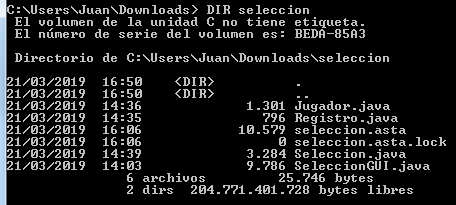
**Conociendo el proyecto** [En lab04.doc]

No olviden respetar los directorios bin docs src

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?



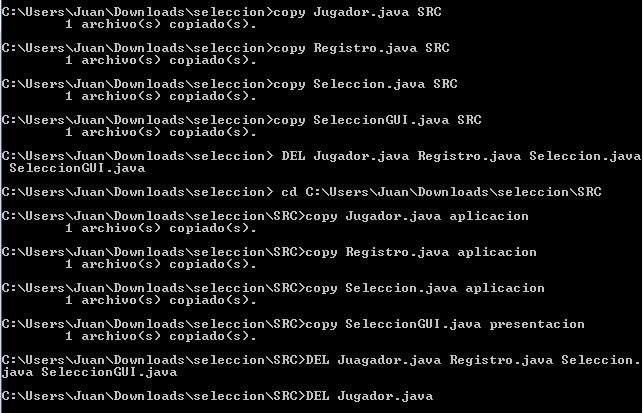
Los archivos .java deberían estar en un directorio src los .asta deberían estar en un directorio docs.

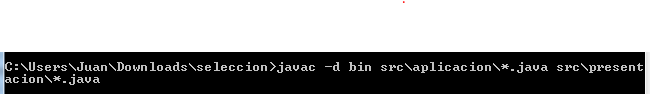
1. Prepare los directorios necesarios para ejecutar el proyecto. ¿qué estructura debe tener? ¿Qué instrucciones debe dar para ejecutarlo?

Compilar: javac -d bin src\aplicacion\\*.java src\presentacion\\*.java

Ejecutar: java -cp bin presentacion.SeleccionGUI

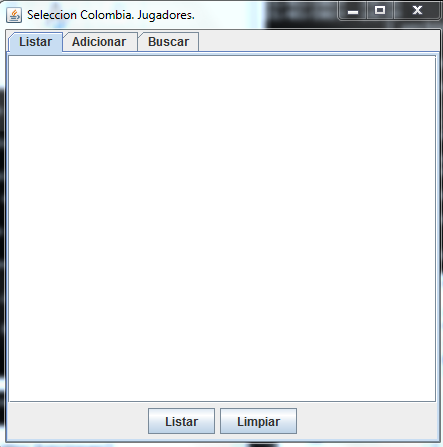






1. Ejecute el proyecto, ¿qué funcionalidades ofrece? ¿cuáles funcionan?





Listar los jugadores existentes, agregar jugadores y buscar jugadores.

1. Revisen el código del proyecto.

¿De dónde salen los jugadores iniciales?

R/. Están definidos por defecto en la clase Selección.

¿Qué clase pide que se adicionen?

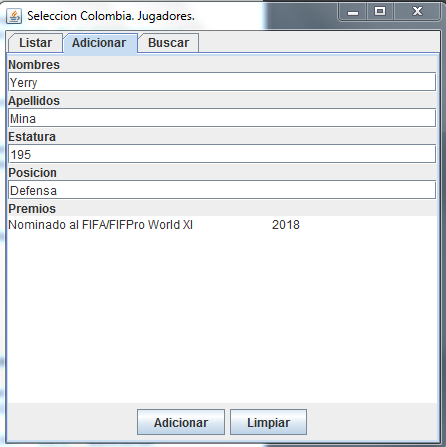
R/. La clase SeleccionGUI.

¿Qué clase los adiciona?

