UNIVERSIDAD DE TALCA
FACULTAD DE INGENIERÍA
ESCUELA DE INGENIERÍA CIVIL EN COMPUTACIÓN

Documento de especificación de casos de prueba

HERRAMIENTA PARA LA OPTIMIZACIÓN DE REDES DE DISTRIBUCIÓN DE AGUA POTABLE

EQUIPO DE DESARROLLO:

Nombre	**Rol**	**Contacto**
Gabriel Sanhueza Fuentes	Administrador, Analista, Diseñador, Implementador y Tester.	gsanhueza15@alumnos .utalca.cl

CONTRAPARTE:

Nombre	**Rol**	**Contacto**
Jimmy H. Gutiérrez-Bahamondes	Cliente/Profesor guía	
Jimmy H. Gutiérrez-Bahamondes	Cliente/Profesor co-guía	

HISTORIAL DE CAMBIOS

Version	**Fecha**	**Modificaciones**
0.1	07/09/2019	
1	15/06/2020	

TABLA DE CONTENIDOS

página

1. Casos de pruebas automatizadas

En este capitulo se presenta la especificación formal de los casos de prueba automatizados.

1.1. *Pump*

En esta sección se presentan las pruebas realizadas para la clase *Pump*.

Test ID:	AT001
Titulo:	Envio de parámetro invalido al método setProperty de la clase *Pump* cuando se usa la clave *PumpProperty.HEAD*.
Característica:	Validar parametro recibido por el método *setProperty*.
Objetivo:	Comprobar que si se pasa un parametro como valor de un tipo distinto a *String*, cuando se usa la clave *PumpProperty.HEAD*, el método lanza una excepción.
Configuración:	Instancia de la clase *Pump* inicializada.
Datos de prueba:	Clave: *PumpProperty.HEAD* Value: Un objeto diferente a un *String*
Acciones de prueba:	1. Inicializar instancia. 2. Pasar una instancia de un objeto distinto de un *String* al método cuando se usa la llave *PumpProperty.HEAD*.
Resultados esperados:	Excepción indicando que el tipo de instancia pasada no es valida.

Test ID:	AT002
Titulo:	Envio de parámetro invalido al método setProperty de la clase *Pump* cuando se usa la clave *PumpProperty.PATTERN*.
Característica:	Validar parametro recibido por el método *setProperty*.
Objetivo:	Comprobar que si se pasa un parametro como valor de un tipo distinto a *String*, cuando se usa la clave *PumpProperty.PATTERN*, el método lanza una excepción.
Configuración:	Instancia de la clase *Pump* inicializada.
Datos de prueba:	Clave: *PumpProperty.PATTERN* Value: Un objeto diferente a un *String*
Acciones de prueba:	1. Inicializar instancia. 2. Pasar una instancia de un objeto distinto de un *String* al método cuando se usa la llave *PumpProperty.HEAD*.
Resultados esperados:	Excepción indicando que el tipo de instancia pasada no es valida.

Test ID:	AT003
Titulo:	Envio de parámetro invalido al método setProperty de la clase *Pump* cuando se usa la clave *PumpProperty.SPEED*.
Característica:	Validar parametro recibido por el método *setProperty*.
Objetivo:	Comprobar que si se pasa un parametro como valor de un tipo distinto a *Double*, cuando se usa la clave *PumpProperty.SPEED*, el método lanza una excepción.
Configuración:	Instancia de la clase *Pump* inicializada.
Datos de prueba:	Clave: *PumpProperty.SPEED* Value: Un *Integer* o alguna instancia de otro objeto distinto de Double.
Acciones de prueba:	1. Inicializar instancia. 2. Pasar una instancia de un objeto distinto de un *Double* al método cuando se usa la llave *PumpProperty.SPEED*.
Resultados esperados:	Excepción indicando que el tipo de instancia pasada no es valida.

Test ID:	AT004
Titulo:	Envio de parámetro invalido al método setProperty de la clase *Pump* cuando se usa la clave *PumpProperty.POWER*.
Característica:	Validar parametro recibido por el método *setProperty*.
Objetivo:	Comprobar que si se pasa un parametro como valor de un tipo distinto a *Double*, cuando se usa la clave *PumpProperty.POWER*, el método lanza una excepción.
Configuración:	Instancia de la clase *Pump* inicializada.
Datos de prueba:	Clave: *PumpProperty.POWER* Value: Un *Integer* o alguna instancia de otro objeto distinto de Double.
Acciones de prueba:	1. Inicializar instancia. 2. Pasar una instancia de un objeto distinto de un *Double* al método cuando se usa la llave *PumpProperty.POWER*.
Resultados esperados:	Excepción indicando que el tipo de instancia pasada no es valida.

Test ID:	AT005
Titulo:	Envio de parámetro válido al método setProperty de la clase *Pump* cuando se usa la clave *PumpProperty.HEAD*.
Característica:	Validar parametro recibido por el método *setProperty*.
Objetivo:	Comprobar que si se pasa un parametro de tipo *String* como valor, cuando se usa la clave *PumpProperty.HEAD*, el método finaliza sin error.
Configuración:	Instancia de la clase *Pump* inicializada.
Datos de prueba:	Clave: *PumpProperty.HEAD* Value: Un *String* no vacío.
Acciones de prueba:	1. Inicializar instancia. 2. Pasar una instancia de un *String* al método cuando se usa la llave *PumpProperty.HEAD*.
Resultados esperados:	Método ejecutado sin errores.

Test ID:	AT006
Titulo:	Envio de parámetro válido al método setProperty de la clase *Pump* cuando se usa la clave *PumpProperty.PATTERN*.
Característica:	Validar parametro recibido por el método *setProperty*.
Objetivo:	Comprobar que si se pasa un parametro de tipo *String* como valor, cuando se usa la clave *PumpProperty.PATTERN*, el método finaliza sin error.
Configuración:	Instancia de la clase *Pump* inicializada.
Datos de prueba:	Clave: *PumpProperty.PATTERN* Value: Un *String* no vacío.
Acciones de prueba:	1. Inicializar instancia. 2. Pasar una instancia de un *String* al método cuando se usa la llave *PumpProperty.HEAD*.
Resultados esperados:	Método ejecutado sin errores.

