Trabajo práctico taller de programación

Grupo 02

Integrantes: Ramiro Cardelli, Joaquin Martinez, Nicolas Herrera y Josefina Frasca Ponce

Índice

1. Aspectos a considerar

- Las clases abstractas *Cliente* y *Usuario* no se testean, justamente por ser abstractas.
    - *Usuario* se testa en sus hijos *Admin y Cliente (pero Cliente es abstracta).*
    - *Usuario* está testeada en su hijo *Empleador* (consideramos que no es necesario testear en su otro hijo *EmpleadoPretenso).*
    - *Cliente* se encuentra testeada en *Empleador* y *EmpleadoPretenso.*

! Una consideración importante, es que la documentación de *Usuario* no es consistente con el código, donde se afirma que esta clase es abstracta, a pesar de que en el código no lo es.

- Los constructores se testean indirectamente en los getters y setters.

2. Casos de prueba de pruebas unitarias caja negra

Clase: Admin
Escenario: Admin sin parámetros

setPassword (string) y getPassword()

Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia
Password (string)	password=null	no	jdoc	1
	password!=null	si	jdoc	2

Número de prueba	Datos de entrada	Valor	Salida Esperada	Clases
1	password	"078hd90"	seteo correcto (que se verifica con el get)	2

setUsserName(string) y getUsserName()

Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia
UsserName (string)	UsserName=null	no	jdoc	1
	UsserName!=null	si	jdoc	2

Número de prueba	Datos de entrada	Valor	Salida Esperada	Clases
1	UsserName	"juanceto01"	seteo correcto (que se verifica con el get)	2

setRealName(string) y getRealName()

Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia
RealName (string)	RealName=null	no	jdoc	1
	RealName!=null	si	jdoc	2

Número de prueba	Datos de entrada	Valor	Salida Esperada	Clases
1	RealName	"Juan Aceto"	seteo correcto (que se verifica con el get)	2

setTelefono(string) y getTelefono()

Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia
Telefono (string)	Telefono=null	no	jdoc	1
	Telefono!=null	si	jdoc	2

Número de prueba	Datos de entrada	Valor	Salida Esperada	Clases
1	Telefono	"2234555645"	seteo correcto (que se verifica con el get)	2

toString(): no puede testearse ya que no se cuenta con la documentacion necesaria para establecer la salida esperada.

Nota: los setters y getters están juntos ya que al testar el set indirectamente también se testea el get.

Clase: ClientePuntaje

Escenario nro. 1: ClientePuntaje con parámetros puntaje y cliente con un objeto tipo EmpleadoPretenso o Empleado.

Nota: En este escenario se testean los getters (que devuelvan el atributo que fue pasado por constructor como parámetro). No se testea el ToString ya que no se cuenta con la documentación necesaria para establecer la salida esperada.

getCliente()

Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia
		si	jdoc	1

Número de prueba	Datos de entrada	Valor	Salida Esperada	Clases
1			referencia la objeto de tipo EmpleadoPretenso pasado en constructor	1

getPuntaje()

Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia
		si	jdoc	1

Número de prueba	Datos de entrada	Valor	Salida Esperada	Clases
1			30.0 (valor de puntaje pasado por constructor)	1

CompareTo: se supone un comportamiento igual al compareTo de la interfaz Comparable (no documentada la sobreescritura del método). De esta forma se compara el puntaje del cliente con el recibido por parámetro, si el puntaje del cliente es mayor al cliente recibido por parámetro retorna 1, si es igual 0 y si es menor -1.

Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia
Objeto	Objeto = ClientePuntaje (puntaje mayor al obj cliente)	si	JDoc I. Comparable	1
	Objeto = ClientePuntaje (puntaje igual al obj cliente)	si	JDoc I. Comparable	2
	Objeto = ClientePuntaje (puntaje menor al obj cliente)	si	JDoc I. Comparable	3
	Objeto = null	si	JDoc I. Comparable	4
	Objeto !=ClientePuntaje	si	JDoc I. Comparable	5

Número de prueba	Datos de entrada	Valor	Salida Esperada	Clases
1	Objeto	ClientePuntaje (puntaje=40)	-1	1
2	Objeto	ClientePuntaje (puntaje=30)	0	2
3	Objeto	ClientePuntaje (puntaje=20)	1	3
4	Objeto	null	throws NullPointerException	4
5	Objeto	Empleador	throws ClassCastException	5

Escenario nro 2: ClientePuntaje sin parámetros por constructor

Nota: Aquí se testean los setters y los getters indirectamente (para corroborar el correcto funcionamiento de los setters). No se testea el ToString ya que no se cuenta con la documentación necesaria para establecer la salida esperada.

setPuntaje()

Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia
Puntaje (double)	Puntaje ∈ R	si	jdoc	1

Número de prueba	Datos de entrada	Valor	Salida Esperada	Clases
1	Puntaje	30	seteo correcto (que se verifica con el get)	1

setCliente()

Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia
Cliente	Objeto de tipo Cliente	si	jdoc	1
	Objeto distinto de tipo Cliente	no	jdoc	2

Número de prueba	Datos de entrada	Valor	Salida Esperada	Clases
1	Cliente	objeto Empleador con atributos vacios	seteo correcto (que se verifica con el get)	1

Clase: Empleador

Escenario nro 1: Empleador con parámetros

Nota: En este escenario, testeamos directamente los getters - que devuelvan los atributos pasados como parámetro en el constructor.

getUssername()

Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia
-	-	si	jdoc	1

Número de prueba	Datos de entrada	Valor	Salida Esperada	Clases
1	-	-	"juancito".- pasado por constructor	1

getPassword()

Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia
-	-	si	jdoc	1

Número de prueba	Datos de entrada	Valor	Salida Esperada	Clases
1	-	-	"qwerty123".- pasada por constructor	1

getRealName()

Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia
-	-	si	jdoc	1

Número de prueba	Datos de entrada	Valor	Salida Esperada	Clases
1	-	-	"+54 9 223 666 - 1234"	1

getRubro()

Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia
-	-	si	jdoc	1

Número de prueba	Datos de entrada	Valor	Salida Esperada	Clases
1	-	-	util.Constantes.SALUD	1

getTipoPersona()

Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia
-	-	si	jdoc	1

Número de prueba	Datos de entrada	Valor	Salida Esperada	Clases
1	-	-	util.Constantes.JURIDICA	1

getTipoPersona()

Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia
-	-	si	jdoc	1

Número de prueba	Datos de entrada	Valor	Salida Esperada	Clases
1	-	-	util.Constantes.JURIDICA	1

calculaComision()

Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia
Ticket	Objeto de tipo Ticket	si	jdoc	1
Ticket	Objeto != de tipo Ticket	no	jdoc	2

Número de prueba	Datos de entrada	Valor	Salida Esperada	Clases
1	Ticket	Ticket(HOME_OFFICE, 80000, JORNADA_COMPLETA, JUNIOR, EXP_MEDIA, TERCIARIOS)	48000	1

Escenario nro 2: Empleador sin parámetros

Nota: En este escenario, testeamos directamente los setters e indirectamente los getters - usando los atributos setteados.

setRubro()

Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia
Rubro	Rubro ∈ util.Constantes (rubros)	si	jdoc	1
	Rubro ∉ util.Constantes(rubros)	no	jdoc	2

Número de prueba	Datos de entrada	Valor	Salida Esperada	Clases
1	Rubro	util.Constantes.SALUD	seteo correcto (que se verifica con el get)	1

setTipoPersona()

Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia
Rubro	Rubro ∈ util.Constantes (tipoPersona)	si	jdoc	1
	Rubro ∉ util.Constantes(tipoPersona)	no	jdoc	2

Número de prueba	Datos de entrada	Valor	Salida Esperada	Clases
1	TipoPersona	util.Constantes.JURIDICA	seteo correcto (que se verifica con el get)	1

setTelefono

Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia
telefono (String)	telefono!=null	si	jdoc	1
	telefono=null	no	jdoc	2

Número de prueba	Datos de entrada	Valor	Salida Esperada	Clases
1	telefono	"+5492236661234"	seteo correcto (que se verifica con el get)	1

setRealName

Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia
realName(String)	realName!=null	si	jdoc	1
	realName=null	no	jdoc	2

Número de prueba	Datos de entrada	Valor	Salida Esperada	Clases
1	realName	"Juan Cito"	seteo correcto (que se verifica con el get)	1

setPassword

Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia
password(String)	password!=null	si	jdoc	1
	password=null	no	jdoc	2

Número de prueba	Datos de entrada	Valor	Salida Esperada	Clases
1	password	"qwerty123"	seteo correcto (que se verifica con el get)	1

setUsserName()

Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia
usserName(String)	usserName!=null	si	jdoc	1
	usserName=null	no	jdoc	2

Número de prueba	Datos de entrada	Valor	Salida Esperada	Clases
1	usserName	"juancito"	seteo correcto (que se verifica con el get)	1

calculaComision()

Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia
Ticket	Objeto de tipo Ticket	si	jdoc	1
Ticket	Objeto != de tipo Ticket	no	jdoc	2

Número de prueba	Datos de entrada	Valor	Salida Esperada	Clases
1	Ticket	Ticket(HOME_OFFICE, 80000, JORNADA_COMPLETA, JUNIOR, EXP_MEDIA, TERCIARIOS)	48000	1

Podriamos obviar este testeo ya que ya fue realizado en el escenario *EmpleadorSinParametros,* pero no está de más realizarlo.

Clase: EmpleadoPretenso

Escenario nro. 1: EmpleadoPretenso sin parámetros.- testeamos los setters (y los getters indirectamente).

setApellido

Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia
apellido(String)	apellido!=null	si	jdoc	1
	apellido=null	no	jdoc	2

Número de prueba	Datos de entrada	Valor	Salida Esperada	Clases
1	apellido	"Rodriguez"	seteo correcto (que se verifica con el get)	1

setEdad

Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia
apellido(String)	edad>0	si	jdoc	1
	edad<0	no	jdoc	2

Número de prueba	Datos de entrada	Valor	Salida Esperada	Clases
1	edad	36	seteo correcto (que se verifica con el get)	1

Escenario nro. 2: EmpleadoPretenso con parámetros.- testeamos los getters (que la salida concuerde con los parámetros pasados por constructor).
Consideraciones: Testeamos en este escenario los métodos de la clase Cliente. El método ToString no será testeado ya que no disponemos de la documentación correspondiente.

getApellido

Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia
-	-	si	jdoc	1

Número de prueba	Datos de entrada	Valor	Salida Esperada	Clases
1	-	-	"Rodriguez" .- se corresponde con parametro apellido del constructor	1

getEdad

Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia
-	-	si	jdoc	1

Número de prueba	Datos de entrada	Valor	Salida Esperada	Clases
1	-	-	36 .- se corresponde con parametro edad del constructor	1

calculaComision()

Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia
Ticket	Objeto de tipo Ticket	si	jdoc	1
Ticket	Objeto != de tipo Ticket	no	jdoc	2

Número de prueba	Datos de entrada	Valor	Salida Esperada	Clases
1	Ticket	Ticket(HOME_OFFICE, 80000, JORNADA_COMPLETA, JUNIOR, EXP_MEDIA, TERCIARIOS)	64000	1

Métodos de la clase Cliente
Testeando los getters se testean indirectamente los setters. (excepto el setListaPostulante que nos parecio interesante documentar el testeo ya que hay 2 posibles clases de equivalencia)

getCandidato()

Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia
-	-	si	jdoc	1

Número de prueba	Datos de entrada	Valor	Salida Esperada	Clases
1	-	-	Candidato = referencia a Empleador seteado	1

setListaPostulantes()

Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia	
ArrayList	ArrayList de tamaño >= 1	si	jdoc	1	
ArrayList	ArrayList de tamaño = 0	si	jdoc	2	

Número de prueba	Datos de entrada	Valor	Salida Esperada	Clases
1	ArrayList con 1 elemento	ArrayList con Empleador("juancito", "qwerty123", "Juan Cito", "+54 9 223 666-1234", util.Constantes.SALUD, util.Constantes.JURIDICA) y puntaje=10	ArrayList no vacio coincidente con el testeado	1
2	ArrayList vacio	ArrayList = " "	ArrayList vacio coincidente con el seteado	2

getListaPostulantes()

Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia	
-	-	si	jdoc	1	

Número de prueba	Datos de entrada	Valor	Salida Esperada	Clases
1	-	-	ArrayList coincidente con el seteado	1

getPuntaje()

Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia
-	-	si	jdoc	1

Número de prueba	Datos de entrada	Valor	Salida Esperada	Clases
1	-	-	puntaje = 28(seteado)	1

getTicket()

Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia
-	-	si	jdoc	1

Número de prueba	Datos de entrada	Valor	Salida Esperada	Clases
1	-	-	Ticekt = referencia a Ticket seteado	1

Clase: Contratación

Escenario nro 1: Contratación con parámetros

En este escenario testeamos los getters (que correspondan con los valores pasados por constructor)

getEmpleado()

Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia
		si	jdoc	1

Número de prueba	Datos de entrada	Valor	Salida Esperada	Clases
1			referencia la objeto de tipo EmpleadoPretenso pasado en constructor	1

getEmpleador()

Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia
		si	jdoc	1

Número de prueba	Datos de entrada	Valor	Salida Esperada	Clases
1			referencia la objeto de tipo Empleador pasado en constructor	1

getFecha()				
Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia
		si	jdoc	1

Número de prueba	Datos de entrada	Valor	Salida Esperada	Clases
1			referencia la objeto Fecha generado en constructor (lo testeamos asegurando que sea != null)	1

Escenario nro 2: Contratación sin parámetros
En este escenario testeamos los setters (y los getters indirectamente).

setEmpleado()				
Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia
empleado	obj tipo EmpleadoPretenso	si	jdoc	1
	obj NO tipo EmpleadoPretenso	no	jdoc	2

Número de prueba	Datos de entrada	Valor	Salida Esperada	Clases
1	empleado	objeto EmpleadoPretenso con atributos vacios	seteo correcto (que se verifica con el get)	1

setEmpleador()				
Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia
empleador	obj tipo Empleador	si	jdoc	1
	obj NO tipo Empleador	no	jdoc	2

Número de prueba	Datos de entrada	Valor	Salida Esperada	Clases
1	empleador	objeto Empleador con atributos vacios	seteo correcto (que se verifica con el get)	1

setfecha()				
Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia
fecha	obj tipo GregorianCalendar	si	jdoc	1
	obj NO tipo GregorianCalendar	no	jdoc	2

Número de prueba	Datos de entrada	Valor	Salida Esperada	Clases
1	fecha	objeto GregorianCalendar vacio	seteo correcto (que se verifica con el get)	1

Nota: En ambos escenarios no se testea el método ToString ya que no disponemos de la documentación correspondiente.

