Taller de programación I
Informe Trabajo Práctico final

Grupo 4
Trinitario Bruno
Juares Alan
Basualdo Juan Ignacio
Olave Juan Ignacio

Indice:

Introducción:

En este informe mostraremos las pruebas realizadas en caja negra, caja blanca y test de GUI para verificar la funcionalidad de un sistema que permite registro de empleados pretensos y empleadores, para conectarlos según las características de cada uno.

Todos los métodos del programa del modelo datos y negocios, han sido testeados con caja negra, a excepción del método aplicaPromo() que ha sido testeado con caja blanca. La interfaz fue testeada con test de GUI.

En el caso del test de caja blanca, utilizamos el método ascendente para crear el grafo, y método general para la obtención de caminos.

Resumen de errores detectados:

Dentro del paquete modelo negocios:

- CalculaPremiosCastigosAsignaciones: error en el escenario 1 cuando debe calcular el puntaje del empleado pretenso con menor puntaje.
- CargarAgencia: setea incorrectamente el límite inferior de remuneración.
- GatillarRonda: error en el cálculo del puntaje del empleado en el escenario 1.
- GatillarRonda: en el escenario 2, cuando a un cliente no se lo elige deberían restarsele de su puntaje 20 puntos, pero se le suman.
- RegistrarUsuario: permite crear un empleador existente, debería lanzar una excepción. También permite registrar un empleador con rubro invalido, los unicos rubros posibles con los que puede ser registrado son: salud, comercio local y comercio internacional (debería lanzar una excepción).

Dentro del paquete modelo de datos:

- En la clase Empleador los atributos Teléfono y RealName no quedan con los resultados esperados, se presume que ambos están intercambiados, ya que cada uno obtiene el valor esperado del otro.
- Dentro de la clase Ticket, el método ComparaciónTotal calcula incorrectamente la comparación del ticket actual con el pasado por parámetro.

En el test de GUI:

- Cuando la Agencia está en estado de contratación no debería permitir crear un ticket a un cliente, sin embargo luego de que hicimos el testeo se crea el ticket.
- Debería mostrar cuando queremos registrar un cliente ya sea empleador o empleado y completar los campos restantes al registrar. Si el campo confirmar contraseña no coincide con el campo contraseña debería lanzar un cartel del tipo “Mensajes.PASS_NO_COINCIDE” pero no lo hace.
- A la hora de registrar un tipo Empleador si el usuario ya existe debería devolver el mensaje “Mensajes.USUARIO_REPETIDO” pero esto no ocurre.

Tests de Caja negra:

Para realizar los test de caja negra, dividimos la bateria de pruebas en cada paquete (modelo Datos, modelo Negocios e interfaz), y así mismo cada uno en cada clase.

Las clases Usuario y Cliente no serán testeadas ya que son abstractas y los métodos a testear fueron sobreescritos por todas las clases hijas.

Modelo Datos:

Empleador:

Métodos

Calcula comision
constructor

Escenarios

Nro Escenario	Descripción
1	No requiere escenario

Tabla de particiones

Método	Condición de entrada	Clase válida	Clase inválida
constructor	los parametros son diferentes de null, rubro y tipoPersona de los tipos esperados (contemplados en la clase Constantes)	ussername != NULL (1)	ussername = NULL
		password != NULL (2)	password = NULL
		RealName != NULL (3)	RealName = NULL
		Telefono != NULL (4)	Telefono = NULL
		Rubro != NULL y existen en la clase constantes (5)	Rubro = NULL o no existen en la clase constantes
		TipoPersona != NULL y existen en la clase constantes (6)	TipoPersona = NULL o no existen en la clase constantes
calculaComision	Objeto ticket valido como entrada	Ticket != null y con atributos internos validos (7)	Ticket == null o parametros internos invalidos

Bateria de pruebas

Método	Tipo de clase	Valores de entrada	Salida esperada	Clases de prueba cubiertas
calculaComision	valida	ticket valido	Se calcula la comision correctamente	7
constructor	validas	ussername != NULL	Se crea el objeto con referencias correctas a los parametros	1
		password != NULL		2
		RealName != NULL		3
		Telefono != NULL		4
		Rubro != NULL y existen en la clase constantes		5
		TipoPersona != NULL y existen en la clase constantes		6

Errores detectados:

Ninguno

EmpleadoPretenso:

Métodos

Constructor
calculaComision

Escenarios

Nro Escenario	Descripción
1	No requiere escenario

Tabla de particiones

Método	Condición de entrada	Clase válida	Clase inválida
Constructor	Los parametros String son	ussername != NULL (1)	ussername = NULL

	distintos de null, la edad es mayor que cero.	password != NULL (2)	password = NULL
		RealName != NULL (3)	RealName = NULL
		Telefono != NULL (4)	Telefono = NULL
		apellido != NULL (5)	apellido != NULL
		edad > 0 (6)	edad <= 0
calculaComision	Objeto ticket valido como entrada	Ticket != null y con atributos internos validos (7)	Ticket == null o parametros internos invalidos

Bateria de pruebas

Método	Tipo de clase	Valores de entrada	Salida esperada	Clases de prueba cubiertas
Constructor	valida	ussername != NULL	Se crea el objeto EmpleadoPretenso con referencias correctas a los parametros	1
		password != NULL		2
		RealName != NULL		3
		Telefono != NULL		4
		apellido != NULL		5
		edad = 55544		6
calculaComision	valida	ticket valido	Se calcula la comision correctamente	7

Errores detectados:

Ninguno

Ticket:

Métodos

getComparacionExperiencia(Ticket otro)
getComparacionEstudios(Ticket otro)
getComparacionJornada(Ticket otro)
getComparacionLocacion(Ticket otro)

getComparacionPuesto(Ticket otro)
getComparacionRemuneracion(Ticket otro)
getComparacionTotal(Ticket otro)
setRemuneracion(int remuneracion)

Escenarios

Nro Escenario	Descripción
1	No requiere escenario

Tabla de particiones

Método	Condicion de entrada	Clase valida	Clase invalida
getComparacionExperiencia(Ticket otro)	el parametro otro es diferente de null	Ticket != null y con atributos internos validos (1)	Ticket == null o parametros internos invalidos
getComparacionEstudios(Ticket otro)	el parametro otro es diferente de null	Ticket != null y con atributos internos validos (2)	Ticket == null o parametros internos invalidos
getComparacionJornada(Ticket otro)	el parametro otro es diferente de null	Ticket != null y con atributos internos validos (3)	Ticket == null o parametros internos invalidos
getComparacionLocacion(Ticket otro)	el parametro otro es diferente de null	Ticket != null y con atributos internos validos (4)	Ticket == null o parametros internos invalidos
getComparacionPuesto(Ticket otro)	el parametro otro es diferente de null	Ticket != null y con atributos internos validos (5)	Ticket == null o parametros internos invalidos
getComparacionRemuneracion(Ticket otro)	el parametro otro es diferente de null	Ticket != null y con atributos internos validos (6)	Ticket == null o parametros internos invalidos
getComparacionTotal(Ticket otro)	el parametro otro es diferente de null	Ticket != null y con atributos internos validos (7)	Ticket == null o parametros internos invalidos
setRemuneracion(int remuneracion)	remuneracion es mayor que cero	remuneracion > 0 (8)	remuneracion <= 0

Bateria de pruebas

Método	Tipo de clase	Valores de entrada	Salida esperada	Clases de prueba cubiertas
getComparacionExperiencia(Ticket otro)	valida	ticket valido	Valor del calculo de comparacion	1
getComparacionEstudios(Ticket otro)		ticket valido	Valor del calculo de comparacion	2
getComparacionJornada(Ticket otro)		ticket valido	Valor del calculo de comparacion	3
getComparacionLocacion(Ticket otro)		ticket valido	Valor del calculo de comparacion	4
getComparacionPuesto(Ticket otro)		ticket valido	Valor del calculo de comparacion	5
getComparacionRemuneracion(Ticket otro)		ticket valido	Valor del calculo de comparacion	6
getComparacionTotal(Ticket otro)		ticket valido	Valor del calculo de comparacion	7
setRemuneracion(int remuneracion)		remuneracion = 23334522	Se modifica el valor de la remuneracion	8

Errores detectados:

getComparacionTotal() no compara bien el total.