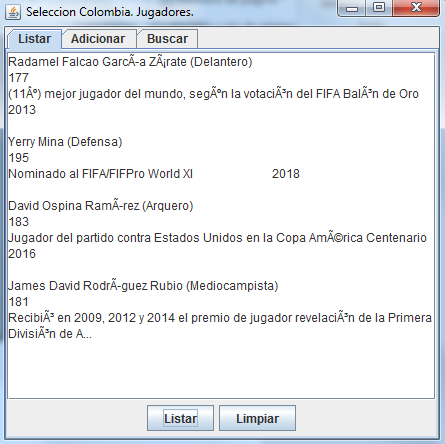
R/. La clase Selección.

1. Adicionen un nuevo jugador

¿Qué ocurre?



¿Cómo lo comprueban? Capturen la pantalla.



¿Es adecuado este comportamiento?

R/. Al adicionar debería limpiar los text box y actualizar la parte de listar.

2. 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?

R/. accionAdicionar() de la clase SeleccionGUI

¿Qué método en la capa de aplicación?

R/. adicione() de la clase Selección.

3. Realicen ingeniería reversa para la capa de aplicación 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 aplicación.

¿Qué método es responsable en la capa de presentación?

R/. accionListar()

¿Qué método en la capa de aplicación?

R/. toString() de la clase Jugador.

5. Realicen ingeniería reversa de la capa de aplicación para listar.

Hecho

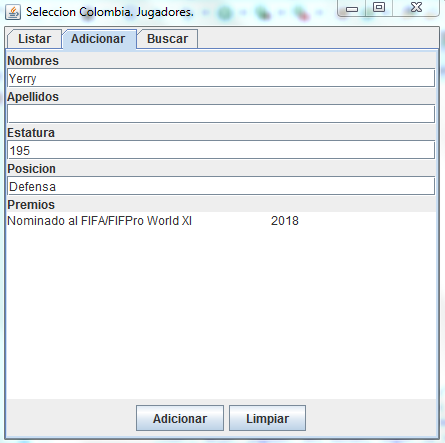
6. Propongan y ejecuten una prueba de aceptación.

Hecho

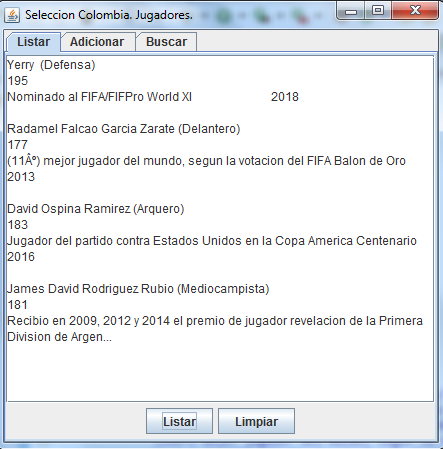
**Adicionar un jugador. ¿Y si no da los apellidos?**

[En lab04.doc, seleccion.asta y \*.java] (NO OLVIDEN BDD – MDD)

1. Adicionen a Yerry sin apellido ¿Qué ocurre?



¿Cómo lo comprueban? Capturen la pantalla.



¿Es adecuado este comportamiento?

R/. No, debería decir que no se puede crear un jugador sin apellido.

2. 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.

Hecho en código

3. Analicen el diseño realizado.

¿Qué método debería lanzar la excepción?

R/. Constructor de Jugador

¿Qué métodos deberían propagarla?

R/. Constructor de jugador y adicione de Selección

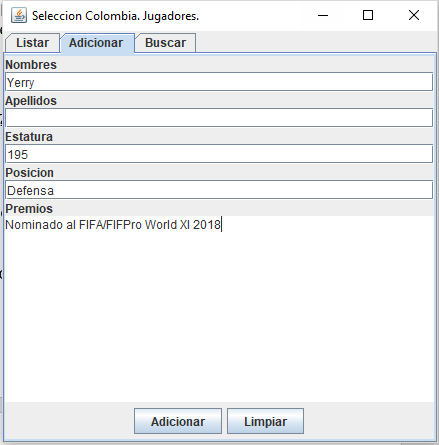
¿Qué método debería atenderla?

