Sistema de gestión multiempresarial - Entrega #2

Integrantes:

- ➔ Juan Fernando Martinez Hidalgo - A00358232
- ➔ Cristhian Camilo Gutierrez Cordoba - A00355902

5. Requerimientos funcionales actualizados.

El programa debe estar en la capacidad de:

I. **Crear** un único holding con nombre, fecha de creación y logo. Solo esta disponible al iniciar el programa por primera vez.

II. **Agregar** una empresa nueva al holding con nombre, NIT y logotipo. Las empresas pertenecientes al holding pueden tener nombres similares pero jamás un mismo número NIT (Número de Identificación Tributaria), sí se intenta registrar un NIT repetido debe generar un aviso de invalidez en la operación, además toda empresa debe tener por lo menos un representante legal. También deberá tener un tipo definido ya que toda empresa se caracteriza por su especialidad.

III. **Vender** una empresa dado su NIT. Se debe brindar la operación de venta de una empresa en específico dado su NIT y respectivamente el costo de la venta. Se debe remover los empleados de la base de datos y ajustar el valor de la venta al balance monetario general del holding empresarial.

IV. **Eliminar** una empresa existente del holding dado su NIT. Se debe realizar una búsqueda en todas las estructuras de almacenamiento de empresas del holding y eliminar la organización en cuestión con el mismo NIT, de no encontrarse dicha empresa debe generar un mensaje donde se explique lo sucedido. Al momento de remover la empresa se debe conservar el buen estado de la estructura de datos donde se encontraba almacenada.

V. **Agregar** un empleado con nombre, ID, dirección de residencia, numero de contacto, puesto, salario acordado, contrato laboral y horario de trabajo. Es posible agregar empleados a cualquier empresa (incluyendo el grupo empresarial), se debe preservar la unicidad del ID, no pueden existir más de dos empleados laborando en el holding con el mismo ID (Nota: Es posible que un mismo empleado pueda laborar en más de una empresa pero solo es posible en horarios diferentes, se debe mostrar una notificación de error si hay cruces con los horarios laborales si se presenta el caso).

VI. **Eliminar** un empleado de una empresa dado su número de identificación. Solo se permite la eliminación si el contrato está dado por terminado o si se incumplío una de las clausulas del mismo. Se debe generar un monto monetario de liquidación si el contrato así lo estipula.

VII. **Eliminar** un empleado según una lista de todos los empleados pertenecientes a una empresa. Se debe generar visualmente una lista con los nombres, números de identificación y cargo de todos los empleados de una empresa en específico, se debe mostrar una notificación de confirmación del procedimiento de remoción o un mensaje de error si el proceso no se hizo correctamente.

VIII. **Agregar** una sede con ubicación, tipo de sede y número de identificación del empleado a cargo. No es posible generar más de una sede con la misma dirección en todo el holding empresarial. Sí se intenta generar una sede con los parámetros incorrectos debe mostrarse una advertencia especificando el error y los pasos a seguir para solucionar dicho problema.

IX. **Crear** un contrato especificando su tipo(no puede haber contrato sin tipo), nombre, fecha de inicio, terminación, descripción de los acuerdos, cláusulas, monto pactado, nombre y número de identificación de los que firman y su respectiva firma (imagen). No puede existir más de un contrato con el mismo número de identificación. En cuanto a los contratos se debe incluir la creación, asignación, almacenamiento y eliminación de los mismos además de la información de las partes involucradas(Solamente puede ser empresa y empleado). Un contrato consta de un nombre descriptivo, fecha de radicación, fecha de inicio y terminación, descripción de los acuerdos, cláusulas, monto pactado, firmantes, número de renovaciones hasta el momento y número de renovaciones máximas.

X. **Dibujar** n firmas en asociación con el contrato. Estas firmas deben ir asociadas a su respectivo firmante (empleado). De no ser dibujadas todas las firmas no se procederá a la validación y ejecución del contrato y se deberá mostrar su respectivo mensaje de error y los pasos a seguir para realizar este proceso adecuadamente.

