**EQUIPOS**

**EN BLUEJ**

**PRACTICANDO MDD Y BDD con EXCEPCIONES**

En este punto vamos a aprender a diseñar, codificar y probar usando excepciones. Para esto se van a trabajar dos métodos de la clase Equipo y la excepción EquipoExcepcion.

1. En su directorio descarguen los archivos contenidos en equipo.zip, revisen el contenido y estudien el diseño estructural de la aplicación.
2. Dadas las pruebas, diseñen y codifiquen el método valorHora.
3. Dada la especificación, diseñen, codifiquen y prueben el método valorHoraEstimado.

**Iemois**

**EN CONSOLA**

**Conociendo el Proyecto**

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

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

Se tienen 3 archivos, no están organizados, deberían estar organizados en las carpetas bin, docs y src.

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

mkdir src

mkdir bin

mkdir docs

cd src

mkdir presentación

mkdir aplicación

mkdir pruebas

cd..

move \*.java src

cd src

move IemoisGUI.java presentación

move \*.java aplicación

cd..

cd bin

mkdir presentación

mkdir aplicación

mkdir pruebas

cd..

cd docs

mkdir presentación

mkdir aplicación

mkdir pruebas

1. Ejecute el proyecto. ¿Qué funcionalidades ofrece? ¿Cuáles funcionan?

Tiene las funcionalidades listar, adicionar y buscar. Funciona adicionar y listar.

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

Los cursos salen de la clase Iemois, tiene un método adicione donde pide ciertos parametros. Los adiciona la clase IemoisGUI.

**Arquitectura**

1. Inicie el diseño con un diagrama de paquetes en el que se presente los componentes y las relaciones entre ellos.
2. Usando ingeniería reversa. Presente el diseño actual de la **capa de aplicación**.
3. Considerando las funcionalidades del sistema. Realicen el diagrama de casos de uso correspondiente.

**Adicionar y listar. Todo OK.**

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

1. Adicionen un nuevo curso

**Machine Learning**

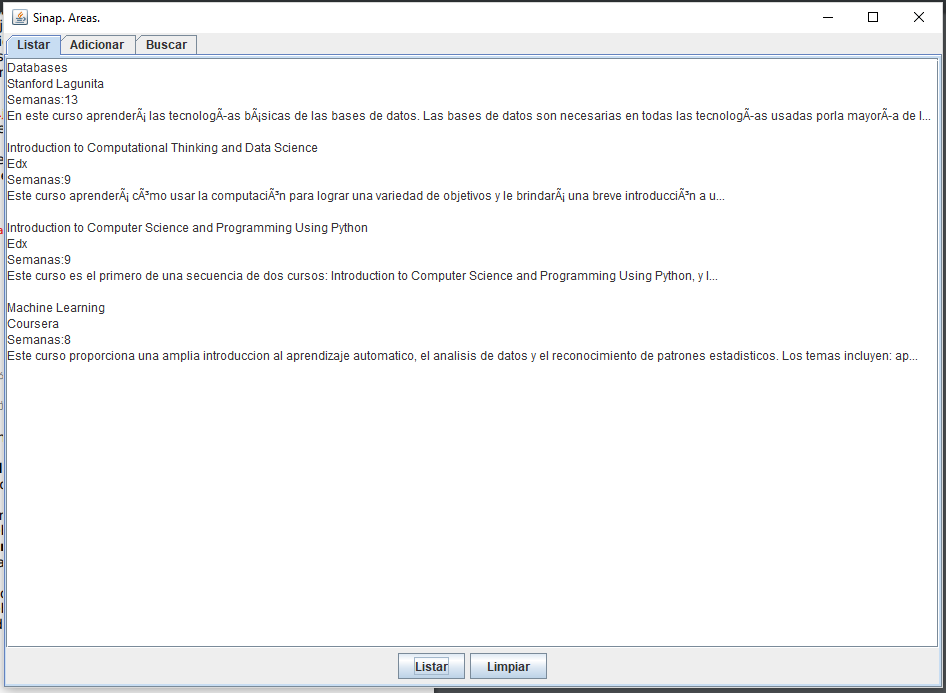
Inteligencia Artificial

Coursera

8

Este curso proporciona una amplia introducción al aprendizaje automático, el análisis de datos y el reconocimiento de patrones estadísticos. Los temas incluyen: aprendizaje supervisado, aprendizaje no supervisado y mejores prácticas en aprendizaje automático.