Clase: Ticket

Nota: En los escenarios nro 1 y nro 2 solamente testeamos los setters y getters ya que los demás métodos se testearán en los escenarios siguientes.

Escenario nro 1: Ticket con parámetros
Testeamos los getters-. que los atributos coincidan con los parámetros pasados por constructor.

getEstudios

Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia
-	-	si	jdoc	1

Número de prueba	Datos de entrada	Valor	Salida Esperada	Clases
1	-	-	util.Constantes.PRIMARIOS	1

getExperiencia

Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia
-	-	si	jdoc	1

Número de prueba	Datos de entrada	Valor	Salida Esperada	Clases
1	-	-	util.Constantes.EXP_MEDIA	1

getJornada

Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia
-	-	si	jdoc	1

Número de prueba	Datos de entrada	Valor	Salida Esperada	Clases
1	-	-	util.Constantes.JORNADA_COMPLETA	1

getLocacion

Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia
-	-	si	jdoc	1

Número de prueba	Datos de entrada	Valor	Salida Esperada	Clases
1	-	-	util.Constantes.PRESENCIAL	1

getPuesto

Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia
-	-	si	jdoc	1

Número de prueba	Datos de entrada	Valor	Salida Esperada	Clases
1	-	-	util.Constantes.JUNIOR	1

getRemuneracion

Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia
-	-	si	jdoc	1

Número de prueba	Datos de entrada	Valor	Salida Esperada	Clases
1	-	-	10000	1

Escenario nro 2: Ticket sin parámetros

Testeamos los setters (e indirectamente los getters)

setExperiencia

Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia
Experiencia	!= null	si	jdoc	1
	= tipo esperado dentro de Constantes	si	jdoc	2
	= null	no	jdoc	3
	!= tipo esperado dentro de Constantes	no	jdoc	4

Número de prueba	Datos de entrada	Valor	Salida Esperada	Clases
1	experiencia	util.Constantes.EXP_MEDIA	setteo correcto testeado con el get	1,2

setEstudios

Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia
Estudios	!= null	si	jdoc	1
	= tipo esperado dentro de Constantes	si	jdoc	2
	= null	no	jdoc	3
	!= tipo esperado dentro de Constantes	no	jdoc	4

Número de prueba	Datos de entrada	Valor	Salida Esperada	Clases
1	estudios	util.Constantes.PRIMARIOS	setteo correcto testeado con el get	1,2

setJornada

Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia
Jornada	!= null	si	jdoc	1
	= tipo esperado dentro de Constantes	si	jdoc	2
	= null	no	jdoc	3
	!= tipo esperado dentro de Constantes	no	jdoc	4

setLocacion

Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia
Locacion	!= null	si	jdoc	1
	= tipo esperado dentro de Constantes	si	jdoc	2
	= null	no	jdoc	3
	!= tipo esperado dentro de Constantes	no	jdoc	4

Número de prueba	Datos de entrada	Valor	Salida Esperada	Clases
1	locacion	util.Constantes.PRESENCIAL	setteo correcto testeado con el get	1,2

setPuesto

Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia
puesto	!= null	si	jdoc	1
	= tipo esperado dentro de Constantes	si	jdoc	2
	= null	no	jdoc	3
	!= tipo esperado dentro de Constantes	no	jdoc	4

Número de prueba	Datos de entrada	Valor	Salida Esperada	Clases
1	puesto	util.Constantes.JUNIOR	setteo correcto testeado con el get	1,2

setRemuneracion

Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia
remuneracion	!= null	si	jdoc	1
	>0	si	jdoc	2
	= null	no	jdoc	3
	<0	no	jdoc	4

Número de prueba	Datos de entrada	Valor	Salida Esperada	Clases
1	remuneracion	10000	setteo correcto testeado con el get	1,2

Nota: En los siguientes escenarios (3 a 12), no se testean los getters y setters ya que han sido testeados previamente y consideramos innecesaria la repetición. En ellos, solo se testean los métodos de comparación entre tickets, de esta forma el escenario se estructura como Ticket XY, que representa un escenario de ticket con atributos X y el cual en sus métodos recibe un ticket a comparar con parámetros Y. Este x e y son número que representan los parámetros de un ticket, de la forma descrita a continuación:

1)Ticket("Home_Office",999,"Jornada_Media","Junior","Exp_Nada","Primarios").
2)Ticket("Presencial",1001,"Jornada_Completa","Senior","Exp_Media","Secundarios").
3) Ticket ("Indistinto",2001,"Jornada_Extendida","Management","Mucha","Terciarios").

De esta forma, si bien no se cubren todas las combinaciones de atributos posibles (resultaría muy extenso e innecesario), se logran cubrir todos los posibles cálculos de comparación posibles entre tickets.

Escenario nro 3: Ticket11

getComparacionLocacion(Ticket)

Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia
Ticket	Ticket!=null	si	jdoc	1
	Ticket=null	no	jdoc	2

Número de prueba	Datos de entrada	Valor	Salida Esperada	Clases
1	Ticket	Ticket(parametros tipo 1)	1	1

getComparacionRemuneracion()

Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia
Ticket	Ticket!=null	si	jdoc	1
	Ticket=null	no	jdoc	2

Número de prueba	Datos de entrada	Valor	Salida Esperada	Clases
1	Ticket	Ticket(parametros tipo 1)	1	1

getComparacionJornada()

Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia
Ticket	Ticket!=null	si	jdoc	1
	Ticket=null	no	jdoc	2

Número de prueba	Datos de entrada	Valor	Salida Esperada	Clases
1	Ticket	Ticket(parametros tipo 1)	1	1

getComparacionPuesto()

Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia
Ticket	Ticket!=null	si	jdoc	1
	Ticket=null	no	jdoc	2

Número de prueba	Datos de entrada	Valor	Salida Esperada	Clases
1	Ticket	Ticket(parametros tipo 1)	1	1

getComparacionExperiencia()

Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia
Ticket	Ticket!=null	si	jdoc	1
	Ticket=null	no	jdoc	2

Número de prueba	Datos de entrada	Valor	Salida Esperada	Clases
1	Ticket	Ticket(parametros tipo 1)	1	1

getComparacionEstudios()

Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia
Ticket	Ticket!=null	si	jdoc	1
	Ticket=null	no	jdoc	2

Número de prueba	Datos de entrada	Valor	Salida Esperada	Clases
1	Ticket	Ticket(parametros tipo 1)	1	1

getComparacionTotal()

Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia
Ticket	Ticket!=null	si	jdoc	1
	Ticket=null	no	jdoc	2

Número de prueba	Datos de entrada	Valor	Salida Esperada	Clases
1	Ticket	Ticket (parametros tipo 1)	6	1

Escenario nro 4: Ticket12

getComparacionLocacion(Ticket)

Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia
Ticket	Ticket!=null	si	jdoc	1
	Ticket=null	no	jdoc	2

Número de prueba	Datos de entrada	Valor	Salida Esperada	Clases
1	Ticket	Ticket(parametros tipo 2)	-1	1

getComparacionRemuneracion()

Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia
Ticket	Ticket!=null	si	jdoc	1
	Ticket=null	no	jdoc	2

Número de prueba	Datos de entrada	Valor	Salida Esperada	Clases
1	Ticket	Ticket(parametros tipo 2)	-0,5	1

getComparacionJornada()

Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia
Ticket	Ticket!=null	si	jdoc	1
	Ticket=null	no	jdoc	2

Número de prueba	Datos de entrada	Valor	Salida Esperada	Clases
1	Ticket	Ticket(parametros tipo 2)	-0,5	1

getComparacionPuesto()

Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia
Ticket	Ticket!=null	si	jdoc	1
	Ticket=null	no	jdoc	2

Número de prueba	Datos de entrada	Valor	Salida Esperada	Clases
1	Ticket	Ticket(parametros tipo 2)	-0,5	1

getComparacionExperiencia()

Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia
Ticket	Ticket!=null	si	jdoc	1
	Ticket=null	no	jdoc	2

Número de prueba	Datos de entrada	Valor	Salida Esperada	Clases
1	Ticket	Ticket(parametros tipo 2)	1,5	1

getComparacionEstudios()

Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia
Ticket	Ticket!=null	si	jdoc	1
	Ticket=null	no	jdoc	2

Número de prueba	Datos de entrada	Valor	Salida Esperada	Clases
1	Ticket	Ticket(parametros tipo 2)	1,5	1

getComparacionTotal()

Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia
Ticket	Ticket!=null	si	jdoc	1
	Ticket=null	no	jdoc	2

Número de prueba	Datos de entrada	Valor	Salida Esperada	Clases
1	Ticket	Ticket(parametros tipo 2)	0,5	1

Escenario nro 5: Ticket13

getComparacionLocacion(Ticket)

Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia
Ticket	Ticket!=null	si	jdoc	1
	Ticket=null	no	jdoc	2

Número de prueba	Datos de entrada	Valor	Salida Esperada	Clases
1	Ticket	Ticket(parametros tipo 3)	1	1

getComparacionRemuneracion()

Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia
Ticket	Ticket!=null	si	jdoc	1
	Ticket=null	no	jdoc	2

Número de prueba	Datos de entrada	Valor	Salida Esperada	Clases
1	Ticket	Ticket(parametros tipo 3)	-1	1

getComparacionJornada()

Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia
Ticket	Ticket!=null	si	jdoc	1
	Ticket=null	no	jdoc	2

Número de prueba	Datos de entrada	Valor	Salida Esperada	Clases
1	Ticket	Ticket(parametros tipo 3)	-1	1

getComparacionPuesto()

Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia
Ticket	Ticket!=null	si	jdoc	1
	Ticket=null	no	jdoc	2

Número de prueba	Datos de entrada	Valor	Salida Esperada	Clases
1	Ticket	Ticket(parametros tipo 3)	-1	1

getComparacionExperiencia()

Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia
Ticket	Ticket!=null	si	jdoc	1
	Ticket=null	no	jdoc	2

Número de prueba	Datos de entrada	Valor	Salida Esperada	Clases
1	Ticket	Ticket(parametros tipo 3)	2	1

getComparacionEstudios()

Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia
Ticket	Ticket!=null	si	jdoc	1
	Ticket=null	no	jdoc	2

Número de prueba	Datos de entrada	Valor	Salida Esperada	Clases
1	Ticket	Ticket(parametros tipo 3)	2	1

getComparacionTotal()

Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia
Ticket	Ticket!=null	si	jdoc	1
	Ticket=null	no	jdoc	2

Número de prueba	Datos de entrada	Valor	Salida Esperada	Clases
1	Ticket	Ticket(parametros tipo 3)	0,5	1

Escenario nro 6: Ticket21

getComparacionLocacion(Ticket)

Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia
Ticket	Ticket!=null	si	jdoc	1
	Ticket=null	no	jdoc	2

Número de prueba	Datos de entrada	Valor	Salida Esperada	Clases
1	Ticket	Ticket(parametros tipo 1)	-1	1

getComparacionRemuneracion()

Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia
Ticket	Ticket!=null	si	jdoc	1
	Ticket=null	no	jdoc	2

Número de prueba	Datos de entrada	Valor	Salida Esperada	Clases
1	Ticket	Ticket(parametros tipo 1)	1	1

getComparacionJornada()

Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia
Ticket	Ticket!=null	si	jdoc	1
	Ticket=null	no	jdoc	2