Contratacion:

Métodos

Constructor

Escenarios

Nro Escenario	Descripción
1	No requiere escenario

Tabla de particiones

Método	Condicion de entrada	Clase valida	Clase invalida
Constructor	Parametros diferentes de null	Empleador != NULL (1)	Empleador = NULL
		Empleador pretenso != NULL (2)	Empleador = NULL

Bateria de pruebas

Método	Tipo de clase	Valores de entrada	Salida esperada	Clases de prueba cubiertas
Constructor	Valida	Empleador != NULL	Se crea el objeto con referencias correctas y fecha correcta	1
		Empeador pretenso != NULL		2

Errores detectados:

Ninguno

ClientePuntaje:

Métodos

Constructor
setCliente

Escenarios

Nro Escenario	Descripción
1	No requiere escenario

Tabla de particiones

Método	Condicion de entrada	Clase valida	Clase invalida
constructor	Usuario diferente de null	Usuario no nulo(1)	NULL

setCliente	Usuario diferente de null	Usuario no nulo(2)	NULL

Bateria de pruebas

Método	Tipo de clase	Valores de entrada	Salida esperada	Clases de prueba cubiertas
constructor	Valida	Usuario no nulo	Se crea ClientePuntaje con referencia correcta de usuario	1
setCliente	Valida	Usuario no nulo	Se crea ClientePuntaje con referencia correcta de usuario	2

Errores detectados:

Ninguno

Modelo Negocios:

Agencia:

Métodos

aplicaPromo
login
calculaPremiosCastigosAsignaciones
cargarAgencia
crearTicketEmpleado
eliminarTicket
gatillarRonda
generaPostulantes
getContratacionEmpleadoPretenso
getContratacionEmpleador
guardarAgencia
match
registroEmpleado
registroEmpleador
setLimitesRemuneracion

Escenarios

Nro Escenario	Descripción
1	La lista de usuarios es nula
2	La lista de usuarios esta vacia
3	La lista de usuarios tiene un usuario empleado con nombre "Pedrito" y contrasena "1234", y otro con nombre vacio y contrasena vacia
4	Las listas de postulantes no estan vacias ni son nulas
5	estadoDeContratación = true y se penaliza a los empleadores que tienen un ticket asociado pero no fueron elegidos en una ronda de contración
6	estadoDeContratación = true y no se penaliza a los empleadores que tienen un ticket asociado ya que fueron elegidos en una ronda de contración
7	estadoDeContratacion = false
8	listas de empleados y empleadores con elementos validos
9	la lista de contrataciones de la agencia tiene a emp1
10	la lista de contrataciones de la agencia tiene a empleador1
11	la IDE tiene permiso para escribir en el disco
12	la IDE no tiene permiso para escribir en el disco
13	el nombre de usuario ya esta en uso
14	nombreUsuario, pass, nombreReal, apellido, telefono son nulos
15	nombreUsuario, pass, nombreReal, tipoPersona, telefono son nulos o el tipo de persona no es "JURIDICA" ni "FISICA"o en caso de que rubro no sea "SALUD" "COMERCIO LOCAL" "COMERCIO INTERNACIONAL"
16	ocurren problemas durante la operacion de lectura del archivo
17	a clase AgenciaDTO no se encuentra en el part durante la deserializacion del objeto

Tabla de particiones

Método	Condicion de entrada	Clase valida	Clase invalida
aplicaPromo(boolean promoPorListaDePo	promoPorListaDePostulantes es	promoPorListaDePostulantes = true (1)	promoPorListaDePostulantes != true o false

stulantes)	true o false	promoPorListaDePostulantes = false (2)	
login(String nombreUsuario, String pass)	nombre de usuario != NULL	Nombre de usuario = "Pedrito"(3), nombre de usuario = ""(3,1)	Nombre de usuario = NULL, nombre de usuario "Pedrito" pero no existe en la lista (3.2)
	contrasena != NULL	Contrasena = "1234" (4), contrasena = "" (4.1)	Contrasena =NULL, contrasena = "1234" pero es incorrecta en el usuario (4.2)
calculaPremiosCastigosAsignaciones()	las listas de postulantes están ordenadas	las listas de postulantes están ordenadas (5)	listas de postulantes desordenadas
cargarAgencia(String nombreArchivo)	nombreArchivo != NULL	nombreArchivo = "taller" (29)	ocurren problemas durante la operacion de lectura del archivo (30) ; la clase AgenciaDTO no se encuentra en el part durante la deserializacion del objeto (31)
cerrarSesion()	No tiene	-	-
crearTicketEmpleado (String locacion, int remuneración, String jornada, String puesto, String experiencia, String estudios, Cliente cliente)	No tiene	el cliente es un empleado y no tiene ticket activo(7) ; el cliente es un empleado y tiene ticket activo(8) ; el cliente es un empleado pero ya ha sido contratado (9)	el cliente no es un empleado (10)
eliminarTicket()	No tiene	hay un cliente logueado en la aplicacion (11)	hay un cliente logueado en la aplicacion y el estado de contratacion no esta en un estado valido (12) ; no hay un cliente logueado en la aplicacion (13)
gatillarRonda()	No tiene	Escenario 5	No tiene
	No tiene	Escenario 6	No tiene
	No tiene	Escenario 7	No tiene
generaPostulantes()	No tiene	Escenario 8	No tiene
getContratacionEmpleadoPretenso()	empleado != NULL y registrado en el sistema	empleado = ep1 y registrado en el sistema (14)	empleado = NULL y registrado en el sistema ; empleado != NULL y no registrado en el sistema ; empleado = NULL y no registrado en el sistema
getContratacionEmpleador()	empleador != NULL y	empleador = empleador1 y registrado en el sistema	empleador = NULL y registrado en el sistema ; empleador !=

