

Trabajo práctico taller de programación

Grupo 02

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



Índice

1. Aspectos a considerar

2.Documentación:

- 2.1 Casos de prueba de pruebas unitarias caja negra
 - 2.1.1 Clase admin
 - 2.1.2 Clase ClientePuntaje
 - 2.1.3 Clase Empleador
 - 2.1.4 Clase EmpleadoPretenso
 - 2.1.5 Contratación
 - 2.1.6 <u>Ticket</u>
- 2.2 Casos de uso y casos de prueba para test de integración Agencia
- 2.3 Prueba unitaria caja blanca
- 2.4 Test de Persitencia
- 2.5 Test de GUI
- 2.6 Excepciones
- 2.7 Recorte

3. Conclusiones



1. Aspectos a considerar

- Las clases abstractas *Cliente* y *Usuario* no se testean, justamente por ser abstractas.
 - o 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).
 - o 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

assword (string) y getPasswo					
Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia	
Password (string)	password=null	no	jdoc	1	
()	password!=null	Sİ	jdoc	2	
Número de prueba	Datos de entrada	\	/alor	Salida Esperada	Clase
1	password	"07	78hd90"	seteo correcto (que se verifica con el get)	2
JsserName(string) y getUsserI	Name()				
Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia	
Haranklana (string)	UsserName=null	no	jdoc	1	
UsserName (string)	UsserName!=null	si	jdoc	2	
	Datos de entrada	\	/alor	Salida Esperada	Clase
Número de prueba					
	UsserName me()		nceto01"	seteo correcto (que se verifica con el get)	2
		"jua	nceto01"	seteo correcto (que se verifica con el get)	2
1 RealName(string) y getRealNa	UsserName me()				2
	UsserName me() Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia	2
1 RealName(string) y getRealNa	UsserName me() Clases de equivalencia RealName=null	Aplica ?	Motivo jdoc	Identificador de clase de equivalencia	2
1 RealName(string) y getRealNa Dato de entrada	UsserName me() Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia	2
1 RealName(string) y getRealNa Dato de entrada	UsserName me() Clases de equivalencia RealName=null	Aplica ?	Motivo jdoc	Identificador de clase de equivalencia	
1 RealName(string) y getRealNa Dato de entrada RealName (string)	me() Clases de equivalencia RealName=null RealName!=null	Aplica ?	Motivo jdoc jdoc	Identificador de clase de equivalencia 1 2	
1 RealName(string) y getRealNa Dato de entrada RealName (string) Número de prueba	me() Clases de equivalencia RealName=null RealName!=null Datos de entrada RealName	Aplica ?	Motivo jdoc jdoc	Identificador de clase de equivalencia 1 2 Salida Esperada	Clase
1 RealName(string) y getRealNa Dato de entrada RealName (string) Número de prueba	UsserName me() Clases de equivalencia RealName=null RealName!=null Datos de entrada RealName	Aplica ?	Motivo jdoc jdoc	Identificador de clase de equivalencia 1 2 Salida Esperada seteo correcto (que se verifica con el get)	Clase
1 RealName(string) y getRealNa Dato de entrada RealName (string) Número de prueba 1 Telefono(string) y getTelefono Dato de entrada	me() Clases de equivalencia RealName=null RealName!=null Datos de entrada RealName	Aplica ? no si	Motivo jdoc jdoc valor nn Aceto"	Identificador de clase de equivalencia 1 2 Salida Esperada	Clase
1 RealName(string) y getRealNa Dato de entrada RealName (string) Número de prueba 1 Felefono(string) y getTelefono	UsserName me() Clases de equivalencia RealName=null RealName!=null Datos de entrada RealName () Clases de equivalencia	Aplica ? no si "Jua	Motivo jdoc jdoc valor in Aceto"	Identificador de clase de equivalencia 1 2 Salida Esperada seteo correcto (que se verifica con el get) Identificador de clase de equivalencia	Clase
1 RealName(string) y getRealNa Dato de entrada RealName (string) Número de prueba 1 Telefono(string) y getTelefono Dato de entrada	UsserName me() Clases de equivalencia RealName=null RealName!=null Datos de entrada RealName () Clases de equivalencia Telefono=null	Aplica ? no si "Jua Aplica ? no si	Motivo jdoc jdoc valor n Aceto" Motivo jdoc	Identificador de clase de equivalencia 1 2 Salida Esperada seteo correcto (que se verifica con el get) Identificador de clase de equivalencia	Clase

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	Va	lor	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	Va	lor	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.

Clases de equivalencia	Aplica?	Motivo	Identificador de clase de equivalencia	
Objeto = ClientePuntaje (puntaje mayor al obj cliente)	si	JDoc I. Comparable	1	
Objeto = ClientePuntaje (puntaje igual al obj cliente)	Sİ	JDoc I. Comparable	2	
Objeto = ClientePuntaje (puntaje menor al obj cliente)	Sİ	JDoc I. Comparable	3	
Objeto = null	si	JDoc I. Comparable	4	
Objeto !=ClientePuntaje	Sİ	JDoc I. Comparable	5	
Datos de entrada		Valor	Salida Esperada	Clases
Objeto	ClientePu		-1	1
Objeto	ClientePu	ntaje (puntaje=30)	0	2
Objeto	ClientePu	ntaje (puntaje=20)	1	3
Objeto		null	throws NullPointerException	4
Objeto	E	mpleador	throws ClassCastException	5
(Objeto = ClientePuntaje (puntaje mayor al obj cliente) Objeto = ClientePuntaje (puntaje igual al obj cliente) Objeto = ClientePuntaje (puntaje igual al obj cliente) Objeto = ClientePuntaje (puntaje menor al obj cliente) Objeto = null Objeto !=ClientePuntaje Datos de entrada Objeto Objeto Objeto Objeto	Objeto = ClientePuntaje (puntaje mayor al obj cliente) Objeto = ClientePuntaje (puntaje igual al obj cliente) Objeto = ClientePuntaje (puntaje igual al obj cliente) Objeto = ClientePuntaje (puntaje menor al obj cliente) Objeto = null si Objeto !=ClientePuntaje si Datos de entrada Objeto Objeto ClientePuntaje Objeto ClientePuntaje	Objeto = ClientePuntaje (puntaje mayor al obj cliente) Objeto = ClientePuntaje (puntaje igual al obj cliente) Objeto = ClientePuntaje (puntaje igual al obj cliente) Objeto = ClientePuntaje (puntaje menor al obj cliente) Objeto = null Objeto !-ClientePuntaje Objeto !-ClientePuntaje Datos de entrada Objeto Objeto Objeto ClientePuntaje (puntaje=40) Objeto Objeto ClientePuntaje (puntaje=20) Objeto Null	Objeto = ClientePuntaje (puntaje mayor al obj cliente) Si JDoc I. Comparable 1 Dipieto = ClientePuntaje (puntaje igual al obj cliente) Objeto = ClientePuntaje (puntaje si JDoc I. Comparable Objeto = ClientePuntaje (puntaje si JDoc I. Comparable Objeto = null si JDoc I. Comparable Objeto = null si JDoc I. Comparable Objeto !=ClientePuntaje si JDoc I. Comparable Objeto !=ClientePuntaje si JDoc I. Comparable Datos de entrada Valor Salida Esperada Objeto ClientePuntaje (puntaje=40) Objeto ClientePuntaje (puntaje=30) Objeto ClientePuntaje (puntaje=20) Objeto null throws NullPointerException

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	Va	lor	Salida Esperada	Clases
1	Puntaje		0	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	
Cliente	Objeto distinto de tipo Cliente	no	jdoc	2	
Número de prueba	Datos de entrada	Va	lor	Salida Esperada	Clases
1	Cliente		mpleador itos 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 ?		Identificador de clase de equivalencia	
	-	si	jdoc	1	
Número de prueba	Datos de entrada	Va	lor	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 do prueba	Dates de entrada	1/2	lor	Calida Esperada	Clean
Número de prueba 1	Datos de entrada	Va	Ю	Salida Esperada "gwerty123" pasada por constructor	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	Va	lor	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	Va	lor	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	Va	lor	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	Va	lor	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	Va	lor	Salida Esperada	Clases
1	Ticket	Ticket(HOME_OFFIC E, 80000, JORNADA_COMPLET A, JUNIOR, EXP_MEDIA, TERCIARIOS)			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	
Kubio	Rubro ∉ util.Constantes(rubros)	no	jdoc	2	
Número de prueba	Datos de entrada	Va	lor	Salida Esperada	Clases
1	Rubro	util.Consta	intes.SALU)	seteo correcto (que se verifica con el get)	1
setTipoPersona()					
Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia	
D. L.	Rubro ∈ util.Constantes (tipoPersona)	si	jdoc	1	
Rubro	Rubro ∉ util.Constantes(tipoPersona)	no	jdoc	2	
Número de prueba	Datos de entrada	Va	lor	Salida Esperada	Clases
1	TipoPersona	util.Consta	antes.JURI CA	seteo correcto (que se verifica con el get)	1



etTelefono					
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	Va	lor	Salida Esperada	Clases
1	telefono	"+549223	6661234"	seteo correcto (que se verifica con el get)	1
etRealName					
Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia	
realName(String)	realName!=null	si	jdoc	1	
realivanie(string)	realName=null	no	jdoc	2	
Número de prueba	Datos de entrada	\/a	lor	Salida Esperada	Clase
1		"Juan		· · · · · · · · · · · · · · · · · · ·	1
1	realName	Juan	Cito	seteo correcto (que se verifica con el get)	
etPassword					
Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia	
11-11-1	password!=null	si	jdoc	1	
password(String)	password=null	no	jdoc	2	
<u> </u>					
Número de prueba	Datos de entrada	Va	lor	Salida Esperada	Clase:
1	password	"qwer	ty123"	seteo correcto (que se verifica con el get)	1
etUsserName()					
Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia	
	usserName!=null	si	jdoc	1	
usserName(String)	usserName=null	no	jdoc	2	
Número de prueba	Datos de entrada	_	lor	Salida Esperada	Clase
1	usserName	"juar	icito"	seteo correcto (que se verifica con el get)	1
alculaComision()					
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	Va	lor	Salida Esperada	Clase
			ME_OFFIC		
			0000,		
1	Ticket		_COMPLET NIOR,	48000	1

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

A, JUNIOR, EXP_MEDIA, TERCIARIOS)

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	
anallida(String)	apellido!=null	si	jdoc	1	
apellido(String)	apellido=null	no	jdoc	2	
Número de prueba	Datos de entrada	Va	lor	Salida Esperada	Clases
1	apellido	"Rodr	iguez"	seteo correcto (que se verifica con el get)	1
setEdad					
Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia	
II: d-(0b-i)	edad>0	si	jdoc	1	
apellido(String)	edad<0	no	jdoc	2	
Número de prueba	Datos de entrada	Va	lor	Salida Esperada	Clases
1	edad	3	6	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	Va	lor	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	Va	lor	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	Va	lor	Salida Esperada	Clases
			ME_OFFIC		
	Ticket		_COMPLET	64000	
1	licket		NIOR,	64000	1
		_	ИEDIA,		
		TERCIA	ARIOS)		

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	Va	lor	Salida Esperada	Clases
1	-		-	Candidato = referencia a Empleador seteado	1



etListaPostulantes()					
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	Va	lor	Salida Esperada	Clase
1	ArrayList con 1 elemento	ArrayList con Empleador("juancit o", "qwerty123", "Juan Cito", "+54 9 223 666-1234", util.Constantes.SALU D, util.Constantes.JURI			1
2	ArrayList vacio	ArrayLi	st = " "	ArrayList vacio coincidente con el seteado	2
etListaPostulantes()					
Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia	
-	- '	si	jdoc	1	
Número de prueba	Datos de entrada	Va	lor	Salida Esperada	Clase
1	-	-		ArrayList coincidente con el seteado	1
etPuntaje()					
Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia	
-	-	si	jdoc	1	
Número de prueba	Datos de entrada	Va	lor	Salida Esperada	Clase
1	-			puntaje = 28(seteado)	1
tTicket()					
Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia	
-	-	si	jdoc	1	
Número de prueba	Datos de entrada	Va	or	Salida Esperada	Clase
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	Va	lor	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	Va	lor	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	Va	lor	Salida Esperada	Clases
				referencia la objeto Fecha generado en	
				constructor (lo testeamos asegurando que	
1				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	
	obj tipo EmpleadoPretenso	si	jdoc	1	
empleado	obj NO tipo EmpleadoPretenso	no	jdoc	2	
Número de prueba	Datos de entrada	Va	lor	Salida Esperada	Clases
		Empleado			
1	empleado	con atribu	tos vacios	seteo correcto (que se verifica con el get)	1
setEmpleador()					
Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia	
	obj tipo Empleador	si	jdoc	1	
empleador	obj NO tipo Empleador	no	jdoc	2	
Número de prueba	Datos de entrada	Va		Salida Esperada	Clases
1	empleador	objeto Er con atribu		seteo correcto (que se verifica con el get)	1
setfecha()					
Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia	
	obj tipo GregorianCalendar	si	jdoc	1	
fecha	obj NO tipo GregorianCalendar	no	jdoc	2	
Número de prueba	Datos de entrada	Va	lor	Salida Esperada	Clases
		obj Gregoriar	eto ıCalendar		·
1	fecha	va	cio	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.