Número de prueba	Datos de entrada	Valor	Salida Esperada	Clases
1	Ticket	Ticket(parametros tipo 1)	-0,5	1

getComparacionPuesto()

Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia
Ticket	Ticket!=null	si	jdoc	1
	Ticket=null	no	jdoc	2

Número de prueba	Datos de entrada	Valor	Salida Esperada	Clases
1	Ticket	Ticket(parametros tipo 1)	-0,5	1

getComparacionExperiencia()

Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia
Ticket	Ticket!=null	si	jdoc	1
	Ticket=null	no	jdoc	2

Número de prueba	Datos de entrada	Valor	Salida Esperada	Clases
1	Ticket	Ticket(parametros tipo 1)	-0,5	1

getComparacionEstudios()

Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia
Ticket	Ticket!=null	si	jdoc	1
	Ticket=null	no	jdoc	2

Número de prueba	Datos de entrada	Valor	Salida Esperada	Clases
1	Ticket	Ticket(parametros tipo 1)	-0,5	1

getComparacionTotal()

Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia
Ticket	Ticket!=null	si	jdoc	1
	Ticket=null	no	jdoc	2

Número de prueba	Datos de entrada	Valor	Salida Esperada	Clases
1	Ticket	Ticket (parametros tipo 1)	-2	1

Escenario nro 7: Ticket22

getComparacionLocacion(Ticket)

Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia
Ticket	Ticket!=null	si	jdoc	1
	Ticket=null	no	jdoc	2

Número de prueba	Datos de entrada	Valor	Salida Esperada	Clases
1	Ticket	Ticket(parametros tipo 2)	1	1

getComparacionRemuneracion()

Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia
Ticket	Ticket!=null	si	jdoc	1
	Ticket=null	no	jdoc	2

Número de prueba	Datos de entrada	Valor	Salida Esperada	Clases
1	Ticket	Ticket(parametros tipo 2)	1	1

getComparacionJornada()

Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia
Ticket	Ticket!=null	si	jdoc	1
	Ticket=null	no	jdoc	2

Número de prueba	Datos de entrada	Valor	Salida Esperada	Clases
1	Ticket	Ticket(parametros tipo 2)	1	1

getComparacionPuesto()

Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia
Ticket	Ticket!=null	si	jdoc	1
	Ticket=null	no	jdoc	2

Número de prueba	Datos de entrada	Valor	Salida Esperada	Clases
1	Ticket	Ticket(parametros tipo 2)	1	1

getComparacionExperiencia()

Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia
Ticket	Ticket!=null	si	jdoc	1
	Ticket=null	no	jdoc	2

Número de prueba	Datos de entrada	Valor	Salida Esperada	Clases
1	Ticket	Ticket(parametros tipo 2)	1	1

getComparacionEstudios()

Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia
Ticket	Ticket!=null	si	jdoc	1
	Ticket=null	no	jdoc	2

Número de prueba	Datos de entrada	Valor	Salida Esperada	Clases
1	Ticket	Ticket(parametros tipo 2)	1	1

getComparacionTotal()

Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia
Ticket	Ticket!=null	si	jdoc	1
	Ticket=null	no	jdoc	2

Número de prueba	Datos de entrada	Valor	Salida Esperada	Clases
1	Ticket	Ticket (parametros tipo 2)	6	1

Escenario nro 8: Ticket23

getComparacionLocacion(Ticket)

Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia
Ticket	Ticket!=null	si	jdoc	1
	Ticket=null	no	jdoc	2

Número de prueba	Datos de entrada	Valor	Salida Esperada	Clases
1	Ticket	Ticket(parametros tipo 13	-1	1

getComparacionRemuneracion()

Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia
Ticket	Ticket!=null	si	jdoc	1
	Ticket=null	no	jdoc	2

Número de prueba	Datos de entrada	Valor	Salida Esperada	Clases
1	Ticket	Ticket(parametros tipo 3)	-0,5	1

getComparacionJornada()

Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia
Ticket	Ticket!=null	si	jdoc	1
	Ticket=null	no	jdoc	2

Número de prueba	Datos de entrada	Valor	Salida Esperada	Clases
1	Ticket	Ticket(parametros tipo 3)	-0,5	1

getComparacionPuesto()

Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia
Ticket	Ticket!=null	si	jdoc	1
	Ticket=null	no	jdoc	2

Número de prueba	Datos de entrada	Valor	Salida Esperada	Clases
1	Ticket	Ticket(parametros tipo 3)	-0,5	1

getComparacionExperiencia()

Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia
Ticket	Ticket!=null	si	jdoc	1
	Ticket=null	no	jdoc	2

Número de prueba	Datos de entrada	Valor	Salida Esperada	Clases
1	Ticket	Ticket(parametros tipo 3)	1,5	1

getComparacionEstudios()

Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia
Ticket	Ticket!=null	si	jdoc	1
	Ticket=null	no	jdoc	2

Número de prueba	Datos de entrada	Valor	Salida Esperada	Clases
1	Ticket	Ticket(parametros tipo 3)	1,5	1

getComparacionTotal()

Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia
Ticket	Ticket!=null	si	jdoc	1
	Ticket=null	no	jdoc	2

Número de prueba	Datos de entrada	Valor	Salida Esperada	Clases
1	Ticket	Ticket (parametros tipo 3	0,5	1

Escenario nro 9: Ticket31

getComparacionLocacion(Ticket)

Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia
Ticket	Ticket!=null	si	jdoc	1
	Ticket=null	no	jdoc	2

Número de prueba	Datos de entrada	Valor	Salida Esperada	Clases
1	Ticket	Ticket (parametros tipo 1	1	1

getComparacionRemuneracion()

Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia
Ticket	Ticket!=null	si	jdoc	1
	Ticket=null	no	jdoc	2

Número de prueba	Datos de entrada	Valor	Salida Esperada	Clases
1	Ticket	Ticket (parametros tipo 1	1	1

getComparacionJornada()

Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia
Ticket	Ticket!=null	si	jdoc	1
	Ticket=null	no	jdoc	2

Número de prueba	Datos de entrada	Valor	Salida Esperada	Clases
1	Ticket	Ticket (parametros tipo 1	-1	1

getComparacionPuesto()

Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia
Ticket	Ticket!=null	si	jdoc	1
	Ticket=null	no	jdoc	2

Número de prueba	Datos de entrada	Valor	Salida Esperada	Clases
1	Ticket	Ticket (parametros tipo 1	-1	1

getComparacionExperiencia()

Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia
Ticket	Ticket!=null	si	jdoc	1
	Ticket=null	no	jdoc	2

Número de prueba	Datos de entrada	Valor	Salida Esperada	Clases
1	Ticket	Ticket (parametros tipo 1	-2	1

getComparacionEstudios()

Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia
Ticket	Ticket!=null	si	jdoc	1
	Ticket=null	no	jdoc	2

Número de prueba	Datos de entrada	Valor	Salida Esperada	Clases
1	Ticket	Ticket (parametros tipo 1	-2	1

getComparacionTotal()

Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia
Ticket	Ticket!=null	si	jdoc	1
	Ticket=null	no	jdoc	2

Número de prueba	Datos de entrada	Valor	Salida Esperada	Clases
1	Ticket	Ticket (parametros tipo 1	0,5	1

Escenario nro 10: Ticket32

getComparacionLocacion(Ticket)

Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia
Ticket	Ticket!=null	si	jdoc	1
	Ticket=null	no	jdoc	2

Número de prueba	Datos de entrada	Valor	Salida Esperada	Clases
1	Ticket	Ticket(parametros tipo 2)	-1	1

getComparacionRemuneracion()

Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia
Ticket	Ticket!=null	si	jdoc	1
	Ticket=null	no	jdoc	2

Número de prueba	Datos de entrada	Valor	Salida Esperada	Clases
1	Ticket	Ticket(parametros tipo 2)	1	1

getComparacionJornada()

Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia
Ticket	Ticket!=null	si	jdoc	1
	Ticket=null	no	jdoc	2

Número de prueba	Datos de entrada	Valor	Salida Esperada	Clases
1	Ticket	Ticket(parametros tipo 2)	1	1

getComparacionPuesto()

Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia
Ticket	Ticket!=null	si	jdoc	1
	Ticket=null	no	jdoc	2

Número de prueba	Datos de entrada	Valor	Salida Esperada	Clases
1	Ticket	Ticket(parametros tipo 2)	1	1

getComparacionExperiencia()

Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia
Ticket	Ticket!=null	si	jdoc	1
	Ticket=null	no	jdoc	2

Número de prueba	Datos de entrada	Valor	Salida Esperada	Clases
1	Ticket	Ticket(parametros tipo 2)	-1,5	1

getComparacionEstudios()

Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia
Ticket	Ticket!=null	si	jdoc	1
	Ticket=null	no	jdoc	2

Número de prueba	Datos de entrada	Valor	Salida Esperada	Clases
1	Ticket	Ticket(parametros tipo 2)	-1,5	1

getComparacionTotal()

Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia
Ticket	Ticket!=null	si	jdoc	1
	Ticket=null	no	jdoc	2

Número de prueba	Datos de entrada	Valor	Salida Esperada	Clases
1	Ticket	Ticket(parametros tipo 2)	0,5	1

Escenario nro 11: Ticket33

getComparacionLocacion(Ticket)

Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia
Ticket	Ticket!=null	si	jdoc	1
	Ticket=null	no	jdoc	2

Número de prueba	Datos de entrada	Valor	Salida Esperada	Clases
1	Ticket	Ticket (parametros tipo 3)	1	1

getComparacionRemuneracion()

Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia
Ticket	Ticket!=null	si	jdoc	1
	Ticket=null	no	jdoc	2

Número de prueba	Datos de entrada	Valor	Salida Esperada	Clases
1	Ticket	Ticket (parametros tipo 3)	1	1

getComparacionJornada()

Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia
Ticket	Ticket!=null	si	jdoc	1
	Ticket=null	no	jdoc	2

Número de prueba	Datos de entrada	Valor	Salida Esperada	Clases
1	Ticket	Ticket (parametros tipo 3)	1	1

getComparacionPuesto()

Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia
Ticket	Ticket!=null	si	jdoc	1
	Ticket=null	no	jdoc	2

Número de prueba	Datos de entrada	Valor	Salida Esperada	Clases
1	Ticket	Ticket (parametros tipo 3)	1	1

getComparacionExperiencia()

Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia
Ticket	Ticket!=null	si	jdoc	1
	Ticket=null	no	jdoc	2

Número de prueba	Datos de entrada	Valor	Salida Esperada	Clases
1	Ticket	Ticket (parametros tipo 3)	1	1

getComparacionEstudios()

Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia
Ticket	Ticket!=null	si	jdoc	1
	Ticket=null	no	jdoc	2

Número de prueba	Datos de entrada	Valor	Salida Esperada	Clases
1	Ticket	Ticket (parametros tipo 3)	1	1

getComparacionTotal()

Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia
Ticket	Ticket!=null	si	jdoc	1
	Ticket=null	no	jdoc	2

Número de prueba	Datos de entrada	Valor	Salida Esperada	Clases
1	Ticket	Ticket (parametros tipo 3)	6	1

Clase: Agencia

Escenario 1: con el objetivo de testear los getter y los setter se plantea una Agencia vacia y en el set up se generan sus listas cargadas con elementos.

setContrataciones

Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia
Contrataciones (ArrayList<Contratacion>)	Contrataciones != null	si	jdoc	1
	Contrataciones == null	no	jdoc (inv clase)	2

Número de prueba	Datos de entrada	Valor	Salida Esperada	Clases
1	Contrataciones	Lista con 1 sola contratacion entre un empleado y empleador	seteo correcto (que se verifica con el get)	1

setCoincidencias

Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia
Coincidencias (ArrayList<Contratacion>)	Coincidencias != null	si	jdoc	1
	Coincidencias == null	no	jdoc (inv clase)	2

Número de prueba	Datos de entrada	Valor	Salida Esperada	Clases
1	Coincidencia	Lista con 1 sola contratacion entre un empleado y empleador	seteo correcto (que se verifica con el get)	1

GetContratacionEmpleador

Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia
Empleador	Empleador!= null, cargado en sistema y con una contratacio	si	jdoc	1
	empleador == null	no	Pre condición	2
	empleador no en sistema	no	Pre condición	3
	empleador sin contratacion	no	No se especifica retorno o lanzamiento de excepcion	4

Número de prueba	Datos de entrada	Valor	Salida Esperada	Clases
1	Empleador	Empleador con atributos vacios	Devuelve la referencia a empleado con el que empleador tenía contratacion	1

GetContratacionEmpleadoPretenso

Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia
Empleador	Empleado!= null, cargado en sistema y con una contratacio	si	jdoc	1
	empleado == null	no	Pre condición	2
	empleado no en sistema	no	Pre condición	3
	empleado sin contratacion	no	No se especifica retorno o lanzamiento de excepcion	4