XI. **Eliminar** un contrato dado su código único. Se debe remover el contrato de la estructura de datos donde se encuentra contenido sin alterar la arquitectura de la misma. Si el contrato es con una persona natural o jurídica que posee un solo contrato con cualquier empresa del holding se procederá a eliminar esta persona de la base de datos de empleados, si tiene más de un contrato solo es necesario recalcular su salario y ajustarlo a la base de datos.

XII. **Generar** un reporte de cualquier tipo. Para generar un reporte se espera que el usuario seleccione el tipo de reporte y a continuación el NIT de la empresa, ID de empleado ó número de identificación del contrato como parámetros únicos de búsqueda para la categoría que aplique. Se debe generar un el reporte y desplegarlo en pantalla si el usuario así lo desea, también se debe brindar la posibilidad de exportarlo en formato txt o csv si es el caso.

XIII. **Mostrar** en todo momento la fecha y hora del sistema la cual deberá actualizarse de manera automática.

XIV. **Mostrar** en pantalla el tiempo que tarda en realizar la última operación

XV. **Buscar** y **mostrar** en pantalla información básica de empleados, sedes y contratos dando su código único de identificación respectivo.

6. Requerimientos no funcionales

1. El programa debe de manejar la información de forma persistente. Todas las clases relacionadas a cada compañia serán almacenadas mediante serialización. La información básica del holding así como la información básica de cada empresa será almacenada mediante archivos de texto.

2. Manejo de excepciones:

    a. Las excepciones relacionadas a la escritura de números como el número de teléfono serán manejadas con NumberFormatException propio de Java.
    b. Las excepciones relacionadas con la carga de archivos se trataran con IOException de Java.
    c. Excepcion java
    d. Escepcion java
    e. Las excepciones relacionadas con el formato del número de identificación serán manejadas con IDFormatException.

f. Las excepciones relacionados a los números de identificación cuando se quiere agregar un empleado será manejado con una excepción propia llamada UsedIDException.
g. Las excepciones relacionadas a la coherencia de fechas de inicio, terminación, hora de apertura y hora de cierre. Serán manejada con una excepción propia llamada DateException y otra llamada TimeException.

3. Holding:

    a. Únicamente habrá un holding disponible para crear, solo habrá un objeto de la clase holding.

4. Empresas:
    a. Las empresas serán almacenadas empleando un árbol binario de búsqueda que usará como criterio de ordenamiento el NIT de cada empresa. Se usará el método compareTo() de la clase String.
    b. Su creación requiere la especificación de un tipo definido de empresa, que se representa con la creación de clases hijas para cada tipo. Si no encaja dentro de alguno de los tipos se puede crear una empresa genérica.
    c. No podrán eliminarse empresas a menos que todos sus empleados hayan sido eliminados.
    d. La búsqueda, adición y eliminación de empresas se realizará con algoritmos recursivos propios de los árboles binarios de búsqueda.
    e. Los tipos de empresa implementan cada una una interfaz que les permite decir como se disminuyen los impuestos por cada tipo determinado

5. Empleados:
    a. Los empleados serán almacenado en un árbol binario de búsqueda que usará como criterio el ID de cada empleado. Se usará el método compareTo() de la clase String.
    b. La búsqueda, adición y eliminación de empleados se realizará con algoritmos recursivos propios de los árboles binarios de búsqueda.
    c. Existen un tipo especial de empleado que es el representante legal de la empresa, hereda de empleado y es único para cada empresa.
6. Contratos:
    a. Serán almacenados utilizando una lista doblemente enlazada.
    b. Podrán ser ordenados empleando el algoritmo burbuja para ordenamiento, usará como criterios la fecha de inicio, la fecha de terminación y el código de contrato.
    c. Cada contrato debe tener necesariamente un tipo, no puede existir un contrato sin tipo por lo que la clase Contrato es abstracta. Los tipos de contratos heredan de la clase Contrato
    d. Se busqueda se realizará usando búsqueda binaria sobre la lista previamente ordenada usando como criterio el id del contrato.

