**Selección Colombia**

**Conociendo el proyecto**

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

Se tienen tres archivos, bin, src y docs; en bin estan los archivos .class, en src estan los archivos .java y el archive docs esta vacio.

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

Deben tener la estructura de un Proyecto de Desarrollo de software (bin, src y docs), la instrucción desde la consola seria:

Java -cp bin presentacion.SeleccionGUI

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

Tiene tres funciones principales (Listar, Adicionar y Buscar), de las cuales funcionan el listar y el adicionar, el buscar no funciona.

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

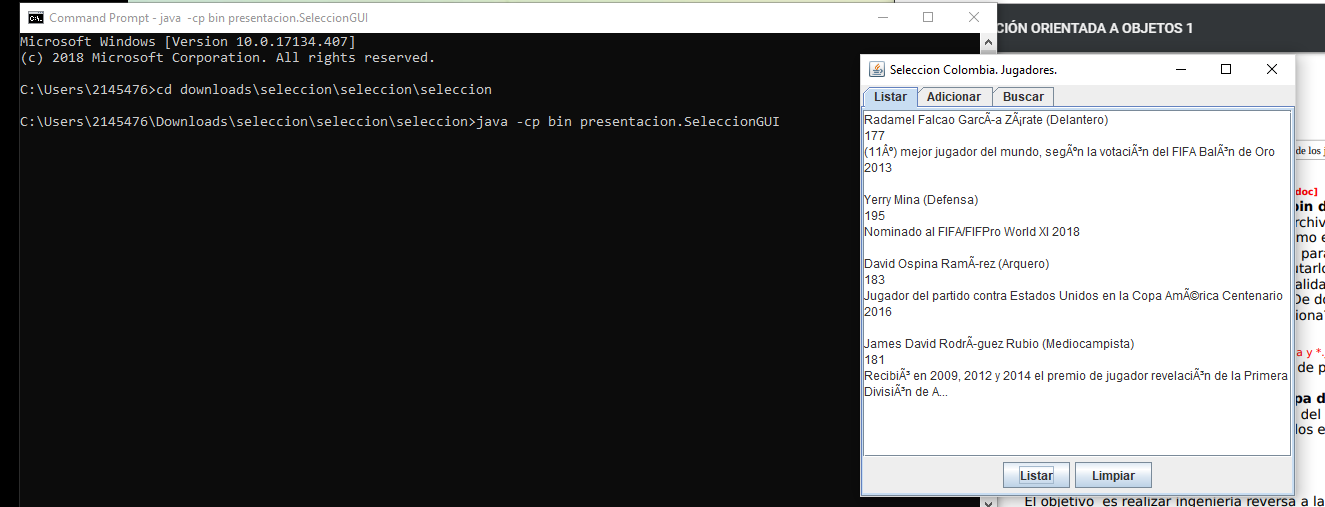
Estan en el codigo de la clase selección ingresados por defecto, la clase SeleccionGUI es la pide que se adicionen y son adicionadas en una LinkedList.

**Adicionar y listar. Todo OK.**

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

1. Adicionen un nuevo jugador (Yerry Mina 195 Defensa Nominado al FIFA/FIFPro World XI 2018) ¿Qué ocurre? ¿Cómo lo comprueban? Capturen la pantalla. ¿Es adecuado este comportamiento?

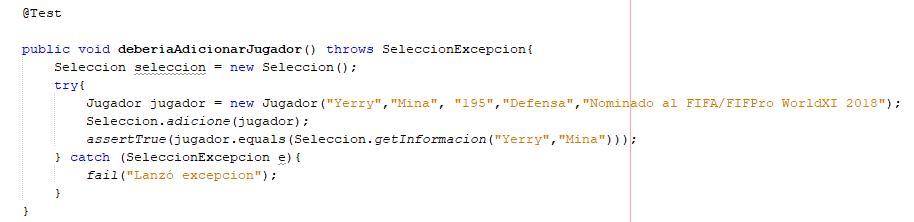
Lo adiciona, lo comprobamos con la función listar, este comportamiento es adecuado.

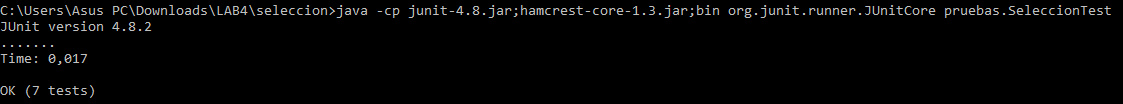


1. Revisen el código asociado a adicionar en la capa de presentación y la capa de aplicación. ¿Qué método es responsable en la capa de presentación? ¿Qué método en la capa de aplicación?

En la capa de presentación el método es accionAdicionar de la clase SeleccionGUI y en la capa de aplicación es el método adicione de la clase Selección.