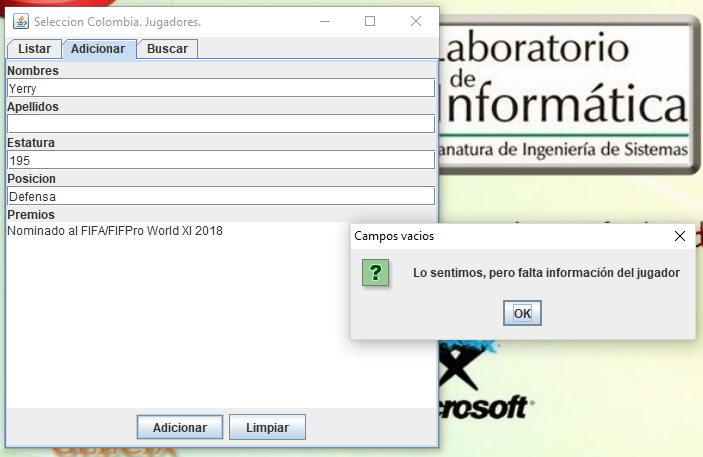
R/. Se atiende en acciónAdicionar de SeleccionGUI

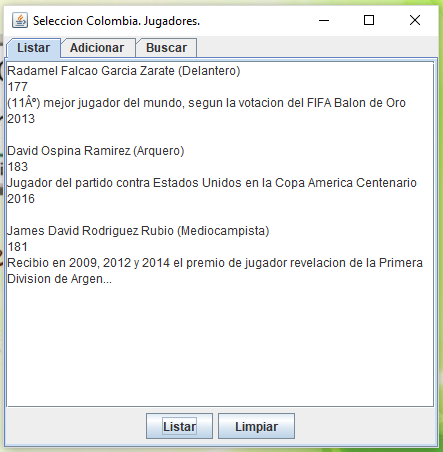
4. Construya la solución propuesta. Capturen los resultados de las pruebas.

Crear pruebas

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

.



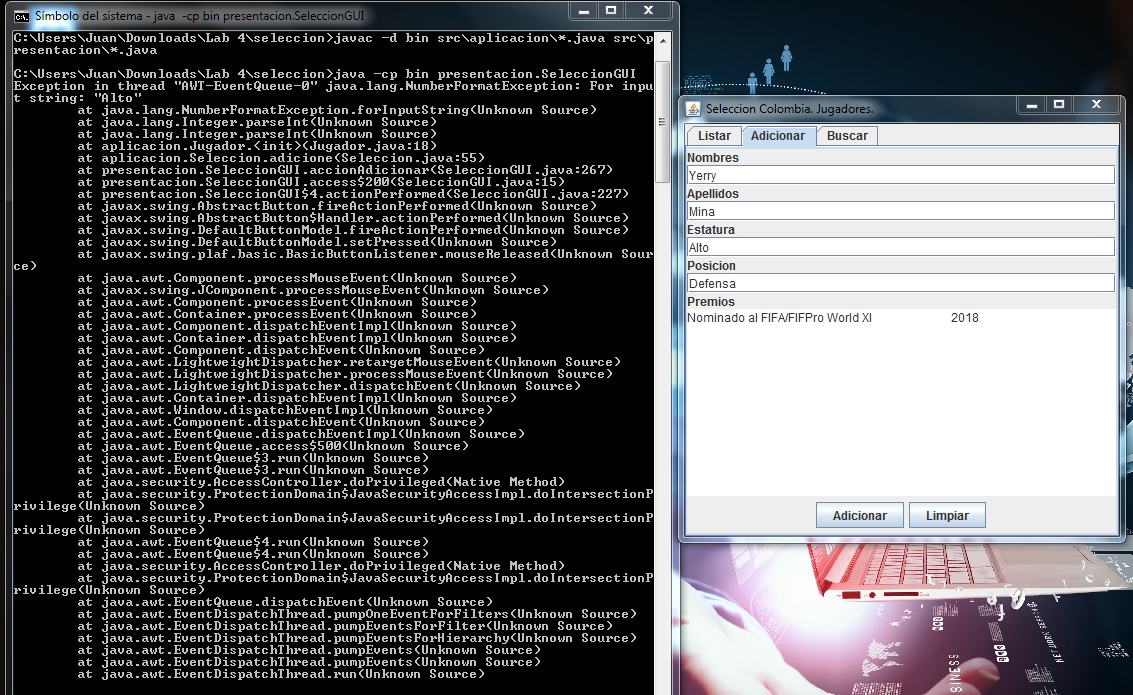


**Adicionar un jugador. ¿Y si da como altura un texto?**

[En lab04.doc, seleccion.asta y \*.java](NO OLVIDEN BDD - MDD)

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, ya que el error le esta saltando al usuario por consola.

