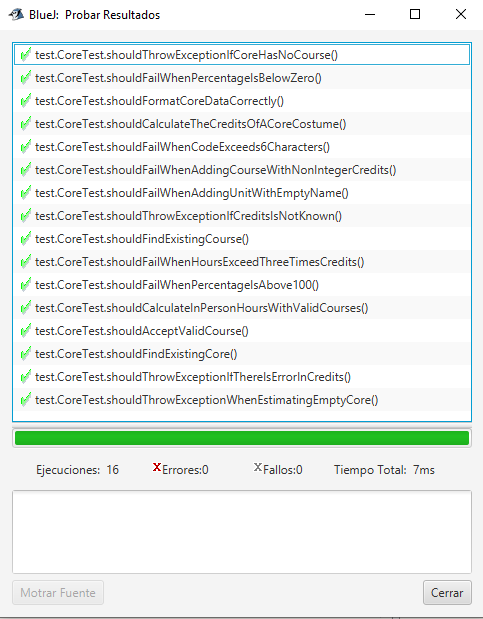
El método asociado a buscar en la capa de presentación (Plan15GUI) es **actionSearch()**, quien está encargado de leer, lo que el usuario ingresa, y así mismo, buscar en los datos de las clases o los cursos, e imprimir la información.

El método asociado a buscar en la capa de dominio es **search()**, en la clase Plan15, esta se encarga de pedir la información a la clase select(), quien devuelve la información necesaria para que Plan15 le pueda enviar la información a Plan15GUI y esta mostrarla

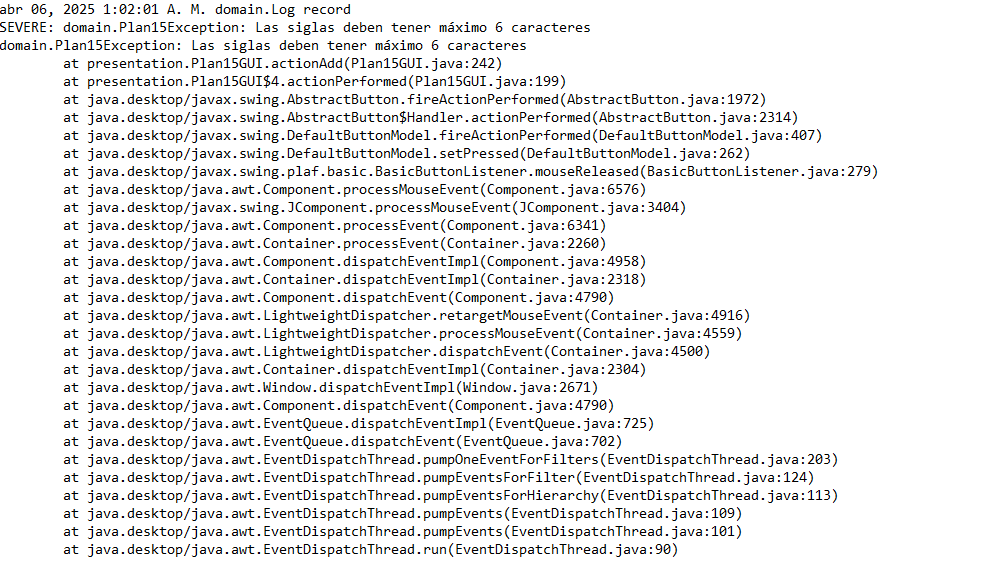
1. **Realicen ingeniería reversa para la capa de dominio para buscar. Capturen los resultados de las pruebas. Deben fallar.**