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

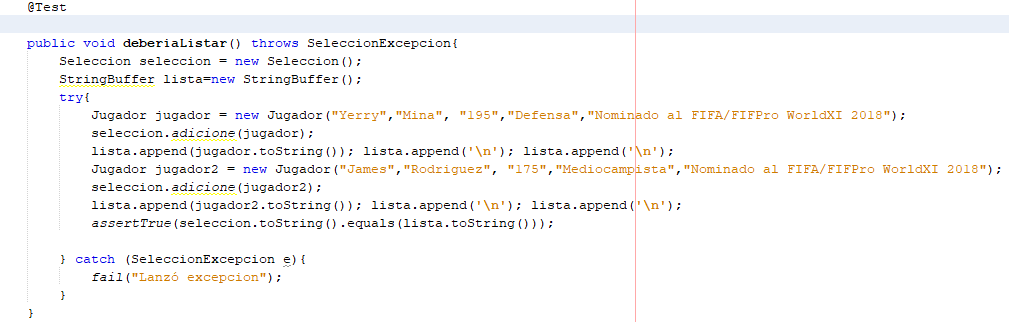


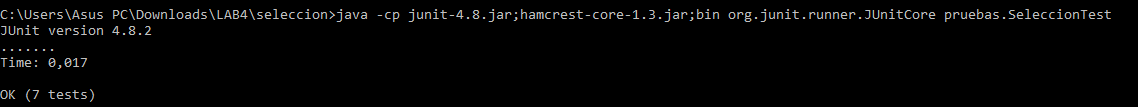


1. Revisen el código asociado a listar en la capa de presentación y la capa de aplicación. ¿Qué método es responsable en la capa de presentación? ¿Qué método en la capa de aplicación?

En la capa de presentación el método es accionListar de la clase SeleccionGUI y en la capa de aplicación es el método toString de la colección de la clase Selección.

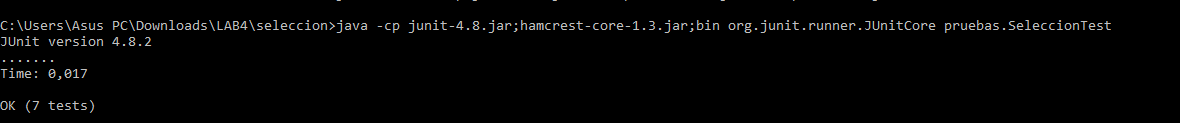
1. Realicen ingeniería reversa de la capa de aplicación para listar. Capturen los resultados de las pruebas de unidad.





1. Propongan y ejecuten una prueba de aceptación.



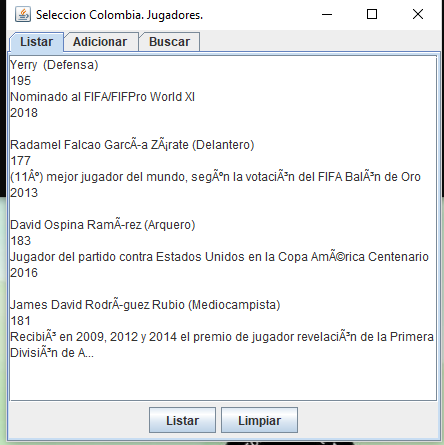


**Adicionar un jugador. ¿Y si no da los apellidos?**

El objetivo es perfeccionar la funcionalidad de adicionar un jugador.

1. Adicionen a Yerry sin apellido ¿Qué ocurre? ¿Cómo lo comprueban? Capturen la pantalla. ¿Es adecuado este comportamiento?

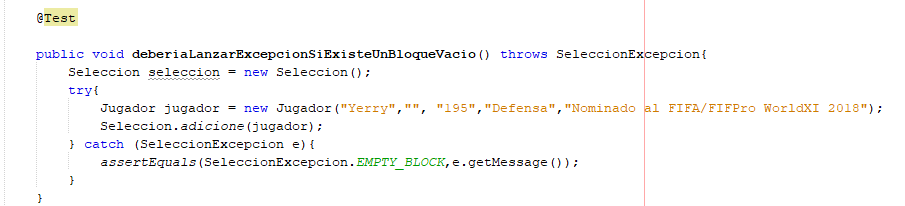
Lo adiciona, lo comprobamos con la función listar, este comportamiento no es adecuado.