etEstudios					
Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia	
-	-	si	jdoc	1	
Número de prueba	Datos de entrada	\/a	lor	Salida Esperada	Cl
Número de prueba	Datos de efficada	Valor		-	Clases
1	-		-	util.Constantes.PRIMARIOS	1
etExperiencia					
Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia	
-	-	si	jdoc	1	
NIZ	Data da saturda			Outido Foresado	
Número de prueba	Datos de entrada		lor	Salida Esperada	Clases
1	-		-	util.Constantes.EXP_MEDIA	1
etJornada					
Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia	
-	-	si	jdoc	1	
Número de prueba	Datos de entrada	Va	lor	Salida Esperada	Clase
1	-		-	util.Constantes.JORNADA_COMPLETA	1
etLocacion					
Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia	
-	-	si	jdoc	1	
Mércan de montes	Data da ante da	17-	1	0-11-5	
Número de prueba	Datos de entrada		lor	Salida Esperada	Clase
1	-		-	util.Constantes.PRESENCIAL	1
etPuesto					
Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia	
-	-	si	jdoc	1	
NIÁmeno do muelo -	Dates de entre de	1.7-	los	Calida Fanarada	
Número de prueba	Datos de entrada	va	lor	Salida Esperada	Clase
1	-		-	util.Constantes.JUNIOR	1
etRemuneracion					
Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia	
-	-	si	jdoc	1	
MAnnan de ancete	Datas da catas de		1	Calida Sanarada	
Número de prueba	Datos de entrada		lor	Salida Esperada	Clase
1	-		-	10000	1

Escenario nro 2: Ticket sin parámetros Testeamos los setters (e indirectamente los getters)

Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia	
!= null	si	jdoc	1	
= tipo esperado dentro de Constantes	si	jdoc	2	
= null	no	jdoc	3	
!= tipo esperado dentro de Constantes	no	jdoc	4	
Datos de entrada	Va	lor	Salida Esperada	Clases
experiencia	util.Constantes.E	XP_MEDIA	setteo correcto testeado con el get	1,2
	I= null = tipo esperado dentro de Constantes = null I= tipo esperado dentro de Constantes Datos de entrada	= null si = tipo esperado dentro de Constantes si = null no = tipo esperado dentro de Constantes no Datos de entrada Va	= null si jdoc = tipo esperado dentro de Constantes si jdoc = null no jdoc = tipo esperado dentro de Constantes no jdoc Datos de entrada Valor	= null



etEstudios					
Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia	
	!= null	si	jdoc	1	
F	= tipo esperado dentro de Constantes	si	jdoc	2	
Estudios	= null	no	jdoc	3	
	!= tipo esperado dentro de Constantes	no	jdoc	4	
Número de prueba	Datos de entrada	Va	lor	Salida Esperada	Clases
1	estudios	util.Constantes.I		setteo correcto testeado con el get	1,2
tJornada					
Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia	
Dato de entrada	!= null	si si	idoc	1	
	= tipo esperado dentro de Constantes	si	jdoc	2	
Jornada	= null	no	idoc	3	
	!= tipo esperado dentro de Constantes	no	jdoc	4	
Al consists	us sonotantes				
tLocacion Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia	
Dato de entrada	!= null	si	idoc	1	
	= tipo esperado dentro de Constantes	si	jdoc	2	
Locacion	= null	no	idoc	3	
	!= tipo esperado dentro de Constantes	no	jdoc	4	
Número de prueba	Datos de entrada	Va	lor	Salida Esperada	Clases
1	locacion	util.Constantes.F	RESENCIAL	setteo correcto testeado con el get	1,2
tPuesto					
Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia	
	!= null	si	idoc	1	
	= tipo esperado dentro de Constantes	si	jdoc	2	
puesto	= null	no	jdoc	3	
	!= tipo esperado dentro de Constantes	no	jdoc	4	
Número de prueba	Datos de entrada	Va	lor	Salida Esperada	Clases
1	puesto	util.Constantes		setteo correcto testeado con el get	1,2
tRemuneracion					
Dato de entrada	Clases de equivalencia != null	Aplica ?	Motivo jdoc	Identificador de clase de equivalencia 1	
romuneracies	>0	si	jdoc	2	
remuneracion	= null	no	jdoc	3	
	<0	no	jdoc	4	
Número de prueba	Datos de entrada	Va	lor	Salida Esperada	Clases
		•••			

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

Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia	
Dato de entrada		•		•	
Ticket	Ticket!=null	Sİ	jdoc	2	
TICKET	Ticket=null	no	jdoc	2	
Número de prueba	Datos de entrada	Valo	r	Salida Esperada	Clases
1	Ticket	Ticket(paramet	ros tipo 1)	1	1
etComparacionRemuneracion	()				
Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia	
	Ticket!=null	si	jdoc	1	
Ticket	Ticket=null	no	jdoc	2	
Mónaga da antaba	Datas da satur da	1/-1-		Onlide Ferrenda	
Número de prueba	Datos de entrada	Valor		Salida Esperada	Clases
1	Ticket	Ticket(paramet	ros tipo 1)	1	1
etComparacionJornada()					
Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia	
	Ticket!=null	si	jdoc	1	
Ticket	Ticket=null	no	jdoc	2	
- Transac	Transaction		,	_	
Número de prueba	Datos de entrada	Valo	r	Salida Esperada	Clases
1	Ticket	Ticket(paramet	ros tipo 1)	1	1
etComparacionPuesto()					
Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia	
	Ticket!=null	si	jdoc	1	
Ticket	Ticket=null	no	jdoc	2	
Número de prueba	Datos de entrada	Valo	r	Salida Esperada	Clase
1	Ticket	Ticket(paramet		Janua Esperada	1
-	Tienet	ricket(paramet	103 tipo 17	-	
etComparacionExperiencia()					
Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia	
	Ticket!=null	si	jdoc	1	
Ticket	Ticket=null	no	jdoc	2	
Número de prueba	Datos de entrada	Valo	r	Salida Esperada	Clase:
1	Ticket	Ticket(paramet	ros tipo 1)	1	1
etComparacionEstudios()	Oleana de controlar i	A = 1: O	Methic		
Dato de entrada	Clases de equivalencia	Aplica ?	Motivo		
	Ticket!=null	Sİ	jdoc	1	
Ticket	Ticket=null	no	jdoc	2	
Número de prueba	Datos de entrada	Valor		Salida Esperada	Clases
1	Ticket	Ticket(paramet		1	1
-		- I paramet			
etComparacionTotal()					
Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia	
	Ticket!=null	si	jdoc	1	
Ticket	Ticket=null	no	jdoc	2	
Número de prueba	Datos de entrada	Valor	r	Salida Esperada	Clases
	Ticket	Ticket (paramet		6	1



Escenario nro 4: Ticket12

etComparacionLocacion(Ticket)	0	A 11 O		11 65 1 1 1 1 1 1 1	
Dato de entrada	Clases de equivalencia	_	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(par	ametros tipo 2)	-1	1
etComparacionRemuneracion()					
Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia	
Dato de elitrada	Ticket!=null			1	
Ticket	Ticket:=null	si no	jdoc jdoc	2	
	Hicket-Hull	110	Juoc	2	
Número de prueba	Datos de entrada	,	Valor	Salida Esperada	Clases
1	Ticket		ametros tipo 2)	-0,5	1
-	. Tonet	Trence (par	e 05 tipo 27	3,2	
etComparacionJornada()					
Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia	
	Ticket!=null	si si	jdoc	1	
Ticket	Ticket=null	no	jdoc	2	
Número de prueba	Datos de entrada	,	Valor	Salida Esperada	Clases
1	Ticket	Ticket(par	ametros tipo 2)	-0,5	1
etComparacionPuesto()					
Dato de entrada	Clases de equivalencia	Aplica?	Motivo	Identificador de clase de equivalencia	
Ti-14	Ticket!=null si jdoc 1	1			
Ticket	Ticket=null	no	jdoc	2	
Número de prueba	Datos de entrada	١	/alor	Salida Esperada	Clases
1	Ticket	Ticket(par	ametros tipo 2)	-0,5	1
etComparacionExperiencia()					
Dato de entrada	Clases de equivalencia	Aplica?	Motivo	Identificador de clase de equivalencia	
Tieleet	Ticket!=null	si	jdoc	1	
Ticket	Ticket=null	no	jdoc	2	
Número de prueba	Datos de entrada	'	Valor	Salida Esperada	Clases
1	Ticket	Ticket(par	ametros tipo 2)	1,5	1
			. ,		-
etComparacionEstudios() Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia	-
Dato de entrada	Ticket!=null	si	Motivo jdoc	1	
			Motivo		-
Dato de entrada Ticket	Ticket!=null Ticket=null	si no	Motivo jdoc jdoc	1 2	
Dato de entrada Ticket Número de prueba	Ticket!=null Ticket=null Datos de entrada	si no	Motivo jdoc jdoc	1 2 Salida Esperada	Clases
Dato de entrada Ticket	Ticket!=null Ticket=null	si no	Motivo jdoc jdoc	1 2	
Dato de entrada Ticket Número de prueba 1	Ticket!=null Ticket=null Datos de entrada	si no	Motivo jdoc jdoc	1 2 Salida Esperada	Clases
Dato de entrada Ticket Número de prueba 1 etComparacionTotal()	Ticket!=null Ticket=null Datos de entrada Ticket	si no Ticket(par	Motivo jdoc jdoc Valor ametros tipo 2)	1 2 Salida Esperada 1,5	Clases
Dato de entrada Ticket Número de prueba 1	Ticket!=null Ticket=null Datos de entrada Ticket Clases de equivalencia	si no Ticket(par	Motivo jdoc jdoc Valor ametros tipo 2)	Salida Esperada 1,5 Identificador de clase de equivalencia	Clases
Dato de entrada Ticket Número de prueba 1 etComparacionTotal()	Ticket!=null Ticket=null Datos de entrada Ticket Clases de equivalencia Ticket!=null	si no Ticket(par Aplica ?	Motivo jdoc jdoc Valor ametros tipo 2) Motivo jdoc	Salida Esperada 1,5 Identificador de clase de equivalencia 1	Clases
Dato de entrada Ticket Número de prueba 1 etComparacionTotal() Dato de entrada	Ticket!=null Ticket=null Datos de entrada Ticket Clases de equivalencia	si no Ticket(par	Motivo jdoc jdoc Valor ametros tipo 2)	Salida Esperada 1,5 Identificador de clase de equivalencia	Clases
Dato de entrada Ticket Número de prueba 1 etComparacionTotal() Dato de entrada Ticket	Ticket!=null Ticket=null Datos de entrada Ticket Clases de equivalencia Ticket!=null Ticket=null	si no Ticket(par Aplica ?	Motivo jdoc jdoc Valor ametros tipo 2) Motivo jdoc jdoc	Salida Esperada 1,5 Identificador de clase de equivalencia 1 2	Clases 1
Número de prueba 1 etComparacionTotal() Dato de entrada	Ticket!=null Ticket=null Datos de entrada Ticket Clases de equivalencia Ticket!=null	si no Ticket(par Aplica ?	Motivo jdoc jdoc Valor ametros tipo 2) Motivo jdoc	Salida Esperada 1,5 Identificador de clase de equivalencia 1	Clases

