Simplex Educativo

Plan de Pruebas

Profesora: Maria Estrada.

Curso: Proyecto de Ingeniería de Software.

6.01.2017

Jose Fernando Molina Chacón

Yordan Jiménez Hernández

1. Índice

1. Introducción

1.1. Propósito

El plan de pruebas tiene el propósito de informar a los miembros del equipo o futuras personas que ingresen a desarrollar dentro del proyecto, acerca de las distintas pruebas desarrolladas dentro del programa para los casos de uso de la primera iteración. Es necesario indicar al lector, que el desarrollo de las pruebas fue en el lenguaje de programación Java aprovechando herramientas disponibles gratuitas para lograr la automatización.

Dentro de las secciones de este documento, se podrá informar acerca de las caracteristicas de las pruebas construidas, ya sea como ejecutarlas, resultados esperados y una breve explicación de cómo está constituida. Además se especificará el tipo que le corresponda, unitaria, de integración o sistema.

1.2. Alcanse

El plan de pruebas es destinado para comunicar acerca de los elementos de validación y verificación definidos por parte del grupo de trabajo para ratificar la funcionalidad de las características presentes en la solución de software propuesta.

A través del plan de pruebas se podrá aclarar el origen de los errores, defecto o fallas que se generan al ejecutar una prueba gracias a la trazabilidad a los componentes pertenecientes al programa y a los casos de uso.

1.3. Estratégia de pruebas

Durante la elaboración y planeamiento de las pruebas se tomó en cuenta la orientación incremental, donde una prueba unitaria forma parte o constituye una de integración, y además las pruebas de integración son parte de una prueba de sistema. Para la construcción de las pruebas se realizaron después de terminar la aplicación esto para prever cambio en interfaces de clases que implicarán cambios en las pruebas construidas.

Además para la construcción de las pruebas se realizaron gracias a herramientas para automatizar estos tipos de archivos, en

nuestro caso JUnit. Gracias a esto la ejecución de las pruebas unitarias y de integración es inmediata retornando resultados de manera fácil y brindando información de ubicación del error.

1.4. Identificación del proyecto

En la tabla siguiente, se especifican los documentos los cuales fueron tomados como apoyo para realizar el plan de pruebas:

Nombre	**Hábil**	**Utilizado**
Especificación de requerimientos del software	Sí	Sí
Documento de Visión	Sí	Sí
Diagrama de interacción	Sí	No
Diagrama de componentes	Sí	No
Diagrama de clases	Sí	Sí
Manual de usuario	No	No
Diagrama de despliegue	Sí	No

La enumeración y descripción de los casos de uso se encuentran dentro del documento de visión del proyecto.

2. Enfoque de las pruebas

En el contenido de esta sección, se definen los distintos tipos de pruebas realizadas para el software, dentro de ellas se detallan característica relevantes que el usuario debe de saber para ejecutar la pruebas e interpretar los resultados que obtuvo. Además para ejecutar cada una se utiliza información llamada “Ground truth” con lo que los resultados obtenidos en cada prueba son comparados para obtener su validez.

2.1. Pruebas por categorías

2.1.1. Pruebas unitarias

Id	PU-1
Nombre	AgregarColumnaTest
Descripción	Comprueba que cuando a una matriz de un tamaño NxM, se le agregan un número Y de columnas, las nuevas dimensiones de las columnas de la matriz sea de Nx(M+Y).
Parámetros	Los parámetros necesarios son: ● Una matriz de NxM, que contenga elementos de la clase AbstractFraccion. ● Número entero que representa la cantidad de columnas a agregar.
Resultados esperados	Una matriz con elementos tipo AbstractFraccion donde las dimensiones de esta, contenga el número de columnas más las indicadas por parámetro, que la original.
Componentes utilizados	AbstractFraccion.java SolucionadorSimplex.java en el método agregarColumna.
Casos de uso implicados	Casos número 1,2,3,4.