7. Sedes:

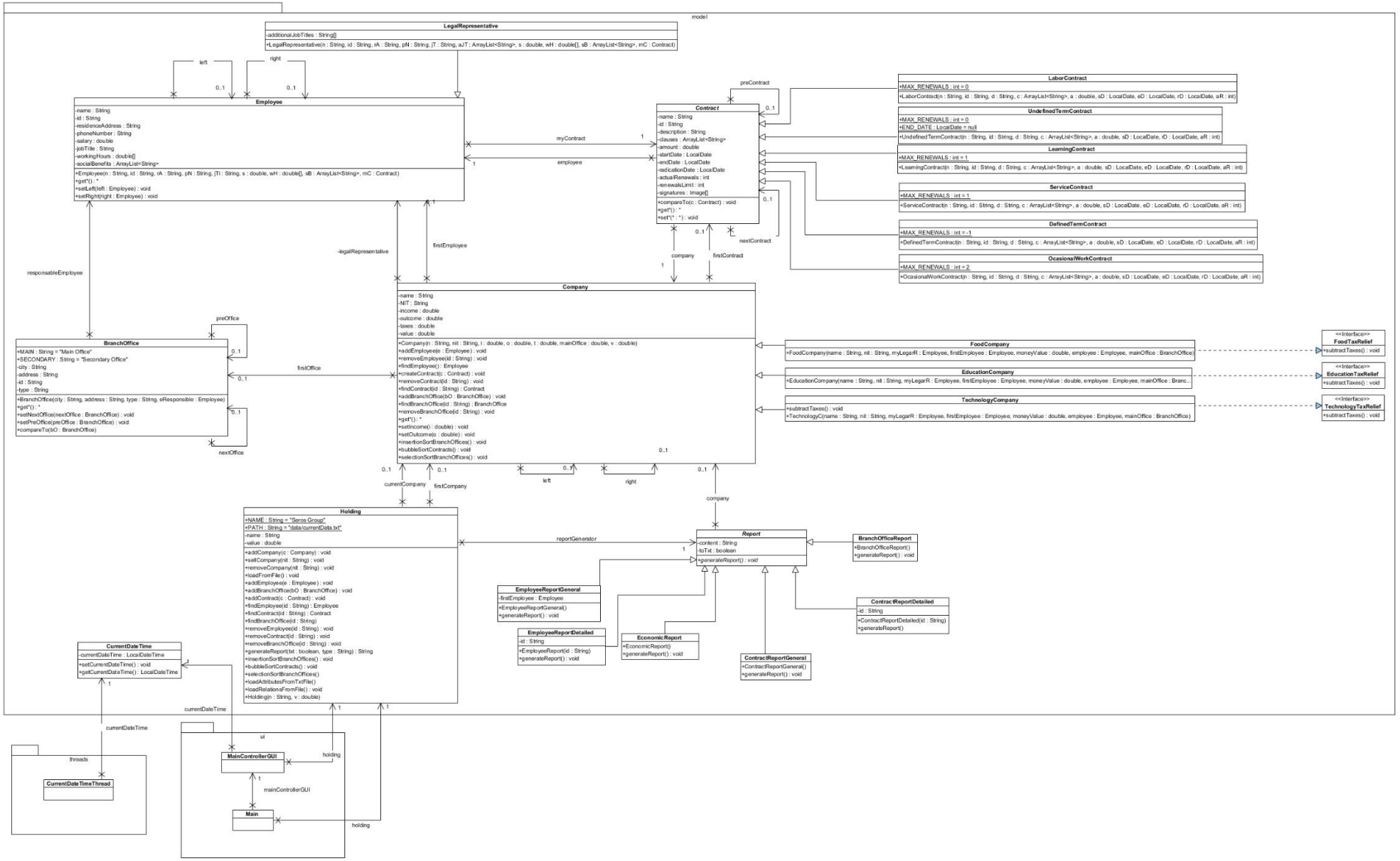
    a. Serán almacenadas utilizando una lista doblemente enlazada.