	registrado en el sistema	(15)	NULL y no registrado en el sistema ; empleador = NULL y no registrado en el sistema
guardarAgencia()	nombreArchivo != NULL	nombreArchivo = "arch" y tiene exito al guardar (devuelve true)(16) ; nombreArchivo = "arch" y fracasa al guardar (devuelve false)(17)	la IDE no tiene permiso para escribir en el disco (18)
match(Empleador empleador, EmpleadoPretenso empleado)	empleador != NULL ; empleado != NULL ; ambos registrados en el sistema	empleador = emp1 ; empleado = emplea1 (19)	No tiene
registroEmpleado(String nombreUsuario, String pass, String nombreReal, String apellido, String telefono, int edad)	No tiene	numbreUsuario = "a1" ; pass = "123" ; nombreReal = "pedro" ; apellido = "zz" ; telefono = 223 ; edad = NULL (20)	nombreUsuario == NULL ; pass == NULL ; nombreReal == NULL ; apellido == NULL ; telefono == NULL ; edad == NULL (22)
			nombreUsuario == NULL ; pass == NULL ; nombreReal == NULL ; apellido == NULL ; telefono == NULL ; edad = 22 (23)
	No tiene	numbreUsuario = "a1" ; pass = "123" ; nombreReal = "pedro" ; apellido = "zz" ; telefono = 223 ; edad = 22 (21)	numbreUsuario = "a1" ; pass = "123" ; nombreReal = "pedro" ; apellido = "zz" ; telefono = 223 ; edad = NULL ; el nombre de usuario ya esta en uso (24)
			numbreUsuario = "a1" ; pass = "123" ; nombreReal = "pedro" ; apellido = "zz" ; telefono = 223 ; edad = NULL ; el nombre de usuario ya esta en uso (25)
registroEmpleador(String nombreUsuario, String pass, String nombreReal, String telefono, String tipoPersona, String rubro)	No tiene	nombreUsuario != NULL ; pass != NULL ; nombreReal != NULL ; telefono != NULL ; tipoPersona = FISICA ; rubro = SALUD (26)	numbreUsuario = "a1" ; pass = "123" ; nombreReal = "pedro" ; telefono = 223 ; tipoPersona = NULL; rubro = NULL ; (27)
			numbreUsuario = "a1" ; pass = "123" ; nombreReal = "pedro" ; telefono = 223 ; tipoPersona = FISICA ; rubro = SALUD ; el nombre de usuario ya esta en uso (28)

Bateria de pruebas

Método	Tipo de clase	Valores de entrada	Salida esperada	Clases de prueba cubiertas
aplicaPromo(boolean promoPorListaDePostulantes)	Valida	promoPorListaDePostulantes = true	Selecciona y beneficia un cliente especifico	1
		promoPorListaDePostulantes = false		2
calculaPremiosCastigosAsignaciones	Valida	las listas de postulantes están ordenadas	otorga premios y castigos a los clientes que participan en el proceso de seleccion	5
		Escenario 4		
login(String nombreUsuario, String pass)	valida	Nombre de usuario = "Pedrito"		3
		Contrasena = "1234"	Loguea correctamente y devuelve 0	4
		Escenario 3		
		Nombre de usuario = ""		3.1
		Contrasena = ""	Loguea correctamente y devuelve 0	4.1
		Escenario 3		
	invalida	Nombre de usuario = ""		
		Contrasena = "1234"	ContraException()	4.2
		Escenario 3		
		Nombre de usuario = "Pedrito"		3.2
		Contrasena = "1234"	NombreUsuarioException()	
		Escenario 1		
		Nombre de usuario = "Pedrito"		3.2
		Contrasena = "1234"	NombreUsuarioException()	
		Escenario 2		
cerrarSesion()	Valida	-	El atributo tipoUsuario en clase Agencia debe ser -1	-
crearTicketEmpleado (String locacion, int remuneración, String jornada, String puesto, String experiencia, String	valida	el cliente es un empleado y no tiene ticket activo	crea un nuevo ticket para un empleado con los datos enviados por parametros	7
		el cliente es un empleado y	borra el ticket anterior y	8

estudios, Cliente cliente)		tiene ticket activo	crea uno nuevo	
		el cliente es un empleado pero ya ha sido contratado	ImposibleModificarTickets Exception()	9
	invalida	el cliente no es un empleado	error	10
eliminarTicket()	valida	hay un cliente logueado en la aplicacion y el estado de contratación esta en un estado valido	elimina el ticket del usuario logueado	11
	invalida	hay un cliente logueado en la aplicacion y el estado de contratación no esta en un estado valido	ImposibleModificarTickets Exception()	12
	invalida	no hay un cliente logueado en la aplicacion	error	13
gatillarRonda()	valida	**Escenario 5**	se penaliza a los empleadores que tienen un toicket asociado pero no fueron elegidos en una ronda de contratacion, y "limpia" o elimina todas las conexiones entre candidatos y puestos de trabajo despues de una ronda de contratacion. Al finalizar la ejecucion estadoDeContratacion = false	
		Escenario 6	"limpia" o elimina todas las conexiones entre candidatos y puestos de trabajo despues de una ronda de contratacion. Al finalizar la ejecucion estadoDeContratacion = false	
		Escenario 7	se generan nuevos postulantes para los trabajos disponibles y se calculan los premios o castigos basados en el resultado de la ronda anterior. Al finalizar la	

			ejecucion estadoDeContratacion = true	
generaPostulantes	valida	**Escenario 8**	establece la lista de EmpleadoPretenso ordenados de mayor a menor como la lista de postulantes para el Empleador y establece una lista de Empleadores ordenada por puntaje de mayor a menor de compatibilidad y la establece como la lista de postulantes para ese EmpleadoPretenso.	
getContratacionEmpleadoPretenso()	valida	empleado = ep1 **Escenario 9**	busca la contratacion asociada (devuelve objeto cliente) a un EmpleadoPretenso especifico en la lista de contrataciones de la agencia.	14
getContratacionEmpleador()	valida	empleador = empleador1 **Escenario 10**	busca la contratacion asociada (devuelve un objeto Usuario) a un Empleador especifico en la lista de contrataciones de la agencia	15
guardarAgencia()	valida	nombreArchivo = "arch" **Escenario 11**	guarda la informacion de una agencia en un archivo utilizando un mecanismo de persistencia.	16
		nombreArchivo = "arch" **Escenario 11**	no guarda el archivo	17
	invalida	nombreArchivo = "arch" **Escenario 12**	IOException	18
match(Empleador empleador, EmpleadoPretenso empleado)	valida	empleador = emp1 ; empleado = emplea1	Los tickets del Empleado y del emepleador se eliminan sin generar penalizaciones.	19

registroEmpleado(String nombreUsuario, String pass, String nombreReal, String apellido, String telefono, int edad)	valida	numbreUsuario = "a1" ; pass = "123" ; nombreReal = "pedro" ; apellido = "zz" ; telefono = 223 ; edad = NULL	se registra un nuevo empleado y se devuelve un objeto Cliente (el empleado registrado)	20
		numbreUsuario = "a1" ; pass = "123" ; nombreReal = "pedro" ; apellido = "zz" ; telefono = 223 ; edad = 22	se registra un nuevo empleado y se devuelve un objeto Cliente (el empleado registrado)	21
	invalida	**Escenario 13**	NewRegisterException	24 y 25
		Escenario 14	ImposibleCrearEmpleado Exception	22 y 23
registroEmpleador(String nombreUsuario, String pass, String nombreReal, String telefono, String tipoPersona, String rubro)	valida	nombreUsuario != NULL ; pass != NULL ; nombreReal != NULL ; telefono != NULL ; tipoPersona = FISICA ; rubro = SALUD (26)	se registra un nuevo Empleador. Se devuelve un objeto Cliente con el Empleador registrado	26
	invalida	**Escenario 15**	ImposibleCrearEmpleador Exception	27
		Escenario 13	NewRegisterException	28
cargarAgencia(String nombreArchivo)	valida	nombreArchivo = "taller"	se carga una instancia de la clase Agencia desde un archivo serializado	29
	invalida	**Escenario 16**	IOException	30
		Escenario 17	ClassNOtFoundExcepti on	31

Errores detectados:

CargarAgencia:

El atributo LimiteInferior no se carga con el valor esperado.