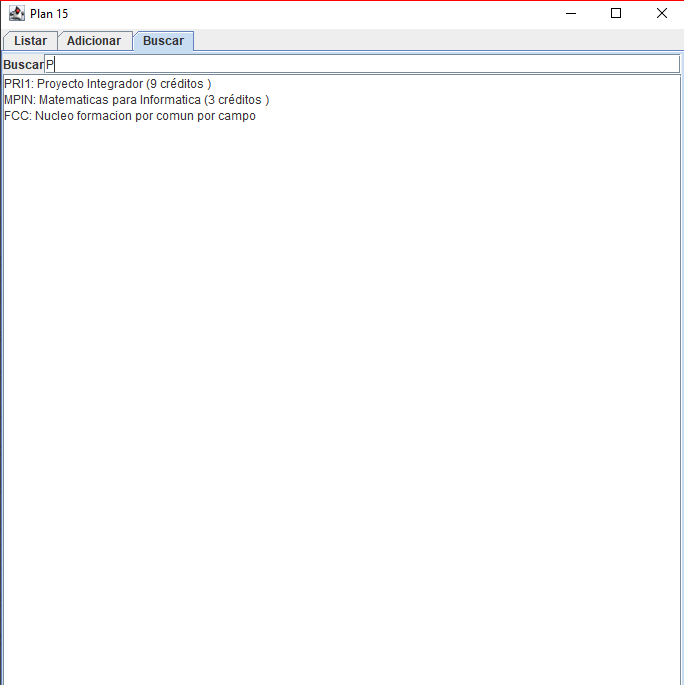
****

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

****

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

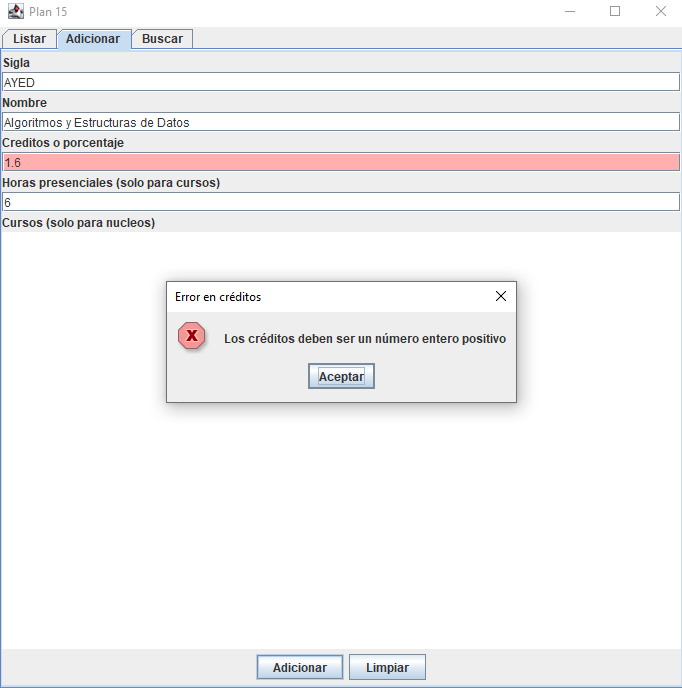
****

****

1. **Refactorice la funcionalidad para que sea más amable con el usuario. ¿Cuál es la propuesta?¿Cómo la implementa?**

Nuestra propuesta consta en mostrarle al usuario donde se generó su error con colores, obviamente también con un mensaje muy claro, donde el usuario pueda entender cuál fue su fallo, añadiendo a esto se generan los mensajes con color rojo.

Nuestra idea era que la interfaz fuera lo más amigable posible, para esto generamos símbolos cuando el usuario se equivoca.



**RETROSPECTIVA**

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

Christian Romero: 20 horas

Anderson Garcia: 20 horas

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

El laboratorio se encuentra completo y funcional. Se implementaron y validaron todos los métodos requeridos, incluyendo addCourse() y inPersonEstimated(), junto con sus respectivas pruebas unitarias. Además, se realizó el diseño del modelo con UML y se generó un diagrama de secuencia para representar el comportamiento de los métodos principales.

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

La práctica más útil fue el desarrollo guiado por pruebas (TDD - Test-Driven Development), ya que permitió validar la funcionalidad del sistema paso a paso. Implementar pruebas unitarias con assert ayudó a detectar errores rápidamente y asegurar que el código cumpliera con los requisitos establecidos.

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

El mayor logro fue la implementación y validación completa del método inPersonEstimated(), ya que implica lógica de negocio compleja con manejo de excepciones, estimaciones condicionales y validación de datos de múltiples objetos. También se acompañó de su correspondiente diagrama de secuencia en UML, facilitando su comprensión y documentación.

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

El mayor problema técnico fue transformar pruebas manuales en pruebas unitarias automáticas con assert y estructura JUnit, ya que inicialmente se usaban impresiones en consola. Para resolverlo, se refactorizó el código de pruebas usando JUnit 5, lo que permitió automatizar la validación de errores esperados y mejorar el mantenimiento del sistema.

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

Se hizo bien la organización del código, la cobertura de pruebas y el modelado UML. Para mejorar, nos comprometemos a integrar herramientas de control de versiones (como Git) desde el inicio del desarrollo, así como documentar mejor los cambios progresivos y aplicar integración continua en futuros laboratorios.

1. **¿Qué referencias usaron? ¿Cuál fue la más útil? Incluyan citas con estándares adecuados.**

Fowler, M. (2004). UML Distilled: A Brief Guide to the Standard Object Modeling Language. Addison-Wesley.

Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (1994). Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley.

Oracle. (2023). JUnit 5 User Guide. Recuperado de: <https://junit.org/junit5/docs/current/user-guide/>

Baeldung. (2023). Guide to JUnit 5. Recuperado de: https://www.baeldung.com/junit-5