   b. Podrán ser ordenadas usando el algoritmo de inserción. Que usará como criterio el código único de las sedes y su ubicación. También podrán ser ordenadas usando solamente como criterio el código.
   c. La búsqueda de información se realizará por medio de una búsqueda binaria que usará como criterio el código único de las sedes.
   d. Tendrán la posibilidad de ser ordenadas por ordenamiento por inserción o selección usando su ciudad e id.
8. Los reportes tendrán un clase abstracta propia llamada Report de la cual heredan todos los tipos de reporte. Cuenta con atributo booleano que permite decir si guarda en formato txt o retorna el reporte como una String para mostrar en pantalla.
9. Implementación de cinco hilos cuyas responsabilidades se especifican a continuación:

   a. Un hilo será el encargado de desplegar la hora actual y tiempo de ejecución sin interrupciones.
   b. Un hilo será el encargado de los respectivos ordenamientos de las estructuras de datos (Empresas, clientes, contratos, etc).
   c. Un hilo será el encargado de realizar la búsqueda de cualquier elemento específico entre todas las estructuras de datos.
   d. Un hilo será el encargado de la gestión de los paneles de dibujo digitales.
   e. Hilo principal.

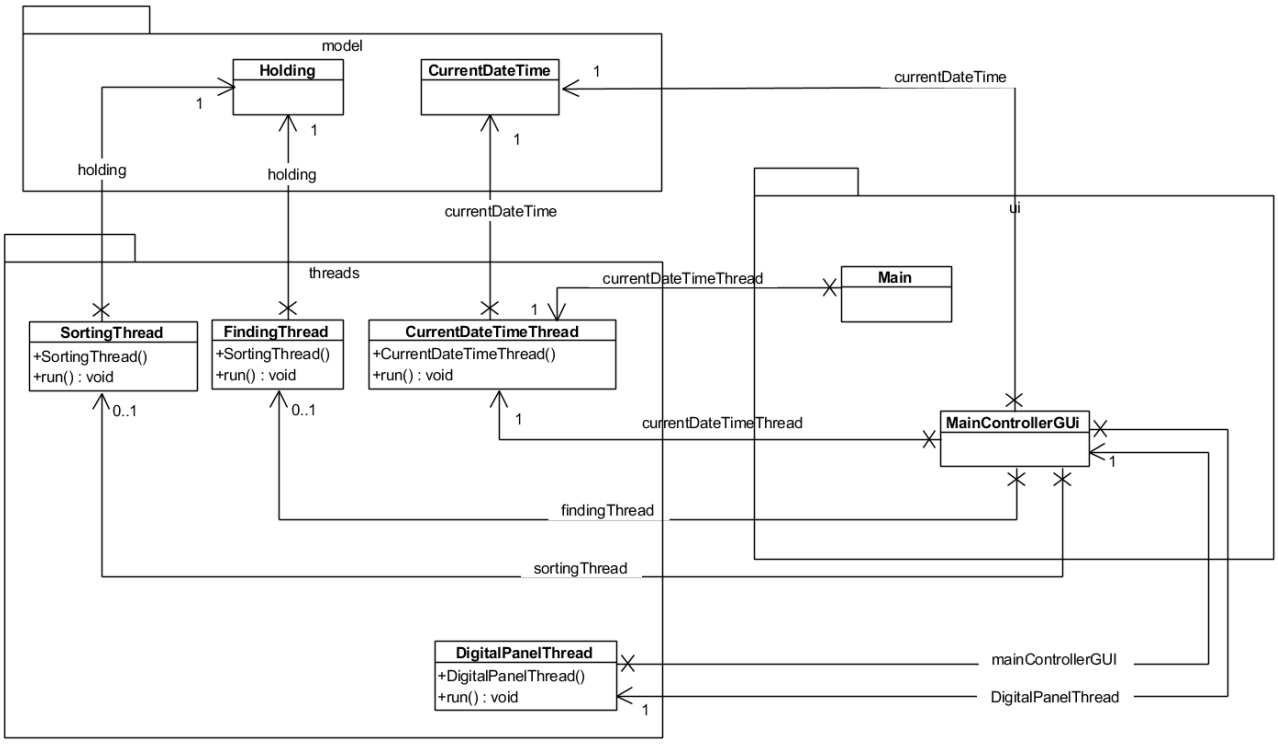
10. Por cada operación (búsqueda, ordenamiento, eliminación y carga de datos.) se debe mostrar en pantalla el tiempo en segundos que el software requirió para concluir un proceso.