Test ID:	AT007
Titulo:	Envio de parámetro válido al método setProperty de la clase *Pump* cuando se usa la clave *PumpProperty.SPEED*.
Característica:	Validar parametro recibido por el método *setProperty*.
Objetivo:	Comprobar que si se pasa un parametro de tipo *Double* como valor, cuando se usa la clave *PumpProperty.SPEED*, el método finaliza sin error.
Configuración:	Instancia de la clase *Pump* inicializada.
Datos de prueba:	Clave: *PumpProperty.SPEED* Value: Un valor *Double*.
Acciones de prueba:	1. Inicializar instancia. 2. Pasar una instancia de un *Double* al método cuando se usa la llave *PumpProperty.SPEED*.
Resultados esperados:	Método ejecutado sin errores.

Test ID:	AT008
Titulo:	Envio de parámetro válido al método setProperty de la clase *Pump* cuando se usa la clave *PumpProperty.POWER*.
Característica:	Validar parametro recibido por el método *setProperty*.
Objetivo:	Comprobar que si se pasa un parametro de tipo *Double* como valor, cuando se usa la clave *PumpProperty.POWER*, el método finaliza sin error.
Configuración:	Instancia de la clase *Pump* inicializada.
Datos de prueba:	Clave: *PumpProperty.POWER* Value: Un valor *Double*.
Acciones de prueba:	1. Inicializar instancia. 2. Pasar una instancia de un *Double* al método cuando se usa la llave *PumpProperty.SPEED*.
Resultados esperados:	Método ejecutado sin errores.

1.2. *GeneticAlgorithm*

En esta sección se especifican las pruebas realizadas para la clase *GeneticAlgorithm*.

Test ID:	AT009
Titulo:	Número máximo de evaluaciones no válido.
Característica:	Validar parámetro *maxEvaluations*.
Objetivo:	Validar que el parámetro *maxEvaluations* no sea negativo. Si el parámetro es negativo debe lanzarse una excepción.
Configuración:	Instancia de la clase *GeneticAlgorithm* inicializada.
Datos de prueba:	Cualquier entero menor que 0.
Acciones de prueba:	Llamar al método *setMaxEvaluations* con un argumento negativo.
Resultados esperados:	Una excepción

Test ID:	AT010
Titulo:	Número máximo de evaluaciones sin mejora no válido.
Característica:	Validar parámetro *maxNumberOfEvaluationsWithoutImprovement*.
Objetivo:	Validar que el parámetro *maxNumberOfEvaluationsWithoutImprovement* no sea negativo. Si el parámetro es negativo debe lanzarse una excepción.
Configuración:	Instancia de la clase *GeneticAlgorithm* inicializada.
Datos de prueba:	Cualquier entero menor que 0.
Acciones de prueba:	Llamar al método *setMaxNumberOfEvaluationsWithoutImprovement* con un argumento negativo.
Resultados esperados:	Una excepción

Test ID:	AT011
Titulo:	Deshabilitar número máximo de evaluaciones sin mejoras cuando se modifica el numero máximo de evaluaciones.
Característica:	Validar parámetro *maxEvaluations*.
Objetivo:	Validar que al modificar el parámetro *maxEvaluations* el parámetro *maxNumberOfEvaluationsWithoutImprovement* cambie a 0 (0 indica que esta deshabilitado).
Configuración:	Instancia de la clase *GeneticAlgorithm* inicializada.
Datos de prueba:	Cualquier entero positivo.
Acciones de prueba:	Llamar al método *setMaxEvaluations* con un entero positivo mayor a 0.
Resultados esperados:	Parámetro *maxNumberOfEvaluationsWithoutImprovement* igual a 0

Test ID:	AT012
Titulo:	Deshabilitar número máximo de evaluaciones cuando se modifica el numero máximo de evaluaciones sin mejoras.
Característica:	Validar parámetro *maxNumberOfEvaluationsWithoutImprovement*.
Objetivo:	Validar que al modificar el parámetro *maxNumberOfEvaluationsWithoutImprovement* el parámetro *maxNumberOfEvaluationsWithoutImprovement* cambie a 0 (0 indica que esta deshabilitado).
Configuración:	Instancia de la clase *GeneticAlgorithm* inicializada.
Datos de prueba:	Cualquier entero positivo.
Acciones de prueba:	Llamar al método *setMaxNumberOfEvaluationsWithoutImprovement* con un entero positivo mayor a 0.
Resultados esperados:	Parámetro *maxEvaluations* igual a 0

1.3. *IntegerRangeRandomMutation*

En esta sección se especifican las pruebas realizadas sobre la clase *IntegerRangeRandomMutation*.

Test ID:	AT013
Titulo:	Probabilidad de mutación no valida.
Característica:	Validar parámetro *mutationProbability*.
Objetivo:	Validar que el parámetro *mutationProbability* no sea negativo. Si el parámetro es negativo debe lanzarse una excepción.
Configuración:	Instancia de la clase *IntegerRangeRandomMutation* inicializada.
Datos de prueba:	Cualquier entero menor que 0.
Acciones de prueba:	Crear instancia de la clase *IntegerRangeRandomMutation* pasando como argumento para el parámetro *mutationProbability* un valor negativo.
Resultados esperados:	Una excepción

Test ID:	AT014
Titulo:	Rango no valido.
Característica:	Validar parámetro *range*.
Objetivo:	Validar que el parámetro *range* no sea negativo. Si el parámetro es negativo debe lanzarse una excepción.
Configuración:	Instancia de la clase *IntegerRangeRandomMutation* inicializada.
Datos de prueba:	Cualquier entero menor que 0.
Acciones de prueba:	Crear instancia de la clase *IntegerRangeRandomMutation* pasando como argumento para el parámetro *range* un valor negativo.
Resultados esperados:	Una excepción

