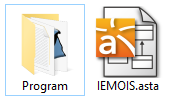
**Laboratorio 04. POOB-02**

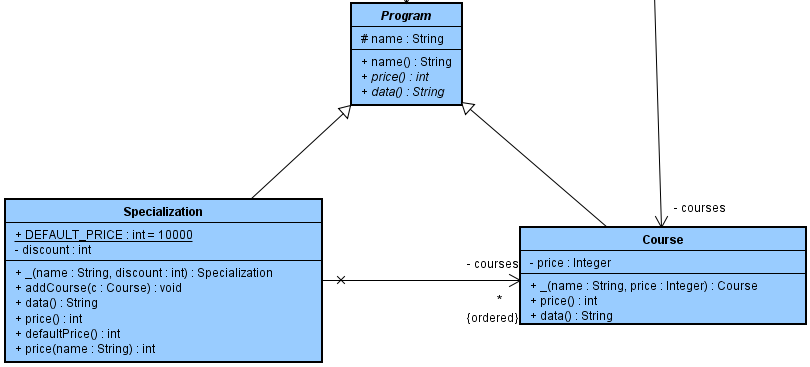
Jefer Alexis González Romero  
Angel Nicolas Cuervo Naranjo

**Combos**

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

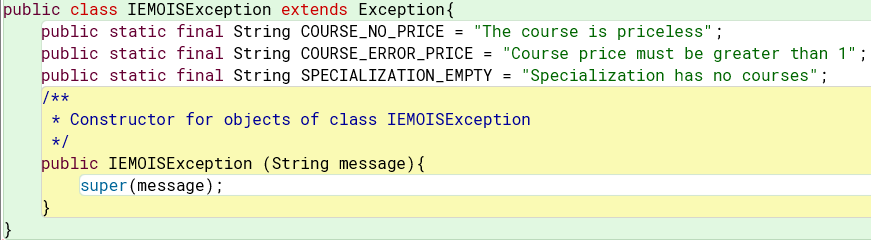
**1.** En su directorio descarguen los archivos contenidos en Program.zip revisen el contenido y estudien el diseño estructural de la aplicación (únicamente la zona en azul).

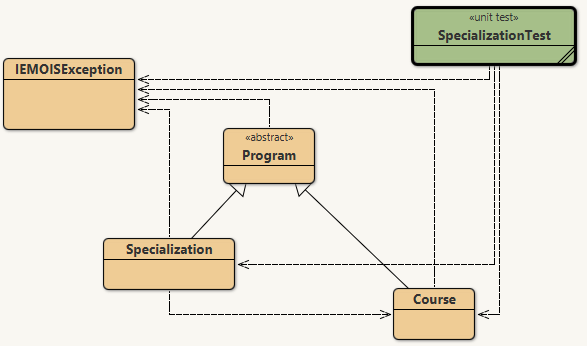




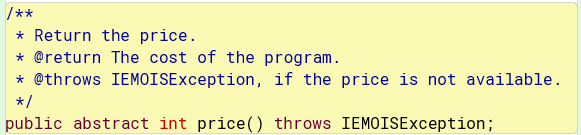
**2.** Expliquen por qué el proyecto no compila. Realicen las adiciones necesarias para lograrlo.

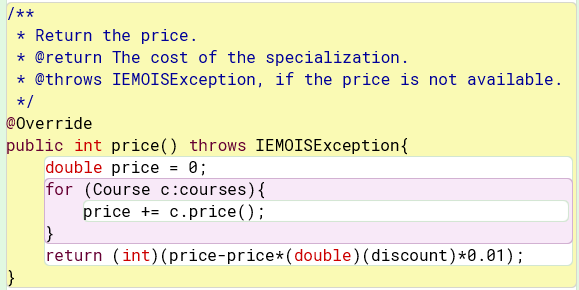
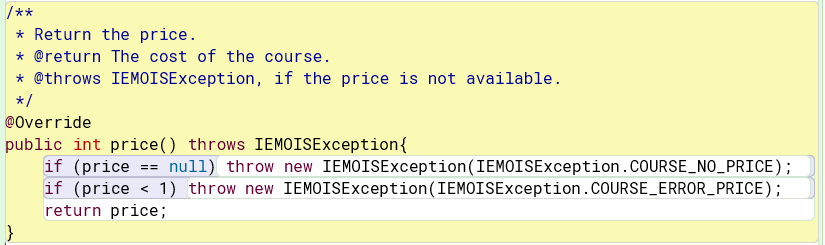
No copila porque aún no está creada la clase de excepciones *IEMOISException* que usan las otras clases. Se necesita adicionar la clase *IEMOISException* con los atributos finales y estáticos que se necesitan en courses y Specialization que son: *COURSE\_NO\_PRICE, COURSE\_ERROR\_PRICE y SPECIALIZATION\_EMPTY*. Además, la clase debe tener el constructor de la clase que debe recibir como parámetros el mensaje.

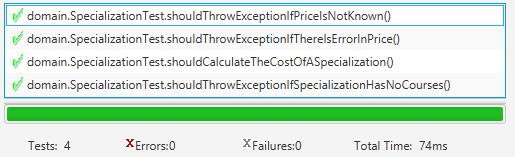




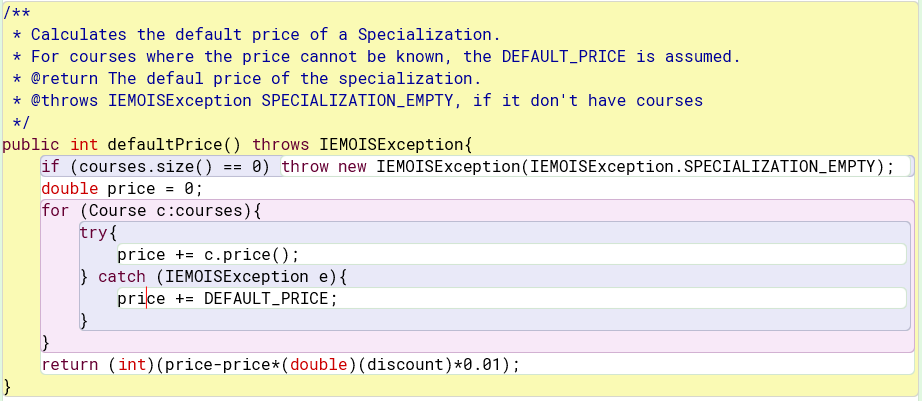
**3.** Dado el diseño y las pruebas, documenten y codifiquen el método price().