Id	PU-2
Nombre	AgregarNombreVariablesTest
Descripción	Agrega a un arreglo existente de elementos tipo String, una cantidad nueva de elementos, donde el prefijo de los nuevos nombres es indicado y el sufijo es la posición dentro del arreglo.
Parámetros	● Arreglo tipo String donde se añadirán los nuevos nombres. ● Número entero que representa la cantidad de elementos a agregar. ● Prefijo tipo String del nombre de los nuevos elementos .
Resultados	Arreglo con el tamaño mayor al original, donde se agregaron

esperados	la cantidad de nuevos nombres que se indicaron.
Componentes utilizados	SolucionadorSimplex.java en el método agregarNombreVariables.
Casos de uso implicados	Casos número 1,2,3,4.

Id	PU-3
Nombre	AgregarNombreWTest
Descripción	Agrega a un arreglo de elementos tipo String, al inicio de este, un nuevo elemento que contendrá el valor de "-w".
Parámetros	● Arreglo tipo String donde se añadirá al inicio el elemento "-w" .
Resultados esperados	Arreglo con el tamaño mayor al original, donde se agregó al inicio de este el elemento "-w".
Componentes utilizados	SolucionadorSimplex.java en el método agregarNombreW.
Casos de uso implicados	Casos número 1,2,3,4.

Id	PU-4
Nombre	AgregarNombresFilaTest
Descripción	Válida que los índices dentro de un arreglo de elementos tipo String, hayan sido intercambiados por un nuevo elemento donde este posee la característica de ser conformado por un sufijo y prefijo que se indican por medio de parámetro.
Parámetros	● Arreglo con elementos tipo String donde se añadirán los nombres generados. ● Arreglo con elementos tipo enteros que representan los índices de los elementos dentro del arreglo a intercambiar. ● Prefijo tipo String del nombre de los nuevos elementos . ● Número entero que indicará el sufijo de los nuevos elementos.

Resultados esperados	Arreglo de elementos tipo String donde se intercambiaron los índice por los nuevos elementos.
Componentes utilizados	SolucionadorSimplex.java en la función agregarNombresFila.
Caso de uso implicados	Casos número 1,2,3,4.

Id	PU-5
Nombre	AgregarUnoMatrizTest
Descripción	Compara si el valor de una posición dentro de una matriz de elementos tipo AbstractFraccion, fue intercambiado por un uno con valor negativo o positivo como se indique.
Parámetros	● Una matriz de NxM, que contenga elementos de la clase AbstractFraccion. ● Número entero que representa a la fila dentro de la matriz. ● Número entero que representa a la columna dentro de la matriz. ● Valor booleano que representa al signo del elemento, verdadero indica que es positivo y al contrario sería negativo.
Resultados esperados	Una matriz de NxM donde se haya intercambiado el valor correcto en la posición indicada.
Componentes utilizados	AbstractFraccion.java SolucionadorSimplex.java en el método agregarUnoMatriz.
Caso de uso implicados	Casos número 1,2,3,4.

Id	PU-6
Nombre	AgregarUnosTest
Descripción	Realiza el proceso de agregar unos positivos o negativos dentro de una matriz de elementos tipo AbstractFraccion en las filas y columnas indicadas, para luego cerciorarse que las posiciones que debieron cambiar, poseen el valor correcto.

Parámetros	● Matriz con elementos tipo AbstractFraccion donde se añadirán los unos. ● Arreglo con elementos tipo enteros que representan los índices de las filas dentro de la matriz donde se intercambiaran su valor. ● Valor booleano que representa al signo del elemento, verdadero indica que es positivo y al contrario sería negativo. ● Número entero que indicará el número de la columna donde se empezará a iterar.
Resultados esperados	Una matriz de NxM donde se haya intercambiado el valor correcto en la posición indicada.
Componentes utilizados	AbstractFraccion.java SolucionadorSimplex.java en el método agregarUnos.
Caso de uso implicados	Casos número 1,2,3,4.

