# Índice de contenido



# Acerca del documento

Éste documento está orientado a toda aquella persona que esté en relación con el desarrollo del Software ORS. Contiene toda la información necesaria que le será de ayuda para lograr una correcta interpretación e implementación del programa.

## Código fuente

En las partes donde sea necesario escribir partes de código, se usará la letra Century SchoolBook URW.

# Preparación del entorno de trabajo

## Introducción

Aunque el código fuente puede abrirse con cualquier editor de texto, el programa se hizo usando

Netbeans 8.0.2.

## Librerías adicionales

### JUnit

Se usa JUnit para realizar los tests unitarios.

## Habilitar los “assert” en Netbeans

En el programa, se usan los `assert` para realizar comprobaciones y avisar cuando hay alguna falla. Cuando el usuario final ejecuta el programa, estos `assert` no se compilan. Sin embargo cuando se está desarrollando el programa, es necesario habilitarlos (Y así poder detectar los errores antes que se distribuya el producto al usuario final).

En NetBeans, se debe ir al proyecto ORS, cliquear en *Properties > Run > VM Options*. Dentro del campo ingresar **`-ea`**.

# Estándar usado en la programación (codificación)

## Introducción

Se sigue el estándar de codificación de Java, con los siguientes agregados para este proyecto

## Constructor

Cuando los parámetros del constructor son iguales a un campo de la clase, a dicho parámetro se le antepone un guión bajo. Por ejemplo, si tenemos el siguiente campo de la clase:

```
private final BigInteger numerador;

private final BigInteger denominador;
```

el constructor debe ser de la forma:

```
public Fraccion( BigInteger _numerador, BigInteger _denominador ){

        //Se desarrolla el código.

}
```

Ésto se hace para diferenciar el nombre de los parámetros y así evitar los posibles problemas de una mala asignación.

# Arquitectura en general

## Estructura

Arquitectónicamente, la aplicación se divide en capas. Hay 3 capas:

- Vista: son los archivos FXML que guardan la estructura de la vista, acciones, etc.
- Presentador: encargado de manejar las acciones desencadenas en la vista
- Clases del dominio: encargadas de implementar la lógica referida a los algoritmos que resuelven los problemas.

Ésto produce que, si se debe modificar la vista (Ya que por ejemplo, se elige un nuevo tipo de interfaz), las clases del dominio no se vean modificadas en absoluto.

## Interfaz gráfica

### Introducción

La aplicación usa JavaFX utilizando el patrón MVP.

### Manejo de vistas

Para el manejo de las vistas, se tomo como referencia el siguiente artículo
https://blogs.oracle.com/acaicedo/entry/managing_multiple_screens_in_javafx1

# Clases

Además de la documentación que existe generada por Javadoc, a continuación se agregará información adicional

## Restricción de dos variables

En el método `public boolean cumpleConLaRestriccion( double x1, double x2 )` se toma que todas las variables deben ser mayor o igual a 0 (Xn >= 0). Si en un futuro el programa se mejora, se debe modificar esa parte.

# Definiciones del dominio

## Forma aumentada

Conjunto de ecuaciones donde se le agregaron las variables suplementarias y se transformaron las inecuaciones (con desigualdades) a ecuaciones.

## Solución aumentada

Es una solución de las variables originales (las variables de decisión) que se aumentó con los valores correspondientes de las variables de holgura.

# Pasos a realizar en la liberación de cada nueva versión

Cada vez que se libera una nueva versión (O incluso puede darse cuando se realice un commit en git sin subir una nueva versión) se deben realizar los siguientes pasos:

- Modificar la carpeta “Ejecutables”, añadiendo los archivos de la nueva versión y modificando el nombre de las carpetas para reflejar cual es la última versión
- Volver a generar el índice de éste documento.
- Volver a generar el PDF de éste documento y del Manual de Usuario.
- Modificar el “Historial de versiones” de éste documento.
- Indicar los Bugs corregidos.
- Volver a generar el JavaDoc.

# Historial de versiones

| Número versión | Fecha lanzamiento | Descripción |
|---|---|---|
| - | - | Versión antigua que estaba desarrollada usando la interfaz Swing.<br>Solamente poseía las siguiente funcionalidades:<br>• Resolución de problemas de optimización.<br>◦ Muestra el paso a paso de la resolución por el método de las 2 fases.<br>◦ Muestra las soluciones. |
| 0.1 | 09/03/2015 | Usa como interfaz JavaFX. Posee las siguientes funcionalidades:<br>• Interfaz rediseñada y mejorada.<br>• Resolución de problemas de optimización<br>◦ Muestra la resolución por el método gráfico.<br>◦ Muestra el paso a paso de la resolución por el método de las 2 fases.<br>◦ Muestra las soluciones<br>• Se agregó la ventana opciones.<br>◦ Posibilidad de elegir en que lenguaje se muestra el programa (En Español o Inglés -Éste último no esta 100% traducido-).<br>• Se agregó la ventana “Acerca de”, en donde se puede encontrar información acerca del programa.<br>• Se agregó la ventana “Ayuda”, en donde se puede encontrar las guías para realizar un correcto uso del software. |

| | | |
|---|---|---|
| 0.1.1 | 13/04/2015 | ***Versión actual.***<br>• Ahora se muestra la solución optima (Valor de Z) en la ventana de las Soluciones de la Optimización.<br>• Posibilidad de elegir si a la solución optima mostrarla en Fracción o en decimales.<br>• Se cambió la forma en que se muestran las fracciones en el método toString(): si el denominador es igual a 1, directamente se muestra el número. Por lo tanto, ahora cada vez que se muestre en una celda del Tableau un número entero, va a dejar de mostrarse como fracción y pasa a mostrarse como entero.<br>• En el caso de que sea un problema no acotado, se permite la visualización en el método gráfico si éste está seleccionado.<br>• Modificación de la documentación en el código fuente.<br>• Correción de bugs. |

# Bugs corregidos

| Versión en la que se corrigió | Descripción |
|---|---|
| 0.1.1 | 1. Corrección del bug en la clase “OrdenadorListasCoordenada” que provocaba que después de sacar las coordenadas repetidas, quedaran guardadas coordenadas null, debido a que creaba un array mayor a lo que se necesitaba (con el mismo tamaño que tenía antes de quitar las repetidas) y por lo tanto llenaba los espacios sobrantes con null.<br>2. Modificador del mensaje de la excepción en el constructor de “Fraccion” donde se enviaba como parámetro un string. Si estaba mal formada el denominador, indicaba que el numerador era el que estaba mal formado. |