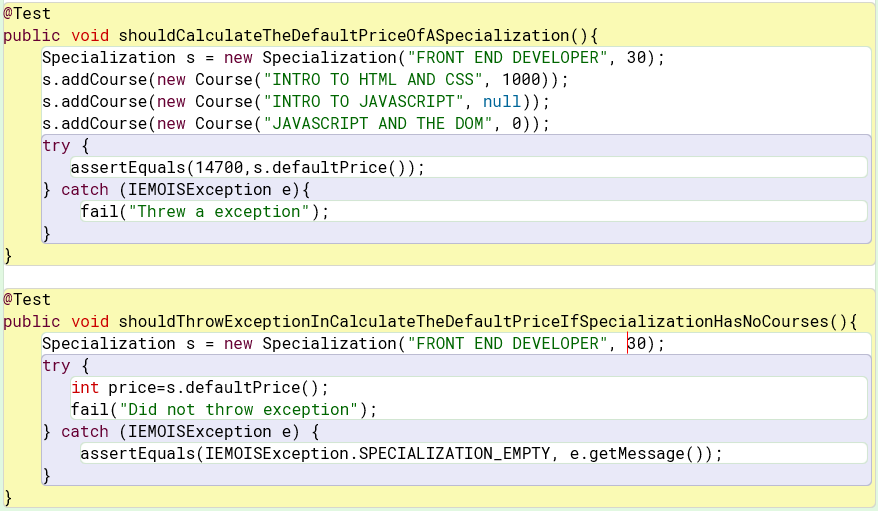


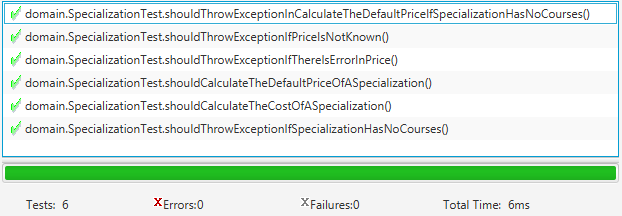




**4.** Dada la documentación y el diseño, codifiquen y prueben el método defaultPrice().



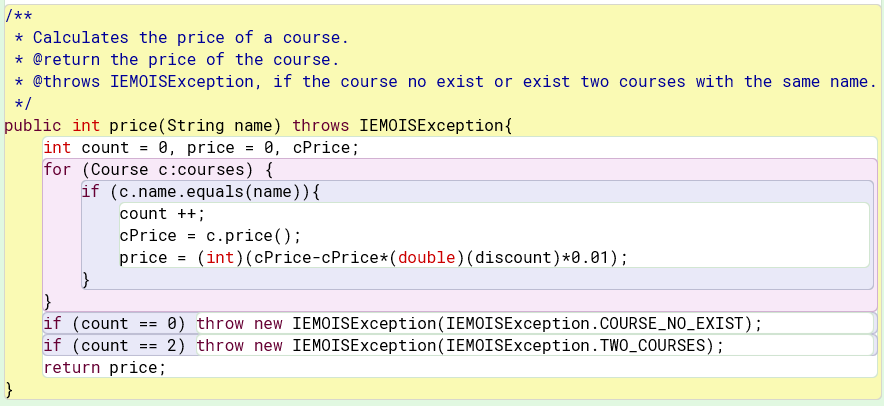


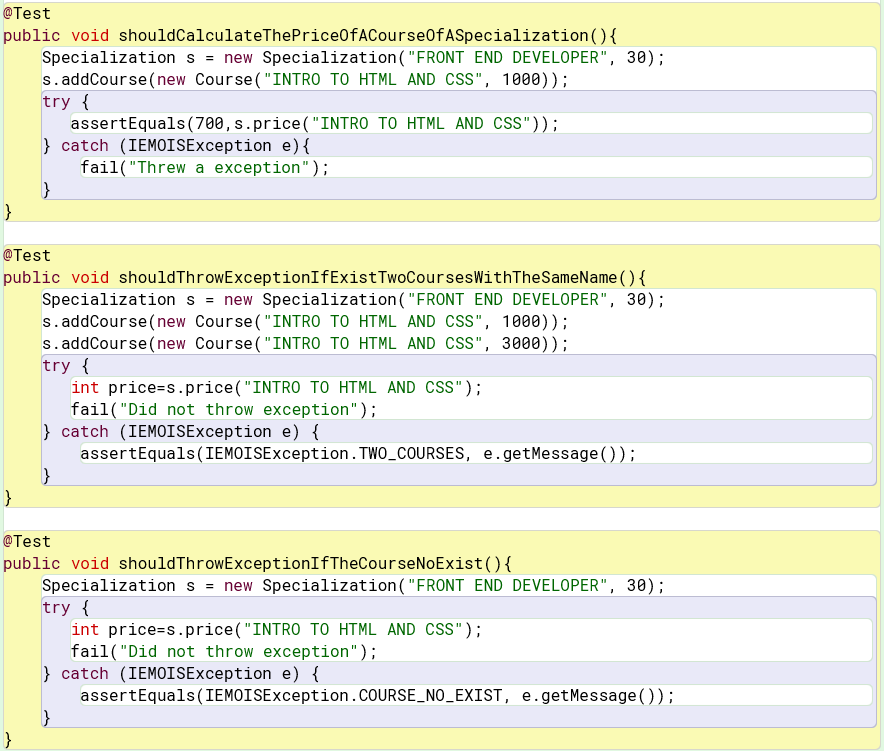


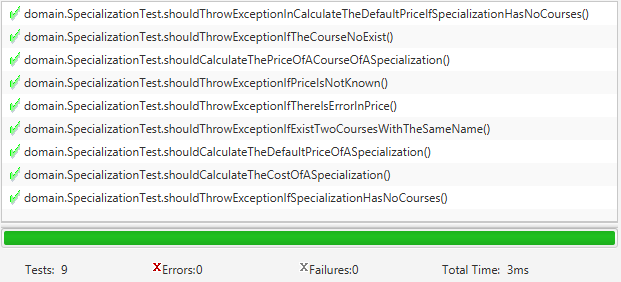
**5.** Documenten, diseñen, codifiquen y prueben el método price(String). Este método calcula el precio de un curso de la especialización considerando el descuento. En este método vamos a incluir dos casos especiales: 1) no existe un curso con ese nombre y 2) existen dos cursos con el mismo nombre

Diagrama

Descripción generada automáticamente





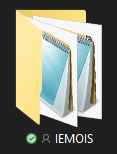


**IEMOIS**

El objetivo de esta aplicación es mantener un catálogo de los MOOC ofrecidos por la decanatura en el período intermedio a sus estudiantes en el proyecto IEMOIS. En este proyecto se ofrecen diferentes programas: cursos y especializaciones.

**Conociendo el proyecto IEMOIS**

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



Texto

Descripción generada automáticamente

Tiene un directorio llamado *IEMOIS* el cual contiene 3 archivos *.java* que no están organizados. Se debería organizar en 3 directorios: *src*donde debe estar todos los códigos fuente *.java, bin*con los *bytecodes* de los códigos fuentes *.class y docs d*ebe contener la documentación. Y Cada uno de ellos por paquetes, uno para la interfaz gráfica con la cual va a interactuar el usuario (*presentation*) y otra para la implementación de la aplicación (*domain*). Además, *src* debe contener otro directorio con las pruebas (*test*).

**2.** 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?

El programa está organizado en dos paquetes, uno es *domain* que contiene 4 clases: *Program, IEMOIS, Course* y *Specialization.* Y el otro es *presentation* que solo tiene una clase que es *IEMOISGUI.* La clase ejecutiva de la aplicación es *IEMOIS.*

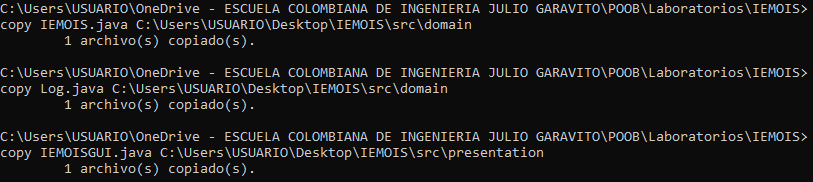
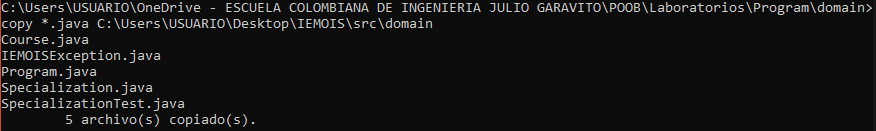
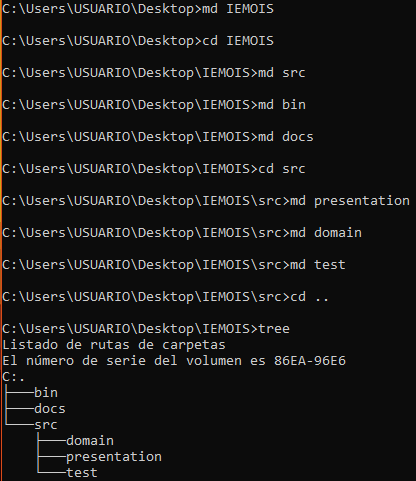
**3.** Prepare los directorios necesarios para ejecutar el proyecto. ¿qué estructura debe tener? ¿qué clases deben tener? ¿dónde están esas clases? ¿qué instrucciones debe dar para ejecutarlo?

La aplicación debe estar dividida en tres directorios *src, bin y docs.* Los cuales a su vez deben estar separados por los paquetes *domain y presentation.* Y *src* debe tener también otro paquete llamado *test.*

En el paquete *domain* deben estar las clases que implementan la aplicación: *Program, IEMOIS, Course* y *Specialization, IEMOIS y Log* (estas clases están en *domain* de *Program* y en la carpeta *IEMOIS).*

En el paquete *presentation* deben estar las clases relacionadas con la interfaz gráfica: *IEMOISGUI* (esta clase está en la carpeta *IEMOIS*).

En el paquete *test* las clases prueba de la aplicación que por ahora solo es: *SpecializationTest* (esta clase está en *domain* de *Program*).



Texto

Descripción generada automáticamenteTexto

Descripción generada automáticamenteTexto

Descripción generada automáticamente

Para copilarlo se usa el comando: *javac -d bin -cp junit-4.13.2.jar src\domain\\*.java src\presentation\\*.java src\test\\*.java*

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



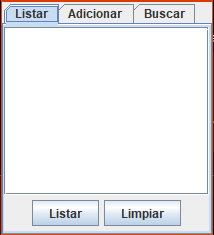
Para documentarlo se usa el comando*: javadoc -d docs src\domain\\*.java src\presentation\\*.java*

**

Para ejecutarlo se usa el comando*: java -cp bin presentation.IEMOISGUI*

**

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



Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamenteInterfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamenteInterfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Ofrece las funcionalidades: Listar programas que tiene dos botones Listar y Limpiar, Adicionar programas que también tiene dos botones Adicionar y Limpiar, y por último Buscar programas.

De los anteriores funcionan Listar y Adicionar con sus respectivos botones.

**5.** ¿De dónde salen los cursos iniciales? Revisen el código y la documentación del proyecto. ¿Qué clase pide que se adicionen? ¿Qué clase los adiciona?

Los cursos iniciales salen del método *addSome()* que está en la clase *IEMOIS.* La clase que piden que se adicionen es *IEMOIS* en el método constructor *IEMOIS()* y esta misma clase es la que los adiciona.

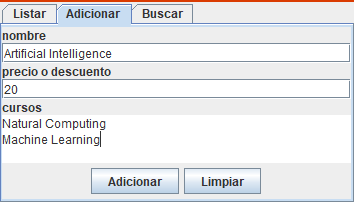
**Adicionar y listar. Todo OK.**

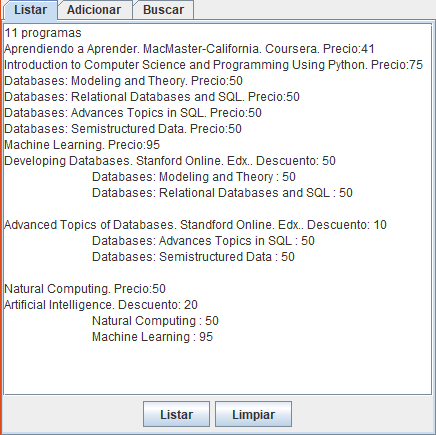
El objetivo es realizar ingeniería reversa a las funciones de adicionar y listar.

**1.** Adicionen un nuevo curso y una nueva especialización Cursol Natural Computing 50 Especialización Artificial Intelligence 20 Natural Computing Machine Learning ¿Qué ocurre? ¿Cómo lo comprueban? Capturen la pantalla. ¿Es adecuado este comportamiento?