Id	PU-7
Nombre	BuscarIndiceTest
Descripción	Se encarga de comparar la búsqueda de un elemento tipo String dentro de un arreglo, retorne la primera posición donde este se encuentre, sino retorne el valor de -1.
Parámetros	● Arreglo con elementos tipo String ● Cadena de texto que se buscará.
Resultados esperados	Índice dentro del arreglo donde se encuentra posicionado el elemento, si no lo encuentra retorn -1.
Componentes utilizados	SolucionadorSimplex.java en el método buscarIndice.
Caso de uso implicados	Casos número 1,2,3,4.

Id	PU-8
Nombre	ConvertirDosFasesTest
Descripción	Valida que una matriz de NxM elementos tipo AbstractFraccion, tome las característica de un problema de

	dos fases, esto agregando la primera fila que representa a la función -w y las variables artificiales dentro de esta, para luego comparar que su nueva estructura sea la correcta.
Parámetros	● Matriz con elementos tipo AbstractFraccion que se va a convertir en un problema de dos fases. ● Número entero que representa la columna donde inician las variables artificiales.
Resultados esperados	Matriz con elementos tipo AbstractFraccion donde contenga la estructura de un problema de programación lineal de dos fases;
Componentes utilizados	SolucionadorSimplex.java en el método convertirDosFases.
Caso de uso implicados	Casos número 1,2,3,4.

Id	PU-9
Nombre	CrearNombreFilaTest
Descripción	Encargada de comparar que la creación del arreglo de nombres de las filas, tenga la cantidad correcta de elementos y posea el primer elemento con el nombre indicado.
Parámetros	● Número entero que representa el tamaño del arreglo. ● Cadena de texto que indica el primer elemento del arreglo resultado.
Resultados esperados	Arreglo de elementos tipo String que contenga la cantidad de elementos correcto y el nombre del primer elemento ingresado.
Componentes utilizados	SolucionadorSimplex.java en el método crearNombreFila.
Caso de uso implicados	Casos número 1,2,3,4.

Id	PU-10
Nombre	EsAcotadoTest

Descripción	Ejecuta la validación de revisar si un arreglo de elementos tipo AbstractFraccion en las posiciones que representan a las restricción de un problema de programación lineal, brinden la característica de que el problema esta acotado o no.
Parámetros	● Arreglo con elementos tipo AbstractFraccion por generar los radios. ● Valor booleano que si es verdadero indica que es un problema de dos fases y al contrario si es falso, que indicará donde inician las restricciones.
Resultados esperados	Valor booleano que si es verdadero representa si un problema está acotado y al contrario si es falso.
Componentes utilizados	SolucionadorSimplex.java en el método esAcotado.
Caso de uso implicados	Casos número 1,2,3,4,5.

Id	PU-11
Nombre	VerificarFactibilidadTest
Descripción	Ejecuta la validación de revisar si un arreglo de elementos tipo AbstractFraccion en las posiciones que representan a las coeficientes de las variables de la función objetivo, brindan la característica de que el problema es factible o no.
Parámetros	● Arreglo con elementos tipo AbstractFraccion por comparar sus elementos.
Resultados esperados	Valor booleano que si es verdadero representa si un problema es factible y al contrario si es falso.
Componentes utilizados	SolucionadorSimplex.java en el método verificarFactibilidad.
Caso de uso implicados	Casos número 1,2,3,4.

2.1.2. Pruebas de integración

Id	PI-1
Nombre	FraccionTest
Descripción	Verifica que las operaciones o métodos definidos dentro de la clase Fraccion, donde se implique el uso o retorno del mismo tipo de dato anteriormente mencionado retornen una instancia con el valor correcto. Se integra entre la misma clase. Se ejecutan las siguientes operaciones: ● Sumar ● Restar ● Multiplicar ● Dividir ● Clonar ● HacerNegativa ● ObtenerInverso ● esCero Donde las primeras 4 operaciones reciben dos parámetros y las últimas 4 solo uno, respectivamente se definen a continuación.
Parámetros	● Fraccion con los datos de numerador y denominador definidos ● Fraccion con los datos de numerador y denominador definidos
Resultados esperados	Fraccion con los datos de numerador y denominador definidos correctamente tras aplicar las operaciones correspondientes.
Componentes utilizados	Fraccion.java
Caso de uso implicados	Casos número 1,2,3,4,5.