7. Diagrama de clases – Model

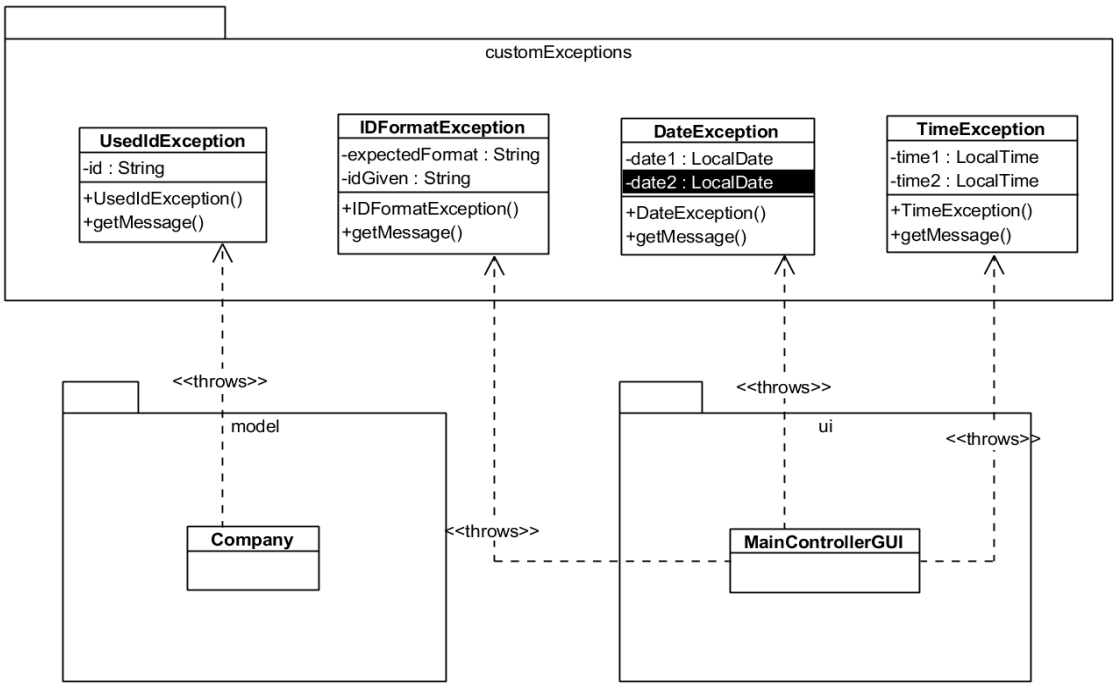


8.Diagrama de clases - Threads

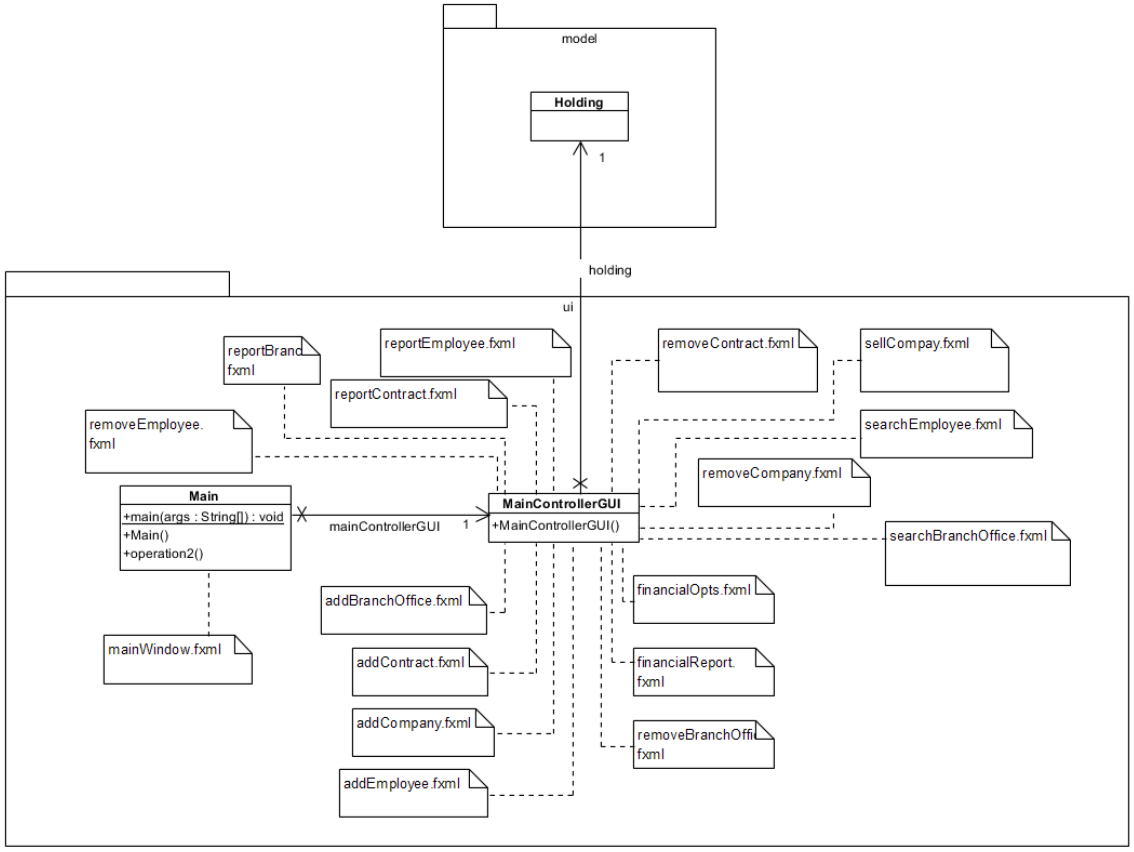
Solo se muestran las relaciones relevantes para el paquete threads, las relaciones entre clases de otros paquetes no necesariamente se muestran.



9. Diagrama de Clases - customExceptions



10.Diagrama de clases - ui



11. Diseño de pruebas unitarias automáticas:

Configuración de escenarios:

Nombre	Clase	Escenario
setupEmployees1	CompanyTest	Compañía con ningún empleado.
setupEmployees2	CompanyTest	Compañía con un empleado con atributos: Name = “Juan” Id = “1234”
setupEmployees3	CompanyTest	Compañía con dos empleados con atributos: Name = “Juan” Id = “1234” Name = “Federico” Id = “2598”
setupEmployees4	CompanyTest	Compañía con 4 empleados con atributos Name = “Juan” Id = “1234” Name = “Federico” Id = “2598”

		Name = “Camila” Id = “0000” Name = “Ana” Id = “1111”
setupContracts1	CompanyTest	Compañía con ningún contrato.
setupContracts2	CompanyTest	Compañía con un contrato con atributos: Name = “c1” Id = “1234”
setupContracts3	CompanyTest	Compañía con dos contratos con atributos: Name = “c1” Id = “C1234” Name = “c2” Id = “C2598”
setupContracts4	CompanyTest	Compañía con 4 contratos con atributos Name = “c1” Id = “C1234” Name = “c2” Id = “C2598” Name = “c3” Id = “C0000” Name = “c4” Id = “C1111”
setupContracts5	CompanyTest	Compañía con dos contratos con atributos: Name = “c2” Id = “C2598” Name = “c1” Id = “C1234”
setupBranchOffices1	CompanyTest	Compañía con ninguna oficina.
setupBranchOffices2	CompanyTest	Compañía con una oficina con id = “O1234”
setupBranchOffices3	CompanyTest	Compañía con dos oficinas con id = “O1234” y id = “O2598”
setupBranchOffices4	CompanyTest	Compañía con 4 oficinas con atributos Id = “O1234” Id = “O2598”