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

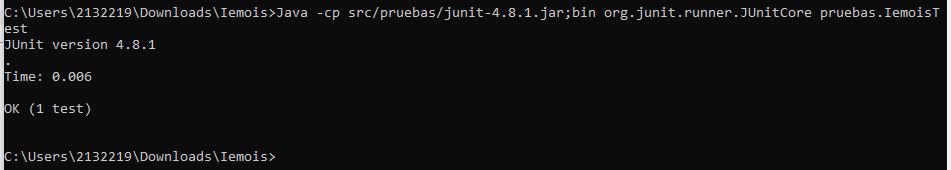


Se adiciona el nuevo curso y para comprobarlo vamos a la pestaña Listar y oprimimos Listar. Es adecuado el comportamiento, ya que se muestra el nuevo curso junto con los anteriores.

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

En aplicación es responsable adicione de la clase Iemois, en presentación es accionAdicionar.

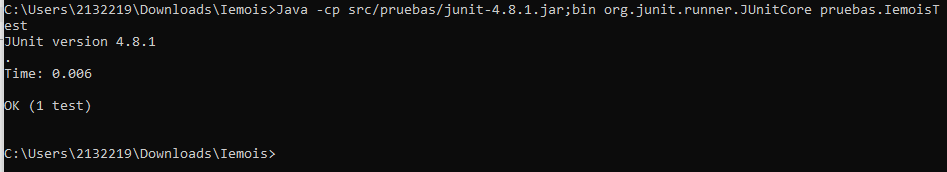
1. Realicen ingeniería reversa para la capa de aplicación de **adicionar**. Capturen los resultados de las pruebas de unidad.



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 aplicación el responsable es toString de la clase Iemois, y en la capa de presentación el responsable es accionListar.

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

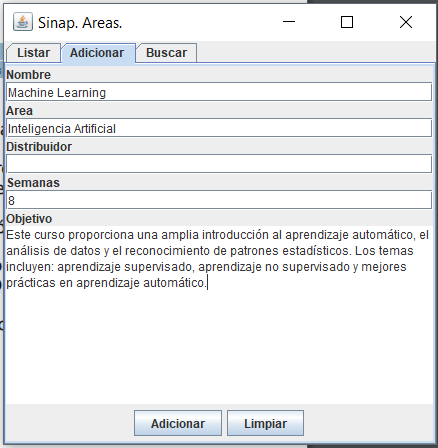


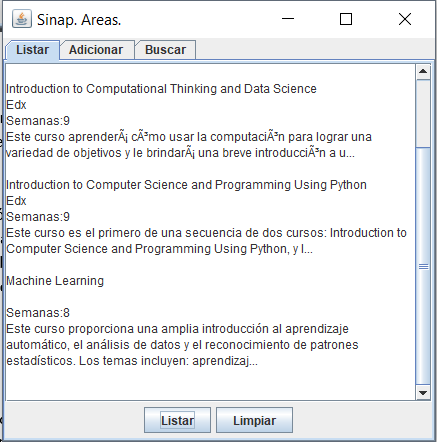
1. Propongan y ejecuten una prueba de aceptación.

**Adicionar un curso. ¿Y si no da un distribuidor?**

El objetivo es perfeccionar la funcionalidad de adicionar un curso.

1. Adicionen el curso Machine Learning sin distribuidor. ¿Qué ocurre? ¿Cómo lo comprueban? Capturen la pantalla. ¿Es adecuado este comportamiento?