2. Analicen el diseño realizado.

¿Qué método lanzan la excepción?

R/. Constructor de Jugador

¿Qué métodos la propagan?

R/. No es propagada.

¿Qué problema tenemos ahí?

R/. Se está tratando de convertir un string a entero por lo que se genera la excepción NumberFormatException.

3. Analicen el diseño realizado.

¿Qué método debería lanzar la excepción propia?

R/. El constructor de Jugador.

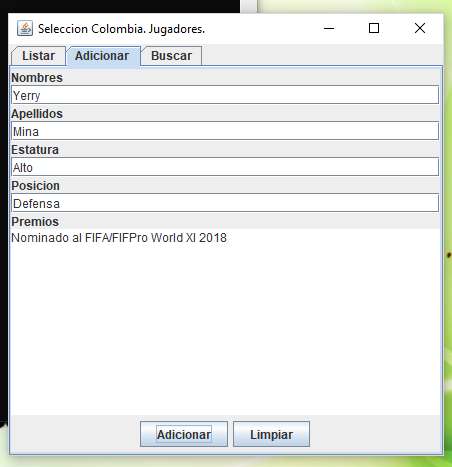
¿Qué métodos deberían propagarla?

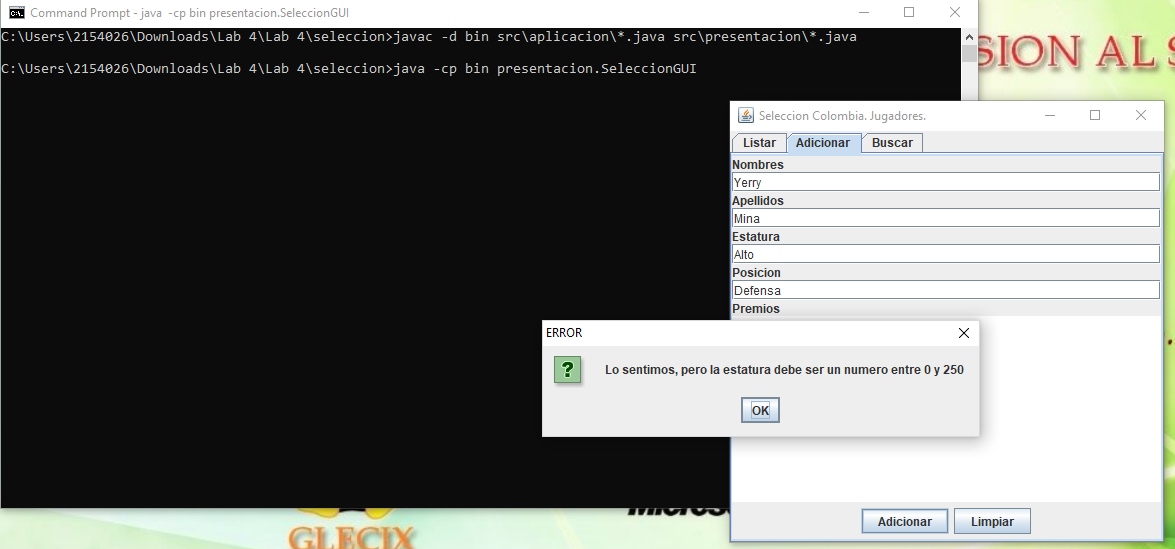
R/. Constructor de jugador y adicione de Selección.

¿Qué método debería atenderla?