CalculaPremiosCastigosAsignaciones:

Error en el escenario 1 cuando debe calcular el puntaje del empleado pretenso con menor puntaje.

GatillarRonda:

Error en el cálculo del puntaje del empleado en el escenario 1.

En el escenario 2, cuando a un cliente no se lo elige deberían restarsele de su puntaje 20 puntos, pero se le suman.

RegistrarUsuario:

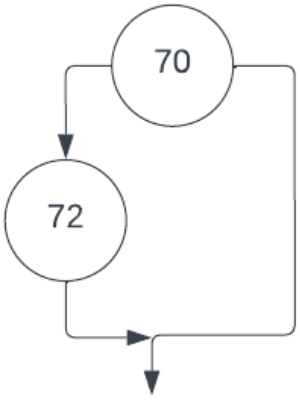
Permite crear un empleador existente, debería lanzar una excepción. También permite registrar un empleador con rubro invalido, los unicos rubros posibles con los que puede ser registrado son: salud, comercio local y comercio internacional (debería lanzar una excepción).

Tests de Caja Blanca:

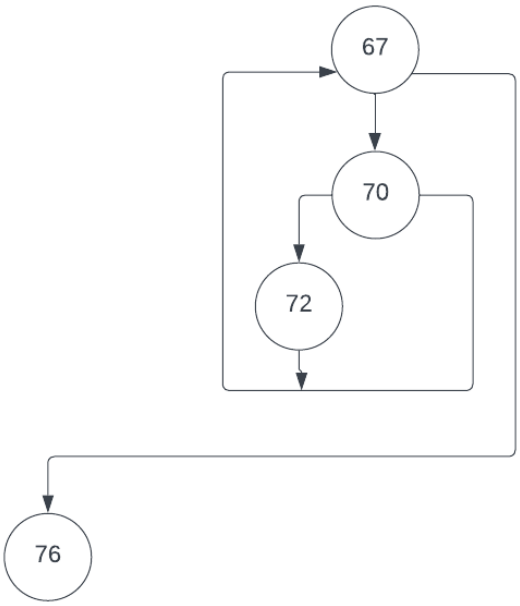
Obtención de grafo (Método ascendente):

Esta táctica inicia con las instrucciones a nivel individual, las cuales se van fusionando para crear las estructuras. Por lo general, se empieza con las instrucciones ubicadas dentro de las iteraciones o condiciones más profundas, avanzando progresivamente hacia el nivel exterior. Si se identifica un condicional, se reemplaza por la estructura correspondiente.

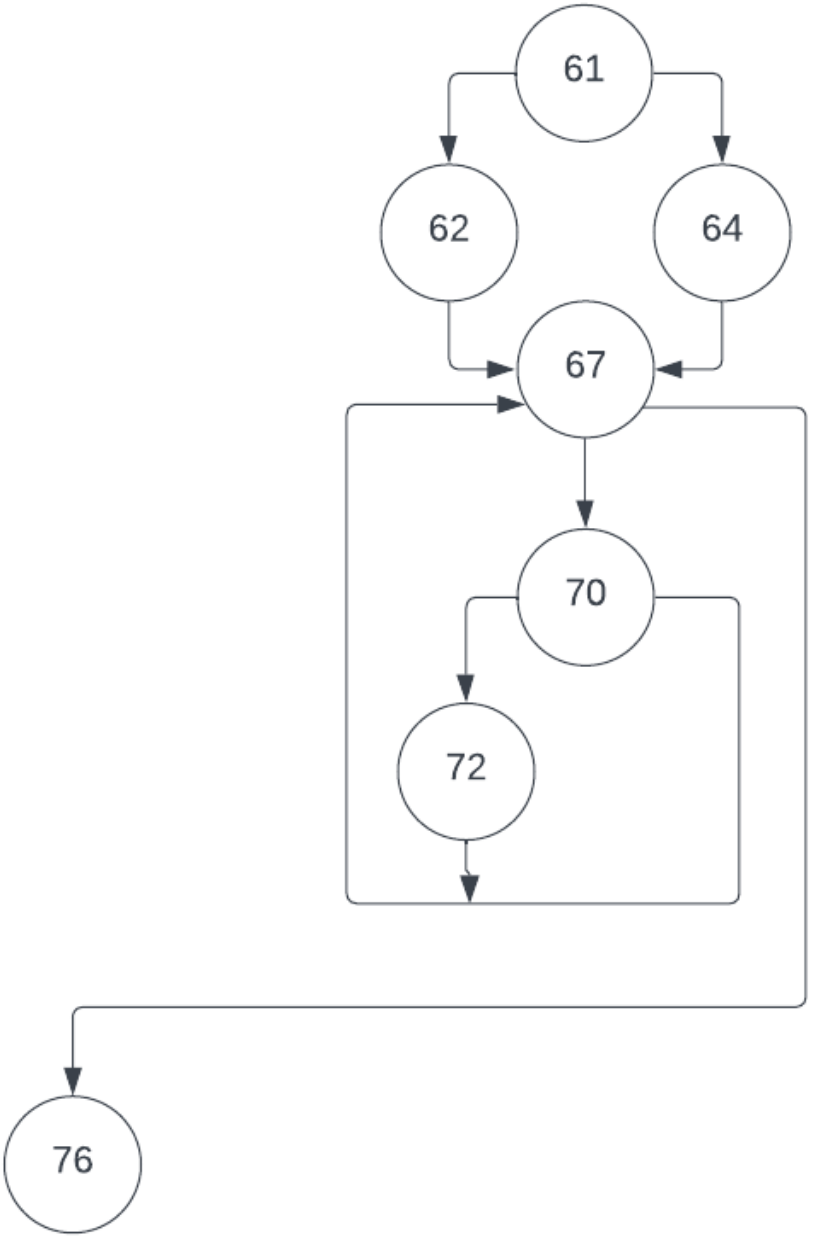
1.