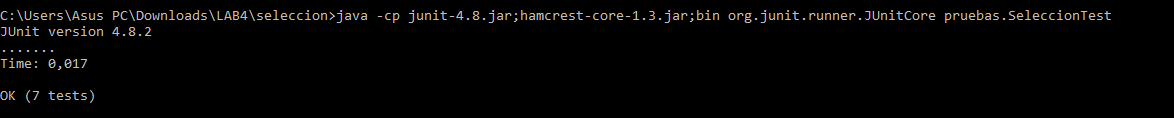


1. Vamos a evitar la creación de áreas con un distribuidor vacío manejando una excepción SeleccionExcepcion. Si el jugador no tiene distribuidor, no lo creamos y se lo comunicamos al usuario1. Para esto lo primero que debemos hacer es crear la nueva clase SeleccionExcepcion considerando este primer mensaje.
2. Analicen el diseño realizado. ¿Qué método debería lanzar la excepción? ¿Qué métodos deberían propagarla? ¿Qué método debería atenderla? Explique claramente.

El método accionAdicionar de la clase SeleccionGUI la lanza y el metodo de prepareAcciones debería atenderla.

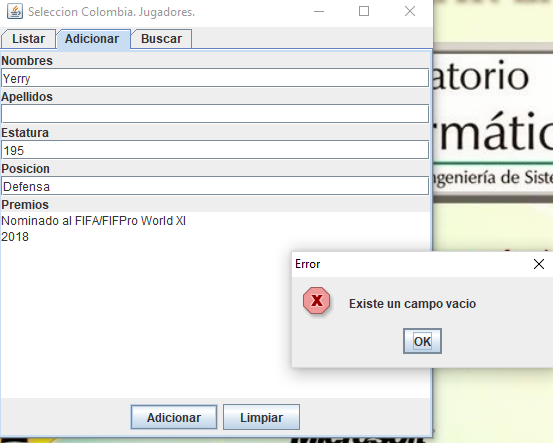
1. Construya la solución propuesta. Capturen los resultados de las pruebas.





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

Ahora genera nuestra excepcion personalizada.

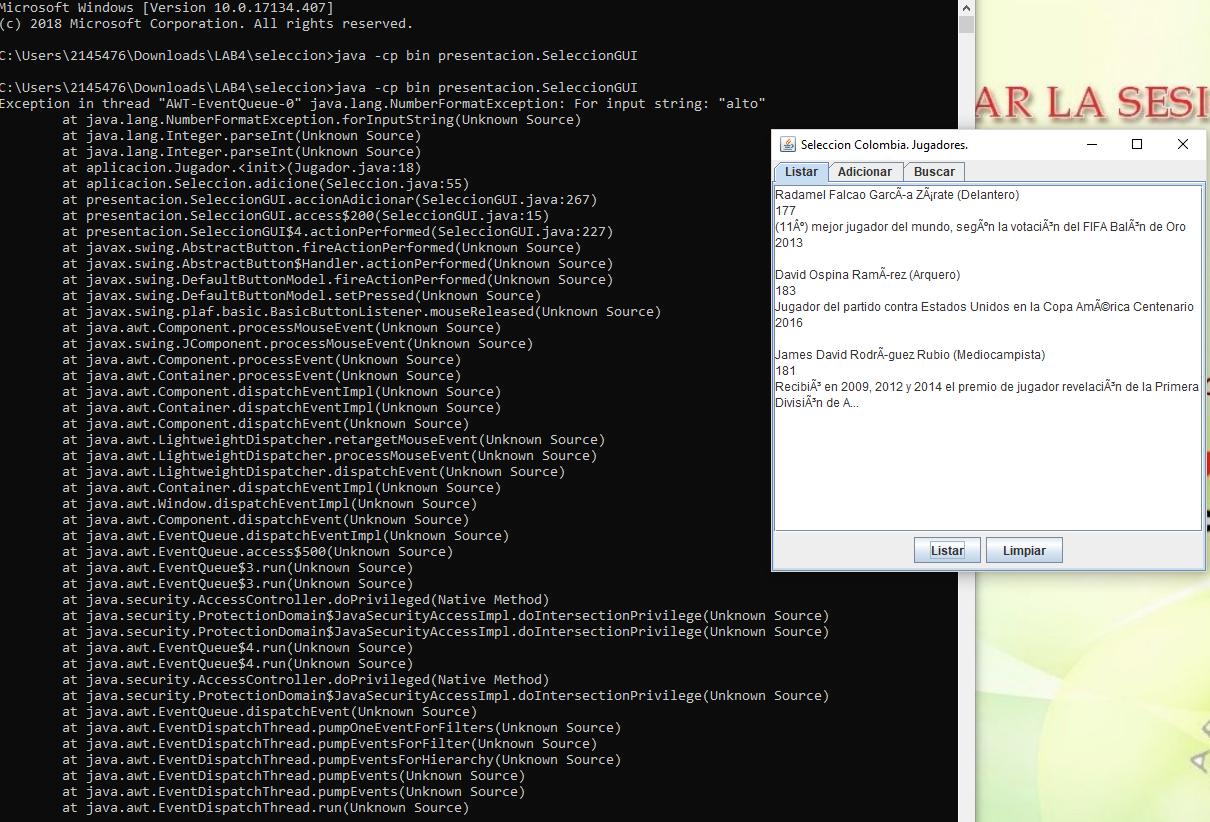


**Adicionar un jugador. ¿Y si da como altura un texto?**

El objetivo es perfeccionar la funcionalidad de adicionar un jugador.