Número de prueba	Datos de entrada	Valor	Salida Esperada	Clases
1	EmpleadoPretenso	Empleado con atributos vacios	Devuelve la referencia a empleador con el que empleador tenía contratacion	1

setComisionesUsuarios()

Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia
comisionesUsuarios (Hashmap<Cliente,Double>	comisionesUsuarios != null	si	jdoc	1
	comisionesUsuarios == null	no	jdoc (inv clase)	2

Número de prueba	Datos de entrada	Valor	Salida Esperada	Clases
1	comisionesUsuarios	Hashmap con un solo elemento (empleado,0.001)	seteo correcto (que se verifica con el get)	1

getComisionUsuario()

Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia
Cliente	Cliente!= null, cargado en sistema y con una comision	si	jdoc	1
	empleado == null	no	No se especifica retorno o lanzamiento de excepcion	2
	empleado no en sistema	no		3
	empleado sin contratacion	no		4

Número de prueba	Datos de entrada	Valor	Salida Esperada	Clases
1	Cliente	Empleado cargado en hashmap con una comision de 0.01	0.01	1

setEmpleadores

Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia
Empleadores (HashMap<String,Empleador>)	Empleadores != null	si	jdoc	1
	Empleadores == null	no	jdoc (inv clase)	2

Número de prueba	Datos de entrada	Valor	Salida Esperada	Clases
1	Empleadores	Hashmap con 1 elemento ("+54 9 223 666-1234,empleador)	seteo correcto (que se verifica con el get)	1

setEmpleados

Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia
mpleados (HashMap<String, EmpleadoPretenso	Empleados != null	si	jdoc	1
	Empleados== null	no	jdoc (inv clase)	2

Número de prueba	Datos de entrada	Valor	Salida Esperada	Clases
1	Empleados	Hashmap con 1 elemento ("+54 9	seteo correcto (que se verifica con el get)	1

getIteratorEmpleadores()

Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia
		si	jdoc	1

Número de prueba	Datos de entrada	Valor	Salida Esperada	Clases
1			Referencia a un iterator que contenga los elementos de la lista de Empleadores	1

getIteratorEmpleadosPretensos()

Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia
		si	jdoc	1

Número de prueba	Datos de entrada	Valor	Salida Esperada	Clases
1			Referencia a un iterator que contenga los elementos de la lista de EmpleadosPretensos	1

Escenario 2: Con el objetivo de testear los getters y setters, se plantea una Agencia vacía y en el setter se generan sus listas sin elementos en ellas.

getEstado()

Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia
		si	jdoc	1

Número de prueba	Datos de entrada	Valor	Salida Esperada	Clases
1			String que según sea F o V el estado en el que se encuentre agencia devuelve Mensaje.AGENCIA_EN CONTRATACION.getValod() o Mensajes.AGENCIA_EN_BUSQUEDA.getValor()	1

getInstance()

Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia
		si	jdoc	1

Número de prueba	Datos de entrada	Valor	Salida Esperada	Clases
1			Debe obtenerse la misma referencia a Agencia que la presente en	1

setCoincidencias ()

Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia
Coincidencias (ArrayList<Contratacion>)	Coincidencias != null	si	jdoc	1
	Coincidencias == null	no	jdoc (inv clase)	2

Número de prueba	Datos de entrada	Valor	Salida Esperada	Clases
1	Coincidencia	Lista sin contrataciones (vacia)	seteo correcto (que se verifica con el get)	1

setComisionesUsuarios()

Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia
comisionesUsuarios (Hashmap<Cliente,Double>	comisionesUsuarios != null	si	jdoc	1
	comisionesUsuarios == null	no	jdoc (inv clase)	2

Número de prueba	Datos de entrada	Valor	Salida Esperada	Clases
1	comisionesUsuarios	Hashmap vacio	seteo correcto (que se verifica con el get)	1

setContrataciones

Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia
Contrataciones (ArrayList<Contratacion>)	Contrataciones != null	si	jdoc	1
	Contrataciones == null	no	jdoc (inv clase)	2

Número de prueba	Datos de entrada	Valor	Salida Esperada	Clases
1	Contrataciones	Lista sin contrataciones (vacia)	seteo correcto (que se verifica con el get)	1

setEmpleadores

Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia
Empleadores (HashMap<String,Empleador>)	Empleadores != null	si	jdoc	1
	Empleadores == null	no	jdoc (inv clase)	2

Número de prueba	Datos de entrada	Valor	Salida Esperada	Clases
1	Empleadores	Hashmap vacio	seteo correcto (que se verifica con el get)	1

setEmpleados

Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia
mpleados (HashMap<String, EmpleadoPretenso	Empleados != null	si	jdoc	1
	Empleados== null	no	jdoc (inv clase)	2

Número de prueba	Datos de entrada	Valor	Salida Esperada	Clases
1	Empleados	Hashmap vacio	seteo correcto (que se verifica con el get)	1

setEstadoContratacion()

Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia
EstadoContratacion	Estado=True	si	jdoc	1
	Estado=False	no	jdoc (inv clase)	2

Número de prueba	Datos de entrada	Valor	Salida Esperada	Clases
1	Estado	True	seteo correcto (que se verifica con el get)	1
1	Estado	False	seteo correcto (que se verifica con el get)	1

setLimitesRemuneración

Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia
limiteInferior	inf<0	si	c	1
	inf>=0	si	c	2
limiteSuperior	sup>=inf	si	c	3
	sup<inf	si	c	4

Número de prueba	Datos de entrada	Valor	Salida Esperada	Clases
1	inf	1	seteo correcto	2,3
	sup	999999		
2	inf	-1	LimiteInferiorRemuneracionInvalidaException	1,3
	sup	999999		
3	inf	2	LimiteSuperiorRemuneracionInvalidaException	2,4
	sup	1		

getLimiteSuperior()

Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia
		si	jdoc	1

Número de prueba	Datos de entrada	Valor	Salida Esperada	Clases
1			Limite superior establecido en agencia (999999)	1

getLimiteInferior()

Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia
		si	jdoc	1

Número de prueba	Datos de entrada	Valor	Salida Esperada	Clases
1			Limite superior establecido en agencia (1)	1

setPersistencia()

Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia
Persistencia (Ipersistencia)	Persistencia != null	si	jdoc	1
	Persistencia == null	si	jdoc	2

Número de prueba	Datos de entrada	Valor	Salida Esperada	Clases
1	Persistencia	PersistenciaXML()	seteo correcto (que se verifica con el get)	1
2	Persistencia	null	seteo correcto (que se verifica con el get)	2

getIteratorEmpleadores()

Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia
		si	jdoc	1

Número de prueba	Datos de entrada	Valor	Salida Esperada	Clases
1			Referencia a un iterator que contenga los elementos de la lista de Empleadores (vacio)	1

getIteratorEmpleadosPretensos()

Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia
		si	jdoc	1

Número de prueba	Datos de entrada	Valor	Salida Esperada	Clases
1			Referencia a un iterator que contenga los elementos de la lista de EmpleadosPretensos (vacio)	1

Escenario 3: con el objetivo de testear métodos de creación de tickets en agencia, se establece un escenario Agencia con su estado en FALSO y con 1 elemento en su lista de empleado y 1 elemento en su lista de empleador. A cada uno de ellos se les crea un ticket siendo entonces ticket empleador {util.Constantes.HOME_OFFICE, 999, util.Constantes.JORNADA_MEDIA, util.Constantes.JUNIOR, util.Constantes.EXP_MEDIA, util.Constantes.PRIMARIOS, empleador} y ticket de empleado {util.Constantes.HOME_OFFICE, 999, util.Constantes.JORNADA_MEDIA, util.Constantes.JUNIOR, util.Constantes.EXP_NADA, util.Constantes.PRIMARIOS, empleado}.
Los test deberán mediante los getter y setter verificar que los tickets del empleado y empleador coinciden con lo creado en el setUp.
!)Nota: como los métodos no establecen como precondiciones las condiciones necesarias para crear un ticket (documentadas en ticket) y tampoco documentan la forma de resolución que adoptan a la hora de incumplir dichas condiciones (lanzamiento de excepción por ejemplo), para reducir la incertidumbre del retorno del método, se toman como precondiciones del método las establecidas para los atributos en ticket (invariantes de clase).

Para el ticket de empleador [empleador.getTicket()]

getEstudios()

Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia
		si	jdoc	1

Número de prueba	Datos de entrada	Valor	Salida Esperada	Clases
1			util.Constantes.PRIMARIOS	1

getExperiencia()

Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia
		si	jdoc	1

Número de prueba	Datos de entrada	Valor	Salida Esperada	Clases
1			util.Constantes.EXP_MEDIA	1

getJornada()

Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia
		si	jdoc	1

Número de prueba	Datos de entrada	Valor	Salida Esperada	Clases
1			util.Constantes.JORNADA_MEDIA	1

getLocacion()

Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia
		si	jdoc	1

Número de prueba	Datos de entrada	Valor	Salida Esperada	Clases
1			util.Constantes.HOME_OFFICE	1

getPuesto()

Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia
		si	jdoc	1

Número de prueba	Datos de entrada	Valor	Salida Esperada	Clases
1			util.Constantes.JUNIOR	1

getRemuneracion()

Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia
		si	jdoc	1

Número de prueba	Datos de entrada	Valor	Salida Esperada	Clases
1			999	1

Para el ticket de empleadoPretenso [empleado.getTicket()]

getEstudios()

Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia
		si	jdoc	1

Número de prueba	Datos de entrada	Valor	Salida Esperada	Clases
1			util.Constantes.PRIMARIOS	1

getExperiencia()

Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia
		si	jdoc	1

Número de prueba	Datos de entrada	Valor	Salida Esperada	Clases
1			util.Constantes.EXP_NADA	1

getJornada()

Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia
		si	jdoc	1

Número de prueba	Datos de entrada	Valor	Salida Esperada	Clases
1			util.Constantes.JORNADA_MEDIA	1

getLocacion()

Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia
		si	jdoc	1

Número de prueba	Datos de entrada	Valor	Salida Esperada	Clases
1			util.Constantes.HOME_OFFICE	1

getPuesto()

Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia
		si	jdoc	1

Número de prueba	Datos de entrada	Valor	Salida Esperada	Clases
1			util.Constantes.JUNIOR	1

getRemuneracion()

Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia
		si	jdoc	1

Número de prueba	Datos de entrada	Valor	Salida Esperada	Clases
1			999	1

Escenario 4: con el objetivo de testear métodos que varían según el estado de la contratación en agencia, se establece un escenario Agencia con su estado en VERDADERO y con 1 elemento en su lista de empleado y 1 elemento en su lista de empleador.

crearTicketEmpleado() ¡! Nota: el metodo no establece como precondiciones las condiciones necesarias para crear un ticket (documentadas en ticket) , pero tampoco documenta la forma de resolucion que adopta a la hora de incumplir dichas condiciones (lanzamiento de excep

Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia
Locacion (string)	Locacion!= null y esta contemplado en la clase Constantes	si	jdoc	1
	String no contemplado en la clase Constantes	no	jdoc (ticket)	2
	Locacion== null	no	jdoc (ticket)	3
remuneracion (int)	Remuneracion >= 0	si	jdoc	4
	Remuneracion < 0	no	jdoc (ticket)	5
jornada (string)	jornada!= null y esta contemplado en la clase Constantes	si	jdoc	6
	String no contemplado en la clase Constantes	no	jdoc (ticket)	7
	jornada== null	no	jdoc (ticket)	8
puesto (string)	puesto!= null y esta contemplado en la clase Constantes	si	jdoc	9
	String no contemplado en la clase Constantes	no	jdoc (ticket)	10
	puesto== null	no	jdoc (ticket)	11
experiencia (string)	experiencia!= null y esta contemplado en la clase Constante	si	jdoc	12
	String no contemplado en la clase Constantes	no	jdoc (ticket)	13
	experiencia== null	no	jdoc (ticket)	14
estudios (string)	estudios!= null y esta contemplado en la clase Constantes	si	jdoc	15
	String no contemplado en la clase Constantes	no	jdoc (ticket)	16
	estudios== null	no	jdoc (ticket)	17
cliente (cliente)	cliente != null y de tipo EmpleadoPretenso	si	jdoc	18
	cliente != null y de tipo != EmpleadoPretenso	no	jdoc	19
	cliente == null	no	jdoc no documenta que ocurre si cliente	20

Número de prueba	Datos de entrada	Valor	Salida Esperada	Clases
1	Locacion	util.Constantes.HOME_OFFICE	Lanzamiento de la excepcion ImposibleModificarTicketException	1,4,6,9,12,15,18
	remuneracion	999		
	jornada	util.Constantes.JORNADA_MEDIA		
	puesto	util.Constantes.JUNIOR		
	experiencia	util.Constantes.EXP_NADA		
	estudios	util.Constantes.PRIMARIOS		
	cliente	empleadoPretenso()		

crearTicketEmpleado() ¡! Nota: el metodo no establece como precondiciones las condiciones necesarias para crear un ticket (documentadas en ticket) , pero tampoco documenta la forma de resolucion que adopta a la hora de incumplir dichas condiciones (lanzamiento de excep

Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia
Locacion (string)	Locacion!= null y esta contemplado en la clase Constantes	si	jdoc	1
	String no contemplado en la clase Constantes	no	jdoc (ticket)	2
	Locacion== null	no	jdoc (ticket)	3
remuneracion (int)	Remuneracion >= 0	si	jdoc	4
	Remuneracion < 0	no	jdoc (ticket)	5
jornada (string)	jornada!= null y esta contemplado en la clase Constantes	si	jdoc	6
	String no contemplado en la clase Constantes	no	jdoc (ticket)	7
	jornada== null	no	jdoc (ticket)	8
puesto (string)	puesto!= null y esta contemplado en la clase Constantes	si	jdoc	9
	String no contemplado en la clase Constantes	no	jdoc (ticket)	10
	puesto== null	no	jdoc (ticket)	11
experiencia (string)	experiencia!= null y esta contemplado en la clase Constante	si	jdoc	12
	String no contemplado en la clase Constantes	no	jdoc (ticket)	13
	experiencia== null	no	jdoc (ticket)	14
estudios (string)	estudios!= null y esta contemplado en la clase Constantes	si	jdoc	15
	String no contemplado en la clase Constantes	no	jdoc (ticket)	16
	estudios== null	no	jdoc (ticket)	17
cliente (cliente)	cliente != null y de tipo Empleador	si	jdoc	18
	cliente != null y de tipo != Empleador	no	jdoc	19
	cliente == null	no	jdoc no documenta que ocurre si cliente	20

Número de prueba	Datos de entrada	Valor	Salida Esperada	Clases
1	Locacion	util.Constantes.HOME_OFFICE	Lanzamiento de la excepcion ImposibleModificarTicketException	1,4,6,9,12,15,18
	remuneracion	999		
	jornada	util.Constantes.JORNADA_MEDIA		
	puesto	util.Constantes.JUNIOR		
	experiencia	util.Constantes.EXP_NADA		
	estudios	util.Constantes.PRIMARIOS		
	cliente	empleador()		

Escenario 5: con el objetivo de testear los metodos de registro de empleado y empleador se plantea un escenario de Agencia con un empleador

Número de prueba	Datos de entrada	Valor	Salida Esperada	Clases
1	nombreUsuario	"Pepegamer"	Creacion correcta de empleador y adicion a la lista de empleadores	1,4,6,8,10,13
	pass	"contrasenia"		
	nombreReal	"Pepe Gomes"		
	telefono	"2234434312"		
	tipoPersona	util.constantes.FISICA		
	rubro	util.constantes.SALUD		
2	nombreUsuario	null	throws ImposibleCrearEmpleadorException()	3,4,6,8,10,13
	pass	"contrasenia"		
	nombreReal	"Pepe Gomes"		
	telefono	"2234434312"		
	tipoPersona	util.constantes.FISICA		
	rubro	util.constantes.SALUD		
3	nombreUsuario	"Pepegamer"	throws ImposibleCrearEmpleadorException()	1,5,6,8,10,13
	pass	null		
	nombreReal	"Pepe Gomes"		
	telefono	"2234434312"		
	tipoPersona	util.constantes.FISICA		
	rubro	util.constantes.SALUD		
4	nombreUsuario	"Pepegamer"	throws ImposibleCrearEmpleadorException()	1,4,7,8,10,13
	pass	"contrasenia"		
	nombreReal	null		
	telefono	"2234434312"		
	tipoPersona	util.constantes.FISICA		
	rubro	util.constantes.SALUD		
5	nombreUsuario	"Pepegamer"	throws ImposibleCrearEmpleadorException()	1,4,6,9,10,13
	pass	"contrasenia"		
	nombreReal	"Pepe Gomes"		
	telefono	null		
	tipoPersona	util.constantes.FISICA		
	rubro	util.constantes.SALUD		
6	nombreUsuario	"Pepegamer"	throws ImposibleCrearEmpleadorException()	1,4,6,8,11,13
	pass	"contrasenia"		
	nombreReal	"Pepe Gomes"		
	telefono	"2234434312"		
	tipoPersona	"NORIMAL"		
	rubro	util.constantes.SALUD		

7	nombreUsuario	"Pepegamer"	throws ImposibleCrearEmpleadorException()	1,4,6,8,12,13
	pass	"contrasenia"		
	nombreReal	"Pepe Gomes"		
	telefono	"2234434312"		
	tipoPersona	null		
	rubro	util.constantes.SALUD		
8	nombreUsuario	"Pepegamer"	throws ImposibleCrearEmpleadorException()	1,4,6,8,10,14
	pass	"contrasenia"		
	nombreReal	"Pepe Gomes"		
	telefono	"2234434312"		
	tipoPersona	util.constantes.FISICA		
	rubro	"INFORMATICA"		
9	nombreUsuario	"Pepegamer"	throws ImposibleCrearEmpleadorException()	1,4,6,8,10,15
	pass	"contrasenia"		
	nombreReal	"Pepe Gomes"		
	telefono	"2234434312"		
	tipoPersona	util.constantes.FISICA		
	rubro	null		
10	nombreUsuario	"Tomasito"	throws NewRegisterException()	1,2,6,8,10,13
	pass	"contrasenia"		
	nombreReal	"Pepe Gomes"		
	telefono	"2234434312"		
	tipoPersona	util.constantes.FISICA		
	rubro	util.constantes.SALUD		

registroEmpleado()

Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia
nombre de usuario (string)	nombreUsuario != null y no se encuentra registrado	si	jdoc	1
	nombreUsuario != null y ya encuentra registrado	si	jdoc	2
	nombreUsuario == null	si	jdoc	3
pass (string)	pass != null	si	jdoc	4
	pass == null	si	jdoc	5
nombre Real (string)	nombreReal != null	si	jdoc	6
	nombreReal == null	si	jdoc	7
apellido (string)	apellido != null	si	jdoc	8
	apellido == null	si	jdoc	9
telefono (string)	telefono != null	si	jdoc	10
	telefono == null	si	jdoc	11
edad (int)	edad ∈ Z	si	jdoc	12

Número de prueba	Datos de entrada	Valor	Salida Esperada	Clases
1	nombreUsuario	"Pepegamer"	Creacion correcta de empleador y adicion a la lista de empleadores	1,4,6,8,10,12
	pass	"contrasenia"		
	nombreReal	"Pepe"		
	apellido	"Gomes"		
	telefono	"2234434312"		
	edad	21		
2	nombreUsuario	null	throws ImposibleCrearEmpleadoException()	3,4,6,8,10,12
	pass	"contrasenia"		
	nombreReal	"Pepe"		
	apellido	"Gomes"		
	telefono	"2234434312"		
	edad	21		
3	nombreUsuario	"Pepegamer"	throws ImposibleCrearEmpleadoException()	1,4,6,8,10,12
	pass	null		
	nombreReal	"Pepe"		
	apellido	"Gomes"		
	telefono	"2234434312"		
	edad	21		
4	nombreUsuario	"Pepegamer"	throws ImposibleCrearEmpleadoException()	1,4,7,8,10,12
	pass	"contrasenia"		
	nombreReal	null		
	apellido	"Gomes"		
	telefono	"2234434312"		
	edad	21		
5	nombreUsuario	"Pepegamer"	throws ImposibleCrearEmpleadoException()	1,4,6,9,10,12
	pass	"contrasenia"		
	nombreReal	"Pepe"		
	apellido	null		
	telefono	"2234434312"		
	edad	21		
6	nombreUsuario	"Pepegamer"	throws ImposibleCrearEmpleadoException()	1,4,6,8,11,12
	pass	"contrasenia"		
	nombreReal	"Pepe"		
	apellido	"Gomes"		
	telefono	null		
	edad	21		
7	nombreUsuario	"Palo"	throws NewRegisterException()	2,4,6,8,10,12
	pass	"contrasenia"		
	nombreReal	"Pepe"		
	apellido	"Gomes"		
	telefono	"2234434312"		
	edad	21		

Escenario 6: con el objetivo de testear el login de un usuario, se establece un escenario de Agencia con un empleador registrado en el sistema (con nombre de usuario "Pepegamer" y contraseña "qwertyuiop")

login()

Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia
nombre de usuario (string)	nombre!=null y usuario registrado	si	precondicion jdoc	1
	nombre!=null y usuario no registrado	si	precondicion jdoc	2
	nombre==null	no	precondicion jdoc	3
contraseña	contraseña!=null y contraseña correcta	si	precondicion jdoc	4
	contraseña!= null y contraseña incorrecta	si	precondicion jdoc	5
	contraseña==null	no	precondicion jdoc	6

Número de prueba	Datos de entrada	Valor	Salida Esperada	Clases
1	nombre usuario	"Pepegamer"	logueo correcto	1,4.
	contraseña	"qwertyuiop"		
2	nombre usuario	"Pepegamer"	throws ContraException()	1,5.
	contraseña	"contraExeception"		
3	nombre usuario	"UsuarioExeception"	throws NombreUsuarioException()	2,4.
	contraseña	"qwertyuiop"		

registroEmpleador()

Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia
nombre de usuario (string)	nombreUsuario != null y no se encuentra registrado	si	jdoc	1
	nombreUsuario != null y ya encuentra registrado	si	jdoc	2
	nombreUsuario == null	si	jdoc	3
pass (string)	pass != null	si	jdoc	4
	pass == null	si	jdoc	5
nombre Real (string)	nombreReal != null	si	jdoc	6
	nombreReal == null	si	jdoc	7
telefono (string)	telefono != null	si	jdoc	8
	telefono == null	si	jdoc	9
tipo persona (string)	tipoPersona != null y contemplada en Constantes	si	jdoc	10
	tipoPersona != null y NO contemplada en Constantes	si	jdoc	11
	tipoPersona == null	si	jdoc	12
rubro (string)	rubro != null y contemplada en Constantes	si	jdoc	13
	rubro != null y NO contemplada en Constantes	si	jdoc	14
	rubro == null	si	jdoc	15

Escenario 6: con el objetivo de testear el login de un usuario, se establece un escenario de Agencia con un empleador registrado en el sistema (con nombre de usuario "Pepegamer" y contraseña "qwertyuiop")

Escenario 7: con el objetivo de testear aquellos métodos que requieren de un usuario logueado para poder funcionar se plantea un escenario Agencia con un usuario registrado y logueado ("Pepegamer","qwertyuiop","Pepe Gomes","2234434312",util.Constantes.FISICA,util.Constantes.SALUD) y que presenta un ticket a su nombre (util.Constantes.HOME_OFFICE,999,util.Constantes.JORNADA_MEDIA,util.Constantes.JUNIOR,util.Constantes.EXP_MEDIA,util.Constantes.PRIMARIOS,empleador).

cerrarCesion()

Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia
		si	jdoc	1

Número de prueba	Datos de entrada	Valor	Salida Esperada	Clases
1			cambio de tipoUsuario por -1 (ningun usuario logueado)	1

getTipoUsuario()

Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia
		si	jdoc	1

Número de prueba	Datos de entrada	Valor	Salida Esperada	Clases
1			tipo de Usuario actual (empleador = 1)	1

Escenario 8: con el objetivo de testear el metodo eliminar ticket se plantea un escenario Agencia con estadoContratacion en FALSO y con un empleado registrado y logueado con un ticket a su nombre

eliminarTicket(). Jdoc omite el descuento del puntaje al eliminar un ticket (o sino no aclara en que estructura se registra la eliminacion del ticket para luego realizar el descuento del puntaje), de igual manera esto se testea ya que esta en el contrato del programa ("Cálculo de Puntaje de Usuario"

Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia
		si	jdoc	1

Número de prueba	Datos de entrada	Valor	Salida Esperada	Clases
1			Se elimina el ticket asociado al usuario logueado actualmente y se resta en 1 su puntaje (2 test distintos para verificar)	1

Escenario 9: con el objetivo de testear el metodo eliminar ticket se plantea un escenario Agencia con estadoContratacion en Verdadero y con un empleado registrado y logueado con un ticket a su nombre.

eliminarTicket()

Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia
		si	jdoc	1

Número de prueba	Datos de entrada	Valor	Salida Esperada	Clases
1			throws ImposibleModificarTicketsException()	1

Escenario 10: con el objetivo de testear el método generar postulantes de Agencia se plantea un escenario con 2 empleados y 2 empleadores cada uno con un ticket. Un par empleado y un empleador tienen tickets idénticos (se los llama empleado1 y empleador1) y el otro par tendrá tickets opuestos a los del primer par. En el set up tambien se llamará al método a testear, y los diferentes test verifican para un empleado y un empleador que se haya generado la lista postulantes correctamente (no vacía y con orden de postulantes correcto).