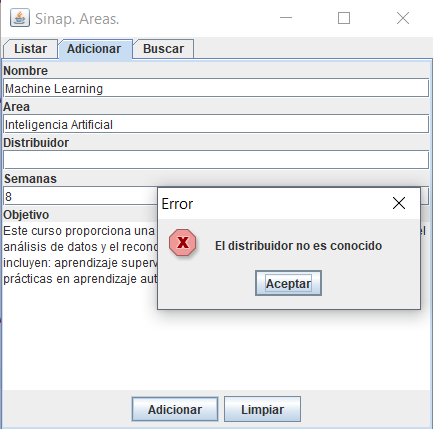


Se adiciona con el distribuidor vacío, se comprueba con el listar como se puede observar en la imagen.

1. Vamos a evitar la creación de áreas con un distribuidor vacío manejando una excepción IemoisExcepcion. Si el curso no tiene distribuidor, no lo creamos y se lo comunicamos al usuario1 . Para esto lo primero que debemos hacer es crear la nueva clase IemoisExcepcion 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.

Debería lanzar la excepción el método adicione, debería propagarla adicione y accionAdicionar (que está en la clase IemoisGUI) y el método que la atiende debería ser accionAdicionar también.

1. Construya la solución propuesta. Capturen los resultados de las pruebas.
2. Ejecuten nuevamente la aplicación con el caso de prueba propuesto en 1, ¿Qué sucede ahora? Capture la pantalla.



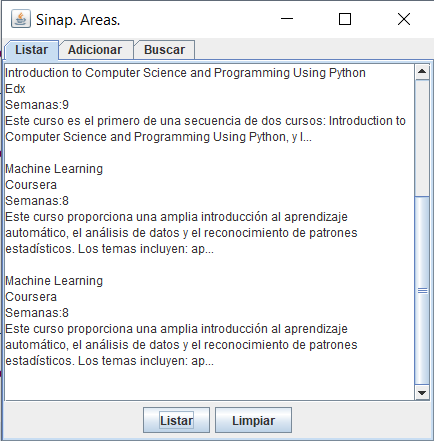
**Adicionar un curso. ¿Y si ya se encuentra?**

El objetivo es perfeccionar la funcionalidad de adicionar un curso.

1. Adicionen dos veces el nuevo curso Machine Learning ¿Qué ocurre? ¿Cómo lo comprueban? Capturen la pantalla. ¿Es adecuado este comportamiento?

Se adicionan estas nuevas áreas, se comprueba cuando se mira en listar.

El comportamiento no es adecuado porque no debería dejar adicionar dos veces el mismo curso.



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

Debería lanzar la excepción el método adicione, debería propagarla adicione y accionAdicionar (que se encuentra en la clase IemoisGUI) y el método que la atrapa debería ser acciónAdicionar.

1. Construya la solución propuesta. Capturen los resultados de las pruebas.
2. Ejecuten nuevamente la aplicación con el caso de prueba propuesto en 1., ¿Qué sucede ahora? Capture la pantalla.

**Adicionar un curso. ¿Y si dan mal las semanas?**

El objetivo es perfeccionar la funcionalidad de adicionar un curso.

1. Adicionen el nuevo curso Machine Learning pero en lugar de 8 semanas colquen B semanas. ¿Qué ocurre? ¿Cómo lo comprueban? Capturen la pantalla. ¿Es adecuado este comportamiento?
2. Analicen el diseño realizado. ¿Qué método debería lanzar la excepción? ¿Qué métodos deberían propagarla? ¿Qué método debería atenderla? Explique claramente.
3. Construya la solución propuesta. Capturen los resultados de las pruebas.
4. Ejecuten nuevamente la aplicación con el caso de prueba propuesto en 1., ¿Qué sucede ahora? Capture la pantalla.

**Adicionar un curso. ¿Otras condiciones?**

El objetivo es perfeccionar la funcionalidad de adicionar un curso.

1. Propongan nuevas condiciones para que la adición de un curso sea más robusta.
2. Construya la solución propuesta. (diseño, prueba de unidad, código) Capturen los resultados de las pruebas.