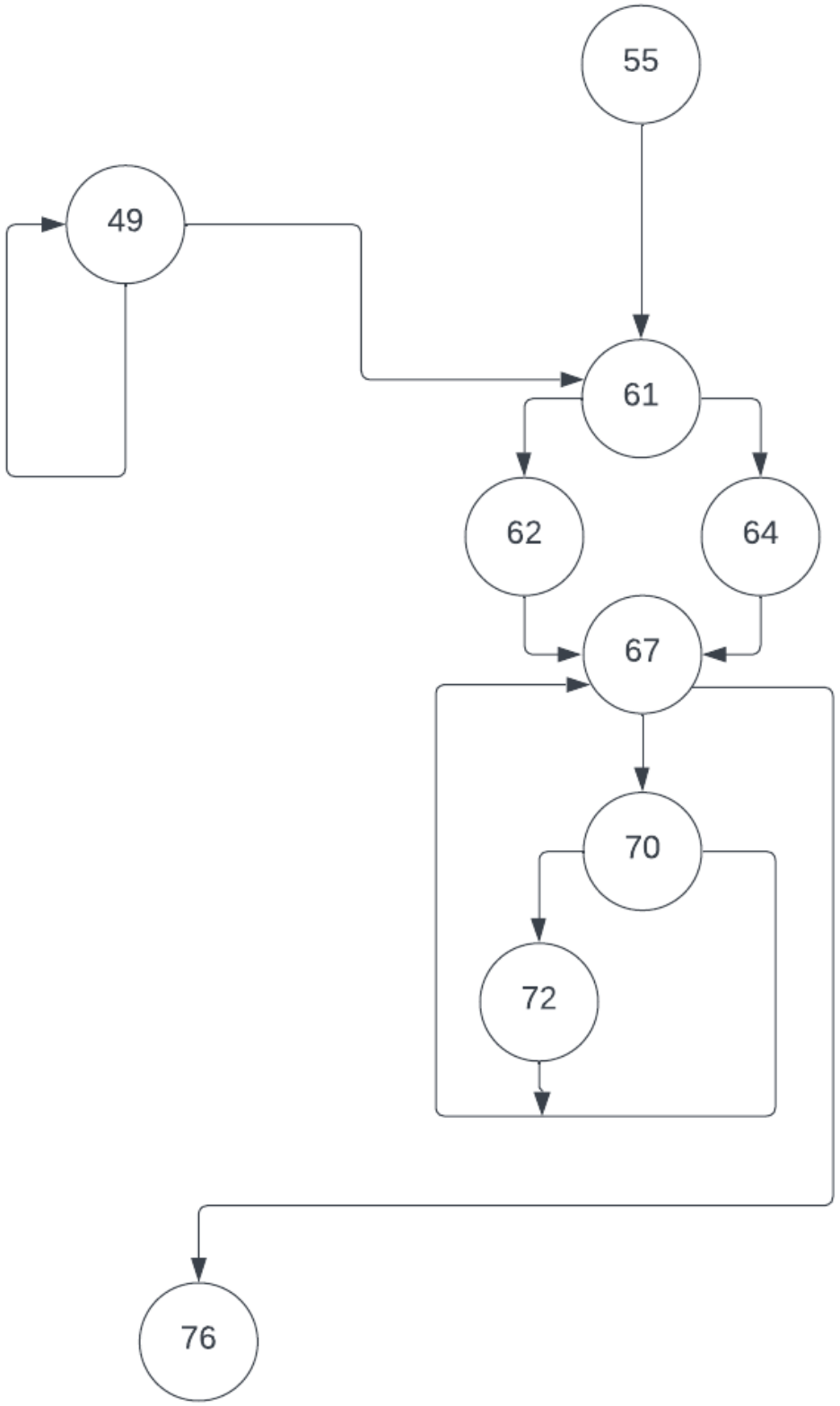
2.



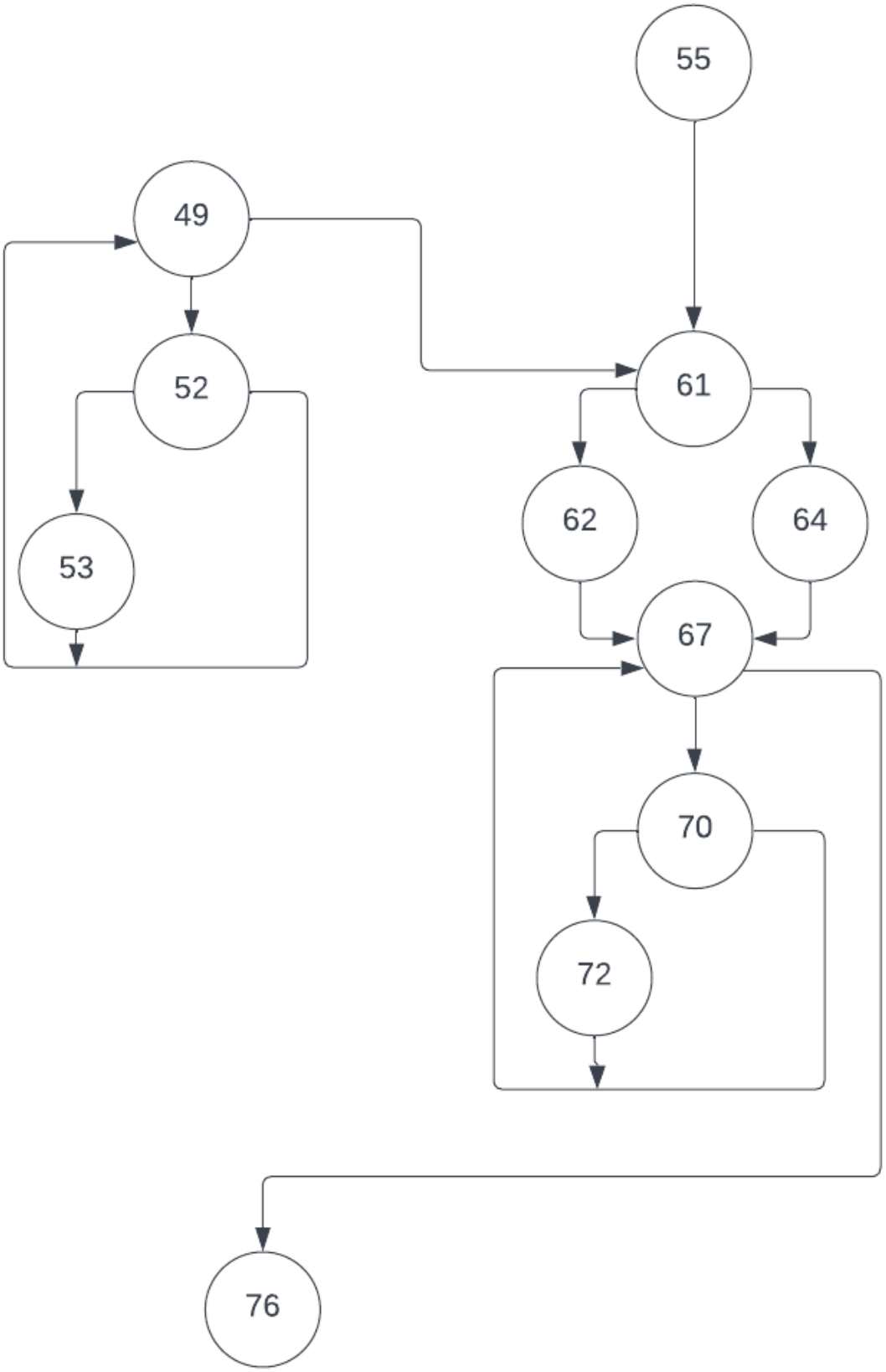
3.