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente



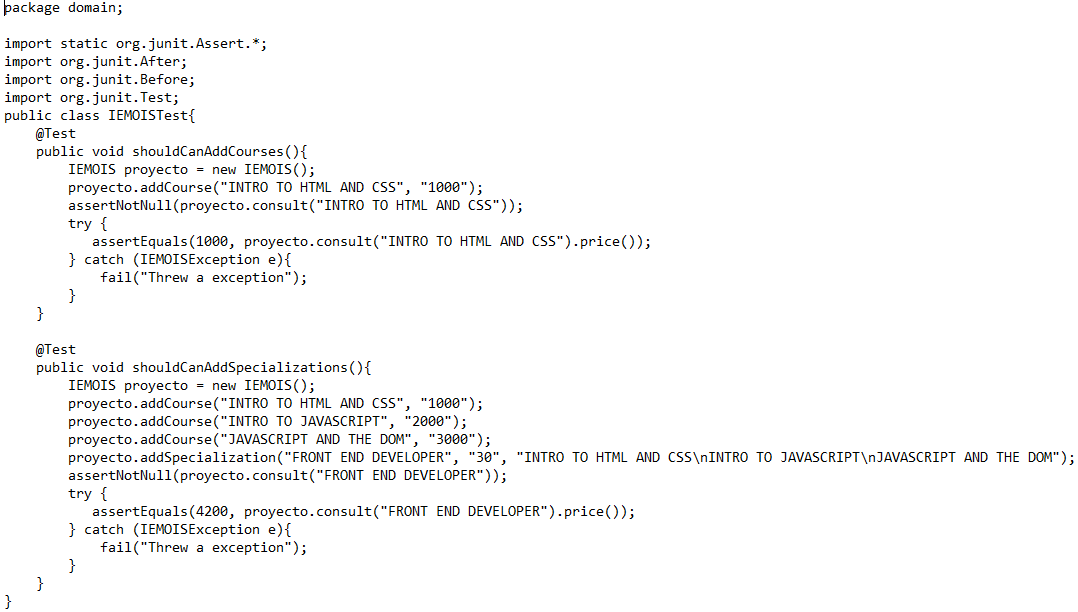


El comportamiento sí es adecuado, ya que se adiciona el curso *Natural Computing* con precio 20 y la especialización *Artificial Intelligence* con un descuento de 20%, que además contiene los cursos de *Natural Computing* y *Machine Learning* que previamente ya se habían adicionado con los precios de 50 y 95 respectivamente. Lo anterior se puede comprobar en *Listar*.

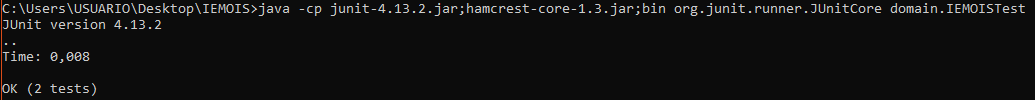
**2.** Revisen el código asociado a adicionar en la capa de presentación y la capa de dominio. ¿Qué método es responsable en la capa de presentación? ¿Qué método en la capa de dominio?

En la capa de presentación los métodos responsables son *actionAdd()* y *prepareAreaAdd()*. En la capa de dominio son los métodos *addSpecialization(String name, String discount, String courses)* y *addCourse(String name, String price).*

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



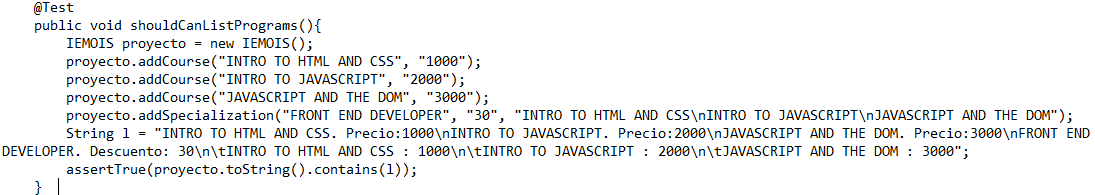
*java -cp junit-4.13.2.jar;hamcrest-core-1.3.jar;bin org.junit.runner.JUnitCore domain.IEMOISTest*

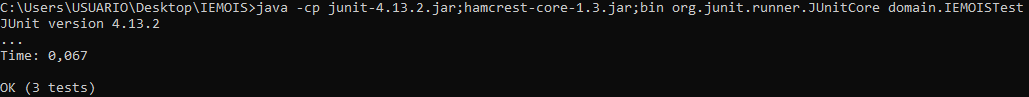


**4.** Revisen el código asociado a listar en la capa de presentación y la capa de dominio. ¿Qué método es responsable en la capa de presentación? ¿Qué método en la capa de dominio?

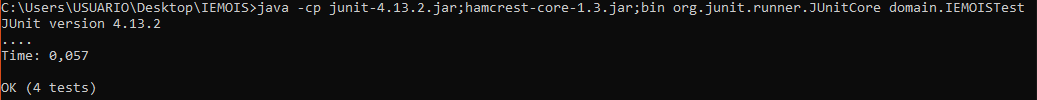
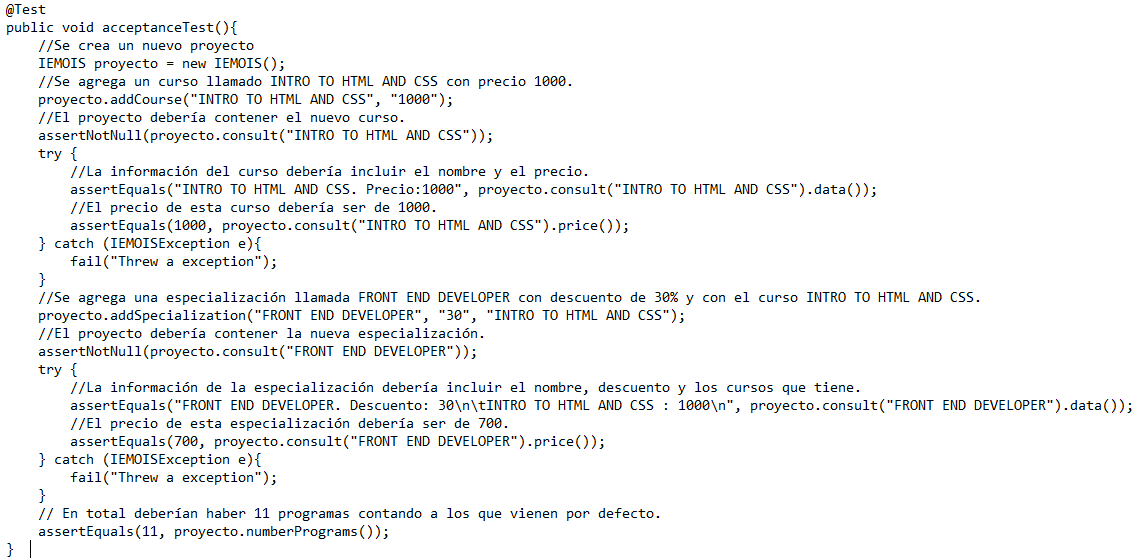
En la capa de presentación los métodos responsables son *actionList()* y *prepareAreaList().* En la capa de dominio es el método *toString().*

**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.



**Adicionar un programa: curso o especialización. Funcionalidad robusto**

El objetivo es perfeccionar la funcionalidad de adicionar un curso para hacerla más robusta. Para cada uno de los siguientes casos realice los pasos del 1 al 4.

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

**2.** Analicen el diseño realizado. Para hacer el software robusto: ¿Qué método debería lanzar la excepción? ¿Qué métodos deberían propagarla? ¿Qué método debería atenderla? Explique claramente.

**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? Capture la pantalla.

**a.** ¿Y si el nombre del curso ya existe?

**1.**

Se adiciona un curso llamado “INTRO TO HTML AND CSS” con precio de 1000 y posteriormente se adiciona otro curso con el mismo nombre “INTRO TO HTML AND CSS” con precio de 2000, al hacer esta última acción se debería presentar un mensaje indicando que no se puede adicionar cursos con nombres que ya existan y no se debería modificar el curso que estaba antes.

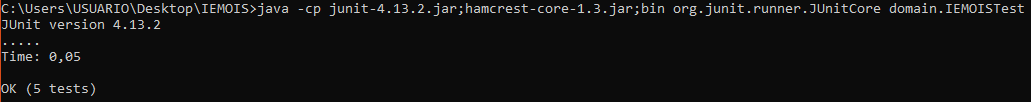
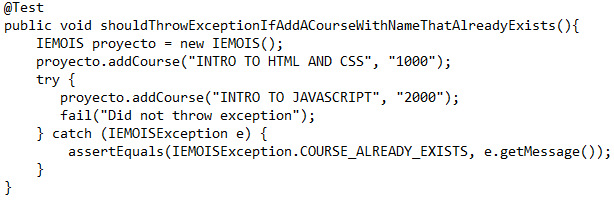
**2.**

**Lanzar:** *addCourse()* de la clase IEMOIS, ya que verifica que los cursos que hay en el proyecto no esté el nombre del curso que se quiere agregar y si pasa se lanza la excepción.

**Propagar:** *addCourse()* de la clase IEMOIS, debido a que se si se lanza una excepción debe ser propagada para que se atienda por el método correspondiente.

**Atender:** *actionAdd()* de la clase IEMOISGUI, porque es la encargada de interactuar con el usuario y si llega agregar un curso con nombre que ya exista le debe aparecer el mensaje.