1. Adicionen el nuevo jugador Yerry Mina, pero en lugar de 195 coloquen alto. ¿Qué ocurre? ¿Cómo lo comprueban? Capturen la pantalla. ¿Es adecuado este comportamiento?

No lo adiciona y genera una excepcion, lo comprobamos con la función listar, este comportamiento no es adecuado.



1. Analicen el diseño realizado. ¿Qué método lanzan la excepción? ¿Qué métodos la propagan? Explique claramente. ¿Qué problema tenemos ahí?

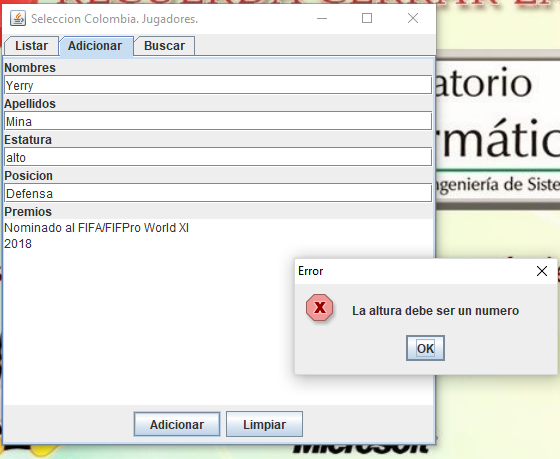
El método que la lanza es el constructor de la clase Jugador, lo propaga los métodos de adición de la clase Selección, los método accionAdicionar y prepareAcciones de la clase SeleccionGUI.

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

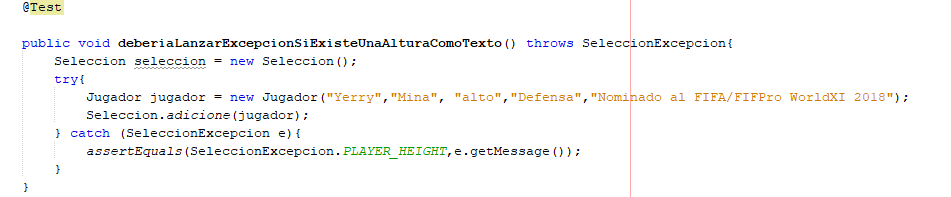
El método que la lanza es el constructor de la clase Jugador, lo propaga los métodos de adición de la clase Selección y el método accionAdicionar de la clase SeleccionGUI; al final lo atiende el método prepareAcciones de la clase SeleccionGUI.

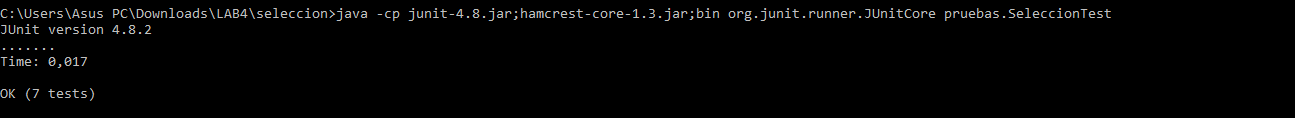
4. Ejecuten nuevamente la aplicación con el caso de prueba propuesto en 1., ¿Qué sucede ahora? Capture la pantalla.

Ahora genera nuestra excepción personalizada.



Pruebas :



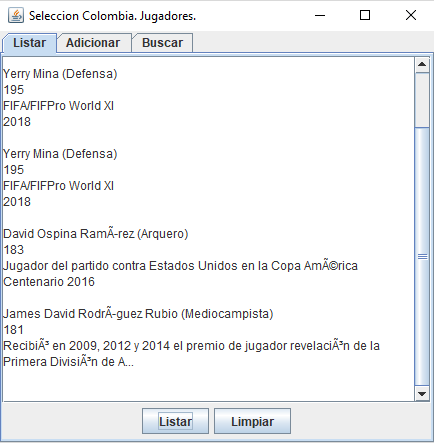


**Adicionar un jugador. ¿Y si ya se encuentra?**

El objetivo es perfeccionar la funcionalidad de adicionar un jugador.