Escenario nro 5: Ticket13



etComparacionLocacion(Ticket)					
Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia	
Tieles	Ticket!=null	si	jdoc	1	
Ticket	Ticket=null	no	jdoc	2	
Número de prueba	Datos de entrada	V	/alor	Salida Esperada	Clases
1	Ticket	Ticket(para	metros tipo 3)	1	1
etComparacionRemuneracion()					
Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia	
	Ticket!=null	si	jdoc	1	
Ticket	Ticket=null	no	jdoc	2	
Número de prueba	Datos de entrada	V	/alor	Salida Esperada	Clases
1	Ticket		metros tipo 3)	-1	1
1	Ticket	пскецрага	imetros tipo 3)	-1	1
etComparacionJornada()					
Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia	
	Ticket!=null	si	jdoc	1	
Ticket	Ticket=null	no	jdoc	2	
			,		
Número de prueba	Datos de entrada	V	/alor	Salida Esperada	Clase
1	Ticket		metros tipo 3)	-1	1
etComparacionPuesto()					
Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia	
	Ticket!=null	si si	jdoc	1	
Ticket	Ticket=null	no	jdoc	2	
			,		
Número de prueba	Datos de entrada	V	/alor	Salida Esperada	Clases
1	Ticket	Ticket(para	metros tipo 3)	-1	1
	•				
etComparacionExperiencia()					
Dato de entrada	Clases de equivalencia	Anlica 2	Motivo	Identificador de clase de equivalencia	
Dato de entrada	Clases de equivalencia		IVIOLIVO	identificador de ciase de equivalencia	
	Ticketl-null		idos	1	
Ticket	Ticket!=null	si	jdoc	1	
Ticket	Ticket!=null Ticket=null	si no	jdoc jdoc	2	
	Ticket=null	no	jdoc	2	Class
Número de prueba	Ticket=null Datos de entrada	no V	jdoc /alor	2 Salida Esperada	
	Ticket=null	no V	jdoc	2	Clase
Número de prueba 1	Ticket=null Datos de entrada	no V	jdoc /alor	2 Salida Esperada	
Número de prueba 1 etComparacionEstudios()	Ticket=null Datos de entrada Ticket	no V Ticket(para	jdoc /alor ametros tipo 3)	2 Salida Esperada 2	
Número de prueba 1 etComparacionEstudios() Dato de entrada	Ticket=null Datos de entrada Ticket Clases de equivalencia	no V Ticket(para	jdoc /alor metros tipo 3) Motivo	Salida Esperada 2 Identificador de clase de equivalencia	Clase 1
Número de prueba 1 etComparacionEstudios()	Ticket=null Datos de entrada Ticket Clases de equivalencia Ticket!=null	no V Ticket(para	jdoc /alor metros tipo 3) Motivo jdoc	Salida Esperada 2 Identificador de clase de equivalencia 1	
Número de prueba 1 etComparacionEstudios() Dato de entrada	Ticket=null Datos de entrada Ticket Clases de equivalencia	no V Ticket(para	jdoc /alor metros tipo 3) Motivo	Salida Esperada 2 Identificador de clase de equivalencia	
Número de prueba 1 etComparacionEstudios() Dato de entrada Ticket	Ticket=null Datos de entrada Ticket Clases de equivalencia Ticket!=null Ticket=null	NO Ticket(para Aplica?	jdoc /alor metros tipo 3) Motivo jdoc jdoc	Salida Esperada 2 Identificador de clase de equivalencia 1 2	1
Número de prueba 1 etComparacionEstudios() Dato de entrada Ticket Número de prueba	Ticket=null Datos de entrada Ticket Clases de equivalencia Ticket!=null Ticket=null Datos de entrada	no V Ticket(para Aplica? si no	jdoc /alor ametros tipo 3) Motivo jdoc jdoc /alor	Salida Esperada 2 Identificador de clase de equivalencia 1 2 Salida Esperada	1 Clase:
Número de prueba 1 etComparacionEstudios() Dato de entrada Ticket	Ticket=null Datos de entrada Ticket Clases de equivalencia Ticket!=null Ticket=null	no V Ticket(para Aplica? si no	jdoc /alor metros tipo 3) Motivo jdoc jdoc	Salida Esperada 2 Identificador de clase de equivalencia 1 2	
Número de prueba 1 etComparacionEstudios() Dato de entrada Ticket Número de prueba 1	Ticket=null Datos de entrada Ticket Clases de equivalencia Ticket!=null Ticket=null Datos de entrada	no V Ticket(para Aplica? si no	jdoc /alor ametros tipo 3) Motivo jdoc jdoc /alor	Salida Esperada 2 Identificador de clase de equivalencia 1 2 Salida Esperada	Clases
Número de prueba 1 etComparacionEstudios() Dato de entrada Ticket Número de prueba 1	Ticket=null Datos de entrada Ticket Clases de equivalencia Ticket!=null Ticket=null Datos de entrada Ticket	Aplica ? si no Violate (para	jdoc /alor ametros tipo 3) Motivo jdoc jdoc /alor ametros tipo 3)	Salida Esperada 2 Identificador de clase de equivalencia 1 2 Salida Esperada 2	1 Clase:
Número de prueba 1 etComparacionEstudios() Dato de entrada Ticket Número de prueba 1 etComparacionTotal() Dato de entrada	Ticket=null Datos de entrada Ticket Clases de equivalencia Ticket!=null Ticket=null Datos de entrada Ticket Clases de equivalencia	Aplica ? Aplica ? Vincket(para	jdoc /alor ametros tipo 3) Motivo jdoc jdoc /alor ametros tipo 3)	Salida Esperada 2 Identificador de clase de equivalencia 1 2 Salida Esperada 2 Identificador de clase de equivalencia	1 Clase:
Número de prueba 1 etComparacionEstudios() Dato de entrada Ticket Número de prueba 1	Ticket=null Datos de entrada Ticket Clases de equivalencia Ticket!=null Ticket=null Datos de entrada Ticket Clases de equivalencia Ticket	Aplica ? Si no Vicket(para	jdoc /alor metros tipo 3) Motivo jdoc /alor metros tipo 3) Motivo jdoc	Salida Esperada 2 Identificador de clase de equivalencia 1 2 Salida Esperada 2 Identificador de clase de equivalencia 1 1 2 Identificador de clase de equivalencia 1	Clases
Número de prueba 1 etComparacionEstudios() Dato de entrada Ticket Número de prueba 1 etComparacionTotal() Dato de entrada	Ticket=null Datos de entrada Ticket Clases de equivalencia Ticket!=null Ticket=null Datos de entrada Ticket Clases de equivalencia	Aplica ? Aplica ? Vincket(para	jdoc /alor ametros tipo 3) Motivo jdoc jdoc /alor ametros tipo 3)	Salida Esperada 2 Identificador de clase de equivalencia 1 2 Salida Esperada 2 Identificador de clase de equivalencia	1 Clase:
Número de prueba 1 etComparacionEstudios() Dato de entrada Ticket Número de prueba 1 etComparacionTotal() Dato de entrada	Ticket=null Datos de entrada Ticket Clases de equivalencia Ticket!=null Ticket=null Datos de entrada Ticket Clases de equivalencia Ticket	Aplica ? Si no Vicket(para	jdoc /alor metros tipo 3) Motivo jdoc /alor metros tipo 3) Motivo jdoc	Salida Esperada 2 Identificador de clase de equivalencia 1 2 Salida Esperada 2 Identificador de clase de equivalencia 1 1 2 Identificador de clase de equivalencia 1	Clases



Escenario nro 6: Ticket21

		_			
Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia	
Ticket	Ticket!=null	si	jdoc	1	
Heret	Ticket=null	no	jdoc	2	
Número de prueba	Datos de entrada	\	/alor	Salida Esperada	Clase
1	Ticket		metros tipo 1)	-1	1
-	Hicket	Ticket(pare	inetros tipo 1 _j	-1	
tComparacionRemuneracion()					
Dato de entrada	Clases de equivalencia	Aplica?	Motivo	Identificador de clase de equivalencia	
Tieleet	Ticket!=null	si	jdoc	1	
Ticket	Ticket=null	no	jdoc	2	
Número de pruebo	Datas de entrada	,	/alor	Calida Esparada	Class
Número de prueba	Datos de entrada		/alor	Salida Esperada	Clase
1	Ticket	Ticket(para	ametros tipo 1)	1	1
tComparacionJornada()					
Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia	
Tiplest	Ticket!=null	si	jdoc	1	
Ticket	Ticket=null	no	jdoc	2	
Número de pruebo	Datas de entrada	,	/alor	Calida Esparada	Class
Número de prueba	Datos de entrada		/alor	Salida Esperada	Clase:
1	Ticket	пскецрага	ametros tipo 1)	-0,5	
tComparacionPuesto()					
Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia	
	Ticket!=null	si	jdoc	1	
Ticket	Ticket=null	no	jdoc	2	
Número de prueba	Datos de entrada		/alor	Salida Esperada	Clase
1	Ticket	Ticket(para	metros tipo 1)	-0,5	1
tComparacion Experiencia()					
Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia	
	Ticket!=null	si si	jdoc	1	
Ticket	Ticket=null	no	jdoc	2	
Número de prueba	Datos de entrada	\	/alor	Salida Esperada	Clase
1	Ticket	Ticket(para	ametros tipo 1)	-0,5	1
tComparacionEstudios()					
tComparacionEstudios() Dato de entrada	Clases de equivalencia	Aplica 2	Motivo	Identificador de clase de equivalencia	
Dato ac cilciaua	Ticket!=null	si	jdoc	1	
Ticket	Ticket=null	no	jdoc	2	
Número de prueba	Datos de entrada		/alor	Salida Esperada	Clases
1	Ticket	Ticket(para	ametros tipo 1)	-0,5	1
atComporacionT-t-I/\					
etComparacionTotal() Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia	
Dato de elitrada	Ticket!=null	Aplica ?	jdoc	1	
Ticket	Ticket:=null	no	jdoc	2	
				0.51.5	
Número de prueba	Datos de entrada	\	√alor	Salida Esperada	Clase



Escenario nro 7: Ticket22

etComparacionLocacion(Ticket)					
Dato de entrada	Clases de equivalencia	Aplica?	Motivo	Identificador de clase de equivalencia	
Ticket	Ticket!=null	si	jdoc	1	
nicket	Ticket=null	no	jdoc	2	
Número de prueba	Datos de entrada	,	√alor	Salida Esperada	Clases
1	Ticket		ametros tipo 2)	1	1
	Herce	Heketipur	unictios tipo 27	-	
etComparacionRemuneracion()					
Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia	
	Ticket!=null	si	jdoc	1	
Ticket	Ticket=null	no	jdoc	2	
Mómene de amobe	Detec de entre de	,	/-l	Calida Farranda	ol.
Número de prueba	Datos de entrada		/alor	Salida Esperada	Clases
1	Ticket	Ticket(par	ametros tipo 2)	1	1
etComparacionJornada()					
Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia	
**** ·	Ticket!=null	si	jdoc	1	
Ticket	Ticket=null	no	jdoc	2	
Móna a de de	Data da da da		/-l	0-54- 5	o.
Número de prueba	Datos de entrada	 	/alor	Salida Esperada	Clases
1	Ticket	Ticket(par	ametros tipo 2)	1	1
etComparacionPuesto()					
Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia	
	Ticket!=null	si	jdoc	1	
Ticket	Ticket=null	no	jdoc	2	
Número de prueba	Datos de entrada	,	√alor	Salida Esperada	Clases
1	Ticket	Ticket(par	ametros tipo 2)	1	1
etComparacionExperiencia()					
Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia	
Dato de entrada	Ticket!=null	si	idoc	1	
Ticket	Ticket=null	no	jdoc	2	
	TIONEE TIGHT		juoo	-	
Número de prueba	Datos de entrada	,	√alor	Salida Esperada	Clases
1	Ticket	Ticket(par	ametros tipo 2)	1	1
atComparacionEstudios()					
etComparacionEstudios() Dato de entrada	Clases de equivalencia	Anlica 2	Motivo	Identificador de clase de equivalencia	
Dato ac cilciaua	Ticket!=null	si	jdoc	1	
Ticket	Ticket=null	no	jdoc	2	
Número de prueba	Datos de entrada		/alor	Salida Esperada	Clases
1	Ticket	Ticket(par	ametros tipo 2)	1	1
etComparacionTotal()					
Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia	
	Ticket!=null	si	jdoc	1	
Ticket	Ticket=null	no	jdoc	2	
Número de prueba	Datos de entrada		/alor	Salida Esperada	Clases
1	Ticket		ametros tipo 2)	6	1



Escenario nro 8: Ticket23

etComparacionLocacion(Ticket					
Dato de entrada	Clases de equivalencia	Aplica?	Motivo	dentificador de clase de equivalencia	
Ti-1 4	Ticket!=null	si	jdoc	1	
Ticket	Ticket=null	no	jdoc	2	
Número de prueba	Datos de entrada	\	/alor	Salida Esperada	Clase
Numero de prueba	Ticket				1
1	licket	icket(para	metros tipo 13	-1	1
etComparacionRemuneracion()				
Dato de entrada	Clases de equivalencia	Anlica 2	Motivo	dentificador de clase de equivalencia	
Date de chirada	Ticket!=null	si si	jdoc	1	
Ticket	Ticket=null	no	jdoc	2	
Número de prueba	Datos de entrada	\	/alor	Salida Esperada	Clase
1	Ticket	Ticket(para	ametros tipo 3)	-0,5	1
otComparacion lorgada/\					
etComparacionJornada() Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	dentificador de clase de equivalencia	
	Ticket!=null	si si	jdoc	1	
Ticket	Ticket=null	no	jdoc	2	
			•		
Número de prueba	Datos de entrada	١	/alor	Salida Esperada	Clase
1	Ticket	Ticket(para	ametros tipo 3)	-0,5	1
atComparacionPuosto()					
etComparacionPuesto() Dato de entrada	Clases de equivalencia	Anlica 2	Motivo	dentificador de clase de equivalencia	
Dato de elitrada	Ticket!=null	si	jdoc	1	
Ticket	Ticket=null	no	jdoc	2	
	TICKE C-Hall	110	Juoc	2	
Número de prueba	Datos de entrada	\ \	/alor	Salida Esperada	Clase
1	Ticket	_	ametros tipo 3)	-0,5	1
etComparacionExperiencia()			4-		
Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	dentificador de clase de equivalencia	
Ticket	Ticket!=null	si	jdoc	1	
	Ticket=null	no	jdoc	2	
Número de prueba	Datos de entrada	١ ١	/alor	Salida Esperada	Clase
1	Ticket		ametros tipo 3)	1,5	1
etComparacionEstudios()	Classa do cavivalentia	Aplica	Motive	Identificador de alors de acciónstante	
Dato de entrada	Clases de equivalencia	_	Motivo	dentificador de clase de equivalencia	
Ticket	Ticket!=null Ticket=null	si no	jdoc jdoc	2	
	Hence-Hull		jaoc	-	
Número de prueba	Datos de entrada	\\	/alor	Salida Esperada	Clase
1	Ticket	Ticket(para	ametros tipo 3)	1,5	1
etComparacionTotal()	Classe de equivalencia	Aplica	Motivo	Identificador de clase de equivalencia	
Dato de entrada	Clases de equivalencia		Motivo	dentificador de clase de equivalencia	
	Ticket!=null	si	jdoc	1	
Ticket	Ticket-null	no.	idos		
Ticket	Ticket=null	no	jdoc	2	
Ticket Número de prueba	Ticket=null Datos de entrada		jdoc /alor	2 Salida Esperada	Clase