Id	PI-2
Nombre	ParserTest
Descripción	Ejecuta todos los métodos que conforma a la clase Parser con el fin de validar el resultado de ejecutar una cadena de texto para obtener una matriz de NxM con elementos tipo AbstractFraccion que representan a los coeficientes del problema ingresado.
Parámetros	● Cadena de texto que representa al problema de

	programación lineal.
Resultados esperados	Matriz de NxM con elementos tipo AbstractFraccion.
Componentes utilizados	IParser.java, Parser.java, SimplexParser.java, SimplexSymbol.java, sym.java
Caso de uso implicados	Casos número 1,2,3,4.

Id	PI-3
Nombre	SolucionadorSimplexCalcularRadiosTest
Descripción	Calcula los radios de una matriz de NxM elementos tipo AbstractFraccion, indicando la columna con la cual se operará para obtener un resultado con el cual comparar la validez. Para esta ejecución el “lado derecho” sería cada último elemento de cada fila.
Parámetros	● Matriz de NxM elementos tipo AbstractFraccion. ● Número entero que representa a la columna.
Resultados esperados	Arreglo de elementos tipo AbstractFraccion que representan los radios.
Componentes utilizados	AbstractFraccion.java SolucionadorSimplex.java en el método calcularRadio.
Caso de uso implicados	Casos número 1,2,3,4,5.

Id	PI-4
Nombre	SolucionadorSimplexCompletarProblemaTest
Descripción	Completa las caracteristicas de un problema de programación lineal, agregando las variables de holgura, artificiales, columnas, convirtiéndolo en un problema de dos fases, para comparar el resultado dependiendo del caso en que se encuentre. La prueba va devolver el resultado dependiendo de las caracteristicas del objeto de transferencia de datos que reciba.
Parámetros	● Objeto de transferencia de datos con las

	características del problema.
Resultados esperados	Objeto de transferencia de datos con las características del problema completado.
Componentes utilizados	AbstractFraccion.java SolucionadorSimplex.java en el método completarProblema.
Caso de uso implicados	Casos número 1,2,3,4.

Id	PI-5
Nombre	SolucionadorSimplexSiguientePasoTest
Descripción	Obtiene un objeto de transferencia de datos donde se aplicaron las operaciones de fila indicadas desde el objeto de transferencia de datos origen que se ingresó. Luego se compara las caracteristicas de ambas instancias.
Parámetros	● Objeto de transferencia de datos con los datos del paso actual.
Resultados esperados	Objeto de transferencia de datos con los datos del paso siguiente.
Componentes utilizados	AbstractFraccion.java SolucionadorSimplex.java en el método siguientePaso.
Caso de uso implicados	Casos número 1,2,3,4.

Id	PI-6
Nombre	SolucionadorSimplexSiguientesOperacionesTest
Descripción	Genera un String con las siguientes operaciones filas que se van a realizar en el problema. Dicho String es comparado para verificar que sea correcto.
Parámetros	● Objeto de transferencia de datos con los datos del paso actual.
Resultados esperados	String con las siguientes operaciones filas que se van a realizar.
Componentes	AbstractFraccion.java

utilizados	SolucionadorSimplex.java en el método siguientesOperaciones.
Caso de uso implicados	Casos número 1,2,3,4,5.

Id	PI-7
Nombre	SolucionadorSimplexSolucionarTest
Descripción	Soluciona un problema de programación lineal indicado por medio un objeto de transferencia de datos, para luego comparar el arreglo de objetos de transferencias de datos y verificar el resultado correcto.
Parámetros	● Objetos de transferencias de datos con los datos del problema.
Resultados esperados	● Arreglo de objetos de transferencia de datos que representa a la cantidad de pasos necesitados para solucionar el problema.
Componentes utilizados	AbstractFraccion.java SolucionadorSimplex.java en el método solucionar.
Caso de uso implicados	Casos número 1,2,3,4.