Test ID:	AT015
Titulo:	Mutar variables.
Característica:	Mutar variables cuando el número generado por el *randomGenerator* es menor que la probabilidad de mutación.
Objetivo:	Validar que la mutación suceda cuando un número generado aleatoriamente sea menor que la probabilidad de mutación.
Configuración:	Solucion con variables preestablecidas. Operador *IntegerRangeRandomMutation* inicializado.
Datos de prueba:	*mutationProbability*: 0.3 *range*: 2 *solution*: [0 ,2, 4] *randomGenerator*: [0.2, 0.2, 0.4] *boundedRandomGenerator*: [2, 0, 5]
Acciones de prueba:	Pasar la solución al método *execute*.
Resultados esperados:	Las variables en la solución despues de realizar la mutación debe ser [2, 0, 4]

Test ID:	AT016
Titulo:	No mutar variables.
Característica:	No mutar variables cuando los números generado por el *randomGenerator* sean mayores que la probabilidad de mutación.
Objetivo:	Validar que la mutación no sucede si el *randomGenerator* devuelve valores mayores a *mutationProbability*.
Configuración:	Solucion con variables preestablecidas. Operador *IntegerRangeRandomMutation* inicializado.
Datos de prueba:	*mutationProbability*: 0.3 *range*: 2 *solution*: [0 ,2, 4] *randomGenerator*: [0.5, 0.4, 0.4] *boundedRandomGenerator*: [2, 0, 5]
Acciones de prueba:	Pasar la solución al método *execute*.
Resultados esperados:	Las variables en la solución despues de realizar la llamada al método *execute* debe ser los valores originales [0, 2, 4].

Test ID:	AT017
Titulo:	No mutar variables cuando no hay un rango de mutación.
Característica:	No mutar variables cuando el rango de las variables a generar es 0.
Objetivo:	Validar que la mutación no sucede si el *range* tiene el valor 0.
Configuración:	Solucion con variables preestablecidas. Operador *IntegerRangeRandomMutation* inicializado.
Datos de prueba:	*mutationProbability*: 0.3 *range*: 0 *solution*: [0 ,2, 4] *randomGenerator*: [0.1, 0.2, 0.3]
Acciones de prueba:	Pasar la solución al método *execute*.
Resultados esperados:	Las variables en la solución despues de realizar la llamada al método *execute* debe ser los valores originales [0, 2, 4].

1.4. *ReflectionUtils*

En esta sección se especifican las pruebas realizadas sobre la clase *ReflectionUtils*.

Durante esta sección nos referiremos a cualquier implementación de las interfaz *Registrable* o sus subinterfaces unicamente como *Registrable*. Es por ello, que cuando se mencione implementar *Registrable* se refiere a implementar ya sea *SingleObjective-Registrable* o *MultiobjectiveRegistrable*.

Test ID:	AT018
Titulo:	Nombre del problema en anotación *@NewProblem*.
Característica:	Obtener nombre del problema de la anotación *@NewProblem*.
Objetivo:	Validar que el método *getNameOfProblem* retorna el nombre asignado en la anotación *@NewProblem*.
Configuración:	Implementación de *Registrable* con la anotación *@NewProblem* en su constructor.
Datos de prueba:	Nombre del problema: "Test"
Acciones de prueba:	Pasar el objeto *Class<?>* que hace referencia al tipo *Registrable* al método *getNameOfProblem*.
Resultados esperados:	"Test"

Test ID:	AT019
Titulo:	Nombre del algoritmo en anotación *@NewProblem*.
Característica:	Obtener nombre del algoritmo de la anotación *@NewProblem*.
Objetivo:	Validar que el método *getNameOfAlgorithm* retorna el nombre asignado en la anotación *@NewProblem*.
Configuración:	Implementación *Registrable* con la anotación *@NewProblem* en su constructor.
Datos de prueba:	Nombre del algoritmo: "NSGAII"
Acciones de prueba:	Pasar el objeto *Class<?>* que hace referencia al tipo *Registrable* al método *getNameOfAlgorithm*.
Resultados esperados:	"NSGAII"

Test ID:	AT020
Titulo:	*Registrable* sin anotaciones.
Característica:	Validar las anotaciones y el tipo de parámetros utilizados en el constructor de la clase *Registrable*.
Objetivo:	Validar que si en el constructor público no tiene ni la anotación *@NewProblem* ni la anotación *Parameters* se lanza una excepción.
Configuración:	Implementación de *Registrable* sin anotaciones.
Datos de prueba:	Objeto *Class<?>* que referencia al tipo *Registrable*.
Acciones de prueba:	Pasar el objeto *Class* que hace referencia al tipo *Registrable* al método *validateRegistrableProblem*.
Resultados esperados:	Una excepción.

Test ID:	AT021
Titulo:	*Registrable* con la anotación *@NewProblem*.
Característica:	Validar las anotaciones y el tipo de parámetros utilizados en el constructor de la clase *Registrable*.
Objetivo:	Validar si en el constructor público tiene ni la anotación *@NewProblem*
Configuración:	Implementación de *Registrable* con la anotación *@NewProblem*.
Datos de prueba:	Objeto *Class<?>* que referencia al tipo *Registrable*.
Acciones de prueba:	Pasar el objeto *Class* que hace referencia al tipo *Registrable* al método *validateRegistrableProblem*.
Resultados esperados:	Método ejecutado sin errores.

Test ID:	AT022
Titulo:	*Registrable* cuyo constructor recibe los parámetros en el orden correcto dependiendo de su tipo.
Característica:	Validar las anotaciones y el tipo de parámetros utilizados en el constructor de la clase *Registrable*.
Objetivo:	Validar que el constructor público con las anotaciones recibe los parametros en el siguiente orden: *Object*, *File*, *(int |Integer |double |Double)*.
Configuración:	Implementación de *Registrable* con las anotaciones *@NewProblem* y *@Parameters*, y con los parámetros recibidos en el orden esperado.
Datos de prueba:	Objeto *Class<?>* que referencia al tipo *Registrable*.
Acciones de prueba:	Pasar el objeto *Class* que hace referencia al tipo *Registrable* al método *validateRegistrableProblem*.
Resultados esperados:	Método ejecutado sin errores.