Escenario nro 9: Ticket31

tComparacionLocacion(Ticket					
Dato de entrada	Clases de equivalencia	Aplica?	Motivo	dentificador de clase de equivalencia	
Ticket	Ticket!=null	si	jdoc	1	
Ticket	Ticket=null	no	jdoc	2	
Número de prueba	Datos de entrada	\	/alor	Salida Esperada	Clase
1	Ticket	licket (par	ametros tipo 1	1	1
tComparacionPemuneracion/					
tComparacionRemuneracion(Dato de entrada	Clases de equivalencia	Anlica 2	Motivo	dentificador de clase de equivalencia	
Dato de elitrada	Ticket!=null	si	jdoc	1	
Ticket	Ticket=null	no	idoc	2	
			,	-	
Número de prueba	Datos de entrada	V	/alor	Salida Esperada	Clases
1	Ticket	Ticket (para	ametros tipo 1	1	1
tComparacionJornada()					
Dato de entrada	Clases de equivalencia	Aplica?	Motivo	dentificador de clase de equivalencia	
Ticket	Ticket!=null	si	jdoc	1	
IICKEL	Ticket=null	no	jdoc	2	
Mérica de la la	Detect to the	1	/alar	0-114- 5	
Número de prueba	Datos de entrada		/alor	Salida Esperada	Clases
1	Ticket	licket (para	ametros tipo 1	-1	1
tComparacionPuesto()					
Dato de entrada	Clases de equivalencia	Anlica 2	Motivo	dentificador de clase de equivalencia	
Dato de entrada	Ticket!=null	si	jdoc	1	
Ticket	Ticket=null	no	jdoc	2	
	TICKEL-HUII	110	juoc	2	
Número de prueba	Datos de entrada	V	/alor	Salida Esperada	Clases
1	Ticket	Ticket (para	ametros tipo 1	-1	1
tComparacionExperiencia()					
Dato de entrada	Clases de equivalencia	Aplica?	Motivo	dentificador de clase de equivalencia	
Ticket	Ticket!=null	si	jdoc	1	
TICKEL	Ticket=null	no	jdoc	2	
Número de prueba	Datos de entrada		/alor	Salida Esperada	Clases
1	Ticket	Ticket (para	ametros tipo 1	-2	1
tComparacionEstudios()					
Dato de entrada	Clases de equivalencia	Aplica 2	Motivo	dentificador de clase de equivalencia	
	Ticket!=null	si	jdoc	1	
Ticket	Ticket=null	no	jdoc	2	
Número de prueba	Datos de entrada	٧	/alor	Salida Esperada	Clases
1	Ticket	Ticket (para	ametros tipo 1	-2	1
tComparacionTotal()	01 1	A 15 C	1.1-2°	Manatificand and a second seco	
Dato de entrada	Clases de equivalencia		Motivo	dentificador de clase de equivalencia	
Ticket	Ticket!=null	si	jdoc	1	
	Ticket=null	no	jdoc	2	
			/alor	0-114- 54-	
Número de prueba	Datos de entrada	\ \	/alor	Salida Esperada	Clase
Número de prueba 1	Datos de entrada Ticket		/alor ametros tipo 1	Salida Esperada 0,5	Clases 1



Escenario nro 10: Ticket32

etComparacionLocacion(Ticket)					
Dato de entrada	Clases de equivalencia	Aplica?	Motivo	dentificador de clase de equivalencia	
Ticket	Ticket!=null	si	jdoc	1	
TICKEL	Ticket=null	no	jdoc	2	
Mómana da amaka	Datas da autor da		/alor	Optida Faranda	
Número de prueba	Datos de entrada	Valor		Salida Esperada	Clases
1	Ticket	Ticket(parametros tipo 2)		-1	1
etComparacionRemuneracion()					
Dato de entrada	Clases de equivalencia	Aplica?	Motivo	dentificador de clase de equivalencia	
Ticket	Ticket!=null	si	jdoc	1	
Ticket	Ticket=null	no	jdoc	2	
Número de prueba	Datos de entrada	\	/alor	Salida Esperada	Clases
1	Ticket	Ticket(parametros tipo 2)		1	1

etComparacionJornada()					
Dato de entrada	Clases de equivalencia	Aplica?	Motivo	dentificador de clase de equivalencia	
Ticket	Ticket!=null	si	jdoc	1	
TIERCE	Ticket=null	no	jdoc	2	
Número de pruebo	Datos do entrada	,	/alor	Calida Fanorada	Classes
Número de prueba	Datos de entrada			Salida Esperada	Clases
1	Ticket	ricket(para	ametros tipo 2	1	1
etComparacionPuesto()					
Dato de entrada	Clases de equivalencia	Aplica?	Motivo	dentificador de clase de equivalencia	
Tielee	Ticket!=null	si	jdoc	1	
Ticket	Ticket=null	no	jdoc	2	
Número de prueba	Datos de entrada	\	/alor	Salida Esperada	Clases
1	Ticket	Ticket(para	ametros tipo 2)	1	1
etComparacionExperiencia()					
Dato de entrada	Clases de equivalencia		Motivo	Identificador de clase de equivalencia	
Ticket	Ticket!=null	si	jdoc	1	
Trence	Ticket=null	no	jdoc	2	
Número de prueba	Datos de entrada	\	/alor	Salida Esperada	Clases
1	Ticket		ametros tipo 2)		1
<u> </u>	TICKEL	Ticket(pare	amedos tipo 2	-1,5	
etComparacionEstudios()					
Dato de entrada	Clases de equivalencia	Aplica?	Motivo	dentificador de clase de equivalencia	
	Ticket!=null	si	jdoc	1	
Ticket	Ticket=null	no	jdoc	2	
Número de prueba	Datos de entrada	\	/alor	Salida Esperada	Clases
1	Ticket	Ticket(para	ametros tipo 2)	-1,5	1
etComparacionTotal()					
Dato de entrada	Clases de equivalencia	Aplica?	Motivo	dentificador de clase de equivalencia	
Ticket	Ticket!=null	si	jdoc	1	
HUNCL	Ticket=null	no	jdoc	2	
Número de prueba	Datos de entrada		/alor	Salida Esperada	Clases
1	Ticket	Ticket(para	ametros tipo 2	0,5	1



Escenario nro 11: Ticket33

ComparacionLocacion(Ticket	1)				
Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	dentificador de clase de equivalencia	
Tieles	Ticket!=null	si	jdoc	1	
Ticket	Ticket=null	no	jdoc	2	
Número de prueba	Dates de entrada	,	/alor	Salida Esperada	Class
Número de prueba	Datos de entrada				Clase
1	Ticket	licket (par	ametros tipo 3	1	1
ComparacionRemuneracion()				
Dato de entrada	Clases de equivalencia	Aplica?	Motivo	dentificador de clase de equivalencia	
Tinker	Ticket!=null	si	jdoc	1	
Ticket	Ticket=null	no	jdoc	2	
Número de pruebo	Datas de entrada	,	/alor	Calida Fanorada	Class
Número de prueba	Datos de entrada	_		Salida Esperada	Clase
1	Ticket	Ticket (par	ametros tipo 3	1	1
ComparacionJornada()					
Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	dentificador de clase de equivalencia	
Tial	Ticket!=null	si	jdoc	1	
Ticket	Ticket=null	no	jdoc	2	
Número de prueba	Datos de entrada		/alor	Salida Esperada	Clase
1	Ticket	Ticket (par	ametros tipo 3	1	1
Dato de entrada	Clases de equivalencia Ticket!=null	Aplica ?	Motivo jdoc	Identificador de clase de equivalencia	
Dato de entrada					
Ticket		 			
	Ticket=null	no	jdoc	2	
Número de prueba	Datos de entrada	V	/alor	Salida Esperada	Clase
1	Ticket	Ticket (par	ametros tipo 3		1
ComparacionExperiencia()	Olasaa da aassissalasada	A 15 O	M-4:	Id-utitidddi	
Dato de entrada	Clases de equivalencia		Motivo	Identificador de clase de equivalencia	
Ticket	Ticket!=null	si	jdoc	1	
	Ticket=null	no	jdoc	2	
Número de prueba	Datos de entrada	V	/alor	Salida Esperada	Clase
4		L			-
1	Ticket	Ticket (par	ametros tipo 3	1	1
_	Ticket	Ticket (par	ametros tipo 3	1	1
ComparacionEstudios()					1
_	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia	1
ComparacionEstudios()	Clases de equivalencia	Aplica ?	Motivo jdoc	dentificador de clase de equivalencia	1
ComparacionEstudios() Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia	1
ComparacionEstudios() Dato de entrada	Clases de equivalencia	Aplica ?	Motivo jdoc	dentificador de clase de equivalencia	
ComparacionEstudios() Dato de entrada Ticket	Clases de equivalencia Ticket!=null Ticket=null	Aplica ?	Motivo jdoc jdoc	dentificador de clase de equivalencia 1 2 Salida Esperada	
ComparacionEstudios() Dato de entrada Ticket Número de prueba 1	Clases de equivalencia Ticket!=null Ticket=null Datos de entrada	Aplica ?	Motivo jdoc jdoc	dentificador de clase de equivalencia 1 2 Salida Esperada	Clase
ComparacionEstudios() Dato de entrada Ticket Número de prueba 1 ComparacionTotal()	Clases de equivalencia Ticket!=null Ticket=null Datos de entrada Ticket	Aplica ? si no V	Motivo jdoc jdoc /alor ametros tipo 3	Identificador de clase de equivalencia 1 2 Salida Esperada 1	Clase
ComparacionEstudios() Dato de entrada Ticket Número de prueba 1	Clases de equivalencia Ticket!=null Ticket=null Datos de entrada Ticket Clases de equivalencia	Aplica ? si no V Ticket (para	Motivo jdoc jdoc /alor ametros tipo 3	Identificador de clase de equivalencia 1 2 Salida Esperada 1 Identificador de clase de equivalencia	Clase
ComparacionEstudios() Dato de entrada Ticket Número de prueba 1 ComparacionTotal()	Clases de equivalencia Ticket!=null Ticket=null Datos de entrada Ticket	Aplica ? si no V	Motivo jdoc jdoc /alor ametros tipo 3	Identificador de clase de equivalencia 1 2 Salida Esperada 1	Clase

Valor

Ticket (parametros tipo 3)

Salida Esperada

Clases

Datos de entrada Ticket

Número de prueba



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.

c. cc. c.p cc	generan sus listas				
etContrataciones					
Dato de entrada	Clases de equivalencia	Aplica?	Motivo	ldentificador de clase de equivalencia	
Contrataciones (ArrayList <contratacion>)</contratacion>	Contrataciones != null	si	jdoc	1	
	Contrataciones == null	no	jdoc (inv clase)	2	
Número de prueba	Datos de entrada		'alor	Salida Esperada	Clases
1	Contrataciones		contratacion entre	seteo correcto (que se verifica con el get)	1
		un emplead	o y empleador	., ,	
etCoincidencias					
Dato de entrada	Clases de equivalencia	Aplica?	Motivo	ldentificador de clase de equivalencia	
	Coincidencias != null	si	jdoc	1	
Coincidencias (ArrayList <contratacion>)</contratacion>	Coincidencias == null	no	jdoc (inv clase)	2	
Número de prueba	Datos de entrada		'alor	Salida Esperada	Clases
radiliero de proeba	Datos de entrada			Oalida Esperada	Oldata
1	Coincidencia		contratacion entre io y empleador	seteo correcto (que se verifica con el get)	1
netContratacionEmpleador					
	mpleador!= 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	·		No se especifica		
	empleador sin contratacion	no	retorno o	4	
	empleador sin contracación	110	lanzamiento	•	1
			de excepcion		
Número de prueba	Datos de entrada		alor	Salida Esperada	Clases
ivumero de prueba	Empleador		n atributos vacios	Devuelve la referencia a empleado con el que empleador tenía contratacion	Clases
ı	Empleador	Empleador co	i atributos vacios	Devoelve la referencia a empleado con el que empleador centa contracación	
GetContratacionEmpleadoPretenso					
Dato de entrada	Clases de equivalencia	Aplica?	Motivo	ldentificador de clase de equivalencia	
	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	
Empleador			No se especifica		
	empleado sin contratacion	no	retorno o	4	
	empresso sin senia accioni		lanzamiento de excepcion	·	
Número de prueba	Datos de entrada		alor	Salida Esperada	Clases
1	EmpleadoPretenso	Empleado cor	atributos vacios	Devuelve la referencia a empleador con el que empleador tenía contratacion	- 1
etComisionesUsuarios()	0				
Dato de entrada	Clases de equivalencia comisionesUsuarios != null	Aplica?	Motivo	ldentificador de clase de equivalencia	
omisionesUsuarios (Hashmap <cliente,double></cliente,double>	comisionesUsuarios := null comisionesUsuarios == null	si no	jdoc jdoc (inv clase)	2	
N/	D. I. I		l-l	O III F	CI.
Número de prueba	Datos de entrada		'alor un solo elemento	Salida Esperada	Clases
1	comisionesUsuarios		ado,0.001)	seteo correcto (que se verifica con el get)	1
etComisionUsuario()					
Dato de entrada	Clases de equivalencia	Aplica?	Motivo	ldentificador de clase de equivalencia	
	Cliente!= null, cargado en sistema y con una comision	si	jdoc	1	
	empleado == null	no	No se especifica	2	
Cliente	empleado no en sistema	no	retorno o	3	
	empleado sin contratacion	no	lanzamiento de excepcion	4	
	_				
Número de prueba 1	Datos de entrada Cliente	Empleado o xeo	alor do en hashmap con	Salida Esperada 0.01	Clases