Empleado1.getListaDePostulantes().size()

Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia
		si	jdoc	1

Número de prueba	Datos de entrada	Valor	Salida Esperada	Clases
1			2 (la lista debe contener a los 2 empleadores)	1

Empleado1.getListaPostulantes().get(0).getCliente(). Con este test se verifica que el primer elemento de la lista de postulantes coincide con el empleador con mayor coincidencia de ticket (puntaje)

Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia
		si	jdoc	1

Número de prueba	Datos de entrada	Valor	Salida Esperada	Clases
1			referencia al objeto Empleador1	1

Empleador1.getListaDePostulantes().size()

Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia
		si	jdoc	1

Número de prueba	Datos de entrada	Valor	Salida Esperada	Clases
1			2 (la lista debe contener a los 2 empleados)	1

Empleador1.getListaPostulantes().get(0).getCliente(). Con este test se verifica que el primer elemento de la lista de postulantes coincide con el empleado con mayor coincidencia de ticket (puntaje)

Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia
		si	jdoc	1

Número de prueba	Datos de entrada	Valor	Salida Esperada	Clases
1			referencia al objeto Empleado1	1

Escenario 11: con el objetivo de testear el metodo calculaPremiosyCastigosAsignaciones se plantea un escenario con 2 empleados y 2 empleadores cada uno con un ticket. Un par empleado y empleador tienen tickets idénticos (se los llama empleado1 y empleador1) y el otro par tendrá tickets opuestos a los del primer par (esto hará que cada empleado y empleador esté en 2 listas, en una primero y en otra última). Además, en el set up se llama al método generarPostulanes para generar ordenadamente dichas listas en cada cliente (precondición del método a testear) y luego se llama al método calculaPremiosyCastigosAsignaciones. Los test verifican para cada empleado y empleador que el cálculo de su puntaje coincide con el esperado en función al contrato del programa.

empleado1.getPuntaje()

Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia
		si	jdoc	1

Número de prueba	Datos de entrada	Valor	Salida Esperada	Clases
1			0 (puntaje= +5 -5 debido a que se encuentra 1ro en la lista del empleador 1 y ultimo en la lista del empleador 2)	1

empleado2.getPuntaje()

Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia
		si	jdoc	1

Número de prueba	Datos de entrada	Valor	Salida Esperada	Clases
1			0 (puntaje= +5 -5 debido a que se encuentra 1ro en la lista del empleador 2 y ultimo en la lista del empleador 1)	1

empleador1.getPuntaje()

Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia
		si	jdoc	1

Número de prueba	Datos de entrada	Valor	Salida Esperada	Clases
1			10 (debido a que se encuentra 1ro en la lista del empleado 1)	1

empleado2.getPuntaje()

Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia
		si	jdoc	1

Número de prueba	Datos de entrada	Valor	Salida Esperada	Clases
1			10 (debido a que se encuentra 1ro en la lista del empleado 2)	1

Escenario 12: con el objetivo de testear el método match de la clase agencia se plantea un escenario de Agencia con un empleado y un empleador registrados, cada uno con un ticket creado. Luego, en el setUp se llama al método a testear y en cada uno de los test se verifica el correcto comportamiento de dicho método (verificar que los cambios sean los esperados)

empleado.getTicket()

Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia
		si	jdoc	1

Número de prueba	Datos de entrada	Valor	Salida Esperada	Clases
1			null	1

empleador.getTicket()

Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia
		si	jdoc	1

Número de prueba	Datos de entrada	Valor	Salida Esperada	Clases
1			null	1

getContratacionEmpleadoPretenso()

Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia
Empleado	Empleado!=null	si	jdoc	1

Número de prueba	Datos de entrada	Valor	Salida Esperada	Clases
1	Empleador	empleado con contratacion vigente	referencia al empleador con el que se tuvo la contratacion	1

getContratacionEmpleador()

Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia
Empleador	Empleador!=null	si	jdoc	1

Número de prueba	Datos de entrada	Valor	Salida Esperada	Clases
1		empleador con contratacion vigente	referencia al empleado con el que se tuvo la contratacion	1

empleado.getPuntaje()

Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia
		si	jdoc	1

Número de prueba	Datos de entrada	Valor	Salida Esperada	Clases
1			10	1

empleador.getPuntaje()

Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia
		si	jdoc	1

Número de prueba	Datos de entrada	Valor	Salida Esperada	Clases
1			50	1

Escenario 13: con el objetivo de testear el método gatillar ronda se plantea un escenario de agencia con el estado de contratación en FALSO y con 2 empleados y 2 empleadores cargados, cada uno con un ticket (el 1er par empleado empleador con tickets iguales y el otro par con sus tickets iguales pero opuestos al 1er par). Luego, en el setup se llama al método gatillar ronda y los distintos test se encargaran de verificar que el comportamiento del método sea el esperado.

Empleado1.getListaDePostulantes().size()

Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia
		si	jdoc	1

Número de prueba	Datos de entrada	Valor	Salida Esperada	Clases
1			2 (la lista debe contener a los 2 empleadores)	1

Empleado1.getListaPostulantes().get(0).getCliente(). Con este test se verifica que el primer elemento de la lista de postulantes coincide con el empleador con mayor coincidencia de ticket (puntaje)

Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia
		si	jdoc	1

Número de prueba	Datos de entrada	Valor	Salida Esperada	Clases
1			referencia al objeto Empleador1	1

Empleador1.getListaDePostulantes().size()

Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia
		si	jdoc	1

Número de prueba	Datos de entrada	Valor	Salida Esperada	Clases
1			2 (la lista debe contener a los 2 empleados)	1

Empleador1.getListaPostulantes().get(0).getCliente(). Con este test se verifica que el primer elemento de la lista de postulantes coincide con el empleado con mayor coincidencia de ticket (puntaje)

Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia
		si	jdoc	1

Número de prueba	Datos de entrada	Valor	Salida Esperada	Clases
1			referencia al objeto Empleado1	1

empleado1.getPuntaje()

Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia
		si	jdoc	1

Número de prueba	Datos de entrada	Valor	Salida Esperada	Clases
1			0 (puntaje= +5 -5 debido a que se encuentra 1ro en la lista del empleador 1 y ultimo en la lista del empleador 2)	1

empleado2.getPuntaje()

Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia
		si	jdoc	1

Número de prueba	Datos de entrada	Valor	Salida Esperada	Clases
1			0 (puntaje= +5 -5 debido a que se encuentra 1ro en la lista del empleador 2 y ultimo en la lista del empleador 1)	1

empleador1.getPuntaje()

Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia
		si	jdoc	1

Número de prueba	Datos de entrada	Valor	Salida Esperada	Clases
1			10 (debido a que se encuentra 1ro en la lista del empleado 1)	1

empleado2.getPuntaje()

Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia
		si	jdoc	1

Número de prueba	Datos de entrada	Valor	Salida Esperada	Clases
1			10 (debido a que se encuentra 1ro en la lista del empleado 2)	1

isEstadoContratacion()

Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia
		si	jdoc	1

Número de prueba	Datos de entrada	Valor	Salida Esperada	Clases
1			True	1

Escenario 14: con el objetivo de testear el método gatillar ronda se plantea un escenario de agencia con el estado de contratación en verdadero y con 1 empleado y 2 empleadores cargados, cada uno con un ticket (el 1er par empleado empleador con tickets iguales y el otro empleador con su tickets opuesto al 1er par). Además, el par empleado y empleador están entre ambos seteados como candidatos y a cada cliente se le genera su lista de postulantes. Por último, en el setUp se llama al método gatillar ronda y los test verificarán que el comportamiento sea el esperado.

empleado1.getPuntaje()

Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia
		si	jdoc	1

Número de prueba	Datos de entrada	Valor	Salida Esperada	Clases
1			10	1

empleador1.getPuntaje()

Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia
		si	jdoc	1

Número de prueba	Datos de entrada	Valor	Salida Esperada	Clases
1			50	1

empleador2.getPuntaje()

Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia
		si	jdoc	1

Número de prueba	Datos de entrada	Valor	Salida Esperada	Clases
1			-20	1

empleado1.getTicket()

Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia
		si	jdoc	1

Número de prueba	Datos de entrada	Valor	Salida Esperada	Clases
1			null	1

empleador1.getTicket()

Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia
		si	jdoc	1

Número de prueba	Datos de entrada	Valor	Salida Esperada	Clases
1			null	1

empleador2.getTicket()

Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia
		si	jdoc	1

Número de prueba	Datos de entrada	Valor	Salida Esperada	Clases
1			null	1

getContratacionEmpleadoPretenso()

Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia
Empleado	Empleado!=null	si	jdoc	1

Número de prueba	Datos de entrada	Valor	Salida Esperada	Clases
1	Empleador	empleado con contratacion vigente	referencia al empleador con el que se tuvo la contratacion	1

getContratacionEmpleador()

Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia
Empleador	Empleador!=null	si	jdoc	1

Número de prueba	Datos de entrada	Valor	Salida Esperada	Clases
1		empleador con contratacion vigente	referencia al empleado con el que se tuvo la contratacion	1

isEstadoContratacion()

Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia
		si	jdoc	1

Número de prueba	Datos de entrada	Valor	Salida Esperada	Clases
1			False	1

2.4 Test de Persitencia

Test 1: el objetivo del test es probar los métodos de abrir en la clase PersistenciaXML, para ello en el escenario se presenta un objeto persistenciaXML().

abrirInput()

Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia
nombre(string)	nombre != null	si	jdoc	1
	nombre == null	no	No especificado el	2

Número de prueba	Datos de entrada	Valor	Salida Esperada	Clases
1	nombre	"Agencia.xml"	Apertura correcta de Input (no lanzamiento de excepcion)	1

abrirOutput()

Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia
nombre (string)	nombre != null	si	jdoc	1
	nombre == null	no	No especificado el	2

Número de prueba	Datos de entrada	Valor	Salida Esperada	Clases
1	nombre	"Agencia.xml"	Apertura correcta de Output (no lanzamiento de excepcion)	1

Test 2: el objetivo del test es probar los métodos de cerrar en la clase Persisitencia XML, para ello en el escenario se presenta un objeto persistenciaXML() con apertura de input y output ya realizadas.

cerrarInput()

Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia
		si	jdoc	1

Número de prueba	Datos de entrada	Valor	Salida Esperada	Clases
1			Cerrado correcto de Input (no lanzamiento de excepcion)	1

cerrarOutput()

Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia
		si	jdoc	1

Número de prueba	Datos de entrada	Valor	Salida Esperada	Clases
1			Cerrado correcto de Output (no lanzamiento de excepcion)	1

Test 3: el objetivo del test es probar todos los métodos de agenciaDTO (que son fundamentalmente setter y getter). Para ello, como escenario se plantea un objetoDTO con atributos vacíos.

getLimiteInferior()

Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia
limiteInferior (int)	limInferior ∈ Z	si	jdoc	1

Número de prueba	Datos de entrada	Valor	Salida Esperada	Clases
1	limiteInferior	999	Seteo correcto (verifica con get)	1

setLimiteSuperior()

Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia
limiteSuperior (int)	limSuperior ∈ Z	si	jdoc	1

Número de prueba	Datos de entrada	Valor	Salida Esperada	Clases
1	limiteSuperior	9999	Seteo correcto (verifica con get)	1

setEstadoContratacion()

Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia
estadoContratacion	estadoContratacion = true	si	jdoc	1
	estadoContratacion = false	si	jdoc	2

Número de prueba	Datos de entrada	Valor	Salida Esperada	Clases
1	estadoContratacion	true	Seteo correcto (verifica con get)	1

setEmpleados()

Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia
mpleados (HashMap<String, EmpleadoPretenso	Empleados != null	si	jdoc	1
	Empleados== null	si	jdoc	2

Número de prueba	Datos de entrada	Valor	Salida Esperada	Clases
1	Empleados	Hashmap vacio	seteo correcto (que se verifica con el get)	1

setEmpleadores()

Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia
Empleadores (HashMap<String,Empleador>)	Empleadores != null	si	jdoc	1
	Empleadores == null	si	jdoc (inv clase)	2

Número de prueba	Datos de entrada	Valor	Salida Esperada	Clases
1	Empleadores	Hashmap vacio	seteo correcto (que se verifica con el get)	1

setContrataciones

Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia
Contrataciones (ArrayList<Contratacion>)	Contrataciones != null	si	jdoc	1
	Contrataciones == null	si	jdoc (inv clase)	2

Número de prueba	Datos de entrada	Valor	Salida Esperada	Clases
1	Contrataciones	Lista sin contrataciones (vacia)	seteo correcto (que se verifica con el get)	1

setComisionesUsuarios()

Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia
comisionesUsuarios (Hashmap<Cliente,Double>	comisionesUsuarios != null	si	jdoc	1
	comisionesUsuarios == null	si	jdoc (inv clase)	2

Número de prueba	Datos de entrada	Valor	Salida Esperada	Clases
1	comisionesUsuarios	Hashmap vacio	seteo correcto (que se verifica con el get)	1

Test 4: el objetivo del test es probar el método agenciaDTOfromAgencia, para ello se plantea como escenario Agencia con listas vacías, su estadoContratacion en true y sus límites (999,9999) y AgenciaDTO obtenida a partir de Agencia a partir del método a testear. Los test (set y get) verificarán que los atributos de agencia hayan sido copiados correctamente en AgenciaDTO.

getComisionesUsuarios()

Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia
		si	jdoc	1

Número de prueba	Datos de entrada	Valor	Salida Esperada	Clases
1			referencia al objeto ComisionesUsuario de Agencia	1

getContrataciones()

Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia
		si	jdoc	1

Número de prueba	Datos de entrada	Valor	Salida Esperada	Clases
1			referencia al objeto Contrataciones de Agencia	1

getEmpleadores()

Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia
		si	jdoc	1

Número de prueba	Datos de entrada	Valor	Salida Esperada	Clases
1			referencia al objeto Empleadores de Agencia	1

getEmpleados()

Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia
		si	jdoc	1

Número de prueba	Datos de entrada	Valor	Salida Esperada	Clases
1			referencia al objeto Empleados de Agencia	1

getLimiteInferior()

Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia
		si	jdoc	1

Número de prueba	Datos de entrada	Valor	Salida Esperada	Clases
1			999	1

getLimiteSuperior()

Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia
		si	jdoc	1

Número de prueba	Datos de entrada	Valor	Salida Esperada	Clases
1			9999	1

isEstadoContratacion()

Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia
		si	jdoc	1

Número de prueba	Datos de entrada	Valor	Salida Esperada	Clases
1			true	1

Test 5: el objetivo del test es probar el metodo agenciafromAgenciaDTO, para ello se plantea como escenario AgenciaDTO con listas vacías, su estadoContratacion en true y sus límites (999,9999) y Agencia obtenida a partir de AgenciaDTO a partir del método a testear. Los test (set y get) verificarán que los atributos de agencia hayan sido copiados correctamente en Agencia.

getComisionesUsuarios()

Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia
		si	jdoc	1

Número de prueba	Datos de entrada	Valor	Salida Esperada	Clases
1			referencia al objeto ComisionesUsuario de AgenciaDTO	1

getContrataciones()

Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia
		si	jdoc	1

Número de prueba	Datos de entrada	Valor	Salida Esperada	Clases
1			referencia al objeto Contrataciones de AgenciaDTO	1

getEmpleadores()

Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia
		si	jdoc	1

Número de prueba	Datos de entrada	Valor	Salida Esperada	Clases
1			referencia al objeto Empleadores de AgenciaDTO	1

getEmpleados()

Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia
		si	jdoc	1

Número de prueba	Datos de entrada	Valor	Salida Esperada	Clases
1			referencia al objeto Empleados de AgenciaDTO	1

getLimiteInferior()

Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia
		si	jdoc	1

Número de prueba	Datos de entrada	Valor	Salida Esperada	Clases
1			999	1

getLimiteSuperior()

Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia
		si	jdoc	1

Número de prueba	Datos de entrada	Valor	Salida Esperada	Clases
1			9999	1

isEstadoContratacion()

Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia
		si	jdoc	1

Número de prueba	Datos de entrada	Valor	Salida Esperada	Clases
1			true	1

Test 6: el objetivo del test es probar el método escribir de PersistenciaXML, para ello se plantea como escenario una Agencia con elementos en sus listas.

escribir()

Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia
		si	jdoc	1

Número de prueba	Datos de entrada	Valor	Salida Esperada	Clases
1			se crea el archivo y se persiste agencia en él.	1

Test 7: Con el objetivo de testear el método leer de PersistenciaXML, se plantea un escenario de Agencia con elementos en sus listas, la cual se persiste en un archivo, luego se vacian las listas y se modifican el resto de atributos y se despersiste la agencia guardada en el archivo creado. Los test, a través de los getter, verifican que lo desperistido coincida con los elementos originales de agencia.
Agencia se encuentra cargada con un empleado y un empleador, un usuario logueado ("PepePretenso","contrasenia") ,una contratación entre ambos, dicha contratación almacenada tambien en coincidencias, en comisiones de usuarios hay una del empleado pretenso con valor 10.0, los límites (999,9999) y el estado de contratación falso.

getLimiteInferior()

Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia
		si	jdoc	1

Número de prueba	Datos de entrada	Valor	Salida Esperada	Clases
1			999	1

getLimiteSuperior()

Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia
		si	jdoc	1

Número de prueba	Datos de entrada	Valor	Salida Esperada	Clases
1			9999	1

getEmpleadores()

Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia
		si	jdoc	1

Número de prueba	Datos de entrada	Valor	Salida Esperada	Clases
1			referencia a hashmap de empleadores con el empleador cargado (no vacio)	1

getEmpleados()

Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia
		si	jdoc	1

Número de prueba	Datos de entrada	Valor	Salida Esperada	Clases
1			referencia a hashmap de empleados con el empleado cargado (no vacio)	1

isEstadoContratacion()

Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia
		si	jdoc	1

Número de prueba	Datos de entrada	Valor	Salida Esperada	Clases
1			false	1

getContrataciones()

Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia
		si	jdoc	1

Número de prueba	Datos de entrada	Valor	Salida Esperada	Clases
1			referencia a lista de contrataciones con la contratacion cargada (no vacia)	1

getComisionesUsuarios()

Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia
		si	jdoc	1

Número de prueba	Datos de entrada	Valor	Salida Esperada	Clases
1			referencia al hashmap de comisionesUsuarios con la comision cargada (no vacio)	1

getCoincidencias()

Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia
		si	jdoc	1

Número de prueba	Datos de entrada	Valor	Salida Esperada	Clases
1			referencia a lista de coincidencias con la coincidencia cargada (no vacia)	1

getTipoUsuario()

Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia
		si	jdoc	1

Número de prueba	Datos de entrada	Valor	Salida Esperada	Clases
1			-1	1

Test 8: con el objetivo de probar los métodos de persistencia de la clase Agencia, se plantea un escenario de Agencia con sus listas vacías.

Escenario con atributo persistencia null

guardarAgencia()

Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia
nombreArchivo (string)	nombre!=null y nombre valido de archivo	si	jdoc	1
	nombre==null	no	jdoc	2

Número de prueba	Datos de entrada	Valor	Salida Esperada	Clases
1	nombre	"AgenciaPersistencia.xml"	False (se carga correctamente)	1

Escenario con atributo persistencia != null (PersistenciaXML)

guardarAgencia()

Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia
nombreArchivo (string)	nombre!=null y nombre valido de archivo	si	jdoc	1
	nombre==null	no	jdoc	2

Número de prueba	Datos de entrada	Valor	Salida Esperada	Clases
1	nombre	"AgenciaPersistencia.xml"	True (se carga correctamente)	1

Escenario con atributo persistencia null

cargarAgencia()

Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia
nombreArchivo (string)	nombre!=null y nombre valido de archivo	si	jdoc	1
	nombre==null	no	jdoc	2

Número de prueba	Datos de entrada	Valor	Salida Esperada	Clases
1	nombre	"AgenciaPersistencia.xml"	True (se carga correctamente)	1

Escenario con atributo persistencia != null (PersistenciaXML)

cargarAgencia()

Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia
nombreArchivo (string)	nombre!=null y nombre valido de archivo	si	jdoc	1
	nombre==null	no	jdoc	2

Número de prueba	Datos de entrada	Valor	Salida Esperada	Clases
1	nombre	"AgenciaPersistencia.xml"	False (se carga correctamente)	1

2.2 Casos de uso prueba integración - Agencia

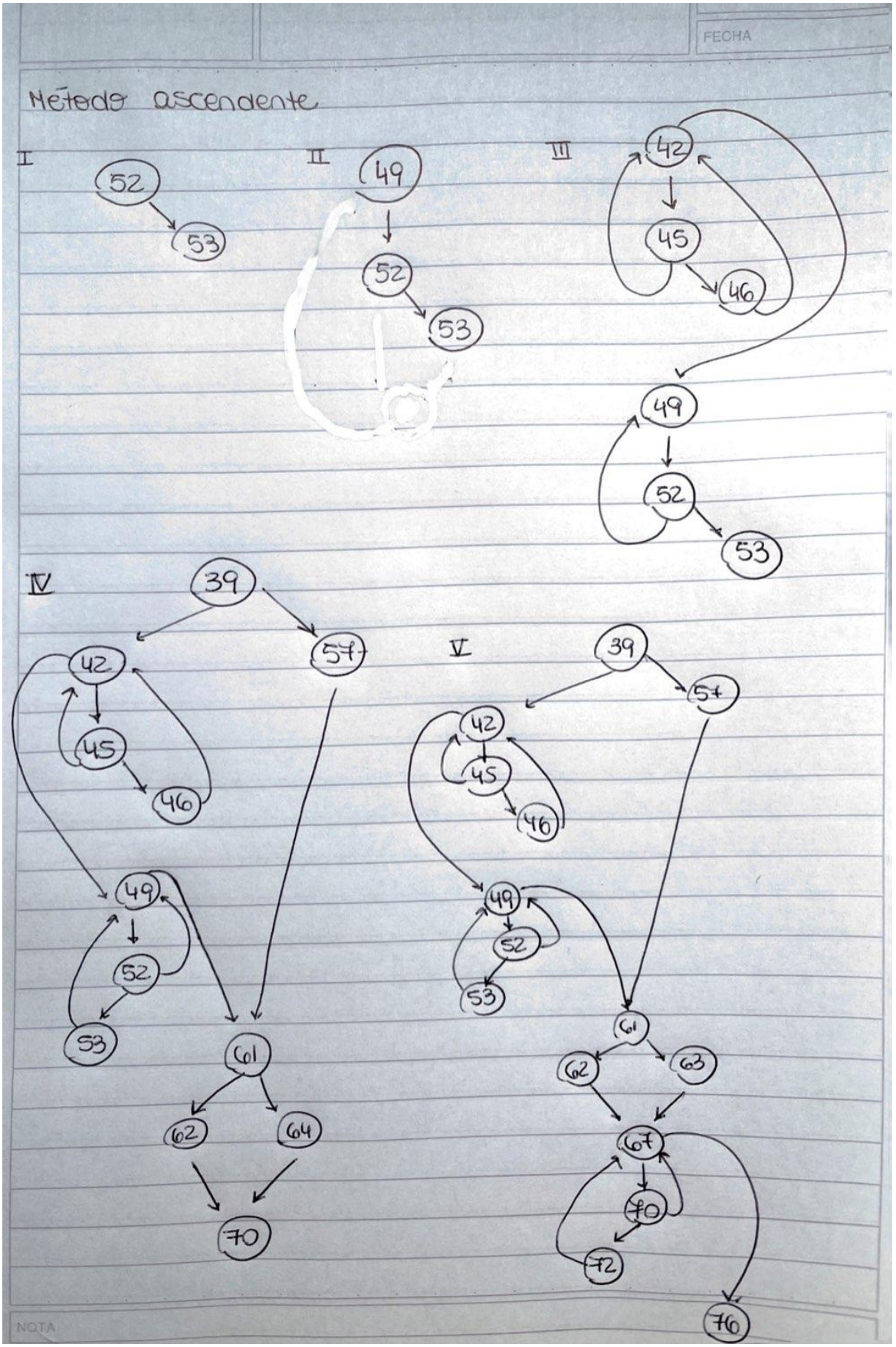
En este apartado, podemos sintetizar en que la prueba de integración estuvo realizada indirectamente en la prueba de la clase Agencia, ya que se esta clase contiene métodos cuyas pruebas comprueban la integración de las pruebas unitarias. Los escenarios del test de agencia se encuentran organizados de forma tal de ir testeando los métodos más elementales, para luego testear haciendo uso de dichos métodos aquellos que son más complejos.