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

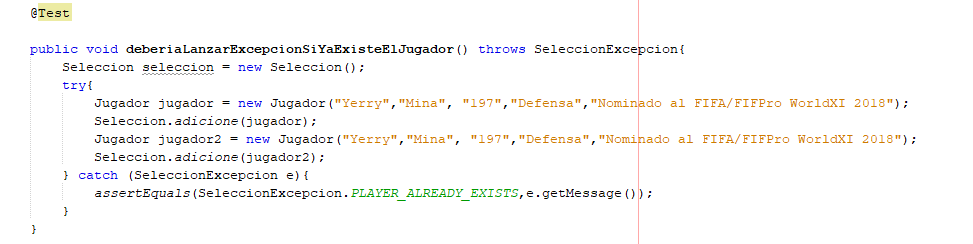
Lo adiciona, lo comprobamos con la función listar, este comportamiento no es adecuado.

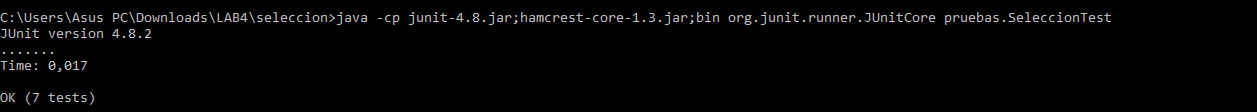


1. Analicen el diseño realizado. ¿Qué método debería lanzar la excepción? ¿Qué métodos deberían propagarla? ¿Qué método debería atenderla? Explique claramente.

El método que la lanza es el adicionar de la clase Selección, lo propaga el método accionAdicionar de la clase SeleccionGUI; al final lo atiende el método prepareAcciones de la clase SeleccionGUI.

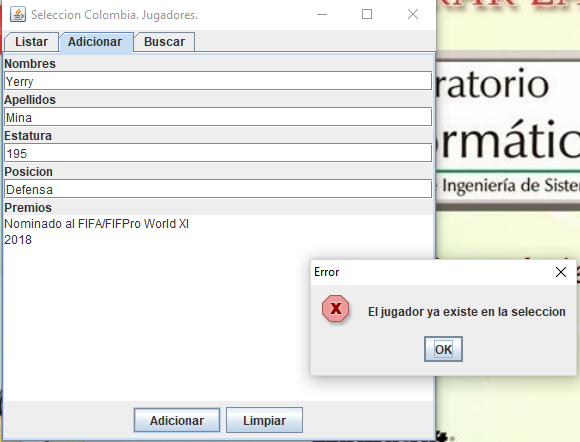
1. Construya la solución propuesta. Capturen los resultados de las pruebas.





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

Ahora genera nuestra excepción personalizada.

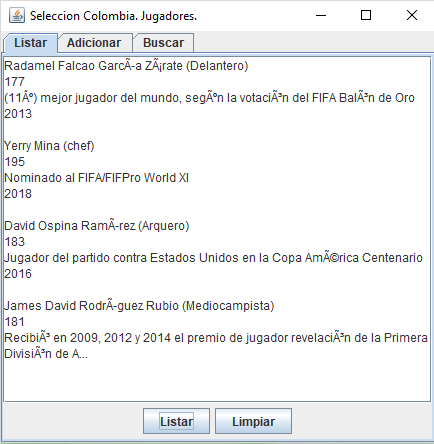


**Adicionar un jugador. ¿Y si dan mal la posición?**

El objetivo es perfeccionar la funcionalidad de adicionar un jugador.

5.Adicionen el nuevo jugador Yerry Mina, pero en lugar de Delantero coloquen chef. ¿Qué ocurre? ¿Cómo lo comprueban? Capturen la pantalla. ¿Es adecuado este comportamiento?

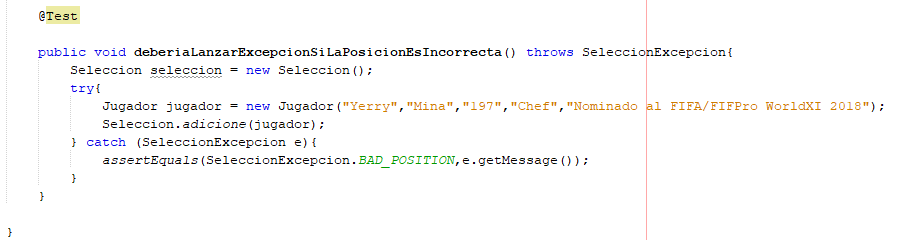
Lo adiciona, lo comprobamos con la función listar, este comportamiento no es adecuado.

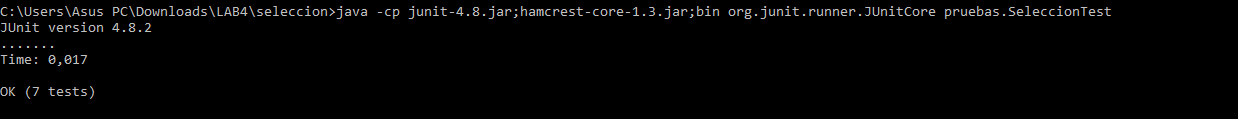


1. Analicen el diseño realizado. ¿Qué método debería lanzar la excepción? ¿Qué métodos deberían propagarla? ¿Qué método debería atenderla? Explique claramente.