**3.**

****

**4.**

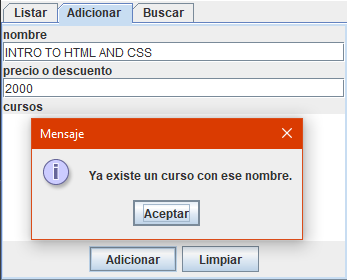
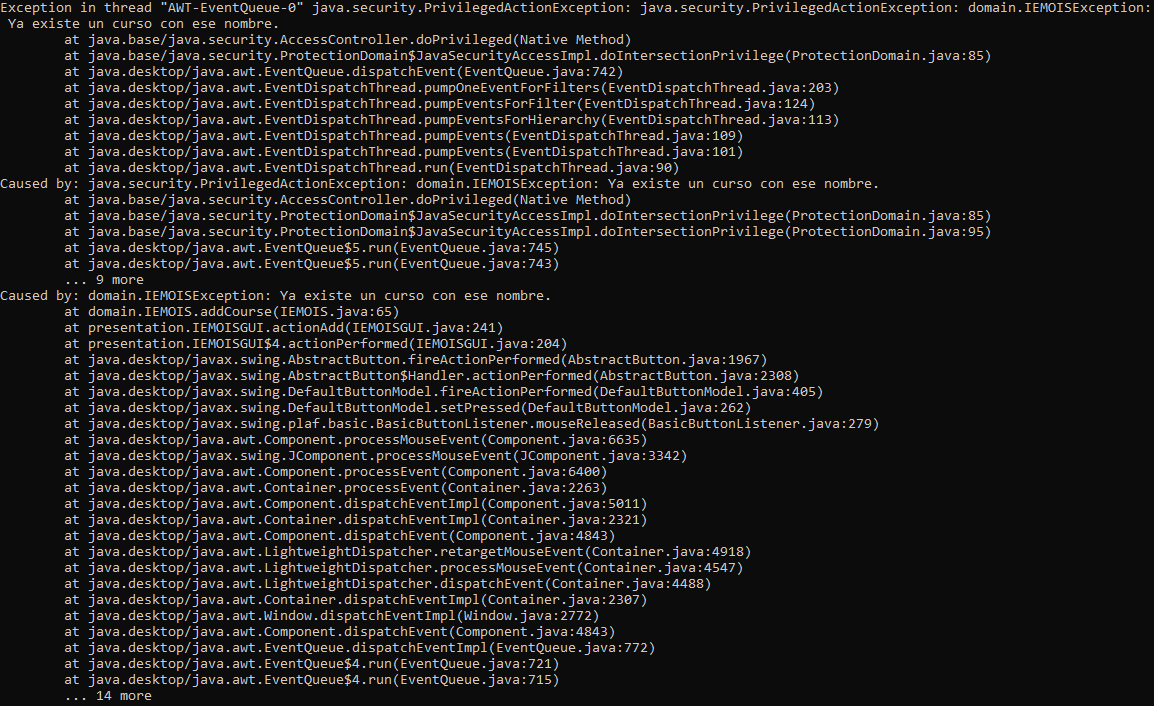
Se adiciona un curso llamado “INTRO TO HTML AND CSS” con precio de 1000.

**Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamenteInterfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente**

Se adiciona otro curso con el mismo nombre “INTRO TO HTML AND CSS” con precio de 2000, al hacer esta última acción se presenta un mensaje indicando ya existe un curso con ese nombre.

No se modifica el curso que estaba antes.

**Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente**

**b.** ¿Y si en precio no da un número? ¿o da un número negativo?

**1.**

Se adiciona un curso llamado “INTRO TO HTML AND CSS” con un precio desconocido, al hacer esto se debería presentar un mensaje indicando que el precio debe ser un número positivo.

Se adiciona un curso llamado "JAVASCRIPT AND THE DOM" con precio “mil”, al hacer esto se debería presentar un mensaje indicando que el precio debe ser un número positivo.

Se agrega otro curso llamado "INTRO TO JAVASCRIPT" con precio -1000, al hacer esto se debería presentar un mensaje indicando que el precio debe ser un número positivo.

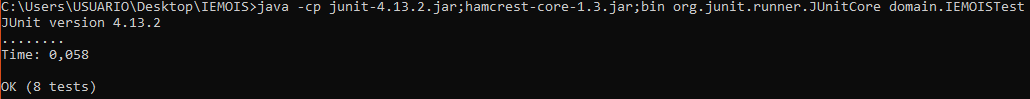
**2.**

**Lanzar:** *addCourse()* de la clase IEMOIS, ya que verifica que el precio dado sea válido para poder así mandar como parámetros el nombre y el precio que debe ser un entero positivo para crear el curso y no genere errores.

**Propagar:** *addCourse()* de la clase IEMOIS, debido a que se si se lanza una excepción debe ser propagada para que se atienda por el método correspondiente.

**Atender:** *actionAdd()* de la clase IEMOISGUI, porque es la encargada de interactuar con el usuario y si llega agregar un curso con precio que no sea válido le debe aparecer el mensaje.

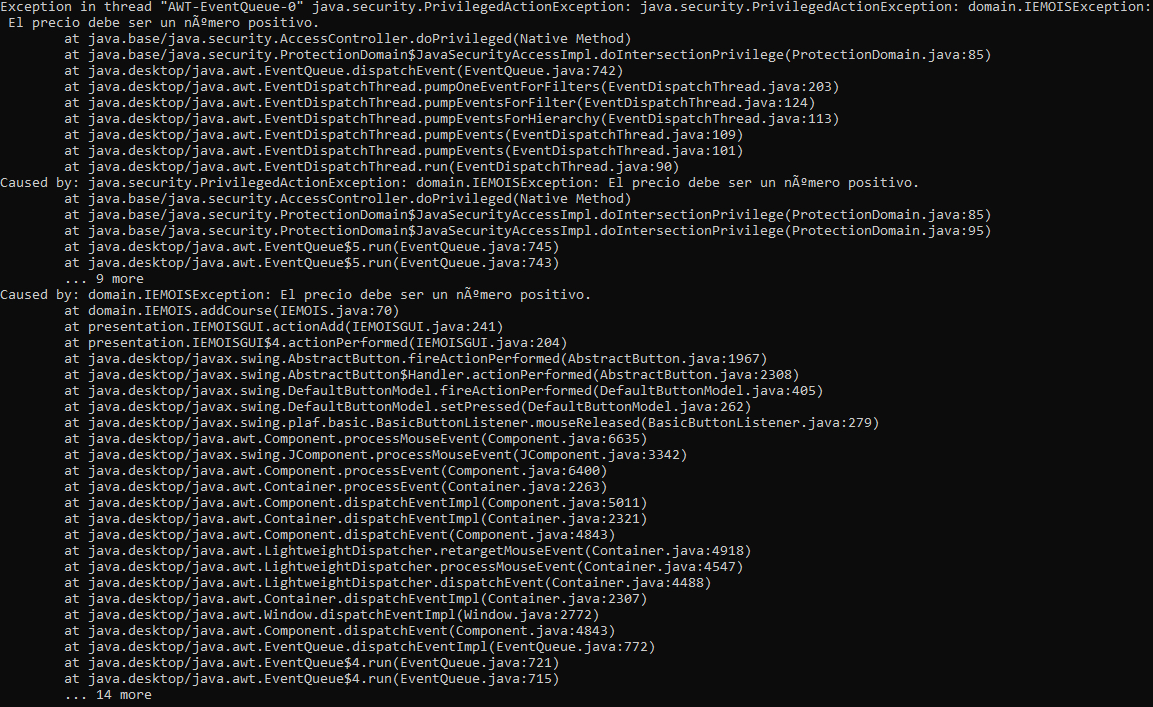
**3.**

****

**4.**

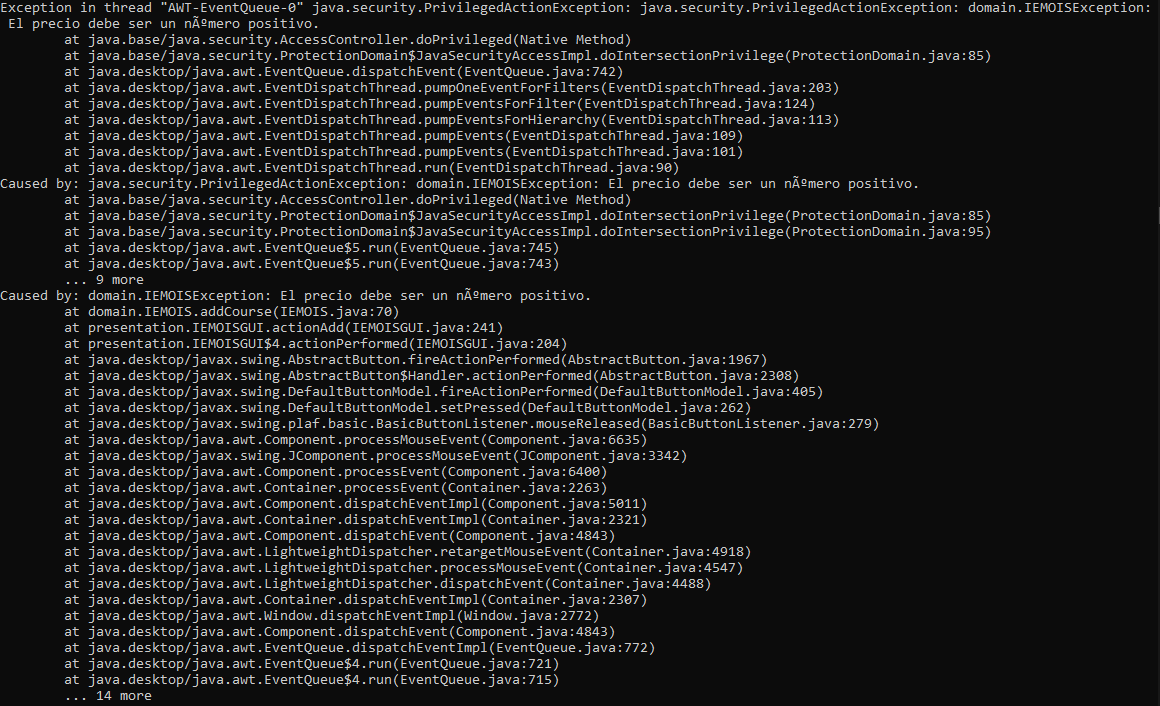
Se adiciona un curso llamado “INTRO TO HTML AND CSS” con un precio desconocido, al hacer esto se presenta un mensaje indicando que el precio debe ser un número positivo.

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente 

Se adiciona un curso llamado "JAVASCRIPT AND THE DOM" con precio “mil”, al hacer esto se presenta un mensaje indicando que el precio debe ser un número positivo.

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Se agrega otro curso llamado "INTRO TO JAVASCRIPT" con precio -1000, al hacer esto se presenta un mensaje indicando que el precio del curso debe ser un número positivo.

**Interfaz de usuario gráfica, Aplicación, Correo electrónico

Descripción generada automáticamente** **Texto

Descripción generada automáticamente**

**c.** Proponga una nueva condición

¿Y si no da el nombre del curso?