2.1.3. Pruebas de Sistema

Id	PS-1
Nombre	PruebaSistema1
Descripción	Prueba desde la interfaz gráfica, la ejecución de un problema no factible.
Parámetros	● String con los datos de un problema no factible.
Resultados esperados	● Reporte de pasos generado para obtener que el problema no es factible.
Componentes utilizados	PantallaPrincipal.java, PantallaPasoIntermedio.java, SimplexControlador.java, SolucionadorSimplex.java, Parser.java, Fraccion.java

Caso de uso implicados	Caso número 1,3

Id	PS-2
Nombre	PruebaSistema2
Descripción	Prueba desde la interfaz gráfica, la ejecución de un problema y obtener su solución óptima con pasos intermedios.
Parámetros	● String con los datos de un problema de programación lineal.
Resultados esperados	● Reporte de pasos generado para obtener la solución del problema.
Componentes utilizados	PantallaPrincipal.java, PantallaPasoIntermedio.java, SimplexControlador.java, SolucionadorSimplex.java, Parser.java, Fraccion.java
Caso de uso implicados	Caso número 3

Id	PS-3
Nombre	PruebaSistema3
Descripción	Prueba desde la interfaz gráfica, la ejecución de insertar una matriz y realizar pivoteos sobre ella.
Parámetros	● String con los datos de la matriz.
Resultados esperados	● Reporte de pasos generados tras pivotear con la matriz.
Componentes utilizados	PantallaPrincipal.java, PantallaPasoIntermedio.java, MatrizControlador.java, SolucionadorSimplex.java, Parser.java, Fraccion.java
Caso de uso implicados	Caso número 5,3

Id	PS-4
Nombre	PruebaSistema4
Descripción	Prueba desde la interfaz gráfica, la ejecución de un problema y obtener su solución óptima inmediata.
Parámetros	● String con los datos de un problema de programación lineal.
Resultados esperados	● Reporte de pasos generado para obtener la solución del problema.
Componentes utilizados	PantallaPrincipal.java, PantallaPasoIntermedio.java, Controlador.java, SolucionadorSimplex.java, Parser.java, Fraccion.java
Caso de uso implicados	Caso número 2

Id	PS-5
Nombre	PruebaSistema5
Descripción	Prueba desde la interfaz gráfica, la ejecución de un problema no acotado.
Parámetros	● String con los datos de un problema no acotado.
Resultados esperados	● Reporte de pasos generado para obtener que el problema no está acotado.
Componentes utilizados	PantallaPrincipal.java, PantallaPasoIntermedio.java, SimplexControlador.java, SolucionadorSimplex.java, Parser.java, Fraccion.java
Caso de uso implicados	Caso número 4,3

2.2. Otros tipos de pruebas

2.2.1. Pruebas de interfaz gráfica

Dentro de esta iteración no se realizaron pruebas de interfaz gráfica.

2.2.2. Pruebas de rendimiento

Como se estableció en el documento de especificación del software, el rendimiento no se encuentra dentro de los requerimientos del sistema, por ello no se presentan pruebas de rendimiento.

3. Requerimientos de ambiente

3.1. Hardware

Equipo	**Características**
Equipo cliente	● Procesador con mínimo un núcleo de 2.0 GHz ● Memoria ram con 1 Gb de memoria. ● También es necesario que el usuario tenga un monitor, teclado, mouse para interactuar con el programa.

3.2. Software

Equipo	**Sistema Operativo**	**Software requerido para las pruebas**
Equipo Cliente	Cualquier sistema operativo que ejecute la máquina virtual de Java.	- Librerías necesarias: ● JUnit ● Hamcrest - Java conjunto a un JDK compatble.

4. Encargados

Plan de pruebas	Fernando Molina
Casos de pruebas	Yordan Jiménez
Automatización de	Yordan Jiménez

pruebas	

5. Aprobación

	Administrador de proyectos	**Administrador de calidad**
Nombre	Fernando Molina	Yordan Jiménez
Firma		