setEmpleadores					
Dato de entrada	Clases de equivalencia	Aplica?	Motivo	ldentificador de clase de equivalencia	
Empleadores (HashMap <string,empleador>)</string,empleador>	Empleadores != null	si	jdoc	1	
Empleadores (Hasili-lapk String,Empleadors)	Empleadores == null	no	jdoc (inv clase)	2	_
Número de prueba	Datos de entrada		/alor	Salida Esperada	Clases
1	Empleadores		1 elemento ("+54 9 34,empleador)	seteo correcto (que se verifica con el get)	1
setEmpleados					
Dato de entrada	Clases de equivalencia	Aplica?	Motivo	ldentificador de clase de equivalencia	
npleados (HashMap <string, empleadopretenso<="" td=""><td>Empleados != null</td><td>si</td><td>jdoc</td><td>1</td><td></td></string,>	Empleados != null	si	jdoc	1	
inpreados (i lasin-lap county, Empreador Texenso	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	ldentificador de clase de equivalencia	
		si	jdoc	1	
Número de prueba	Datos de entrada		/alor	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	ldentificador de clase de equivalencia	
		si	jdoc	1	_
Número de prueba	Datos de entrada		/alor	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 jdoc	ldentificador de clase de equivalencia 1	
Número de prueba 1	Datos de entrada	\	/alor	<u>Salida Esperada</u> String que según sea F o V el estado en el que se encuentre agencia devuelve Mensaje AGENCIA, EN CONTRATACION.gev Vaoloj (o Mensajes AGENCIA, EN, BUSQUEDA get Valor[)	Clases 1
				THE STATE OF THE PROPERTY OF THE STATE OF TH	
getInstance() Dato de entrada	Clases de equivalencia	Aplica ?	Motivo jdoc	Identificador de clase de equivalencia	
Número de prueba	Datos de entrada	-	/alor	Salida Esperada	Clases
1				Debe obtenerse la misma referencia a Agencia que la presente en	1
setCoincidencias ()	Classed an inclassio	A-6 2	Maxima	l de a siút e adre de a la constituel e a cir.	
Dato de entrada Coincidencias (ArrayList <contratacion>)</contratacion>	Clases de equivalencia Coincidencias != null Coincidencias == null	Aplica? si no	Motivo jdoc jdoc (inv clase)	Identificador de clase de equivalencia 1 2	
Número de prueba	Datos de entrada		/alor	- Salida Esperada	Clases
1	Coincidencia	Lista sin contr	ataciones (vacia)	seteo correcto (que se verifica con el get)	1
setComisionesUsuarios()	Constitution	A-1: 2	M.C.	Heat (foods a dealers de sector les etc.)	_
Dato de entrada comisionesUsuarios (Hashmap <cliente,double></cliente,double>	Clases de equivalencia comisionesUsuarios != null comisionesUsuarios == null	Aplica? si	Motivo jdoc jdoc (inv clase)	Identificador de clase de equivalencia 1 2	
Número de prueba	Datos de entrada		/alor	Salida Esperada	Clases
1	comisionesUsuarios		nap 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>)</contratacion>	Contrataciones != null Contrataciones == null	si no	jdoc jdoc (inv clase)	nder kindaduri de diase de equivalenda 2	
Número de prueba	Datos de entrada		/alor	Salida Esperada	Clases
1	Contrataciones	Lista sin contr	ataciones (vacia)	seteo correcto (que se verifica con el get)	1
setEmpleadores					
Dato de entrada Empleadores (HashMap <string,empleador>)</string,empleador>	Clases de equivalencia Empleadores != null Empleadores == null	Aplica?	Motivo jdoc	Identificador de clase de equivalencia 1 2	
Número de prueba	Datos de entrada	no	jdoc (inv clase) /alor	2 Salida Esperada	Clases
1 1	Empleadores	Hashr	nap vacio	Saltoa Esperatua seteo correcto (que se verifica con el get)	1
setEmpleados					
Dato de entrada mpleados (HashMapkString, EmpleadoPretenso	Clases de equivalencia Empleados != null	Aplica? si	Motivo jdoc	ldentificador de clase de equivalencia 1	
	Empleados== null	no	jdoc (inv clase)	2	
Número de prueba 1	Datos de entrada Empleados		/alor nap vacio	Salida Esperada seteo correcto (que se verifica con el get)	Clases 1
5.10					
setEstadoContratacion() Dato de entrada	Clases de equivalencia Estado: True	Aplica?	Motivo jdoc	Identificador de clase de equivalencia	
EstadoContratacion	Estado=False	no	jdoc (inv clase)	2	
Número de prueba 1	<u>Datos de entrada</u> Estado		/alor True	Salida Esperada seteo correcto (que se verifica con el get)	Clases 1
1	Estado	F	alse	seteo correcto (que se verifica con el get)	1
setLimitesRemuneración Dato de entrada	Clases de equivalencia inf<0	Aplica ?	Motivo c	ldentificador de clase de equivalencia 1	
limiteInferior	inf>=0 sup>=inf	si si	0 0	2 3	
limiteSuperior	supkinf	si	С	4	
Número de prueba	Datos de entrada inf	Valor 1		Salida Esperada seteo correcto	Clases 2,3
2	sup inf	999999		LimiteInferiorRemuneracionInvalidaException	1,3
3	sup inf sup	999999 2 1		LimiteSuperiorRemuneracionInvalidaException	2,4
getLimiteSuperior() Dato de entrada	Clases de equivalencia	Aplica?	Motivo	ldentificador de clase de equivalencia	
		si	jdoc	1	
Número de prueba 1	Datos de entrada	 	'alor	Salida Esperada Limite superior establecido en agencia (999999)	Clases 1
getLimiteInferior()					
Dato de entrada	Clases de equivalencia	Aplica ?	Motivo jdoc	Identificador de clase de equivalencia	1
Número de prueba	Datos de entrada		'alor	Salida Esperada	Clases
1	Balos de erinada			Limite superior establecido en agencia (1)	1
setPersistencia()					
Dato de entrada Persistencia (Ipersistencia)	Clases de equivalencia Persistencia != null	Aplica ?	Motivo jdoc	Identificador de clase de equivalencia	
(y	Persistencia == null	si	jdoc	2	
Número de prueba	Datos de entrada Persistencia		'alor enciaXML()	Salida Esperada seteo correcto (que se verifica con el get)	Clases 1
2	Persistencia		null	seteo correcto (que se verifica con el get)	2
getkeratorEmpleadores()			<u> </u>		
Dato de entrada	Clases de equivalencia	Aplica ?	Motivo jdoc	ldentificador de clase de equivalencia 1	-
Número de prueba	Datos de entrada		/alor	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 jdoc	ldentificador de clase de equivalencia 1	
Número de prueba	Datos de entrada		/alor	Salida Esperada	Clases
1				Referencia a un iterator que contenga los elementos de la lista de Empleados Pretensos (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 {util.Constantes.HOME OFFICE, siendo entonces ticket empleador util.Constantes.JORNADA_MEDIA, util.Constantes.JUNIOR, util.Constantes.EXP_MEDIA, util.Constantes.PRIMARIOS, empleador} ticket de empleado У {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).

ra el ticket de empleador [empleador.getTicket(JJ				
Estudios() Dato de entrada	Clases de equivalencia	Aplica?	Motivo	Identificador de clase de equivalencia	
		si	jdoc	1	
Número de prueba	Datos de entrada	V	alor	Salida Esperada	Clase:
1				util.Constantes.PRIMARIOS	1
Experiencia() Dato de entrada	Clases de equivalencia	Aplica?	Motivo	Identificador de clase de equivalencia	
Date de entrada	Crases de equivalencia	si	idoc	1	
Número de prueba	Datos de entrada		alor	Salida Esperada	Clase
1				util.Constantes.EXP_MEDIA	1
Jornada() Dato de entrada	Clases de equivalencia	Aplica?	Motivo	Identificador de clase de equivalencia	
Dato de entrada	Crases de equivalencia	Apilica :	idoc	identificador de clase de equivalencia 1	
Número de prueba	Datos de entrada		alor	Salida Esperada	Clase
1				Salida Esperada util.Constantes.JORNADA_MEDIA	1
Locacion()					
Dato de entrada	Clases de equivalencia	Aplica ?	Motivo jdoc	ldentificador de clase de equivalencia 1	
Número de prueba	Datos de entrada		alor	Salida Esperada	Clase
ivumero de prueba 1	Datos de entrada	<u> </u>		Salida Esperada util.Constantes.HOME_OFFICE	Clase 1
Puesto()					
Dato de entrada	Clases de equivalencia	Aplica ?	Motivo jdoc	ldentificador de clase de equivalencia	
			·	·	
Número de prueba 1	Datos de entrada	 -	alor	Salida Esperada util.Constantes.JUNIOR	Clase:
					-
Remuneracion() Dato de entrada	Classe de accióndacio	Aplica?	Motivo	laboritico ador do ologo do ou incloso in	
Dato de entrada	Clases de equivalencia	si si	jdoc	Identificador de clase de equivalencia 1	
Número de prueba	Datos de entrada		alor	Salida Esperada	Clases
1				999	1
	T				
ra el ticket de empleadoPretenso (empleado.ge	ricket[j]				
Estudios() Dato de entrada	Clases de equivalencia	Aplica?	Motivo	ldentificador de clase de equivalencia	
		si	jdoc	1	
Número de prueba	Datos de entrada		alor	Salida Esperada util.Constantes.PRIMARIOS	Clases
1				util.Constantes.PRIMARIOS	
Experiencia() Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia	
Date de entrada	Crases de equivalencia	si	jdoc	ider kincador de crase de equivalencia	
Número de prueba	Datos de entrada	v	alor	Salida Esperada util.Constantes.EXP NADA	Clase:
1				util.Constantes.EXP_NADA	1
Jornada()	Classic desired	A-1:2	Mark	Harden de de de la companya de la companya	
Dato de entrada	Clases de equivalencia	Aplica ?	Motivo jdoc	ldentificador de clase de equivalencia 1	
Número de prueba	Datos de entrada	V	alor	Salida Esperada	Clases
1		1		Salida Esperada util.Constantes.JORNADA_MEDIA	1
.ocacion()					
Dato de entrada	Clases de equivalencia	Aplica ?	Motivo jdoc	ldentificador de clase de equivalencia 1	
Número de prueba	Datos de entrada		alor	Salida Esperada	Clase
1	Datos de entrada	<u> </u>		Salida Esperada util.Constantes.HOME_OFFICE	1
esto()					
Dato de entrada	Clases de equivalencia	Aplica ?	Motivo idoc	ldentificador de clase de equivalencia 1	
Número de prueba	Datos de entrada		alor	Salida Esperada util.Constantes.JUNIOR	Cla
nuneracion()					
Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia 1	
		si	jdoc	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.