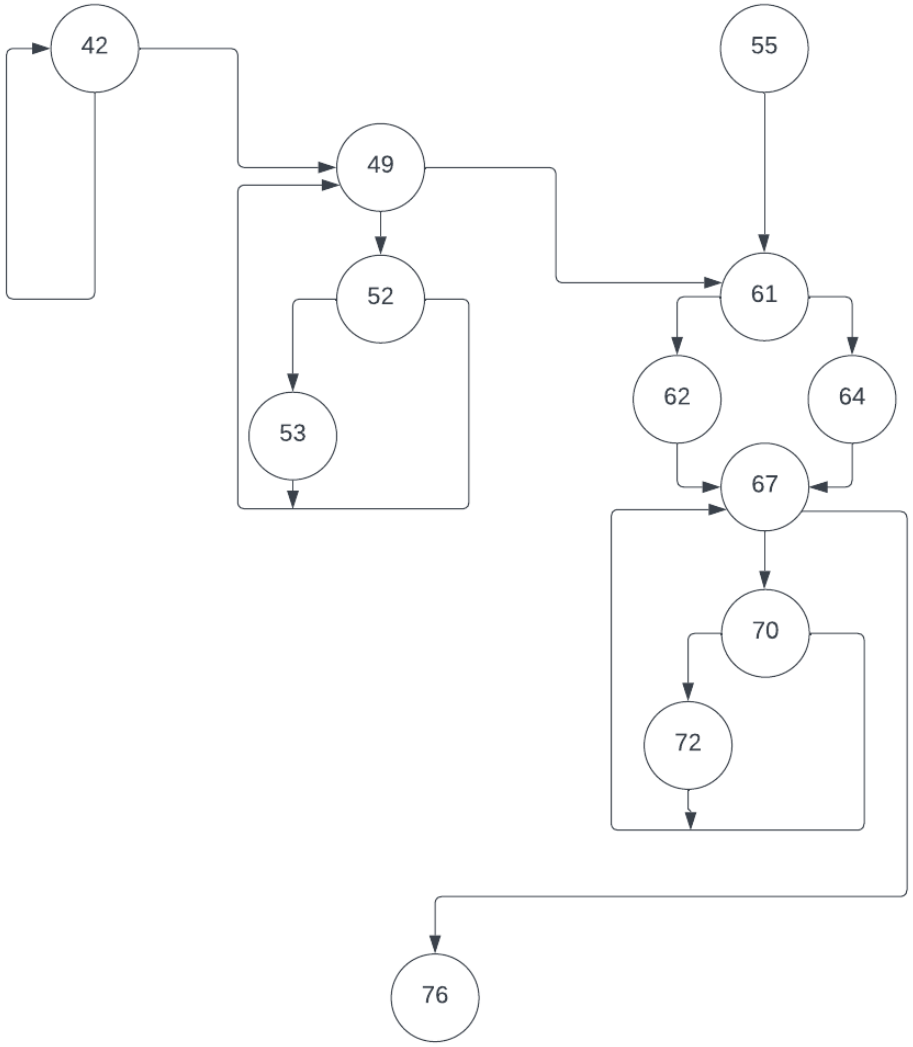
4.



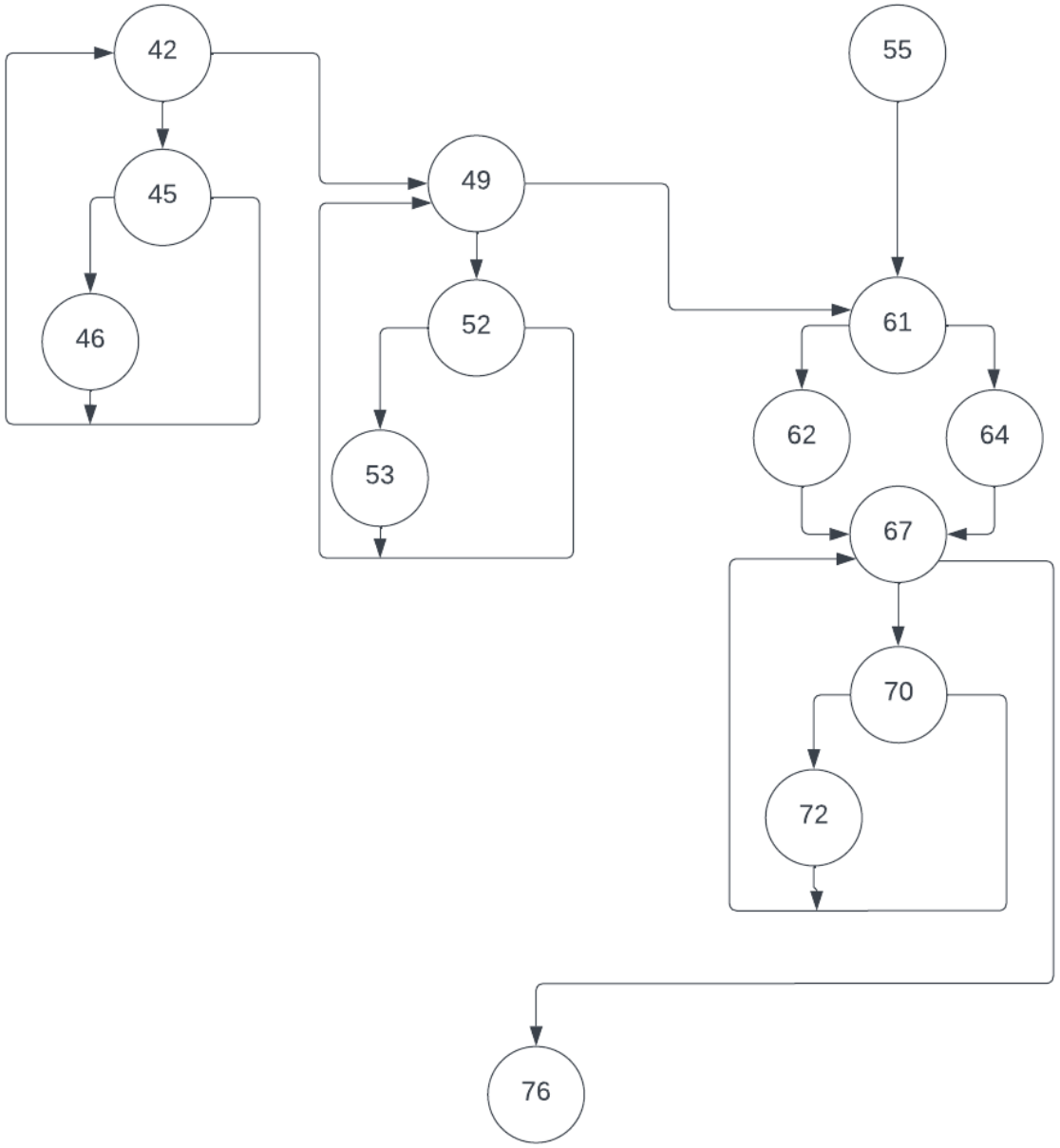
5.