Test ID:	AT023
Titulo:	*Registrable* cuyo constructor recibe los parámetros en un orden incorrecto.
Característica:	Validar las anotaciones y el tipo de parámetros utilizados en el constructor de la clase *Registrable*.
Objetivo:	Validar que el constructor público con las anotaciones recibe los parametros en un orden distinto a: *Object*, *File*, *(int |Integer |double |Double)*.
Configuración:	Implementación de *Registrable* con las anotaciones *@NewProblem* y *@Parameters*, y con los parámetros recibidos en un orden distinto al especificado.
Datos de prueba:	Objeto *Class<?>* que referencia al tipo *Registrable*.
Acciones de prueba:	Pasar el objeto *Class* que hace referencia al tipo *Registrable* al método *validateRegistrableProblem*.
Resultados esperados:	Una excepción.

Test ID:	AT024
Titulo:	*Registrable* cuyo constructor recibe una cantidad de parámetros distintas a la esperada de acuerdo a la anotación *@Parameters.*
Característica:	Validar las anotaciones y el tipo de parámetros utilizados en el constructor de la clase *Registrable.*
Objetivo:	Validar que si el constructor público recibe más parámetros que los indicados en *@Parameters*, lanza una excepción.
Configuración:	Implementación de *Registrable* con las anotaciones *@NewProblem* y *@Parameters*, y con un parámetro extra en el constructor al indicado en la anotación *@Parameters.*
Datos de prueba:	Objeto *Class<?>* que referencia al tipo *Registrable.*
Acciones de prueba:	Pasar el objeto *Class* que hace referencia al tipo *Registrable* al método *validateRegistrableProblem.*
Resultados esperados:	Una excepción.

Test ID:	AT025
Titulo:	*Registrable* con una anotación extra en *@Parameters*.
Característica:	Validar las anotaciones y el tipo de parámetros utilizados en el constructor de la clase *Registrable*.
Objetivo:	Validar que si *@Parameters* tiene mas anotaciones que el número de parámetros en el constructor, se lanza una excepción.
Configuración:	Implementación de *Registrable* con las anotaciones *@NewProblem* y *@Parameters*, ésta última con una anotación extra.
Datos de prueba:	Objeto *Class<?>* que referencia al tipo *Registrable*.
Acciones de prueba:	Pasar el objeto *Class* que hace referencia al tipo *Registrable* al método *validateRegistrableProblem*.
Resultados esperados:	Una excepción.

Test ID:	AT026
Titulo:	*Registrable* con dos constructores.
Característica:	Validar las anotaciones y el tipo de parámetros utilizados en el constructor de la clase *Registrable*.
Objetivo:	Validar que si *Registrable* tiene dos constructores se lanza una excepción.
Configuración:	Implementación de *Registrable* con dos constructor, uno de ellos con la anotaciones *@NewProblem* y *@Parameters*.
Datos de prueba:	Objeto *Class<?>* que referencia al tipo *Registrable*.
Acciones de prueba:	Pasar el objeto *Class* que hace referencia al tipo *Registrable* al método *validateRegistrableProblem*.
Resultados esperados:	Una excepción.

Test ID:	AT027
Titulo:	Id del grupo usado por *@NumberToggleInput* secuencial.
Característica:	Validar las anotaciones y el tipo de parámetros utilizados en el constructor de la clase *Registrable*.
Objetivo:	Validar que si *Registrable*, en el elemento *numberToggle* de la anotación *@Parameters* recibe las anotaciones *@NumberToggleInput* de manera secuencial no se lanza una excepción. Con secuencial se refiere a que *@NumberToggleInput* con el mismo *groupID* deben estar juntos.
Configuración:	Implementación de *Registrable* las anotaciones *@NewProblem* y *@Parameters*. El elemento *numberToggle* de *@Parameters* tiene las anotaciones *@NumberToggleInput* con el mismo *groupID* juntos.
Datos de prueba:	Objeto *Class<?>* que referencia al tipo *Registrable*.
Acciones de prueba:	Pasar el objeto *Class* que hace referencia al tipo *Registrable* al método *validateRegistrableProblem*.
Resultados esperados:	Método termina sin error.

Test ID:	AT028
Titulo:	Id del grupo usado por *@NumberToggleInput* no secuencial.
Característica:	Validar las anotaciones y el tipo de parámetros utilizados en el constructor de la clase *Registrable*.
Objetivo:	Validar que si *Registrable*, en el elemento *numberToggle* de la anotación *@Parameters* recibe las anotaciones *@NumberToggleInput* de manera no secuencial se lanza una excepción. Con secuencial se refiere a que *@NumberToggleInput* con el mismo *groupID* deben estar juntos.
Configuración:	Implementación de *Registrable* las anotaciones *@NewProblem* y *@Parameters*. El elemento *numberToggle* de *@Parameters* no tiene las anotaciones *@NumberToggleInput* con el mismo *groupID* juntos.
Datos de prueba:	Objeto *Class<?>* que referencia al tipo *Registrable*.
Acciones de prueba:	Pasar el objeto *Class* que hace referencia al tipo *Registrable* al método *validateRegistrableProblem*.
Resultados esperados:	Método termina sin error.

Test ID:	AT029
Titulo:	*Registrable* con parámetros en el constructor que no están definidos en la anotación *@Parameters*.
Característica:	Validar las anotaciones y el tipo de parámetros utilizados en el constructor de la clase *Registrable*.
Objetivo:	Validar que si el constructor *Registrable* tiene un parámetro no correspondiente al indicado en la anotación *@Parameters* se lanza una excepción.
Configuración:	Implementación de *Registrable* las anotaciones *@NewProblem* y *@Parameters*. El constructor indica que recibirá un *File* mientras que la anotación *@Parameters* indica que en esa posición debería recibirse un *Object*, el cual hace referencia a un operador de la metaheurística.
Datos de prueba:	Objeto *Class<?>* que referencia al tipo *Registrable*.
Acciones de prueba:	Pasar el objeto *Class* que hace referencia al tipo *Registrable* al método *validateRegistrableProblem*.
Resultados esperados:	Una excepción.