El método que la lanza es el constructor de la clase Jugador, lo propaga los métodos de adición de la clase Selección y el método accionAdicionar de la clase SeleccionGUI; al final lo atiende el método prepareAcciones de la clase SeleccionGUI.

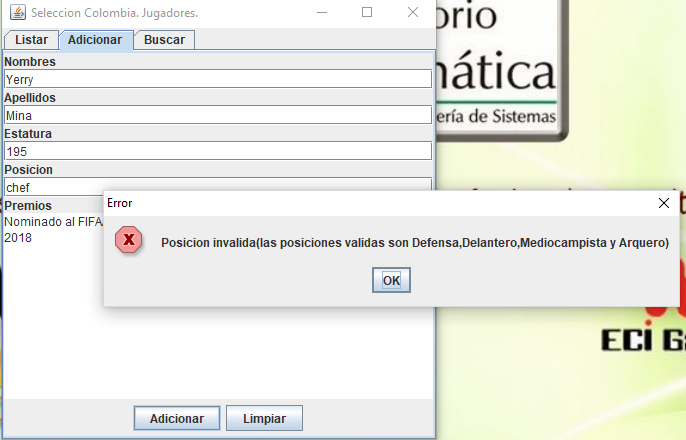
1. Construya la solución propuesta. Capturen los resultados de las pruebas.





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

Ahora genera nuestra excepción personalizada.



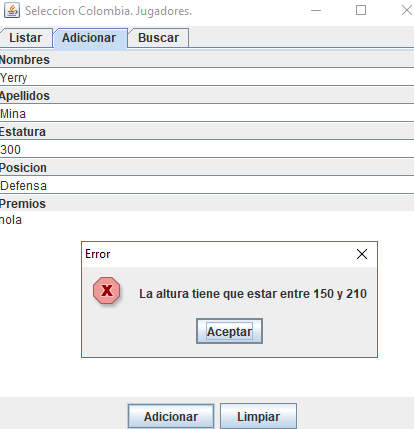
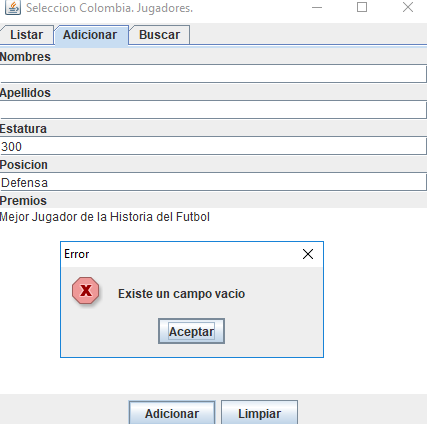
**Adicionar un jugador. ¿Otras condiciones?**

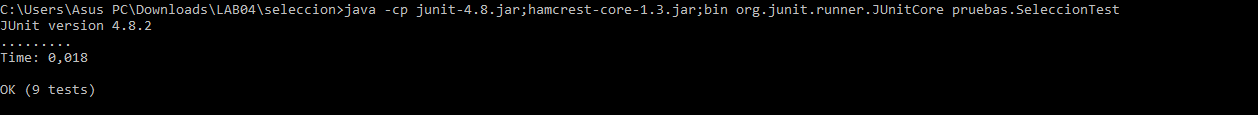
Para presentar los mensajes de error al usuario use el método de clase de JOptionPane public static void showMessageDialog (Component parentComponent, Object message, String title, int messageType) throws HeadlessException Con componente padre: este mensaje: la cadena correspondiente al mensaje de error de la excepcion correspondiente, título: ERROR y tipo de mensaje: JOptionPane.ERROR\_MESSAGE (NO OLVIDEN BDD – MDD) El objetivo es perfeccionar la funcionalidad de adicionar un jugador.

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

No se pueden dejar nombre, apellidos, estatura y posicion vacias; ademas la estatura debe estar entre 150 y 210 centímetros.