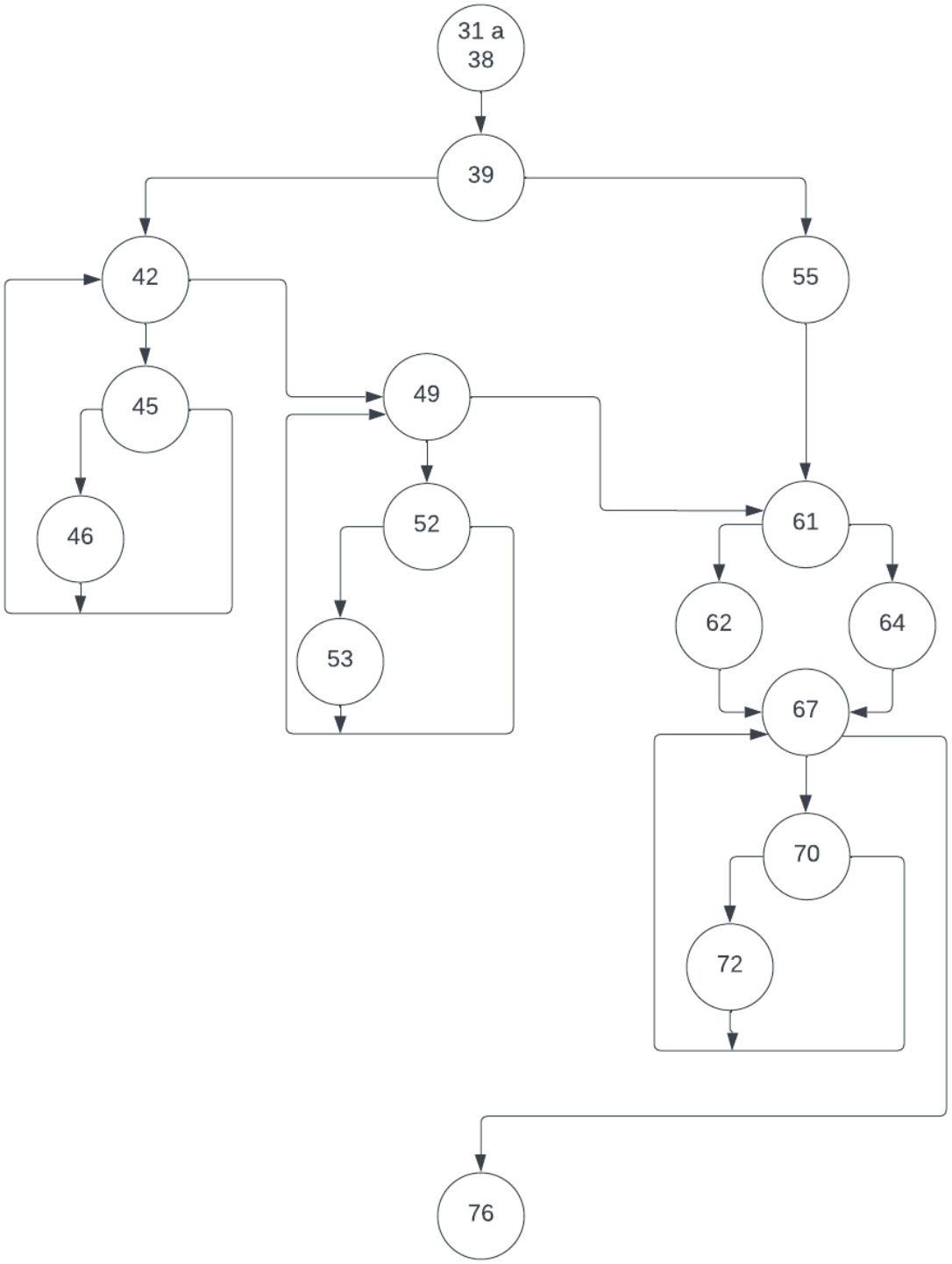
6.



7.



8.



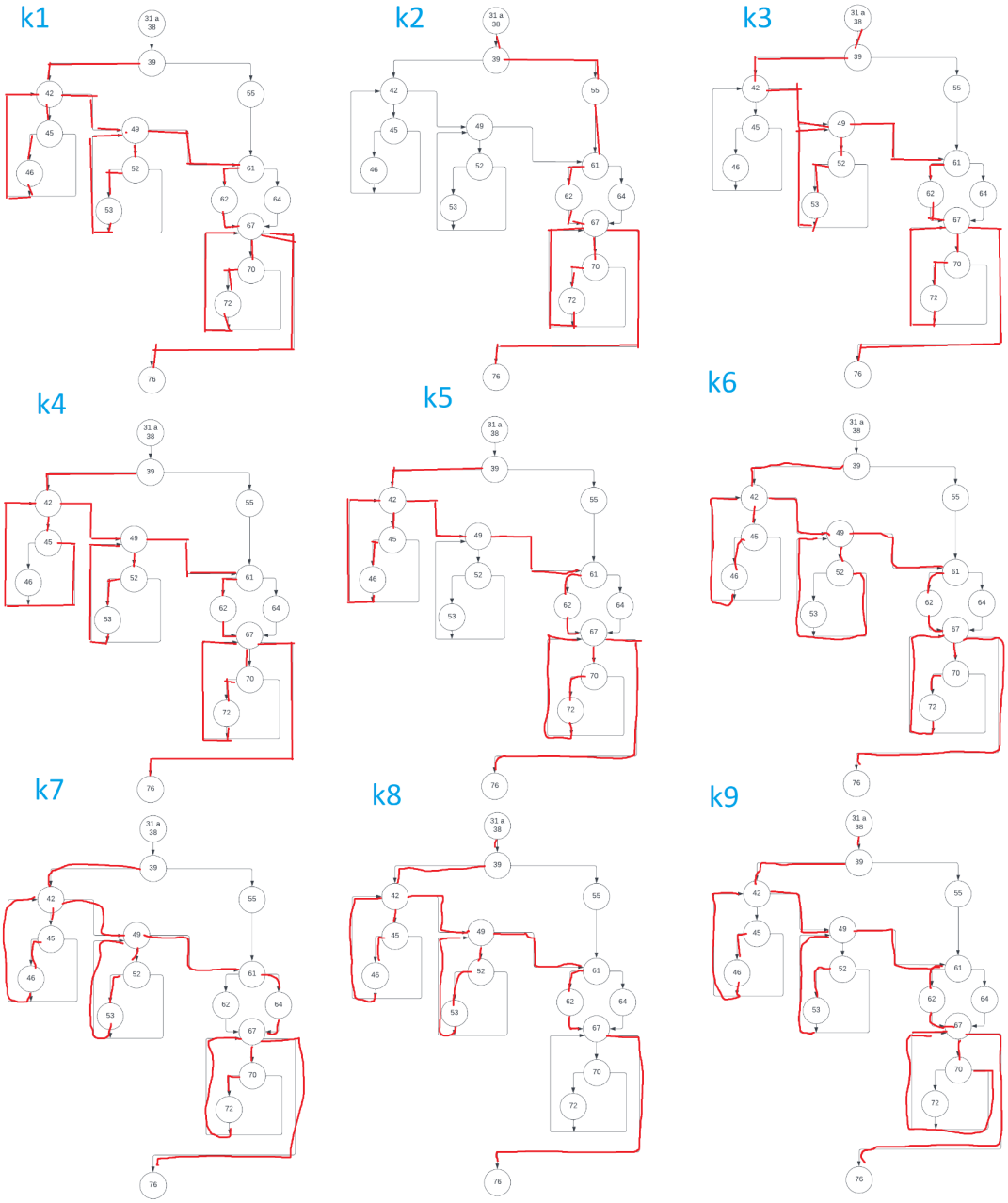
Obtención de caminos (Método general):

En este método se elige el primer camino que tenga sentido funcional, es decir, que represente la operación normal del software. Estos caminos omiten, en primera instancia, las salidas de error y las excepciones y recorre al menos una vez cada ciclo. A partir de ese camino inicial, se va variando una parte cada vez hasta completar los caminos necesarios.