1.5. *ResultSimutation*

En esta sección se especifica los test automatizados realizados sobre la clase *ResultSimulation*.

Test ID:	AT030
Titulo:	*timeInSeconds* guardado correctamente.
Característica:	Validar que la clase abstracta guarda el parametro *timeInSeconds* correctamente.
Objetivo:	Validar que al pasar el tiempo en segundos a la superclase *ResultSimulation* se guarda el tiempo correctamente en el campo *timeInSeconds*.
Configuración:	Clase que hereda de *ResultSimulation* cuyo constructor delega el parámetro *timeInSeconds* para que sea guardado en la superclase.
Datos de prueba:	*timeInSeconds*: 72000
Acciones de prueba:	Pasar al constructor de la superclase abstracta *ResultSimulation* el tiempo en segundo de la simulación.
Resultados esperados:	*timeInSeconds* igual a 72000.

Test ID:	AT031
Titulo:	*String* que representa el valor de *timeInSeconds* en formato HH:mm:ss.
Característica:	Validar el método *getTimeString*.
Objetivo:	Validar que al pasar el tiempo en segundos a la superclase *ResultSimulation* y llamar al método *getTimeString* se retorna un *String* con la hora en formato HH:mm:ss.
Configuración:	Clase que hereda de *ResultSimulation* cuyo constructor delega el parámetro *timeInSeconds* para que sea guardado en la superclase.
Datos de prueba:	*timeInSeconds*: 72000, 0, 1, 1000, 86159, 86399.
Acciones de prueba:	Pasar al constructor de la superclase abstracta *ResultSimulation* el tiempo en segundo de la simulación.
Resultados esperados:	Los *String* "20:00:00", "00:00:00", "00:00:01", "00:30:00", "23:55:59","23:59:59".

Test ID:	AT032
Titulo:	*timeInSeconds* fuera del rango de simulación.
Característica:	Validar el intervalo permitido para *timeInSeconds*
Objetivo:	Validar que si *timeInSeconds* esta fuera del rango de las 24 horas se lanza una excepción.
Configuración:	Clase que hereda de *ResultSimulation* cuyo constructor delega el parámetro *timeInSeconds* para que sea guardado en la superclase.
Datos de prueba:	*timeInSeconds*: -2, -1, 86400(24:00:00), 86401 (24:00:01).
Acciones de prueba:	Pasar al constructor de la superclase abstracta *ResultSimulation* el tiempo en segundo de la simulación.
Resultados esperados:	Una excepción para cualquiera de los valores de *timeInSeconds* usados.

1.6. *JsonSimpleReader*

En esta sección se especifican los casos de prueba para la clase *JsonSimpleReader*.

Test ID:	AT033
Titulo:	Leer un entero (*int*) desde un json.
Característica:	Leer valores desde un json.
Objetivo:	Validar que el método *getInt* retorna un *int* cuando se lee un número desde una propiedad en json.
Configuración:	Instancia de *JsonSimpleReader* inicializado y listo para leer.

Datos de prueba:	Un json con los siguientes valores:
	`{`
	`"int": 5,`
	`"double": 2.5,`
	`"ints": [0,1,2,3,4,5],`
	`"doubles": [0.1,0.2,2.3,2.5,2.5],`
	`"doubleMatrix":`
	`[[0.2,0.3,0.4],[1.2,1.3,1.5]],`
	`"intMatrix": [[0,1,2],[3,4,5]],`
	`"string": "A simple string",`
	`"boolean": true`
	`}`

Acciones de prueba:	Llamar al método *getInt*.
Resultados esperados:	El entero leido desde el json con clave "*int*".

Test ID:	AT034
Titulo:	Leer un numero decimal (*double*) desde un json.
Característica:	Leer valores desde un json.
Objetivo:	Validar que el método *getDouble* retorna un *double* cuando se lee un número desde una propiedad en json.
Configuración:	Instancia de *JsonSimpleReader* inicializado y listo para leer.

Datos de prueba:	Un json con los siguientes valores:
	`{`
	`"int": 5,`
	`"double": 2.5,`
	`"ints": [0,1,2,3,4,5],`
	`"doubles": [0.1,0.2,2.3,2.5,2.5],`
	`"doubleMatrix":`
	`[[0.2,0.3,0.4],[1.2,1.3,1.5]],`
	`"intMatrix": [[0,1,2],[3,4,5]],`
	`"string": "A simple string",`
	`"boolean": true`
	`}`

Acciones de prueba:	Llamar al método *getDouble*.
Resultados esperados:	El double leido desde el json con clave "*double*".

Test ID:	AT035
Titulo:	Leer un valor booleano (*boolean*) desde un json.
Característica:	Leer valores desde un json.
Objetivo:	Validar que el método *getBoolean* retorna un *boolean* cuando se lee una propiedad booleana desde el json.
Configuración:	Instancia de *JsonSimpleReader* inicializado y listo para leer.

Datos de prueba:	Un json con los siguientes valores:
	`{`
	`"int": 5,`
	`"double": 2.5,`
	`"ints": [0,1,2,3,4,5],`
	`"doubles": [0.1,0.2,2.3,2.5,2.5],`
	`"doubleMatrix":`
	`[[0.2,0.3,0.4],[1.2,1.3,1.5]],`
	`"intMatrix": [[0,1,2],[3,4,5]],`
	`"string": "A simple string",`
	`"boolean": true`
	`}`

Acciones de prueba:	Llamar al método *getBoolean*.
Resultados esperados:	El boleano leido desde el json con clave "*boolean*".