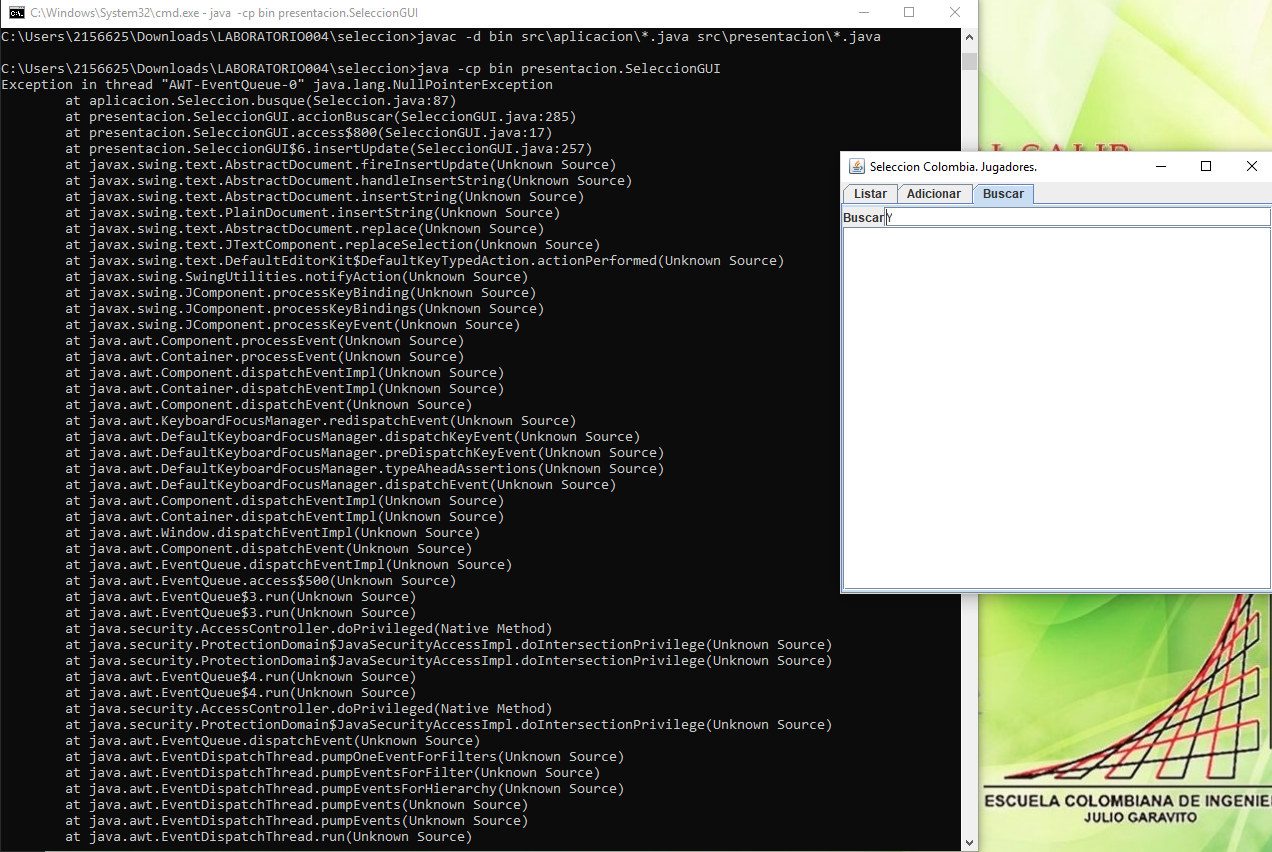
1. Construya la solución propuesta. (diseño, prueba de unidad, código) Capturen los resultados de las pruebas.





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

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



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

Guarda las excepciones realizadas en un documento.

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.

Cada vez que se manda una excepcion al realizar uno de los tres métodos, también debe añadirse al registro.

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 el mensaje “Hubo un error inesperado al buscar, mire el registro”

La aplicación si termina

En el log se guarda, la fecha donde ocurrió la excepcion, el tipo de excepcion y la ruta donde se encontró la excepcion.

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

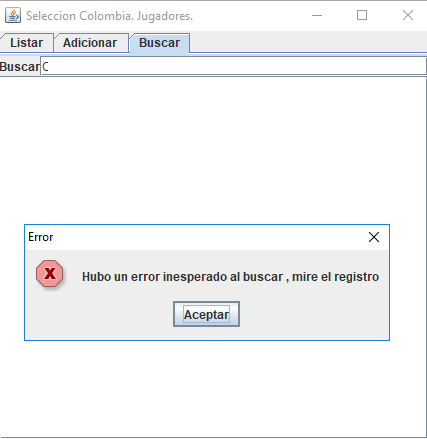
Para la acción de buscar, sería adecuado no terminar la ejecución.

Dependería del alcance del método que se realiza, si afecta a los demás atributos debería parar.

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.