**1.**

Se adiciona un curso sin nombre con un precio 1000, al hacer esto se debería presentar un mensaje indicando que el curso debe tener nombre.

**2.**

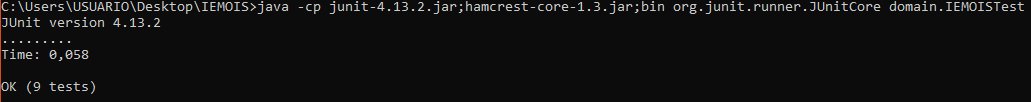
**Lanzar:** *addCourse()* de la clase IEMOIS, ya que verifica que se de el nombre, para poder así mandar como parámetros el nombre y el precio para crear el curso.

**Propagar:** *addCourse()* de la clase IEMOIS, debido a que se si se lanza una excepción debe ser propagada para que se atienda por el método correspondiente.

**Atender:** *actionAdd()* de la clase IEMOISGUI, porque es la encargada de interactuar con el usuario y si llega agregar un curso sin nombre le debe aparecer el mensaje.

**3.**

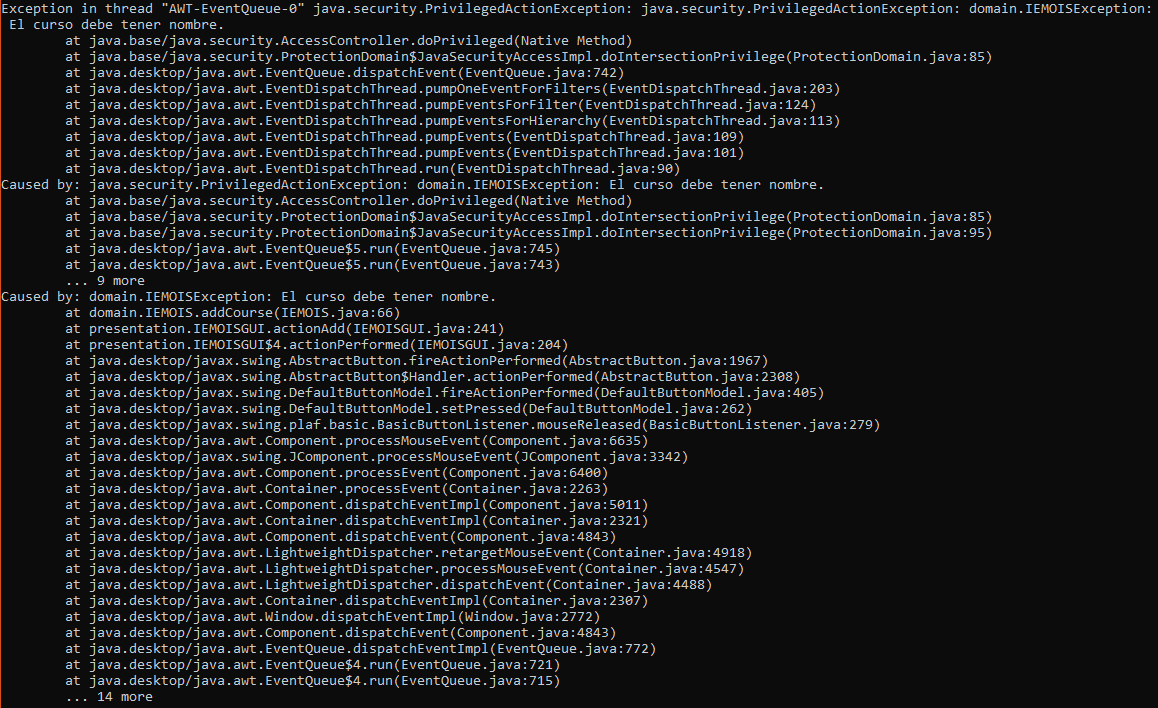
**Texto, Carta

Descripción generada automáticamente**

**4.**

Se adiciona un curso sin nombre con un precio 1000, al hacer esto se presenta un mensaje indicando que el curso debe tener nombre.

**Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente**

El objetivo es perfeccionar la funcionalidad de adicionar una especialización para hacerla más robusta. Para cada uno de los siguientes casos realice los pasos del 1 al 4.

**a**. ¿Y si el nombre de la especialización ya existe?

**1.**

Se adicionan los cursos “INTRO TO JAVASCRIPT", "JAVASCRIPT AND THE DOM" y "INTRO TO HTML AND CSS” con precios 2000, 3000 y 1000 respectivamente, posteriormente se agrega una especialización llamada "FRONT END DEVELOPER" con descuento de 30% y cursos "INTRO TO HTML AND CSS” y “INTRO TO JAVASCRIPT". Ahora se adiciona otra especialización con el mismo nombre ("FRONT END DEVELOPER") con descuento de 40% y con el curso "JAVASCRIPT AND THE DOM", al hacer esta última acción se debería presentar un mensaje indicando que no se puede adicionar especializaciones con nombres que ya existan y no se debería modificar la especialización que estaba antes.

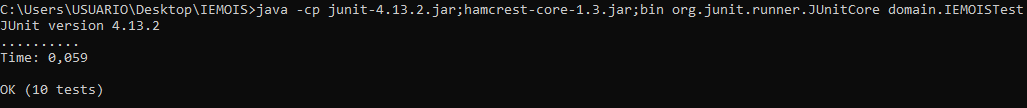
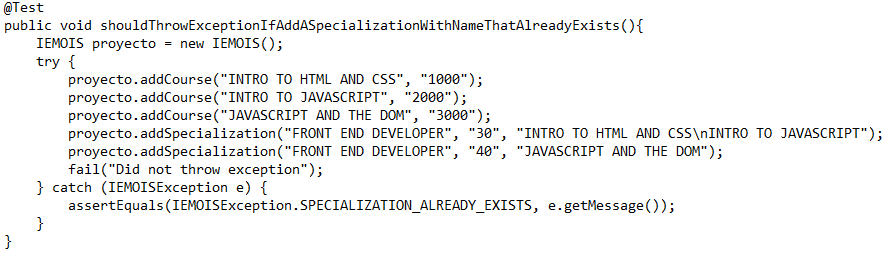
**2.**

**Lanzar:** *addSpecialization()* de la clase IEMOIS, ya que verifica que as especializaciones que hay en el proyecto no esté el nombre de la especialización que se quiere agregar y si pasa se lanza la excepción.

**Propagar:** *addSpecialization()* de la clase IEMOIS, debido a que se si se lanza una excepción debe ser propagada para que se atienda por el método correspondiente.

**Atender:** *actionAdd()* de la clase IEMOISGUI, porque es la encargada de interactuar con el usuario y si llega agregar una especialización con nombre que ya exista le debe aparecer el mensaje.

**3.**

****

**4.**

Se adicionan los cursos “INTRO TO JAVASCRIPT", "JAVASCRIPT AND THE DOM" y "INTRO TO HTML AND CSS” con precios 2000, 3000 y 1000 respectivamente.

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamenteInterfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamenteInterfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamenteInterfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

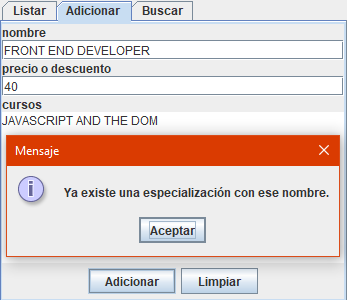
Se agrega una especialización llamada "FRONT END DEVELOPER" con descuento de 30% y cursos "INTRO TO HTML AND CSS” y “INTRO TO JAVASCRIPT".

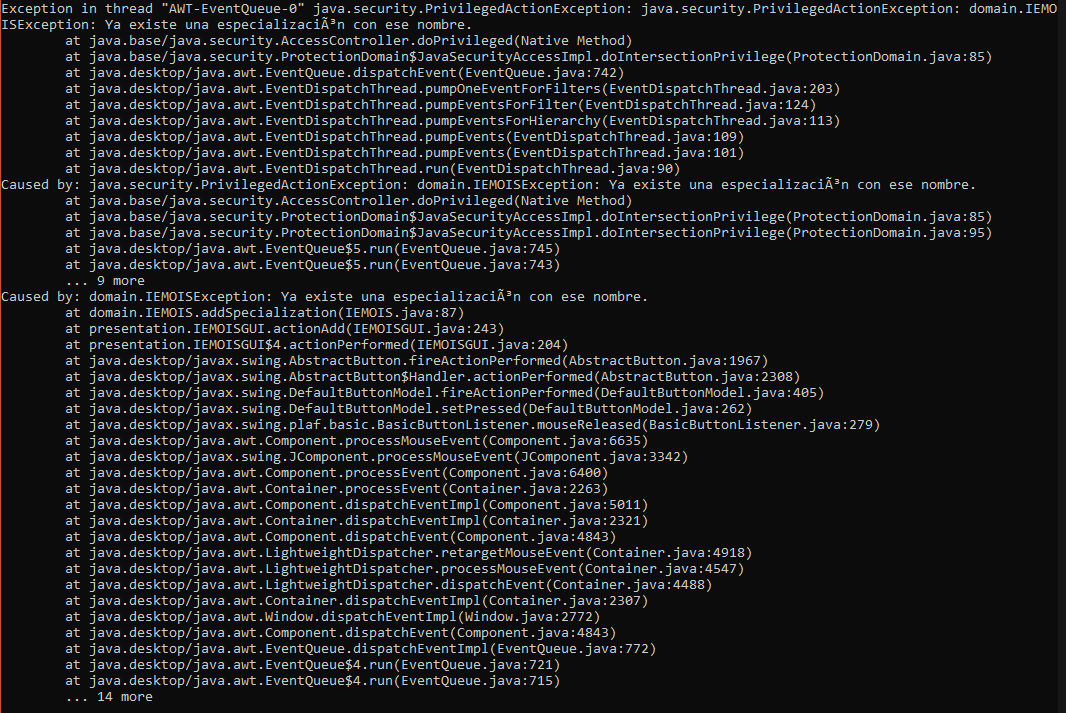
Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente Texto

Descripción generada automáticamente

Ahora se adiciona otra especialización con el mismo nombre ("FRONT END DEVELOPER") con descuento de 40% y con el curso "JAVASCRIPT AND THE DOM", al hacer esta última acción se presenta un mensaje indicando que no se puede adicionar cursos con nombres que ya existan y no se modifica la especialización que estaba antes.