Test ID:	AT036
Titulo:	Leer un arreglo de enteros (*int*[]) desde un json.
Característica:	Leer valores desde un json.
Objetivo:	Validar que el método *getIntegerArray* retorna un *int*[] cuando se lee un arreglo desde el json.
Configuración:	Instancia de *JsonSimpleReader* inicializado y listo para leer.

Datos de prueba:	Un json con los siguientes valores:
	`{`
	`"int": 5,`
	`"double": 2.5,`
	`"ints": [0,1,2,3,4,5],`
	`"doubles": [0.1,0.2,2.3,2.5,2.5],`
	`"doubleMatrix":`
	`[[0.2,0.3,0.4],[1.2,1.3,1.5]],`
	`"intMatrix": [[0,1,2],[3,4,5]],`
	`"string": "A simple string",`
	`"boolean": true`
	`}`

Acciones de prueba:	Llamar al método *getIntegerArray*.
Resultados esperados:	El arreglo de enteros desde el json con clave "*ints*".

Test ID:	AT037
Titulo:	Leer un arreglo de números decimales (*double*[]) desde un json.
Característica:	Leer valores desde un json.
Objetivo:	Validar que el método *getDoubleArray* retorna un *double*[] cuando se lee un arreglo desde el json.
Configuración:	Instancia de *JsonSimpleReader* inicializado y listo para leer.

Datos de prueba:	Un json con los siguientes valores:
	`{`
	`"int": 5,`
	`"double": 2.5,`
	`"ints": [0,1,2,3,4,5],`
	`"doubles": [0.1,0.2,2.3,2.5,2.5],`
	`"doubleMatrix":`
	`[[0.2,0.3,0.4],[1.2,1.3,1.5]],`
	`"intMatrix": [[0,1,2],[3,4,5]],`
	`"string": "A simple string",`
	`"boolean": true`
	`}`

Acciones de prueba:	Llamar al método *getDoubleArray*.
Resultados esperados:	El arreglo de valores decimales leido desde el json con clave "*doubles*".

Test ID:	AT038
Titulo:	Leer un matriz de enteros (*int*[][]) desde un json.
Característica:	Leer valores desde un json.
Objetivo:	Validar que el método *getIntegerMatrix* retorna un *int*[][] cuando se lee un matriz desde el json.
Configuración:	Instancia de *JsonSimpleReader* inicializado y listo para leer.

Datos de prueba:	Un json con los siguientes valores:
	`{`
	`"int": 5,`
	`"double": 2.5,`
	`"ints": [0,1,2,3,4,5],`
	`"doubles": [0.1,0.2,2.3,2.5,2.5],`
	`"doubleMatrix":`
	`[[0.2,0.3,0.4],[1.2,1.3,1.5]],`
	`"intMatrix": [[0,1,2],[3,4,5]],`
	`"string": "A simple string",`
	`"boolean": true`
	`}`

Acciones de prueba:	Llamar al método *getIntegerMatrix*.
Resultados esperados:	La matriz de valores enteros leido desde el json con clave "*intMatrix*".

Test ID:	AT039
Titulo:	Leer un matriz de decimales (*double*[][]) desde un json.
Característica:	Leer valores desde un json.
Objetivo:	Validar que el método *getDoubleMatrix* retorna un *double*[][] cuando se lee un matriz desde el json.
Configuración:	Instancia de *JsonSimpleReader* inicializado y listo para leer.

Datos de prueba:	Un json con los siguientes valores:
	`{`
	`"int": 5,`
	`"double": 2.5,`
	`"ints": [0,1,2,3,4,5],`
	`"doubles": [0.1,0.2,2.3,2.5,2.5],`
	`"doubleMatrix":`
	`[[0.2,0.3,0.4],[1.2,1.3,1.5]],`
	`"intMatrix": [[0,1,2],[3,4,5]],`
	`"string": "A simple string",`
	`"boolean": true`
	`}`

Acciones de prueba:	Llamar al método *getDoubleMatrix*.
Resultados esperados:	La matriz de valores decimales leido desde el json con clave "*doubleMatrix*".

2. Casos de pruebas manuales

En este capitulo se presenta la especificación formal de los casos de prueba manuales llevados que son llevados a cabo sobre el software.

Test ID:	MT001
Titulo:	Visualización de la red.
Característica:	Mostrar visualización de la red.
Objetivo:	Confirmar que la red puede ser leida desde un archivo ".inp" y ser visualizada en la applicación.
Configuración:	El equipo tiene la applicación JHawanetFramework lista para ejecutar.
Datos de prueba:	inp: hanoi-Frankenstein.inp
Acciones de prueba:	1. Abrir JHawanetFramework 2. Cargar archivo de red.
Resultados esperados:	El sistema muestra la red leida desde un archivo inp gráficamente en la applicación.

Test ID:	MT002
Titulo:	Optimización monoobjetivo realizada completamente.
Característica:	Realizar simulación monoobjetivo.
Objetivo:	Confirmar que se puede llevar a cabo la resolución del problema *PipeOptimizing* sobre la red abierta.
Configuración:	El equipo tiene la applicación JHawanetFramework lista para ejecutar.
Datos de prueba:	independentRun = 10 Archivo inp: hanoi-Frankenstein.inp Archivo gama: hanoiHW.Gama
Acciones de prueba:	1. Abrir JHawanetFramework 2. Cargar archivo de red. 3. Seleccionar el problema *PipeOptimizing del menú.* 4. Configurar el problema usando la ventana de configuración. 5. Realizar la optimización.
Resultados esperados:	Al terminar la optimización el sistema muestra una interfaz con las soluciones generadas por la optimización. Debe haber tantas soluciones como el numero de configuraciones independientes (*independentRun*).