R/. Se atiende en acciónAdicionar de SeleccionGUI.

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





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

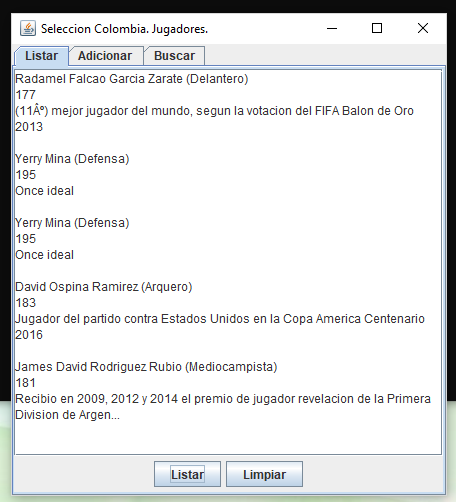
[En lab04.doc, seleccion.asta y \*.java] (NO OLVIDEN BDD - MDD)

El objetivo es perfeccionar la funcionalidad de adicionar un jugador.

1. Adicionen dos veces el nuevo jugador ¿Qué ocurre?

R/. Se agrega dos veces el mismo jugador.

¿Cómo lo comprueban? Capturen la pantalla.



¿Es adecuado este comportamiento?

R/. No, debería decir que el jugador ya existe.

2. Analicen el diseño realizado. ¿Qué método debería lanzar la excepción?

R/. Adicione de la clase Selección.

¿Qué métodos deberían propagarla?

R/. Adicione de la clase Selección.

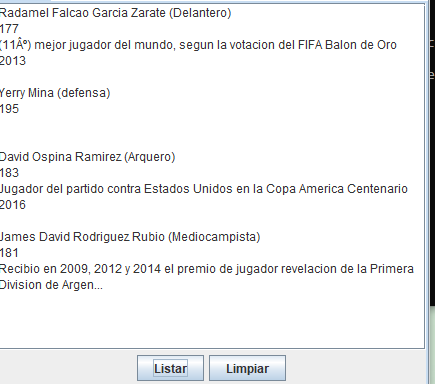
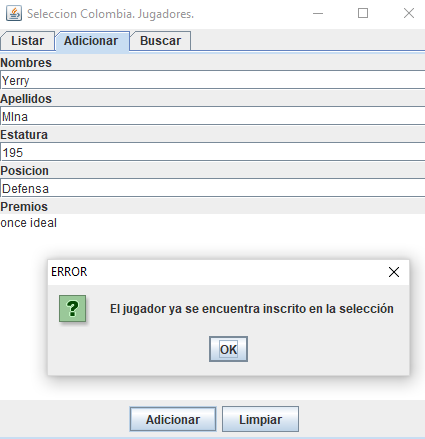
¿Qué método debería atenderla?

R/. accionAdicionar de SeleccionGUI.

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

Crear pruebas

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

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

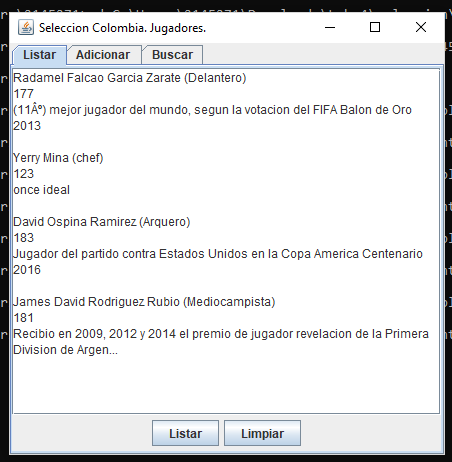
[En lab04.doc, seleccion.asta y \*.java] (NO OLVIDEN BDD - MDD)

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?

R/. Acepta al jugador con esa posición.

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



R/. No.

6. Analicen el diseño realizado.

¿Qué método debería lanzar la excepción?

R/. Constructor de Jugador.

¿Qué métodos deberían propagarla?

R/. adicionar

¿Qué método debería atenderla? Explique claramente.

R/. accionAdicione

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

R/.

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