**** **Texto

Descripción generada automáticamente** 

**b.** ¿Y si el descuento no da un número? ¿o da un número negativo?

**1.**

Se adiciona un curso llamado “INTRO TO HTML AND CSS” con precio 1000, posteriormente se agrega una especialización llamada "FRONT END DEVELOPER" con descuento desconocido y curso "INTRO TO HTML AND CSS” , al hacer esto se debería presentar un mensaje indicando que el descuento debe ser un número positivo.

Se agrega una especialización llamada "FRONT END DEVELOPER" con descuento “treinta” y curso "INTRO TO HTML AND CSS”, al hacer esto se debería presentar un mensaje indicando que el descuento debe ser un número positivo.

Se adiciona una especialización llamada "FRONT END DEVELOPER" con descuento -30 y curso "INTRO TO HTML AND CSS”, al hacer esto se debería presentar un mensaje indicando que el descuento debe ser un número positivo.

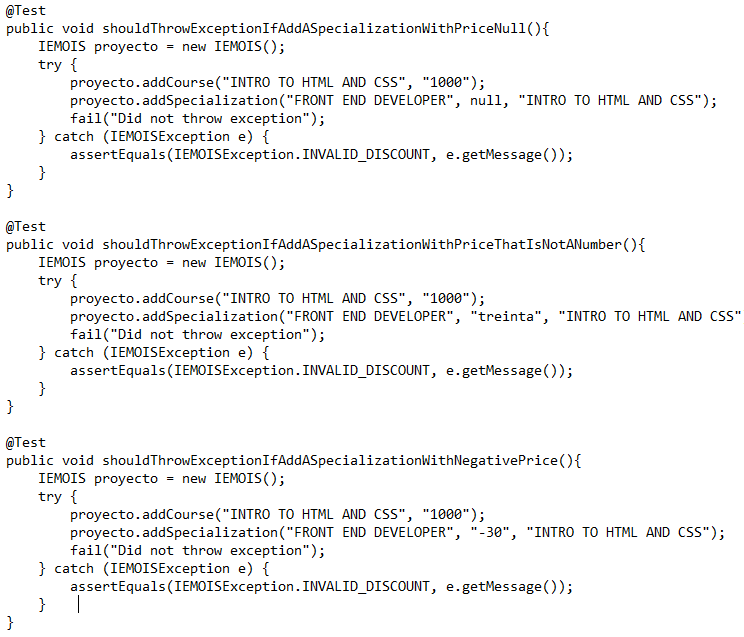
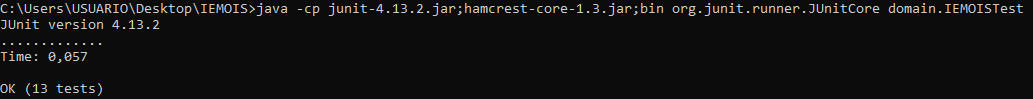
**2.**

**Lanzar:** *addSpecialization()* de la clase IEMOIS, ya que verifica que el descuento dado sea válido para poder así mandar como parámetros el nombre, el descuento que debe ser un entero positivo y los cursos para crear la especialización y no genere errores.

**Propagar:** *addSpecialization()* de la clase IEMOIS, debido a que se si se lanza una excepción debe ser propagada para que se atienda por el método correspondiente.

**Atender:** *actionAdd()* de la clase IEMOISGUI, porque es la encargada de interactuar con el usuario y si llega agregar una especialización con descuento que no sea válido le debe aparecer el mensaje.

**3.**

**** ****

**4.**

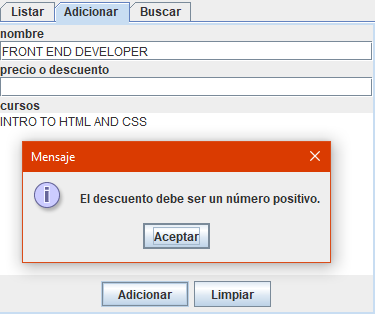
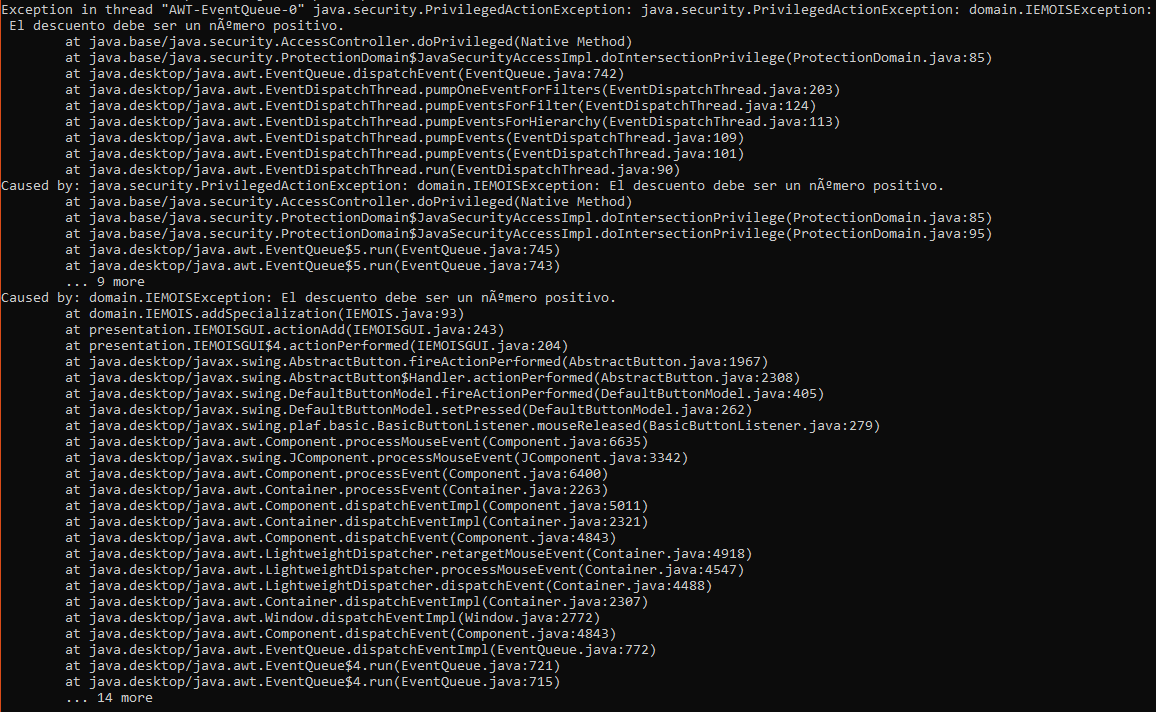
Se adiciona un curso llamado “INTRO TO HTML AND CSS” con precio 1000.

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

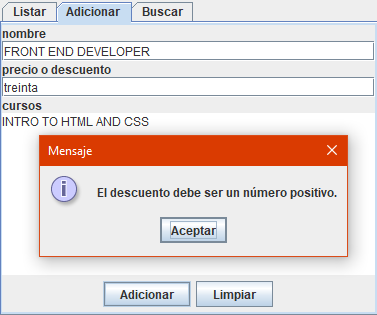
Descripción generada automáticamenteInterfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Se agrega una especialización llamada "FRONT END DEVELOPER" con descuento desconocido y curso "INTRO TO HTML AND CSS” , al hacer se presenta un mensaje indicando que el descuento debe ser un número positivo.

Se agrega una especialización llamada "FRONT END DEVELOPER" con descuento “treinta” y curso "INTRO TO HTML AND CSS”, al hacer esto se presenta un mensaje indicando que el descuento debe ser un número positivo.

 Texto

Descripción generada automáticamente

Se adiciona una especialización llamada "FRONT END DEVELOPER" con descuento -30 y curso "INTRO TO HTML AND CSS”, al hacer esto se presenta un mensaje indicando que el descuento debe ser un número positivo.

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente Texto

Descripción generada automáticamente

**c.** ¿Y si alguno de los cursos que la componen no existen?

**1.**

Se agrega una especialización llamada "FRONT END DEVELOPER" con descuento 30% y curso "INTRO TO HTML AND CSS”, al hacer esto se debería presentar un mensaje indicando algún curso de la especialización no existe.

**2.**

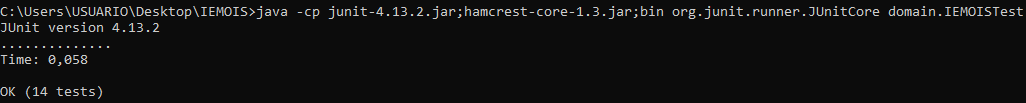
**Lanzar:** *addSpecialization()* de la clase IEMOIS, ya que verifica que los cursos que la componen estén para poder así mandar como parámetros el nombre, el precio y los cursos para crear la especialización.

**Propagar:** *addSpecialization()* de la clase IEMOIS, debido a que se si se lanza una excepción debe ser propagada para que se atienda por el método correspondiente.

**Atender:** *actionAdd()* de la clase IEMOISGUI, porque es la encargada de interactuar con el usuario y si llega agregar una especialización con algún curso que no esté le debe aparecer el mensaje.