Test ID:	MT003
Titulo:	Optimización monoobjetivo cancelada.
Característica:	Realizar simulación monoobjetivo.
Objetivo:	Confirmar que se puede llevar a cabo la resolución del problema *PipeOptimizing* sobre la red abierta y que se puede cancelar el proceso cerrando la ventana o pulsando cancelar.
Configuración:	El equipo tiene la applicación JHawanetFramework lista para ejecutar.
Datos de prueba:	Archivo inp: hanoi-Frankenstein.inp Archivo gama: hanoiHW.Gama
Acciones de prueba:	1. Abrir JHawanetFramework 2. Cargar archivo de red. 3. Seleccionar el problema *PipeOptimizing del menú*. 4. Configurar el problema usando la ventana de configuración. 5. Realizar la optimización.
Resultados esperados:	Al cancelar la busqueda de soluciones la ventana de estado indica que la optimización a sido detenida.

Test ID:	MT004
Titulo:	Optimización multiobjetivo realizada completamente.
Característica:	Realizar simulación multiobjetivo.
Objetivo:	Confirmar que se puede llevar a cabo la resolución del problema *PumpingScheduling* sobre la red abierta.
Configuración:	El equipo tiene la applicación JHawanetFramework lista para ejecutar.
Datos de prueba:	Archivo inp: Vanzyl.inp Archivo gama: VanzylConfiguration.json
Acciones de prueba:	1. Abrir JHawanetFramework 2. Cargar archivo de red. 3. Seleccionar el problema *PumpingScheduling del menú.* 4. Seleccionar el operador NSGAII. 5. Configurar el problema usando la ventana de configuración. 6. Realizar la optimización.
Resultados esperados:	Al terminar la optimización el sistema muestra una interfaz con las soluciones generadas por la optimización. Estas soluciones corresponden a la frontera de pareto del Experimento.

Test ID:	MT005
Titulo:	Optimización multiobjetivo cancelada.
Característica:	Realizar simulación multiobjetivo.
Objetivo:	Confirmar que se puede llevar a cabo la resolución del problema *PumpingScheduling* sobre la red abierta y que se puede cancelar el proceso cerrando la ventana o pulsando cancelar.
Configuración:	El equipo tiene la applicación JHawanetFramework lista para ejecutar.
Datos de prueba:	Archivo inp: Vanzyl.inp Archivo gama: VanzylConfiguration.json
Acciones de prueba:	1. Abrir JHawanetFramework 2. Cargar archivo de red. 3. Seleccionar el problema *PumpingScheduling del menú.* 4. Seleccionar el operador NSGAII. 5. Configurar el problema usando la ventana de configuración. 6. Realizar la optimización.
Resultados esperados:	Al cancelar la busqueda de soluciones la ventana de estado indica que la optimización a sido detenida.

Test ID:	MT006
Titulo:	Ver soluciones de los problemas monoobjetivos gráficamente a medida que se ejecuta la optimización.
Característica:	Visualizar gráficamente las soluciones.
Objetivo:	Comprobar que a medida que algoritmo va generando soluciones estas pueden ser visualizadas. El grafico es Objetivo vs Numero de generaciones.
Configuración:	El equipo tiene la applicación JHawanetFramework lista para ejecutar.
Datos de prueba:	Archivo inp: hanoi-Frankenstein.inp Archivo gama: hanoiHW.Gama
Acciones de prueba:	1. Abrir JHawanetFramework 2. Cargar archivo de red. 3. Escoger el problema monoobjetivo 4. Configurar el problema monoobjetivo y ejecutar 5. Ir a la ventana del gráfico.
Resultados esperados:	Un gráfico de dos dimensiones en el que se muestre los resultados de las soluciones generadas por cada generación de cada una de las ejecuciones independientes.

Test ID:	MT007
Titulo:	Ver soluciones del problema multiobjetivos, con dos objetivos, gráficamente a medida que se ejecuta la optimización.
Característica:	Visualizar gráficamente las soluciones
Objetivo:	Comprobar que a medida que algoritmo va generando soluciones estas pueden ser visualizadas. El grafico es Objetivo1 vs Objetivo2.
Configuración:	El equipo tiene la applicación JHawanetFramework lista para ejecutar.
Datos de prueba:	Archivo inp: Vanzyl.inp Archivo gama: VanzylConfiguration.json
Acciones de prueba:	1. Abrir JHawanetFramework 2. Cargar archivo de red. 3. Escoger el problema monoobjetivo 4. Configurar el problema monoobjetivo y ejecutar 5. Ir a la ventana del gráfico.
Resultados esperados:	Un gráfico de dos dimensiones en el que se muestre los resultados de las soluciones generadas por cada generación de cada una de las ejecuciones independientes.

Test ID:	MT008
Titulo:	Guardar la gráfica de las soluciones del problema mono-objetivo como png.
Característica:	Guardar gráfica de soluciones.
Objetivo:	Confirmar que se puede guardar el gráfico de soluciones como un png.
Configuración:	El equipo tiene la applicación JHawanetFramework lista para ejecutar.
Datos de prueba:	Archivo inp: hanoi-Frankenstein.inp Archivo gama: hanoiHW.Gama
Acciones de prueba:	1. Abrir JHawanetFramework 2. Cargar archivo de red. 3. Escoger el problema monoobjetivo 4. Configurar el problema monoobjetivo y ejecutar 5. Ir a la ventana del gráfico. 6. Pulsar el boton para guardar una captura del gráfico. 7. Configurar donde guardar la captura.
Resultados esperados:	Generación de un archivo png en el equipo.

Test ID:	MT009
Titulo:	Guardar la gráfica de soluciones de problemas de 2 objetivos como png.
Característica:	Guardar gráfica de soluciones.
Objetivo:	Confirmar que se puede guardar el gráfico de soluciones como un png.
Configuración:	El equipo tiene la applicación JHawanetFramework lista para ejecutar.
Datos de prueba:	Archivo inp: Vanzyl.inp Archivo gama: VanzylConfiguration.json
Acciones de prueba:	1. Abrir JHawanetFramework 2. Cargar archivo de red. 3. Escoger el problema monoobjetivo 4. Configurar el problema monoobjetivo y ejecutar 5. Ir a la ventana del gráfico. 6. Pulsar el boton para guardar una captura del gráfico. 7. Configurar donde guardar la captura.
Resultados esperados:	Generación de un archivo png en el equipo.