Dato de entrada	Clases de equivalencia	Aplica?	Motivo	ldentificador de clase de equivalencia	7
	Locacion!= null y esta contemplado en la clase Constantes	si	jdoc	1	7
Locacion (string)	String no contemplado en la clase Constantes	no	jdoc (ticket)	2	1
	Locacion== null	no	jdoc (ticket)	3	1
remuneracion (int)	Remuneracion >= 0	si	jdoc	4	7
remuneracion (int)	Remuneracion < 0	no	jdoc (ticket)	5	1
	jornadal= null y esta contemplado en la clase Constantes	si	jdoc	6	7
jornada (string)	String no contemplado en la clase Constantes	no	jdoc (ticket)	7	
	jornada== null	no	jdoc (ticket)	8	
	puestol= null y esta contemplado en la clase Constantes	si	jdoc	3	
puesto (string)	String no contemplado en la clase Constantes	no	jdoc (ticket)	10	
	puesto== null	no	jdoc (ticket)	11	
	«periencial» null y esta contemplado en la clase Constante	si	jdoc	12	
experiencia (string)	String no contemplado en la clase Constantes	no	jdoc (ticket)	13	
	experiencia== null	no	jdoc (ticket)	14	
	estudios!= null y esta contemplado en la clase Constantes	si	jdoc	15	
estudios (string)	String no contemplado en la clase Constantes	no	jdoc (ticket)	16	
	estudios== null	no	jdoc (ticket)	17	
	cliente != null y de tipo EmpleadoPretenso	si	jdoc	18	
	cliente != null y de tipo != EmpleadoPretenso	no	idoc	19	
oliente (oliente)	cliente == null	no	documenta que ocurre si cliente	20	
Número de prueba	Datos de entrada		/alor	Salida Esperada	1
ryumero de prueba	Locacion		s.HOME OFFICE	Salida Esperada	+
	remuneracion		SHOME_OFFICE		
	iornada		JORNADA MEDIA		
1	puesto		antes.JUNIOR	Lanzamiento de la excepcion Imposible/ModificarTicketException	1.4
•	experiencia		tes.EXP NADA	contamination of the strong potential postulation in the description	1"
	estudios		tes.PRIMARIOS		
	cliente		oPretenso()		1

i icketEmpleado∏ ∏Nota: el metodo r	io establece como precondiciones las condiciones necesarias p			ticket) , pero tampoco documenta la forma de resolucion que adopta a la hora de incumplir dichas condiciones (lan:	zamiento de
Dato de entrada	Clases de equivalencia	Aplica?	Motivo	Identificador de clase de equivalencia	
	Locacion!: null y esta contemplado en la clase Constantes	si	jdoc	1	
Locacion (string)	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 (int)	Remuneracion < 0	no	jdoc (ticket)	5	
	jornada!= null y esta contemplado en la clase Constantes	si	jdoc	6	
jornada (string)	String no contemplado en la clase Constantes	no	jdoc (ticket)	7	
	jornada== null	no	jdoc (ticket)	8	
	puestol: null y esta contemplado en la clase Constantes	si	jdoc	9	
puesto (string)	String no contemplado en la clase Constantes	no	jdoc (ticket)	10	
	puesto== null	no	jdoc (ticket)	11	
	«periencial» null y esta contemplado en la clase Constante	si	jdoc	12	
experiencia (string)	String no contemplado en la clase Constantes	no	jdoc (ticket)	13	
	experiencia== null	no	jdoc (ticket)	14	
	estudios!= null y esta contemplado en la clase Constantes	si	jdoc	15	
estudios (string)	String no contemplado en la clase Constantes	no	jdoc (ticket)	16	
	estudios== null	no	jdoc (ticket)	17	
	cliente != null y de tipo Empleador	si	jdoc	18	
	cliente != null y de tipo != Empleador	no	jdoc	19	
cliente (cliente)	cliente == null	no	documenta que ocurre si cliente	20	
			"		
Número de prueba	Datos de entrada		/alor	Salida Esperada	С
	Locacion	util.Constante:	s.HOME_OFFICE		
	remuneracion		999		
	jornada	util.Constantes.	JORNADA_MEDIA		
1	puesto	util.Consta	antes.JUNIOR	Lanzamiento de la excepcion ImposibleModificarTicketException	1,4,6
	experiencia	util.Constant	tes.EXP_NADA		
	estudios	util.Constant	tes.PRIMARIOS		- 1
	cliente	omn.	leador()		1

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
	nombreUsuario	"Pepegamer"		
	pass	"contrasenia"		
1	nombreReal	"Pepe Gomes"	Creacion correcta de empleador y adicion a la lista de empleadores	1,4,6,8,10,13
,	telefono	"2234434312"	are an area are empressed in a second contract of the contract	1,10,0,10,10
	tipoPersona	util.constantes.FISICA		
	rubro	util.constantes.SALUD		
	nombreUsuario	null		
	pass	"contrasenia"		
2	nombreReal	"Pepe Gomes"	throws ImposibleCrearEmpleadorException()	3,4,6,8,10,13
2	telefono	"2234434312"	(mows imposibleCrealEmpleadorException()	3,4,0,0,10,13
	tipoPersona	util.constantes.FISICA		
	rubro	util.constantes.SALUD		
	nombreUsuario	"Pepegamer"		
	pass	null	$throws \ Imposible Crear Emple ador Exception ()$	
3	nombreReal	"Pepe Gomes"		
3	telefono	"2234434312"		
	tipoPersona	util.constantes.FISICA		
	rubro	util.constantes.SALUD		
	nombreUsuario	"Pepegamer"		
	pass	"contrasenia"		
	nombreReal	null	throws ImposibleCrearEmpleadorException()	1,4,7,8,10,13
•	telefono	"2234434312"	(nrows imposibleCrearEmpleadorException()	1,4,7,8,10,13
	tipoPersona	util.constantes.FISICA		
	rubro	util.constantes.SALUD		
	nombreUsuario	"Pepegamer"		
	pass	"contrasenia"		
5	nombreReal	"Pepe Gomes"	throws ImposibleCrearEmpleadorException()	*****
5	telefono	null	throws imposibleCrearEmpleadorException()	1,4,6,9,10,13
	tipoPersona	util.constantes.FISICA		
	rubro	util.constantes.SALUD		
	nombreUsuario	"Pepegamer"		
	pass	"contrasenia"		
	nombreReal	"Pepe Gomes"	d	
6	telefono	"2234434312"	throws ImposibleCrearEmpleadorException()	1,4,6,8,11,13
	tipoPersona	"NORMAL"		
	rubro	util.constantes.SALUD		
		UD II		



	nombreUsuario	"Pepegamer"		
	pass	"contrasenia"		
7	nombreReal	"Pepe Gomes"	throws ImposibleCrearEmpleadorException()	1,4,6,8,12,13
r	telefono	"2234434312"	throws imposibleCreatEmpleadorException()	1,4,0,0,12,13
	tipoPersona	null		
	rubro	util.constantes.SALUD		
	nombreUsuario	"Pepegamer"		
	Dass			
8	nombreReal	"Pepe Gomes"	Non-the-side Confederation (1,4,6,8,10,14
	telefono	"2234434312"	throws ImposibleCrearEmpleadorException()	
	tipoPersona	util.constantes.FISICA		
	rubro	"INFORMATICA"		
	nombreUsuario	"Pepegamer"		
	pass	"contrasenia"		
9	nombreReal	"Pepe Gomes"	throws ImposibleCrearEmpleadorException()	1,4,6,8,10,15
•	telefono	"2234434312"	throws impositive create inheadore sception()	1,4,0,0,10,10
	tipoPersona	util.constantes.FISICA		
	rubro	null		
	nombreUsuario	"Tomasito"		
	pass	"contrasenia"		
10	nombreReal	"Pepe Gomes"		1,2,6,8,10,13
IU	telefono	"2234434312"	throws NewRegisterException()	1,2,0,8,10,13
	tipoPersona	util.constantes.FISICA		
	rubro	util.constantes.SALUD		

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

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

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")

ogin()					
Dato de entrada	Clases de equivalencia	Aplica?	Motivo	ldentificador de clase de equivalencia	1
	nombre!= null y usuario registrado	si	precondicion idoc	1	
nombre de usuario (string)	nombre!= null y usuario no registrado	si	precondicion idoc	2	
	nombre==null	no	precondicion jdoc	3	
	contraseñal=null y contraseña correcta	si	precondicion idoc	4	
contraseña	contraseña!= null y contraseña incorrecta	si	precondicion idoc		1
	contraseña==null	no	precondicion jdoc	6	
Número de prueba	Datos de entrada	1	/alor	Salida Esperada	Clases
	nombre usuario	"Pep	egamer"	In a company	1.4.
ļ.	contraseña	"qwe	ertyuiop"	logueo correcto	1,4.
	nombre usuario	"Pep	egamer"	throws ContraException()	45
2	contraseña	"contra	Exception"	throws ContraException()	1,5.
	nombre usuario	"Usuario	oException"		
3	contraseña	"awa	ertuuiop"	throws NombreUsuarioException()	2,4.



egistroEmpleador()				
Dato de entrada	Clases de equivalencia	Aplica?	Motivo	ldentificador de clase de equivalencia
	nombreUsuario != null y no se encuentra registrado	si	jdoc	1
nombre de usuario (string)	nombreUsuario != null y ya encuentra registrado	si	idoc	2
	nombreUsuario == null	si	jdoc	3
pass (string)	pass != null	si	jdoc	4
pass (sung)	pass == null	si	jdoc	5
nombre Real (string)	nombreReal != null	si	jdoc	6
nombre near (string)	nombreReal == null	si	jdoc	7
telefono (string)	telefono != null	si	jdoc	8
telerono (string)	telefono == null	si	jdoc	9
	tipoPersona != null y contemplada en Constantes	si	jdoc	10
tipo persona (string)	tipoPersona!= null y NO contemplada en Constantes	si	jdoc	11
	tipoPersona == null	si	jdoc	12
	rubro != null y contemplada en Constantes	si	jdoc	13
rubro (string)	rubro != null y NO contemplada en Constantes	si	idoc	14
·	rubro == null	si	idoc	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.J UNIOR,util.Constantes.EXP_MEDIA,util.Constantes.PRIMARIOS,empleador).

cerrarCesion()					
Dato de entrada	Clases de equivalencia	Aplica?	Motivo	ldentificador de clase de equivalencia	1
	1	si	jdoc	1	
Número de prueba	Datos de entrada	1 1	/alor	Salida Esperada	Clases
1	<u> </u>	1		cambio de tipoUsuario por -1 (ningun usuario logueado)	1
qetTipoUsuario()					
Dato de entrada	Clases de equivalencia	Aplica?	Motivo	ldentificador de clase de equivalencia	
		si	jdoc	1	
				0.115	
Número de prueba	Datos de entrada	- '	/alor	Salida Esperada	Clases
1				tipo de Usuario actual (empleador = 1)	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

e <mark>limina Ticket(). Udoo omite el descuento del pun</mark> taje al elimina run 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 Usuari								
Dato de entrada	Clases de equivalencia	Aplica?	Motivo	ldentificador 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 loqueado actualmente u 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	ldentificador 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).

Clases de equivalencia	Aplica?	Motivo	Identificador de clase de equivalencia	
	si	jdoc	1	
Datos de entrada	V.	alor		Clases
			2 (la lista debe contener a los 2 empleadores)	1
Con este test se verifica que el primer elemento de la li-	sta de postulantes	coincide con el empleador o	on mayor coincidencia de ticket (puntaje)	
Clases de equivalencia	Aplica?	Motivo	Identificador de clase de equivalencia	
	si	jdoc	1	
Datos de entrada	V.	alor		Clases
			referencia al objeto Empleador1	1
Clases de equivalencia	Aplica?	Motivo	ldentificador de clase de equivalencia	
	si	jdoc	1	
D. I. I	U	-l	0115	Clases
Datos de entrada	V-	alor		Liases
		_	z (la lista debe contener a los z empleados)	
Clases de equivalencia			ldentificador de clase de equivalencia	
	si	idoc	1	
Datos de entrada	V.	alor	Salida Fenerada	Clases
			referencia al obieto Empleado1	
	Datos de entrada Con este test se verifica que el primer elemento de la l Clases de equivalencia Datos de entrada Clases de equivalencia	Datos de entrada V Con este test se verifica que el primer elemento de la lista de postulantes Classes de equivalencia Aplica ? Datos de entrada V Classes de equivalencia Aplica ? Si Datos de entrada V Classes de equivalencia Aplica ? Si Con este test se verifica que el primer elemento de la lista de postulantes Classes de equivalencia Aplica ? Aplica ? Aplica ?	Datos de entrada Valor Con este test se verifica que el primer elemento de la lista de postulantes coincide con el empleador or Clases de equivalencia Aplica ? Motivo si Idoo Datos de entrada Valor Clases de equivalencia Aplica ? Motivo si Idoo Datos de entrada Valor Clases de equivalencia Aplica ? Motivo si Idoo Datos de entrada Valor Clases de equivalencia Aplica ? Motivo si Idoo Datos de entrada Valor	Batos de entrada Valor Salida Esperada 2 (la lista debe contener a los 2 empleadores) con este test se verifica que el primer elemento de la lista de postulantes coincide con el empleador con magor coincidencia de ticket (purtaje) Clases de equivalencia Aplica ? Motivo Datos de entrada Valor Salida Esperada referencia al objeto Empleador 1 Clases de equivalencia Aplica ? Motivo Identificador de clase de equivalencia Salida Esperada referencia al objeto Empleador 1 Clases de equivalencia Aplica ? Motivo Identificador de clase de equivalencia si joloc Clases de entrada Valor Salida Esperada referencia al objeto Empleador 1 Clases de entrada Valor Salida Esperada (2 (la lista debe contener a los 2 empleados) Con este test se verifica que el primer elemento de la lista de postutantes coincide con el empleado con magor coincidencia de ticket (puntaje) Clases de equivalencia si joloc Clases de equivalencia si joloc Lidentificador de clase de equivalencia

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.