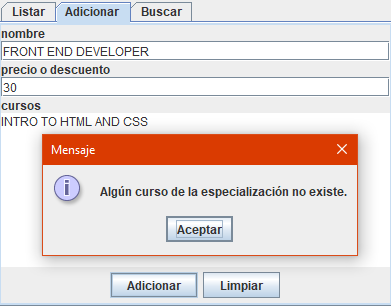
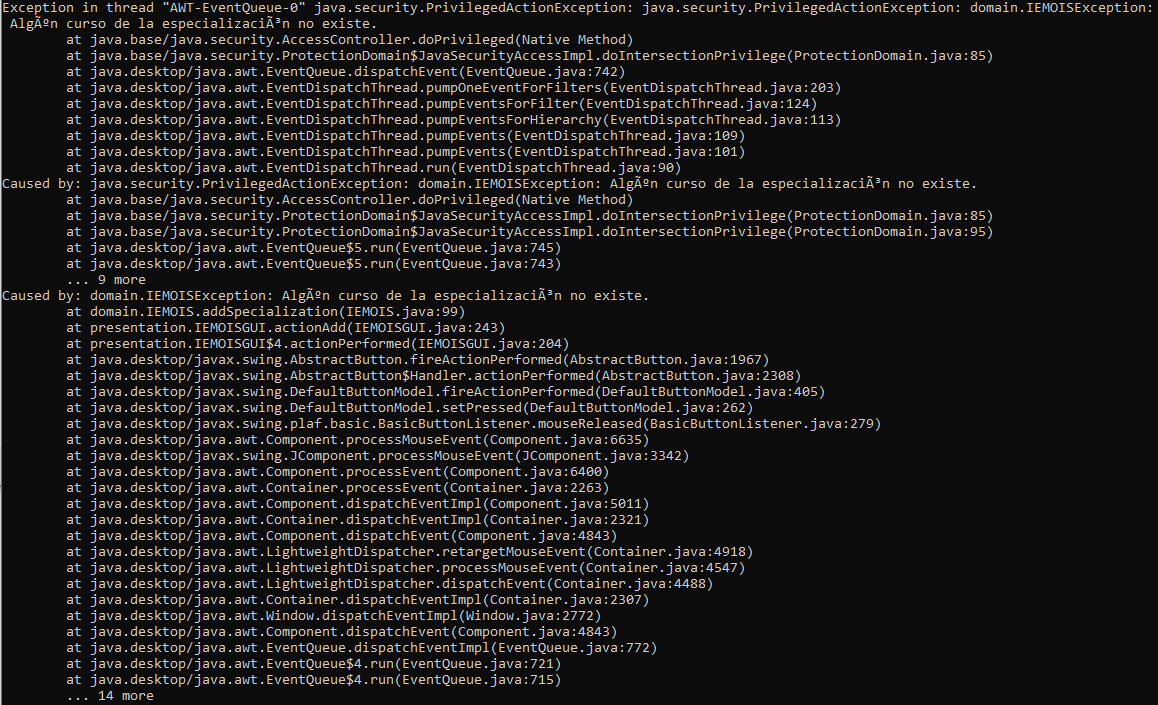
**3.**

**Un conjunto de letras blancas en un fondo blanco

Descripción generada automáticamente con confianza baja**

**4.**

Se agrega una especialización llamada "FRONT END DEVELOPER" con descuento 30% y curso "INTRO TO HTML AND CSS”, al hacer esto se presentar un mensaje indicando algún curso de la especialización no existe.

**** ****

**d.** Proponga una nueva condición

¿Y si el descuento es mayor a 100%?

**1.**

Se adiciona un curso llamado “INTRO TO HTML AND CSS” con precio 1000, posteriormente se agrega una especialización llamada "FRONT END DEVELOPER" con descuento de 101% y curso "INTRO TO HTML AND CSS”, al hacer esto se debería presentar un mensaje indicando que el descuento debe ser menor o igual a 100%.

**2.**

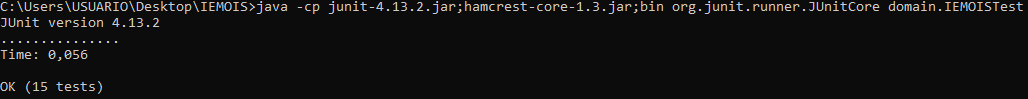
**Lanzar:** *addSpecialization()* de la clase IEMOIS, ya que verifica que el descuento dado sea válido para poder así mandar como parámetros el nombre, el descuento que debe ser menor a 100% y los cursos para crear la especialización.

**Propagar:** *addSpecialization()* de la clase IEMOIS, debido a que se si se lanza una excepción debe ser propagada para que se atienda por el método correspondiente.

**Atender:** *actionAdd()* de la clase IEMOISGUI, porque es la encargada de interactuar con el usuario y si llega agregar una especialización con descuento que no sea válido le debe aparecer el mensaje.

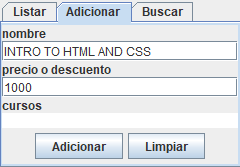
**3.**

**Texto

Descripción generada automáticamente**

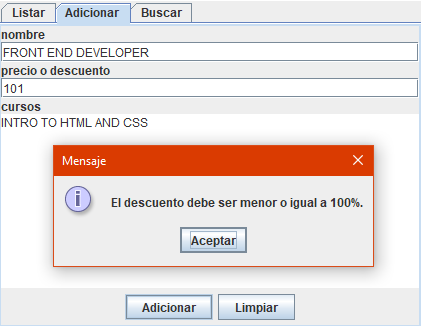
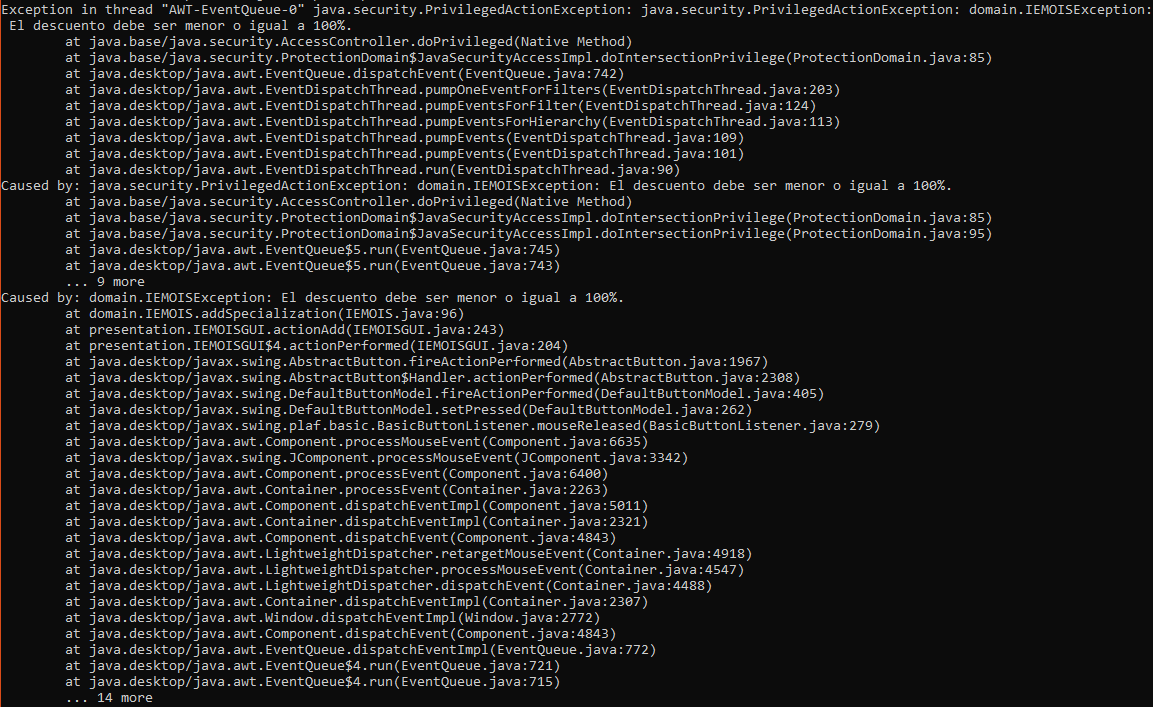
**4.**

Se adiciona un curso llamado “INTRO TO HTML AND CSS” con precio 1000.

**** **Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente**

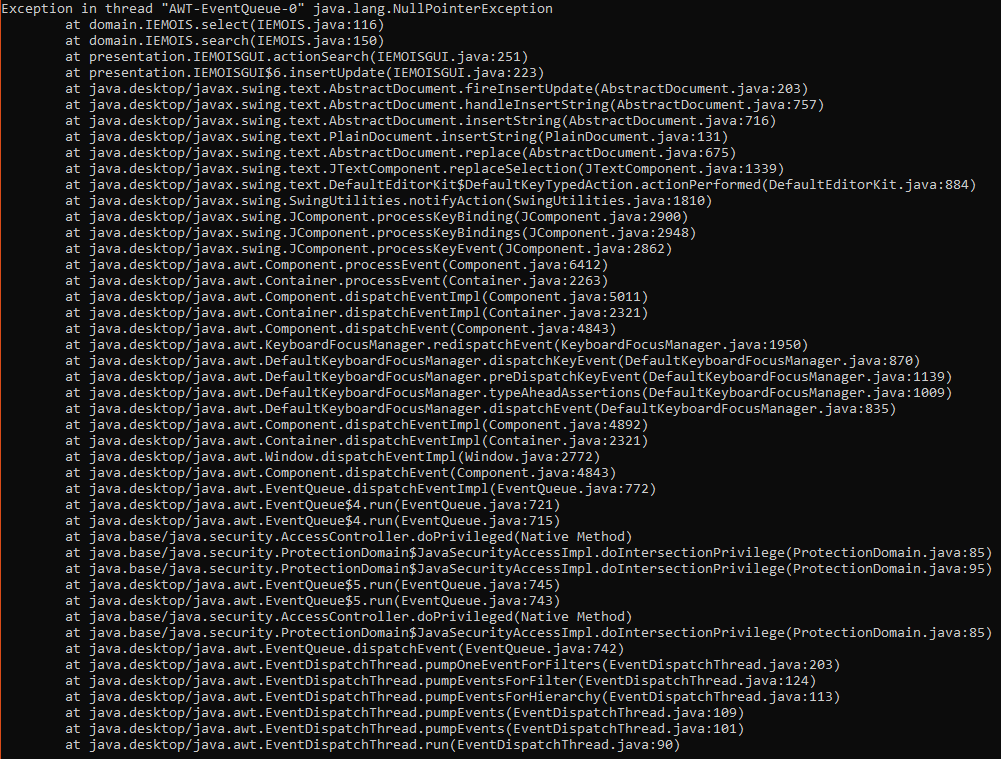
Se agrega una especialización llamada "FRONT END DEVELOPER" con descuento de 101% y curso "INTRO TO HTML AND CSS”, al hacer esto se presenta un mensaje indicando que el descuento debe ser menor o igual a 100%.