Complejidad ciclomática: condicionales + 1 = 8 + 1 = 9

Por lo tanto habran a lo sumo 9 caminos:
(Ver ilustraciones de caminos en la pagina siguiente)

Camino	Recorrido
K1	39-42-45-46-42-49-52-53-49-61-62-67-70-72-67-76
K2	39-55-61-62-67-70-72-67-76
K3	39-42-49-52-53-61-62-67-70-72-67-76
K4	39-42-45-42-49-52-53-49-61-62-67-70-72-76
K5	39-42-45-46-42-49-61-62-67-70-72-67-76
K6	39-42-45-46-42-49-52-49-61-62-67-70-72-76
K7	39-42-45-46-42-49-52-53-49-61-64-67-70-72-67-76
K8	39-42-45-46-42-49-52-53-49-61-62-67-76
K9	39-42-45-46-42-49-52-53-49-61-62-67-70-67-76



El camino 3 no es recorrible, ya que si el hashmap empleadores no contiene elementos (nodo 42), nunca podra suceder que contadorEmpleador > contadorEmpleadoPretenso (nodo 61) . Por lo tanto para poder ir por el false del nodo 42, se modifica la condicion 61.

El camino 8 tampoco es recorrible, porque si contadorEmpleador > contadorEmpleadoPretenso (nodo 61), no puede suceder que clientes tiene elementos (nodo 67). Por lo tanto se modifica los valores de entrada para poder recorrerlo.

Batería de pruebas:

Camino	nodo condicion	Valores de entrada	Salida esperada
K1	39 == True	promoPorListaDePostulantes == true	Devuelve el cliente beneficiario de la promo
	42 == true	Hashmap empleadores tiene elementos	
	45 == true	empleador tiene lista de postulantes	
	49 == true	Hashmap empleados tiene elementos	
	52== true	empleado tiene lista de postulantes	
	61= true	contadorEmpleador > contadorEmpleadoPretenso	
	67 == true	clientes tiene elementos	
	70 == true	cl.getPuntaje() > puntajeMaximo	
K2	39 == false	promoPorListaDePostulantes == false	Devuelve el cliente beneficiario de la promo
	61= true	contadorEmpleador > contadorEmpleadoPretenso	
	67 == true	clientes tiene elementos	
	70 == true	cl.getPuntaje() > puntajeMaximo	
K3	39 == True	promoPorListaDePostulantes == true	Devuelve el cliente beneficiario de la promo
	42 == false	Hashmap empleadores no tiene elementos	
	49 == true	Hashmap empleados tiene elementos	
	52== true	empleado tiene lista de postulantes	
	61= false	contadorEmpleador <= contadorEmpleadoPretenso	

	67 == true	clientes tiene elementos	
	70 == true	cl.getPuntaje() > puntajeMaximo	
K4	39 == True	promoPorListaDePostulantes == true	Devuelve el cliente beneficiario de la promo
	42 == true	Hashmap empleadores tiene elementos	
	45 == false	empleador no tiene lista de postulantes	
	49 == true	Hashmap empleados tiene elementos	
	52== true	empleado tiene lista de postulantes	
	61= true	contadorEmpleador > contadorEmpleadoPretenso	
	67 == true	clientes tiene elementos	
	70 == true	cl.getPuntaje() > puntajeMaximo	
K5	39 == True	promoPorListaDePostulantes == true	Devuelve el cliente beneficiario de la promo
	42 == true	Hashmap empleadores tiene elementos	
	45 == true	empleador tiene lista de postulantes	
	49 == false	Hashmap empleados no tiene elementos	
	61= true	contadorEmpleador > contadorEmpleadoPretenso	
	67 == true	clientes tiene elementos	
	70 == true	cl.getPuntaje() > puntajeMaximo	
K6	39 == True	promoPorListaDePostulantes == true	Devuelve el cliente beneficiario de la promo
	42 == true	Hashmap empleadores tiene elementos	
	45 == true	empleador tiene lista de postulantes	
	49 == true	Hashmap empleados tiene elementos	
	52== false	empleado no tiene lista de postulantes	
	61= true	contadorEmpleador > contadorEmpleadoPretenso	

	67 == true	clientes tiene elementos	
	70 == true	cl.getPuntaje() > puntajeMaximo	
K7	39 == True	promoPorListaDePostulantes == true	Devuelve el cliente beneficiario de la promo
	42 == true	Hashmap empleadores tiene elementos	
	45 == true	empleador tiene lista de postulantes	
	49 == true	Hashmap empleados tiene elementos	
	52== true	empleado tiene lista de postulantes	
	61= false	contadorEmpleador <= contadorEmpleadoPretenso	
	67 == true	clientes tiene elementos	
	70 == true	cl.getPuntaje() > puntajeMaximo	
K8	39 == True	promoPorListaDePostulantes == true	Devuelve NULL
	42 == false	Hashmap empleadores no tiene elementos	
	45 == false	empleador no tiene lista de postulantes	
	61= false	contadorEmpleador<= contadorEmpleadoPretenso	
	67 == false	clientes no tiene elementos	
K9	39 == True	promoPorListaDePostulantes == true	Devuelve el cliente beneficiario de la promo
	42 == true	Hashmap empleadores tiene elementos	
	45 == true	empleador tiene lista de postulantes	
	49 == true	Hashmap empleados tiene elementos	
	52== true	empleado tiene lista de postulantes	
	61= true	contadorEmpleador > contadorEmpleadoPretenso	
	67 == true	clientes tiene elementos	
	70 == false	cl.getPuntaje() <= puntajeMaximo	

Errores detectados:

Ninguno

Test de GUI:

Errores detectados:

1. En la ventana de registro de usuario_debería mostrar cuando queremos registrar un cliente ya sea empleador o empleado y completar los campos restantes al registrar. Si el campo confirmar contraseña no coincide con el campo contraseña debería lanzar un cartel del tipo “Mensajes.PASS_NO_COINCIDE” pero no lo hace.

2. Al registrar un Empleador, es posible registrar varios con el mismo nombre de usuario, sobreescribiendo el anterior

3. Cuando la Agencia está en estado de contratación no debería permitir crear un ticket a un cliente, sin embargo luego de que hicimos el testeo se crea el ticket.

4. A la hora de registrar un tipo Empleador si el usuario ya existe debería devolver el mensaje “Mensajes.USUARIO_REPETIDO” pero esto no ocurre.

Conclusiones:

Hemos abordado de manera exhaustiva el proceso de prueba del sistema. Se han aplicado pruebas de caja negra, caja blanca y pruebas de interfaz gráfica para evaluar la funcionalidad del programa en diferentes niveles.

En las pruebas de caja negra, se identificaron varios errores en el código, como problemas en el cálculo del puntaje del empleado pretenso, asignaciones incorrectas en la carga de la agencia, y errores en la penalización de empleadores no elegidos. Además, se detectaron fallos en la interfaz gráfica, como la creación indebida de tickets en estado de contratación y la falta de mensajes de error al registrar usuarios.

En las pruebas de caja blanca, se realizaron análisis detallados del flujo del programa, identificando caminos recorribles y no recorribles. Se observaron dificultades en la obtención de caminos debido a condiciones que no pueden cumplirse simultáneamente, lo que sugiere posibles problemas lógicos en la implementación.

El trabajo llevó tiempo y fue crucial la coordinación de los integrantes del grupo, lo que nos proporcionó mucha experiencia en trabajos grupales. La dificultad fue moderada, ya que la documentación del sistema no era muy clara pero tampoco era muy compleja, por lo que luego de varias lecturas pudimos entender como funcionaba el software. Las herramientas vistas en la teoría de la materia fueron suficientes para poder realizar todo el trabajo práctico.