ado1.getPuntaje()					
Dato de entrada	Clases de equivalencia	Aplica?	Motivo	ldentificador de clase de equivalencia	
		si	jdoc	1	
Número de prueba	Datos de entrada	V	alor	Salida Esperada	Cla
1		7		0 (puntaje= +5-5 debido a que se encuentra fro en la lista del empleador 1 y ultimo en la lista del empleador 2)	-
ado2.getPuntaje()					
Dato de entrada	Clases de equivalencia	Aplica?	Motivo	ldentificador de clase de equivalencia	1
		si	jdoc	1	1
Número de prueba	Datos de entrada	V	alor	Salida Esperada	Cla
1				0 (puntaje= +5 -5 debido a que se encuentra fro en la lista del empleador 2 y ultimo en la lista del empleador 1)	
ador1.getPuntaie()					
Dato de entrada	Clases de equivalencia	Aplica?	Motivo	ldentificador de clase de equivalencia	1
		si	jdoc	1	
Número de prueba	Datos de entrada	V	alor	Salida Esperada	Cla
1				10 (debido a que se encuentra fro en la lista del empleado 1)	
ado2.getPuntaje()					
Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	Identificador de clase de equivalencia	
		si	jdoc	1	4
Número de prueba	Datos de entrada	v	alor	Salida Esperada	Cla
-				10 (debido a que se encuentra fro en la lista del empleado 2)	

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)



					npleado.getTicket()
	ldentificador de clase de equivalencia	Motivo	Aplica?	Clases de equivalencia	Dato de entrada
	1	jdoc	si		
Clases	Salida Esperada	alor	V	Datos de entrada	Número de prueba
1	null				1
					npleador.getTicket()
	ldentificador de clase de equivalencia	Motivo idoc	Aplica?	Clases de equivalencia	Dato de entrada
Clase:	Salida Esperada	alor		Datos de entrada	Número de prueba
1	null		<u> </u>	Datos de entrada	1 1
					ContratacionEmpleadoPretenso()
	ldentificador de clase de equivalencia	Motivo	Aplica?	Clases de equivalencia	Dato de entrada
	1	jdoc	si	Empleado!=null	Empleado
Clase	Salida Esperada	alor		Datos de entrada	Número de prueba
1	referencia al empleador con el que se tuvo la contratacion	ontratacion vigente	empleado con co	Empleador	1
					ContratacionEmpleador()
	ldentificador de clase de equivalencia	Motivo	Aplica?	Clases de equivalencia	Dato de entrada
	1	jdoc	si	Empleador!=null	Empleador
Clase	Salida Esperada	alor	V	Datos de entrada	Número de prueba
1	referencia al empleado con el que se tuvo la contratacion	ontratacion vigente	empleador con ci		1
					pleado.getPuntaje()
	Identificador de clase de equivalencia	Motivo idoc	Aplica?	Clases de equivalencia	Dato de entrada
Clas	Salida Esperada	alor		Datos de entrada	Número de prueba
1	Janua Esperada 10			Datos de entrada	1 1
					pleador.getPuntaje()
	ldentificador de clase de equivalencia 1	Motivo idoc	Aplica?	Clases de equivalencia	Dato de entrada
	· ·				
Clase	Salida Esperada	alor		Datos de entrada	Número de prueba

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 jdoc	Identificador de clase de equivalencia	
		SI	laoc	l l	
Número de prueba	Datos de entrada	V	alor	Salida Esperada	Clases
1				2 (la lista debe contener a los 2 empleadores)	1
impleado1.getListaPostulantes().get(0).getCliente().					
Dato de entrada	Clases de equivalencia	Aplica?	Motivo jdoc	ldentificador de clase de equivalencia 1	
Número de prueba 1	Datos de entrada	v	alor	Salida Esperada referencia al objeto Empleador1	Clases 1
mpleador1.getListaDePostulantes().size()					
Dato de entrada	Clases de equivalencia	Aplica?	Motivo jdoc	Identificador de clase de equivalencia 1	
Número de prueba	Datos de entrada	V	alor	Salida Esperada	Clases
1 _				2 (la lista debe contener a los 2 empleados)	1
mpleador1.getListaPostulantes().get(0).getCliente()					
Dato de entrada	Clases de equivalencia	Aplica ?	Motivo jdoc	Identificador de clase de equivalencia 1	
Número de prueba	Datos de entrada	V	alor	Salida Esperada	Clases
1				referencia al objeto Empleado1	1
mpleado1.getPuntaje()	-				
Dato de entrada	Clases de equivalencia	Aplica?	Motivo jdoc	Identificador de clase de equivalencia 1	
Número de prueba	Datos de entrada		alor	Salida Esperada 0 (puntaje= •5 -5 debido a que se encuentra fro en la lista del empleador 1 y ultimo en la lista del empleador 2)	Clases
empleado2.getPuntaje() Dato de entrada	Clases de equivalencia	Aplica?	Motivo	Identificador de clase de equivalencia	1
Balo de cimada	olases ac equilateriola	si	jdoc	1	1
Número de prueba	Datos de entrada		alor	Salida Esperada 0 (puntaje: +5-5 debido a que se encuentra fro en la lista del empleador 2 y ultimo en la lista del empleador 1)	Clases
				o (pulsales - o -o debido a que se encuenta a lo ema assa derempieador 2 y distino em a assa derempieador i)	<u> </u>
empleador1.getPuntaje() Dato de entrada	Clases de equivalencia	Aplica?	Motivo	Identificador de clase de equivalencia	1
		si	jdoc	1	
Número de prueba	Datos de entrada		alor	Salida Esperada	Clases
1 -				10 (debido a que se encuentra fro en la lista del empleado 1)	1
empleado2.getPuntaje()					
Dato de entrada	Clases de equivalencia	Aplica?	Motivo jdoc	Identificador de clase de equivalencia 1	1
Número de prueba	Datos de entrada	v	alor	Salida Esperada 10 (debido a que se encuentra liro en la lista del empleado 2)	Clases 1
sEstadoContratacion()					
Dato de entrada	Clases de equivalencia	Aplica?	Motivo	ldentificador de clase de equivalencia	1
N/	D. I. I		jdoc	1	Class
Número de prueba 1	Datos de entrada	- 	alor	Salida Esperada True	Clases 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 jdoc	ldentificador de clase de equivalencia 1	
Número de prueba 1	Datos de entrada		alor	Salida Esperada 10	Clases 1
empleador1.getPuntaje()					
Dato de entrada	Clases de equivalencia	Aplica?	Motivo jdoc	ldentificador de clase de equivalencia 1	
Número de prueba 1	Datos de entrada		alor	<u>Salida Esperada</u> 50	Clases 1
empleador2.getPuntaje()					
Dato de entrada	Clases de equivalencia	Aplica?	Motivo jdoc	ldentificador de clase de equivalencia 1	
Número de prueba	Datos de entrada	V	/alor	Salida Esperada	Clases
1				-20	
empleado1.getTicket() Dato de entrada	Clases de equivalencia	Aplica ?	Motivo jdoc	ldentificador de clase de equivalencia	
Número de prueba	Datos de entrada		alor		Clases
1	Datos de el mada			oaiwa Esperada null	1
empleadorf.getTicket() Dato de entrada	Clases de equivalencia	Aplica?	Motivo	ldentificador de clase de equivalencia	
Dato de entrada	Clases de equivalencia	si si	idoc	ider kilicador de clase de equivaler icia 1	
Número de prueba 1	Datos de entrada	V	alor	<u>Salida Esperada</u> null	Clases 1
empleador2.getTicket() Dato de entrada	Clases de equivalencia	Aplica?	Motivo jdoc	Identificador de clase de equivalencia	
Número de prueba	Datos de entrada		alor	' Salida Esperada	Clases
Numero de prueba 1	Datos de entrada		101	Salida Esperada null	1
getContratacionEmpleadoPretenso()					
Dato de entrada Empleado	Clases de equivalencia Empleado!= null	Aplica ?	Motivo jdoc	ldentificador de clase de equivalencia 1	
Número de prueba 1	Datos de entrada Empleador		alor ontratacion vigente	Salida Esperada referencia al empleador con el que se tuvo la contratacion	Clases 1
getContratacionEmpleador()					
Dato de entrada	Clases de equivalencia	Aplica?	Motivo	ldentificador de clase de equivalencia	
Empleador	Empleador!=null	si	jdoc	· ·	
Número de prueba 1	Datos de entrada		alor ontratacion vigente	Salida Esperada referencia al empleado con el que se tuvo la contratacion	Clases 1
sEstadoContratacion()					
Dato de entrada	Clases de equivalencia	Aplica ?	Motivo jdoc	ldentificador de clase de equivalencia 1	
Número de prueba	Datos de entrada		/alor	Salida Esperada False	Clases
				1 Mars	

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 != null	si	jdoc	1	
nombre(string)	nombre == null	no	el el	2	
Número de prueba	Datos de entrada	1	/alor	Salida Esperada	Clases
1	nombre	"Age	ncia.xml"	Apertura correcta de Input (no lanzamiento de excepcion)	1
abrirOutput()					
Dato de entrada	Clases de equivalencia	Aplica?	Motivo	ldentificador de clase de equivalencia	
	nombre != null	si	jdoc	1	
nombre (string)	nombre == null	no	el el	2	
Número de prueba	Datos de entrada	1	/alor	Salida Esperada	Clases
1	nombre	"Age	ncia.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	ldentificador de clase de equivalencia	
		si	jdoc	1	
Número de prueba	Datos de entrada	Valor		Salida Esperada	Clases
1		7	Cerrado correcto de Input (no lanzamiento de excepcion)		1
cerrarOutput()					
Dato de entrada	Clases de equivalencia	Aplica?	Motivo	ldentificador 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.

etLimiteInferior()					
Dato de entrada	Clases de equivalencia	Aplica?	Motivo	ldentificador de clase de equivalencia	
limiteInferior (int)	limInferior ∈ Z	si	jdoc	1	
Número de prueba	Datos de entrada		/alor	Salida Esperada	Clase
1	limiteInferior		999	Seteo correcto (verifica con get)	1
LimiteSuperior()					
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	1	/alor	Salida Esperada	Class
1	limiteSuperior		9999	Seteo correcto (verifica con get)	1
EstadoContratacion()					
Dato de entrada	Clases de equivalencia	Aplica?	Motivo	ldentificador de clase de equivalencia	
	estadoContratacion = true	si	idoc	1	
estadoContratacion	estadoContratacion = false	si	jdoc	2	
Número de prueba	Datos de entrada estadoContratacion		/alor true	Salida Esperada Seteo correcto (verifica con get)	Clase
'	estadocontratación		ude	Select confecto (Vennos con ger)	'
tEmpleados()					
Dato de entrada	Clases de equivalencia	Aplica?	Motivo	ldentificador de clase de equivalencia	
leados (HashMap <string, empleadopretenso<="" td=""><td>Empleados != null Empleados== null</td><td>si si</td><td>jdoc jdoc</td><td>1 2</td><td></td></string,>	Empleados != null Empleados== null	si si	jdoc jdoc	1 2	
Número de prueba	Datos de entrada	-	/alor	Salida Esperada	Clase
1 1	Empleados		nap vacio	seteo correcto (que se verifica con el get)	1
tEmpleadores()					
Dato de entrada	Clases de equivalencia	Aplica?	Motivo	ldentificador de clase de equivalencia	
mpleadores (HashMap <string,empleador>)</string,empleador>	Empleadores != null	si	jdoc	1	
(1000)	Empleadores == null	si	jdoc (inv clase)	2	
Número de prueba	Datos de entrada	1	/alor	Salida Esperada	Clase
1	Empleadores		nap vacio	seteo correcto (que se verifica con el get)	1
tContrataciones					
Dato de entrada	Clases de equivalencia	Aplica?	Motivo	Identificador de clase de equivalencia	
Contrataciones (ArrayList <contratacion>)</contratacion>	Contrataciones != null Contrataciones == null	si si	jdoc jdoc (inv clase)	1 2	
Número de prueba	Datos de entrada	,	/alor	Salida Esperada	Clase
1	Contrataciones		ataciones (vacia)	seteo correcto (que se verifica con el get)	1
(ComisionesUsuarios()					
Dato de entrada	Clases de equivalencia comisionesUsuarios l= null	Aplica?	Motivo	Identificador de clase de equivalencia	
nisionesUsuarios (Hashmap <cliente,double></cliente,double>	comisionesUsuarios := null comisionesUsuarios == null	SI Si	jdoc jdoc (inv clase)	1 2	
Número de prueba	Datos de entrada		/alor	Salida Esperada	Clase