2.3 Prueba unitaria caja blanca

Tomando la siguiente enumeración como referencia:

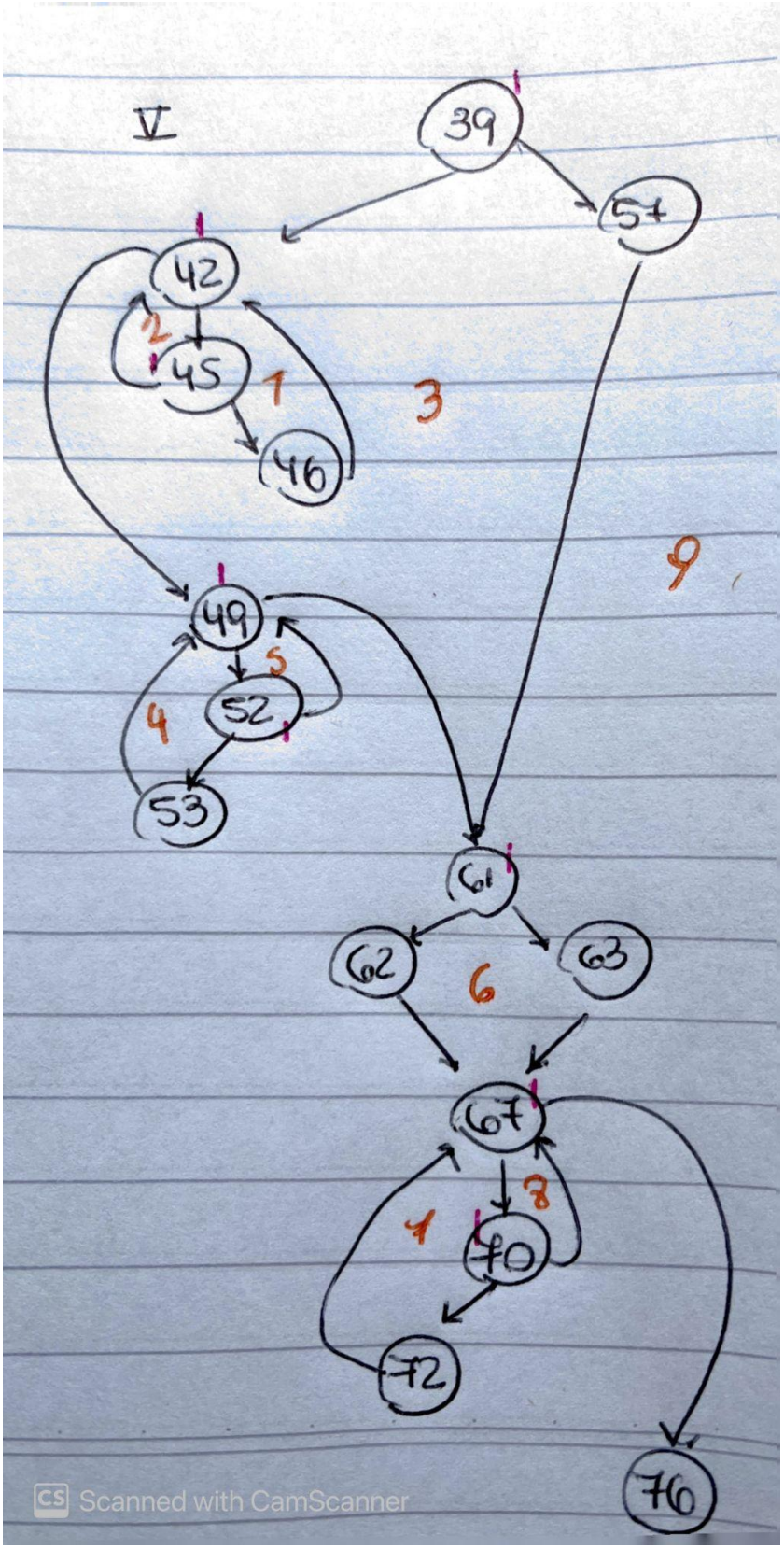
```java
public Cliente aplicaPromo(boolean promoPorListaDePostulantes,
        HashMap<String, EmpleadoPretenso> empleados, HashMap<String, Empleador> empleadores)
{
    Iterator clientes = null;
    int contadorEmpleador = 0;
    int contadorEmpleadoPretenso = 0;
    Cliente clienteBeneficiado = null;

    if (promoPorListaDePostulantes)
    {
        Iterator<Empleador> itEmpleadores = empleadores.values().iterator();
        while (itEmpleadores.hasNext())
        {
            Empleador empleador = itEmpleadores.next();
            if (empleador.getListaDePostulantes() != null)
                contadorEmpleador += empleador.getListaDePostulantes().size();
        }
        Iterator<EmpleadoPretenso> itEmpleados = empleados.values().iterator();
        while (itEmpleados.hasNext())
        {
            EmpleadoPretenso empleadoPretenso = itEmpleados.next();
            if (empleadoPretenso.getListaDePostulantes() != null)
                contadorEmpleadoPretenso += empleadoPretenso.getListaDePostulantes().size();
        }
    } else

    {
        contadorEmpleador = empleadores.size();
        contadorEmpleadoPretenso = empleados.size();
    }

    if (contadorEmpleador > contadorEmpleadoPretenso)
        clientes = empleadores.values().iterator();
    else
        clientes = empleados.values().iterator();

    int puntajeMaximo = Integer.MIN_VALUE;
    while (clientes.hasNext())
    {
        Cliente cl = (Cliente) clientes.next();
        if (cl.getPuntaje() > puntajeMaximo)
        {
            puntajeMaximo = cl.getPuntaje();
            clienteBeneficiado = cl;
        }
    }
    return clienteBeneficiado;
}
```

Grafo de control utilizando el método ascendente:



Cálculo de complejidad ciclomática

CC = nro de aristas - nro de nodos + 2 = 22 - 15 + 2 = 9
CC = nro de regiones cerradas + región total = 8 + 1 = 9
CC = nro de nodos condición + 1 = 8 + 1 = 9



Conjunto básico de caminos independientes usando **el método general**

El camino más significativo puede ser:

C1: 39 - 42 - 45 - 46 - 42 - 49 - 52 - 53 - 49 - 61 - 62 - 67 - 70 - 72 - 67 - 76

Si iniciamos con ese camino, el nodo 45 tiene una salida no ejercitada

C2: 39 - 42 - 45 - 42 - 49 - 52 - 53 - 49 - 61 - 62 - 67 - 70 - 72 - 67 - 76

De nuevo siguiendo por el primer camino, el nodo 42 tiene una salida no ejercitada

C3: 39 - 42 - 49 - 52 - 53 - 49 - 61 - 62 - 67 - 70 - 72 - 67 - 76

De nuevo siguiendo por el primer camino, el nodo 52 tiene una salida no ejercitada

C4: 39 - 42 - 45 - 46 - 42 - 49 - 52 - 49 - 61 - 62 - 67 - 70 - 72 - 67 - 76

De nuevo siguiendo por el primer camino, el nodo 49 tiene una salida no ejercitada

C5: 39 - 42 - 45 - 46 - 42 - 49 - 61 - 62 - 67 - 70 - 72 - 67 - 76

De nuevo siguiendo por el primer camino, el nodo 61 tiene una salida no ejecutada

C6: 39 - 42 - 45 - 46 - 42 - 49 - 52 - 53 - 49 - 61 - 63 - 67 - 70 - 72 - 67 - 76

De nuevo siguiendo por el primer camino, el nodo 70 tiene una salida no ejecutada

C7: 39 - 42 - 45 - 46 - 49 - 52 - 53 - 49 - 61 - 62 - 67 - 70 - 67 - 76

De nuevo siguiendo por el primer camino, el nodo 67 tiene una salida no ejecutada

C8: 39 - 42 - 45 - 46 - 49 - 52 - 53 - 49 - 61 - 62 - 67 - 76

Finalmente, siguiendo el primer camino, el nodo 39 tiene una salida no ejecutada.

C9: 39 - 57 - 61 - 62 - 67 -76

Lo que nos da un total de 9 caminos.

Nro de escenario	Descripción
1	Ambos HashMap vacíos
2	HashMap empleados vacío
3	HashMap empleadores vacío
4	Ambos HashMap no vacíos

Camino independiente	Parámetros de entrada	Resultado esperado	Escenario	
N/A	HashMap empleados vacío HashMap empleadores vacio promoPorListaDePostulantes=true	null	1	
C1	HashMap empleados con 2 postulantes en total HashMap empleadores con 4 postulantes en total promoPorListaDePostulantes=true	Empleador de mayor puntaje	4	
C2	HashMap empleados con 3 postulantes en total HashMap empleadores con 0 postulantes en total promoPorListaDePostulantes=true	Empleador de mayor puntaje	4	No aplica - se llega a una contradicción en el código
C3	HashMap empleados con 2 postulantes en total HashMap empleadores vacía promoPorListaDePostulantes=true	Empleador de mayor puntaje	3	No aplica - se llega a una contradicción en el código
C4	HashMap empleados con 0 postulantes en total HashMap empleadores con 3 postulantes en total promoPorListaDePostulantes=true	Empleador de mayor puntaje	4	
C5	HashMap empleados vacío HashMap empleadores con 2 postulantes en total promoPorListaDePostulantes=true	Empleador de mayor puntaje	2	
C6	HashMap empleados con 4 postulantes en total HashMap empleadores con 2 postulantes en total promoPorListaDePostulantes=true	EmpleadoPretenso de mayor puntaje	4	

C7	HashMap empleados con 2 postulantes en total HashMap empleadores con 3 postulantes en total promoPorListaDePostulantes=true	Empleador de mayor puntaje	4	
C8	HashMap empleados con 2 postulantes en total HashMap empleadores con 3 postulantes en total promoPorListaDePostulantes=true	Empleador de mayor puntaje	4	
C9	HashMap empleados de tamaño 1 HashMap empleadores de tamaño 2 promoPorListaDePostulantes=false	Empleador de mayor puntaje	4	

2.5 Test de GUI.

En el test de GUI (Graphical User Interface) se testea la capa de presentación. Se testea que la ventana responda de acuerdo a lo requerido en la SRS. Se testea que los botones se activan y se desactiven según la secuencia del programa o los campos requeridos, que los recuadros de texto tengan o no permitida la escritura, además se testea que la ventana responda de manera correcta ante los eventos que surgen desde la capa de modelo, como por ejemplo: errores a la hora de procesar información o usuarios repetidos (en el caso de este TP).

Para realizar este tipo de testeo se utiliza la clase Robot del JUnit. Esta clase, tal y como su nombre lo dice, es un Robot que ejecuta acciones como un humano sobre la ventana. Es una herramienta clave para este tipo de testeo dado que automatiza todo el proceso y permite al programador llevar un control exacto del testing que se va realizando.

Además , el programa debe haber sido realizado pensando en el testing posterior, teniendo cada objeto con el que el usuario puede interactuar una variable con su nombre. Este nombre va a ser luego buscado en todo el árbol de componentes para obtener la referencia a ese objeto para poder ser utilizado en el testing con la clase Robot.

Para testear los Option Pane que se muestran cada vez que hay un problema o un evento que surge desde la capa de modelo, el controlador debe tener un setter para poder asignarle un “Falso” Option Pane que nos permita ver que el mensaje mostrado es el que debe mostrar.

En este TP se testean todas las ventanas, la activación de los botones según su especificación y el contenido de los Option Pane con mensajes que surgen desde la capa de modelo. Se utilizó la clase TestUtils provista por el profesor Guillermo Lazzurri que incluía tipeo y borrado en los recuadros de texto, click en los componentes y búsqueda de componentes en el árbol de componentes.

Debido a la gran cantidad de casos y lo costoso de realizar este tipo de testeo se realizaron una serie de “recortes”, todos orientados a evitar caminos muy raros y que los caminos fueran lo más genéricos posibles:

1. En el registro no se testean todas las combinaciones de campos vacíos y llenos, solo las más relevantes. Un solo recuadro de texto vacío y que se tengan que ir completando cada uno de los recuadros para permitir el registro. También se testeo el borrado luego de la escritura de los campos de texto
2. En el Login se testean todos los casos debido a los pocos casos que hay.
3. En la sesión abierta de empleado y empleador se testea que los botones de Nuevo Ticket y aceptar se activen y desactiven. Se testea el caso general de una remuneración normal y los radioButtons al azar.
4. En el Administrador se testean los límites , el botón de Cambiar y el de cerrar sesión.

2.6 Excepciones

A lo largo de este informe y la presentación de los clases y batería de pruebas, se ha incluido el testeo de las excepciones. Esto abarca tanto el chequeo de que estas se lancen debidamente, como comprobar que NO se lancen incorrectamente. Entre las excepciones que fueron testeadas se encuentran:

- Limite superior remuneración invalida: cuando se manda un límite superior menor al inferior.
- Limite inferior remuneración invalida: cuando se manda un límite inferior negativo.
- imposiblemodificartickets: cuando se intenta crear o eliminar un ticket y el estado de la contratación está en Verdadero
- NewRegisterException: nombre de usuario ya usado cuando se quiere registrar un cliente.
- ImposibleCrear empleado/empleador: al metodo le llega un parámetro en null o no en constantes.
- ContraException: contraseña incorrecta para el login
- NombreUsuarioExc: no se encontró usuario con el nombre para el login

2.7 Recorte

En el testing es imposible tener una cobertura de los errores absoluta del código, y por ende debemos ir definiendo el alcance de los test acorde a los recursos y el tiempo. En esta ocasión, optamos por no repetir cada caso de prueba en cada escenario, si no que los que métodos cuyos resultados no varíen, testear el método en uno solo de los escenarios.

3.Conclusiones:

El testing es una herramienta muy poderosa a la hora de buscar y encontrar errores en el código. Estos errores deben ser encontrados rápidamente en etapas previas a la puesta en producción del código para que el cliente pueda hacer un uso del software de acuerdo a lo acordado con nosotros o nuestra empresa. Un buen testing además otorga

calidad a nuestro software debido a que podemos garantizar, con un gran nivel de confianza, que el software no le va a generar problemas a futuro a nuestros usuarios.
Particularmente, el test de GUI es muy engorroso de realizar. Programar este tipo de testeos es una tarea que demanda mucho tiempo y una cantidad de código muy grande. Es por eso que no es recomendable testear todos los casos, solo los caminos más comunes.
Cabe destacar, que para la correcta realización del testing es fundamental una adecuada documentación, ya que sin esta es imposible testear y verificar que se cumplan los contratos.
En conclusión, podemos afirmar que mediante un buen testing y una buena documentación se asegura la calidad y confiabilidad , aspecto fundamental a la hora de desarrollar un sistema.