Test ID:	MT010
Titulo:	Guardar las solucion seleccionada como inp.
Característica:	Guardar resultados de la optimización.
Objetivo:	Confirmar que se puede exportar los resultados de la solución sobre el archivo de red (inp).
Configuración:	El equipo tiene la applicación JHawanetFramework lista para ejecutar.
Datos de prueba:	Archivo inp: hanoi-Frankenstein.inp Archivo gama: hanoiHW.Gama
Acciones de prueba:	1. Abrir JHawanetFramework 2. Cargar archivo de red. 3. Escoger el problema monoobjetivo 4. Configurar el problema monoobjetivo y ejecutar 5. Esperar que la ejecucion finalice. 6. Seleccionar una solución. 7. Pulsar el boton guardar como inp. 8. Configurar donde guardar el archivo.
Resultados esperados:	Generación del archivo de configuración de la red (inp) con la solución aplicada.

Test ID:	MT011
Titulo:	Guardar las soluciones en archivos tsv.
Característica:	Guardar resultados de la optimización.
Objetivo:	Confirmar que se puede exportar las soluciones a dos archivos tsv. Uno para las variables y el otro para los objetivos.
Configuración:	El equipo tiene la applicación JHawanetFramework lista para ejecutar.
Datos de prueba:	Archivo inp: hanoi-Frankenstein.inp Archivo gama: hanoiHW.Gama
Acciones de prueba:	1. Abrir JHawanetFramework 2. Cargar archivo de red. 3. Escoger el problema monoobjetivo 4. Configurar el problema monoobjetivo y ejecutar 5. Esperar que la ejecucion finalice. 6. Pulsar el boton guardar tabla. 7. Configurar donde guardar los archivo.
Resultados esperados:	Generación del 2 archivos .tsv. Uno con el prefijo FUN_ y otro con el prefijo VAR_. El archivo FUN contiene el valor de los objetivos de las soluciones. El archivo VAR contiene las variables de la soluciones.

Test ID:	MT012
Titulo:	Guardar las soluciones en un excel.
Característica:	Guardar resultados de la optimización.
Objetivo:	Confirmar que se puede exportar la tabla de resultados completas a un excel.
Configuración:	El equipo tiene la applicación JHawanetFramework lista para ejecutar.
Datos de prueba:	Archivo inp: hanoi-Frankenstein.inp Archivo gama: hanoiHW.Gama
Acciones de prueba:	1. Abrir JHawanetFramework 2. Cargar archivo de red. 3. Escoger el problema monoobjetivo 4. Configurar el problema monoobjetivo y ejecutar 5. Esperar que la ejecucion finalice. 6. Pulsar el boton guardar tabla como excel. 7. Configurar donde guardar el archivo.
Resultados esperados:	Un archivo excel con los mismos datos que la tabla de resultados de la applicación.

Test ID:	MT013
Titulo:	Simulación hidráulica sobre red de un solo tiempo.
Característica:	Realizar simulación con configuración de archivo de red.
Objetivo:	Confirmar que se puede realizar la simulación utilizando los valores del archivo de red.
Configuración:	El equipo tiene la applicación JHawanetFramework lista para ejecutar.
Datos de prueba:	Archivo inp: hanoi-Frankenstein.inp Archivo gama: hanoiHW.Gama
Acciones de prueba:	1. Abrir JHawanetFramework 2. Cargar archivo de red. 3. Pulsar botón *Execute* de la ventana principal.
Resultados esperados:	Ejecución realizada sin ningun error.

Test ID:	MT014
Titulo:	Ver resultados de la simulación hidráulica de un solo tiempo
Característica:	Realizar simulación con configuración de archivo de red.
Objetivo:	Confirmar que se puede visualizar los resultados de la simulación realizada.
Configuración:	El equipo tiene la applicación JHawanetFramework lista para ejecutar.
Datos de prueba:	Archivo inp: hanoi-Frankenstein.inp Archivo gama: hanoiHW.Gama
Acciones de prueba:	1. Abrir JHawanetFramework 2. Cargar archivo de red. 3. Pulsar botón *Execute* de la ventana principal. 4. Pulsar botón *Results* de la ventana principal.
Resultados esperados:	Una interfaz que permite seleccionar si se quieren ver los resultados para los enlaces o los nodos.

Test ID:	MT015
Titulo:	Simulación hidráulica sobre red de mas de un tiempo.
Característica:	Realizar simulación con configuración de archivo de red.
Objetivo:	Confirmar que se puede realizar la simulación utilizando los valores del archivo de red.
Configuración:	El equipo tiene la applicación JHawanetFramework lista para ejecutar.
Datos de prueba:	Archivo inp: Vanzyl.inp Archivo gama: VanzylConfiguration.json
Acciones de prueba:	1. Abrir JHawanetFramework 2. Cargar archivo de red. 3. Pulsar boton *Execute* de la ventana principal.
Resultados esperados:	Ejecución realizada sin ningun error.

Test ID:	MT016
Titulo:	Ver resultados de la simulación hidráulica de mas de un tiempo
Característica:	Realizar simulación con configuración de archivo de red.
Objetivo:	Confirmar que se puede visualizar los resultados de la simulación realizada.
Configuración:	El equipo tiene la applicación JHawanetFramework lista para ejecutar.
Datos de prueba:	Archivo inp: Vanzyl.inp Archivo gama: VanzylConfiguration.json
Acciones de prueba:	1. Abrir JHawanetFramework 2. Cargar archivo de red. 3. Pulsar botón *Execute* de la ventana principal. 4. Pulsar botón *Results* de la ventana principal.
Resultados esperados:	Una interfaz que permite seleccionar si se quieren ver los resultados para los enlaces o los nodos. Adicionalmente, permite escoger el tiempo de simulación y listar los resultados para todos los tiempos de un elemento de la red específico.