Al realizar el método de búsqueda no se debería realizar System.exit(0)

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 el mensaje “Hubo un error inesperado al buscar, mire el registro”

La aplicación no termina

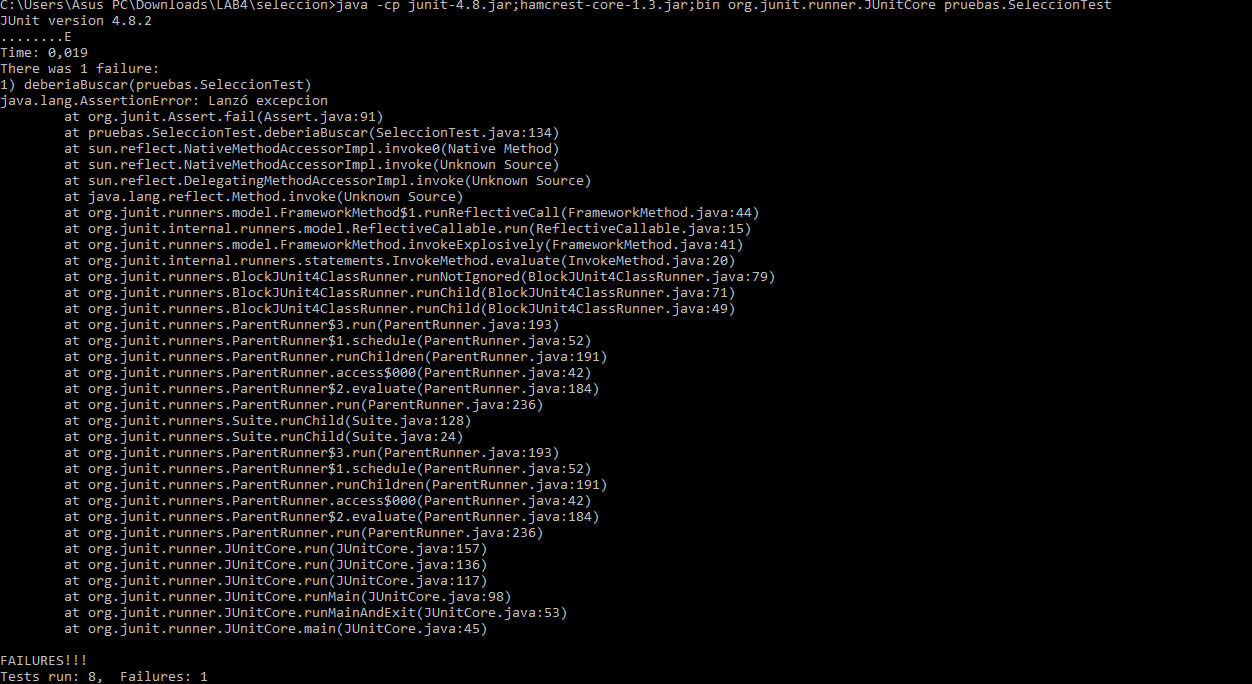
En el log se guarda, la fecha donde ocurrió la excepcion, el tipo de excepcion y la ruta donde se

encontró la excepcion.

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

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

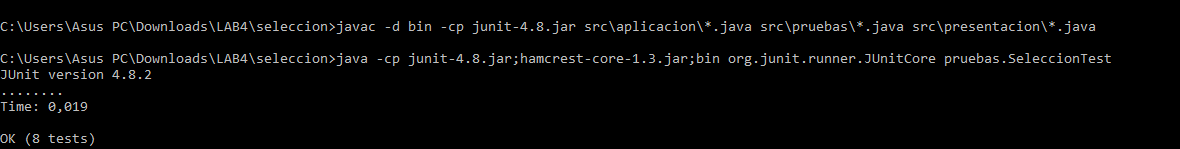
En la capa de presentación el método es accionBuscar de la clase SeleccionGUI y en la capa de aplicación es el método busque de la clase Selección.

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

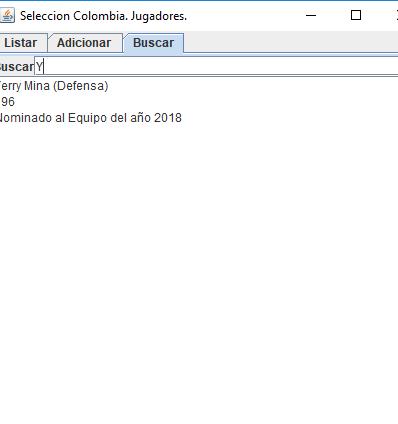
3. ¿Cuál es el error? Soluciónenlo. Capturen los resultados de las pruebas.

Que, al instanciar los resultados, se inician con un valor nulo en vez de un constructor del arrraylist.





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



El archivo de registro no guardo nada.

**RETROSPECTIVA**

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

8 horas por hombre .

1. ¿Cuál es el estado actual del laboratorio? ¿Por qué?
2. Considerando las prácticas XP del laboratorio. ¿cuál fue la más útil? ¿por qué?

La práctica más útil fue simplicidad ya que simper mantuvimos los metodos sencillos y entendibles.

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

Lograr que las pruebas compilaran y corrieran desde la consola.

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

La compilación de las pruebas y la ambigüedad del texto del pdf.

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

Trabajamos la mayoría del laboratorio presencialmente. Así , logramos reducir la probabilidad de cometer errores y pudimos complementar de mejor manera nuestras habilidades y conocimientos.