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	V	alor	Salida Esperada	Clases
1	Datos de entrada	<u> </u>		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	V	alor	Salida Esperada	Clases
1				referencia al objeto Contrataciones de Agencia	1
getEmpleadores()					
Dato de entrada	Clases de equivalencia	Aplica ?	Motivo jdoc	Identificador de clase de equivalencia	
			1900		
Número de prueba	Datos de entrada	V	alor	Salida Esperada	Clases
1				referencia al objeto Empleadores de Agencia	1
getEmpleados()					
Dato de entrada	Clases de equivalencia	Aplica ?	Motivo jdoc	Identificador de clase de equivalencia 1	
		-	1000	·	
Número de prueba	Datos de entrada	V	alor	Salida Esperada	Clases
1				referencia al objeto Empleados de Agencia	1
getLimiteInferior()					
Dato de entrada	Clases de equivalencia	Aplica ?	Motivo jdoc	ldentificador de clase de equivalencia 1	
Número de prueba 1	Datos de entrada	V.	alor	Salida Esperada 989	Clases
				333	
getLimiteSuperior()					
Dato de entrada	Clases de equivalencia	Aplica ?	Motivo jdoc	ldentificador de clase de equivalencia 1	
Número de prueba 1	Datos de entrada	- 	alor	Salida Esperada 9999	Clases 1
isEstadoContratacion() Dato de entrada	Clases de equivalencia	Aplica?	Motivo	Identificador de clase de equivalencia	
Dato de entrada	Crases de équivalencia	si si	jdoc	identinicador de cidad de équivalencia 1	
Número de prueba 1	Datos de entrada	-1 _	alor	Salida Esperada true	Clases 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	l v	alor	Salida Esperada	Clases
1				referencia al objeto ComisionesUsuario de AgenciaDTO	1
qetContrataciones()					
Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	ldentificador de clase de equivalencia	
		si	jdoc	1	
Número de prueba	Datos de entrada	V	alor	Salida Esperada	Clases
1				referencia al objeto Contrataciones de AgenciaDTO	1
getEmpleadores()					
Dato de entrada	Clases de equivalencia	Aplica ?	Motivo jdoc	Identificador de clase de equivalencia	
		SI	laoc	l .	
Número de prueba	Datos de entrada	-	alor	Salida Esperada	Clases
1	Battos de entrado			referencia al objeto Empleadores de AgenciaDTO	1
				• • • •	
getEmpleados()					
Dato de entrada	Clases de equivalencia	Aplica?	Motivo	ldentificador de clase de equivalencia	
		si	jdoc	1	
Número de prueba	5. 1 1		alor	Salida Esperada	Clases
ivumero de prueba	Datos de entrada	-+ `	alui	referencia al objeto Empleados de AgenciaDTO	Clases
· ·				reference at outro Empleados de Agenciad To	
qetLimiteInferior()					
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? Mo	vo Identificador de clase de equivalencia	
		si jd	0 1	
Número de prueba	Datos de entrada	Valor	Salida Esperada	Clases
1			9999	1
isEstadoContratacion()				
Dato de entrada	Clases de equivalencia	Aplica? Mo	vo Identificador de clase de equivalencia	
		si jd	c 1	
Número de prueba	Datos de entrada	Valor	Salida Esperada	Clases
1			true	1 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	ldentificador de clase de equivalencia	
		si	jdoc	1	
Número de prueba	Datos de entrada	V	'alor	Salida Esperada	Clases
1		1		se orea el archivo y se persiste agencia en él.	

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.

LimiteInferior()					
Dato de entrada	Clases de equivalencia	Aplica?	Motivo	ldentificador de clase de equivalencia	
		si	jdoc	1	
Número de prueba	Datos de entrada		/alor	Salida Esperada	Clase:
1				999	1
.imiteSuperior()					
Dato de entrada	Clases de equivalencia	Aplica?	Motivo	ldentificador de clase de equivalencia	
		si	jdoc	1	
Número de prueba	Datos de entrada	V	/alor	Salida Esperada	Clase
1		1		9999	1
Empleadores()					
Dato de entrada	Clases de equivalencia	Aplica?	Motivo	Identificador de clase de equivalencia	
		si	jdoc	1	
Número de prueba	Datos de entrada	- V	/alor	Salida Esperada	Clase
1				referencia a hashmap de empleadores con el empleador cargado (no vacio)	1
Empleados()					
Dato de entrada	Clases de equivalencia	Aplica ?	Motivo jdoc	Identificador de clase de equivalencia	
		31	Juou	1	
Número de prueba	Datos de entrada		/alor	Salida Esperada	Clase
1				referencia a hashmap de empleados con el empleado cargado (no vacio)	1 1
tadoContratacion()					
Dato de entrada	Clases de equivalencia	Aplica ?	Motivo	ldentificador de clase de equivalencia	
		si	jdoc	1	



Número de prueba	Datos de entrada	Valor	Salida Esperada	Clase
1			false	1
Contrataciones()				
Dato de entrada	Clases de equivalencia	Aplica? Motiv	ldentificador de clase de equivalencia	
		si jdoo	1	
		Valor		
Número de prueba	Datos de entrada	Valor	Salida Esperada referencia a lista de contrataciones con la contratacion cargada (no vacia)	Clase
1			referencia a lista de contrataciones con la contratación cargada (no vacia)	-+-
ComisionesUsuarios()				
Dato de entrada	Clases de equivalencia	Aplica? Motiv	o Identificador de clase de equivalencia	
		si jdoo	1	
Número de prueba	Datos de entrada	Valor	Salida Esperada	Clase
1			referencia al hashmap de comisionesUsuarios con la comision cargada (no vacio)	1
Coincidencias()				
Dato de entrada	Clases de equivalencia	Aplica? Motiv	dentificador de clase de equivalencia	
		si jdoo		
		Valor		
Número de prueba 1	Datos de entrada	Valor	Salida Esperada referencia a lista de coincidencias con la coincidencia cargada (no vacia)	Clase 1
TipoUsuario()				
Dato de entrada	Clases de equivalencia	Aplica? Motiv		_
Número de prueba	Datos de entrada	Valor	Salida Esperada	Clas

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.

enario con atributo persistencia null					
cenario con atributo persistencia nuii ardarAgencia∩					
Dato de entrada	Clases de equivalencia	Aplica?	Motivo	11.29 1.1.1.1.1.1	
Dato de entrada				Identificador de clase de equivalencia	
nombreArchivo (string)	nombre!=null y nombre valido de archivo	si	jdoc idoc	2	
	nombre==null	no	laoc	۷	
Número de prueba	Datos de entrada	V	alor	Salida Esperada	Clase
1	nombre	"AgenciaPe	rsistencia.xml"	False (se carga correctamente)	1
				` · · · · · · · · · · · · · · · · · · ·	
cenario con atributo persistencia != null (Pers	I SAN SINGS				
cenano con acibuto persistencia := nui (r eis ardarAgencia∏	SISTERICIANIVIL)				
Dato de entrada	Clases de equivalencia	Aplica?	Motivo	ldentificador de clase de equivalencia	
nombreArchivo (string)	nombre!=null y nombre valido de archivo	si	jdoc	1	
nombrewichieo (saing)	nombre==null	no	jdoc	2	
Número de prueba	Datos de entrada		alor	Salida Esperada	Clase
1 Ivailleio de pideba	nombre	"AgenciaPersistencia.xml"		True (se carga correctamente)	1
· · · · · · · · · · · · · · · · · · ·	TOTAL	rigenoisi	I DID CHOICE III	rior (or varga some varieties)	
cenario con atributo persistencia null rgarAgencia()					
Dato de entrada	Clases de equivalencia	Aplica?	Motivo	Identificador de clase de equivalencia	
	nombrel=null y nombre valido de archivo	si	idoc	1	
nombreArchivo (string)	nombre==null	no	jdoc	2	
Número de prueba	Datos de entrada		alor	Salida Esperada	Clase
1 1	nombre		rsistencia.xml"	True (se carga correctamente)	- Ciase
<u> </u>	nombre	Agericiai e	I SISTERICIO.AITH	rrue (se varga correctamente)	
cenario con atributo persistencia != null (Pers rgarAgencia()	sistenciaXML)				
Dato de entrada	Clases de equivalencia	Aplica?	Motivo	Identificador de clase de equivalencia	
	nombre!=null y nombre valido de archivo	si	idoc	1	
nombreArchivo (string)	nombre==null	no	jdoc	2	
Número de prueba	Datos de entrada		alor	Salida Esperada	Clase
radinero de proeba	nombre		rsistencia.xml"	False (se carga correctamente)	Clase

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.

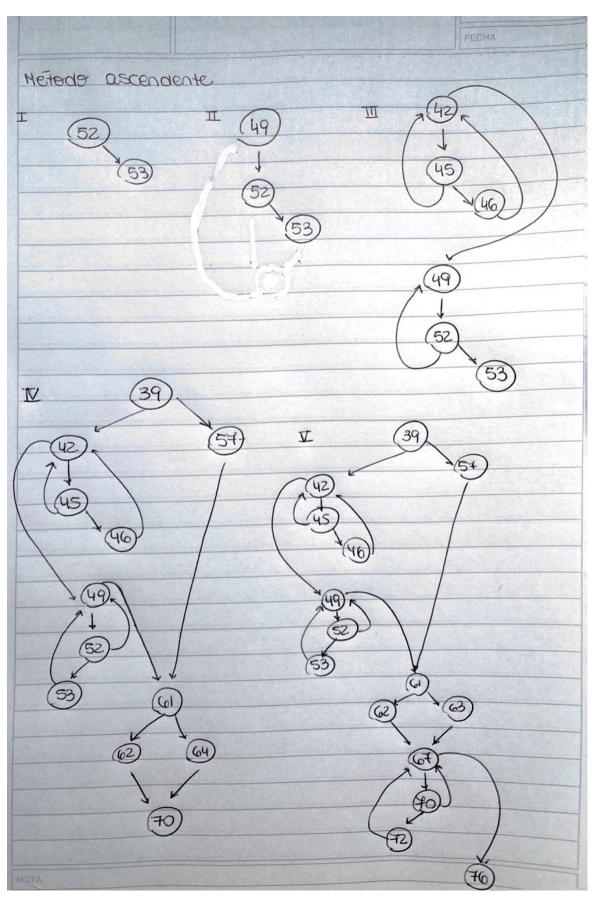
2.3 Prueba unitaria caja blanca

Tomando la siguiente enumeración como referencia:



```
310
       public Cliente aplicaPromo(boolean promoPorListaDePostulantes,
32
                HashMap<String, EmpleadoPretenso> empleados, HashMap<String, Empleador> empleadores)
33
34
           Iterator clientes = null;
35
           int contadorEmpleador = 0;
36
           int contadorEmpleadoPretenso = 0;
37
           Cliente clienteBeneficiado = null;
38
39
           if (promoPorListaDePostulantes)
40
41
               Iterator<Empleador> itEmpleadores = empleadores.values().iterator();
42
               while (itEmpleadores.hasNext())
43
44
                    Empleador empleador = itEmpleadores.next();
45
                    if (empleador.getListaDePostulantes() != null)
46
                        contadorEmpleador += empleador.getListaDePostulantes().size();
47
48
               Iterator<EmpleadoPretenso> itEmpleados = empleados.values().iterator();
49
               while (itEmpleados.hasNext())
50
51
                    EmpleadoPretenso empleadoPretenso = itEmpleados.next();
52
                    if (empleadoPretenso.getListaDePostulantes() != null)
53
                        contadorEmpleadoPretenso += empleadoPretenso.getListaDePostulantes().size();
54
55
           } else
56
           -{
57
               contadorEmpleador = empleadores.size();
               contadorEmpleadoPretenso = empleados.size();
58
59
           }
60
61
           if (contadorEmpleador > contadorEmpleadoPretenso)
62
               clientes = empleadores.values().iterator();
63
64
               clientes = empleados.values().iterator();
65
66
           int puntajeMaximo = Integer.MIN VALUE;
67
           while (clientes.hasNext())
68
69
               Cliente cl = (Cliente) clientes.next();
70
               if (cl.getPuntaje() > puntajeMaximo)
71
72
73
                   puntajeMaximo = cl.getPuntaje();
                   clienteBeneficiado = cl;
75
           1
76
           return clienteBeneficiado;
77
```





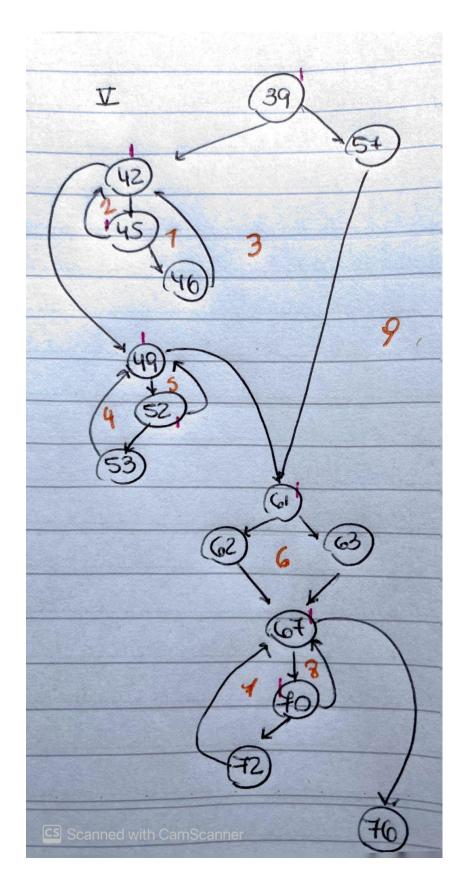
Cálculo de complejidad ciclomática



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:

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

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

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

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

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

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

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

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

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



C7	postulantes en total	con 2 con 3 =true	Empleador mayor puntaje	de	4	
C8	postulantes en total	con 2 con 3 =true	Empleador mayor puntaje	de	4	
C9	HashMap empleados de tama HashMap empleadores de tam promoPorListaDePostulantes=	Empleador mayor puntaje	de	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.