** **

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

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

Interfaz de usuario gráfica, Aplicación, Word

Descripción generada automáticamente 

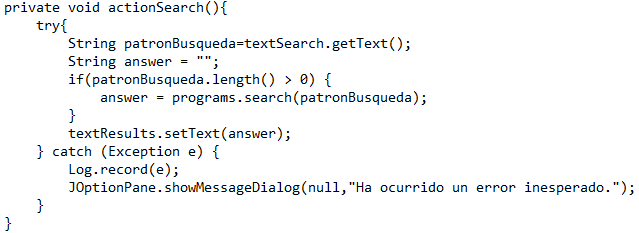
Al intentar consultar "INTRO TO” aparece un a excepción *java.lang.NullPointerException*, creemos que se intentó usar una referencia a un objeto que tenía un valor nulo. Esta información la debe conocer el programador, los usuarios no necesita tener conocimiento de esta.

**2.** Exploren el método record de la clase Log ¿Qué servicio presta?

El método *record* se encarga de tomar las excepciones obtenidas en un *stack* y registrarlas en un archivo plano llamado *IEMOIS.log* con el fin de que el desarrollador conozca que puede estar saliendo mal con la aplicación.

**3.** 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.

Se tiene que atender los errores en el método *actionSearch()* de la clase *IEMOISGUI*, de modo que cuando ocurra cualquier excepción se registrará y se le mostrará un mensaje al usuario, sin dejar de ejecutar.

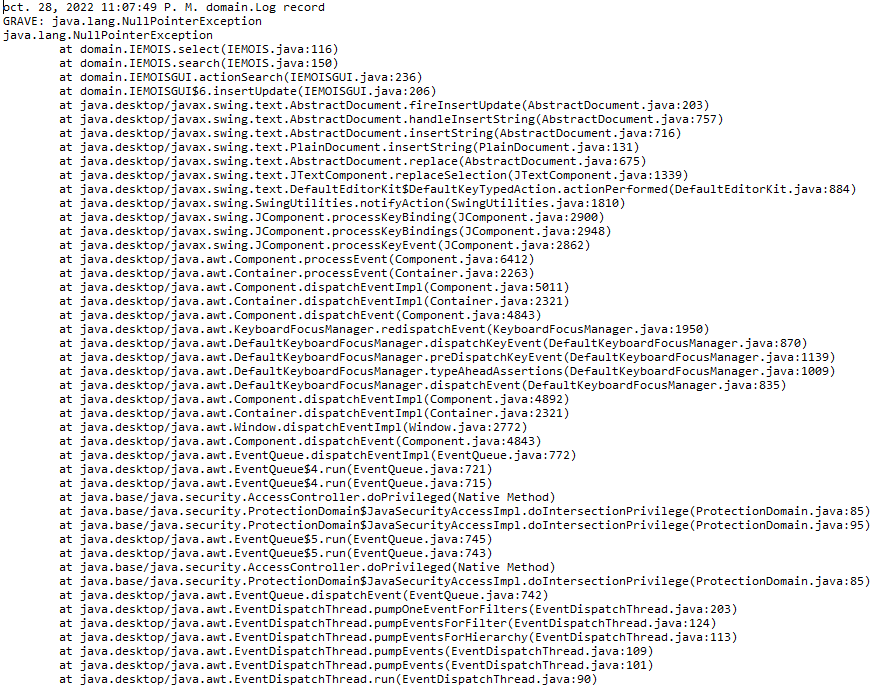


**4.** Ejecuten nuevamente la aplicación con el caso propuesto en 1. ¿Qué mensaje salió en pantalla? ¿La aplicación termina? ¿Qué información tiene el archivo de errores?

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Salió en la pantalla el mensaje “Ha ocurrido un error inesperado”. No se termina la aplicación.



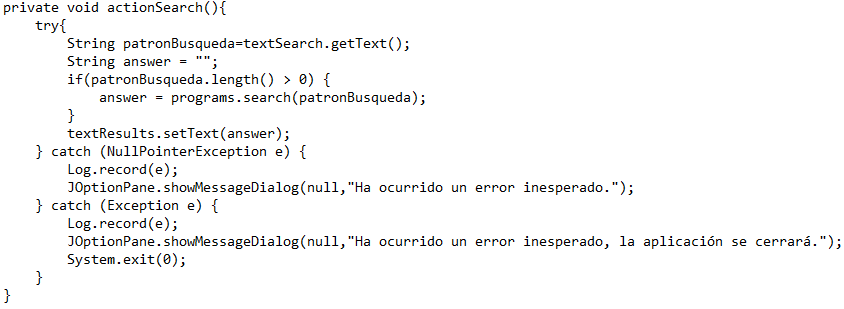
Tiene la fecha en la que se registro el error junto con la hora y el tipo de excepción con su información.

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

Si la aplicación tiene un riesgo de integridad lo más adecuado sería que terminara la ejecución, pero antes se le debería mostrar un mensaje al usuario que la aplicación debe detenerse por un error. En este caso lo más adecuado es continuar explicándole al usuario que hubo un fallo.

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

Para las excepciones de tipo *NullPointerException* se registrarán y se le mostrará un mensaje al usuario, pero para cualquier otra excepción se hará lo mismo pero el mensaje indicará que se cerrará la aplicación y se cierra.



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

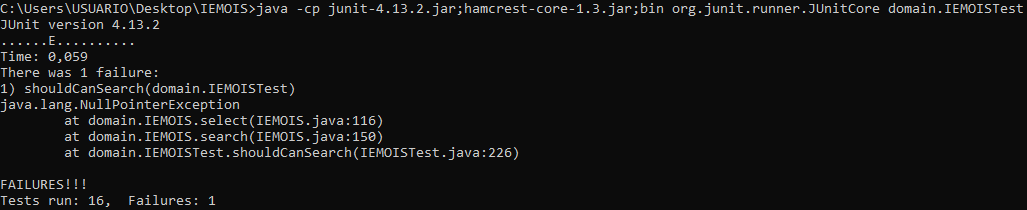
**1.** Revisen el código asociado a buscar en la capa de presentación y la capa de dominio. ¿Qué método es responsable en la capa de presentación? ¿Qué método es responsable en la capa de dominio?

En la capa de presentación los métodos responsables son *actionSearch()* y *prepareSearchArea ()*. En la capa de dominio es el método *search(String prefix).*

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

Interfaz de usuario gráfica

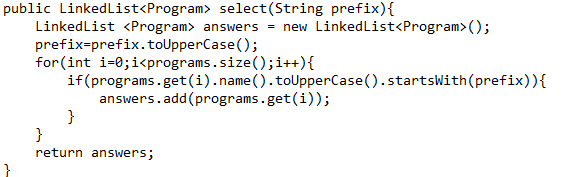
Descripción generada automáticamente con confianza baja



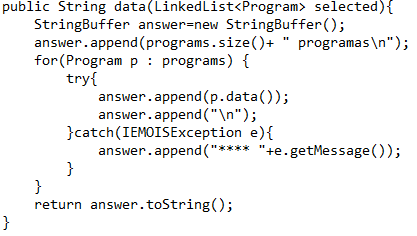
**3.** ¿Cuál es el error? Soluciónenlo. Capturen los resultados de las pruebas.

El error está en el método *select,* ya que en él se está instanciando de forma errónea el *LinkedList<Program>* llamado *answer* se le está asignado *null* cuando debería ser *new LinkedList<Program>().*

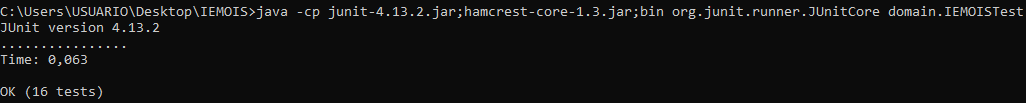
Texto

Descripción generada automáticamente 

Y en el método *data*, se está incluyendo todos los programas que tiene el proyecto, solo se debe tener en cuenta los que se reciben en la *LinkedList* llamada *selected,* para esto se debe cambiar *programs* por *selected.*

 Texto, Carta

Descripción generada automáticamente

**

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

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Ahora si deja escribir completamente la búsqueda, ya no aparece ningún error. El archivo de errores tiene el error que se obtuvo al principio, y no se agregó ninguno con la última consulta.

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

Para las consultas se van a listar los programas enumerándolos, para que el usuario pueda distinguirlos fácilmente y pueda ver cuales están disponibles, de igual forma los cursos de las especializaciones estarán numerados. Para su implementación se modifica los métodos *data()* de *IEMOIS y Specialization,* de modo que se incluya un número al comenzar la información del curso.

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

**RETROSPECTIVA**

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

14 horas cada integrante.

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

El laboratorio esta completo.

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

*Coding standards* ha sido la practica XP más útil ya que esta nos invita mantener una escritura de código consistente y fácil de leer entre los dos a la hora de realizar cualquier tipo de avance. En caso de detectarse alguna falencia o mejora que se pueda realizar, la finalidad principal del código no se verá afectada por alguna clase de mal entendido.

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

Pasar de BlueJ a consola, realizar los cambios al código, ejecutar las pruebas y el programa, fue un cambio que no tenía contemplado.

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

Compilar y ejecutar el código sin un IDE (BlueJ) escribiéndolo en un note pad y realizando las instrucciones previamente descritas a través de consola.

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

Analizar detalladamente el camino de ejecución de este laboratorio para especificar correctamente las excepciones y la forman en la que son tratadas. Nos comprometemos a continuar dedicando los tiempos apropiados a estas prácticas para así obtener mejores resultados.