		Id = “O0000” Id = “O1111”
setupBranchOffices5	CompanyTest	Compañía con dos oficinas con id = “O2598” y id = “O1234”
setupCompanies1	HoldingTest	Holding con ninguna compañía.
setupCompanies2	HoldingTest	Holding con una compañía con atributos Name = “Norma” NIT = “N1234”
setupCompanies3	HoldingTest	Holding con dos compañías con atributos Name = “Norma” Id = “N1234” Name = “Carvajal” Id = “N2598”
setupCompanies4	HoldingTest	Holding con 4 compañías con atributos Name = “Norma” Id = “N1234” Name = “Carvajal” Id = “N2598” Name = “Bavaria” Id = “N0000” Name = “Reprograf” Id = “N1111”

Casos de prueba:

Objetivo de la prueba: Determinar si el método addEmployee agrega correctamente a un empleado nuevo a un árbol binario de búsqueda de empleados.

Clase	Método	Escenario	Valores de Entrada	Resultado
Company	**addEmployee**	setupEmployees1	Objeto empleado con atributos Name = “Camilo Id = “2222”	**El empleado se agrega como primer empleado**
Company	**addEmployee**	setupEmployees2	Objeto empleado con atributos Name = “Camilo Id = “2222”	**El empleado se agrega a la derecha del primer empleado.**
Company	**addEmployee**	setupEmployees3	Objeto empleado con atributos Name = “Camilo	**El empleado se agrega a la**

			Id = “2222”	**izquierda del empleado “Federico”.**
Company	**addEmployee**	setupEmployees4	Objeto empleado con atributos Name = “Camilo” Id = “2222”	**El empleado se agrega a la izquierda del empleado “Federico”**
Company	**addEmployee**	setupEmployees4	Objeto empleado con atributos Name = “Camilo” Id = “1125”	**El empleado se agrega a la derecha del empleado “Ana”**

Objetivo de la prueba: Determinar si el método findEmployee encuentra correctamente un empleado dado su id, y retorna null si no lo encuentra

Clase	Método	Escenario	Valores de Entrada	Resultado
Company	**findEmployee**	setupEmployees1	Id = “2222”	**Retorna nulo**
Company	**findEmployee**	setupEmployees2	Id = “2222”	**Retorna nulo**
Company	**findEmployee**	setupEmployees2	Id = “1234”	**Retorna una referencia al empleado “Juan”con id = “1234”**
Company	**findEmployee**	setupEmployees3	Id = “2222”	**Retorna nulo**
Company	**findEmployee**	setupEmployees3	Id = “2598”	**Retorna una referencia al empleado “Federico” con id = “2598”**
Company	**findEmployee**	setupEmployees4	Id = “2222”	**Retorna nulo**
Company	**findEmployee**	setupEmployees4	Id = “0000”	**Retorna una referencia al empleado “Camila” con id = “0000”**

Objetivo de la prueba: Determinar si el método removeEmployee remueve correctamente a un empleado de un árbol binario de búsqueda de empleados.

Clase	Método	Escenario	Valores de Entrada	Resultado
Company	**removeEmployee**	setupEmployees1	Id = “2222”	**El primer empleado sigue siendo null.**
Company	**removeEmployee**	setupEmployees2	Id = “2222”	**El primer empleado sigue siendo “Juan” con id = “1234”**
Company	**removeEmployee**	setupEmployees2	Id = “1234”	**El primer empleado ahora es null.**
Company	**removeEmployee**	setupEmployees3	Id = “2222”	**El primer empleado sigue siendo “Juan” con id = “1234”**
Company	**removeEmployee**	setupEmployees3	Id = “1234”	**El primer empleado ahora es “Federico” con id = “2598”**
Company	**removeEmployee**	setupEmployees4	Id = “2222”	**El primer empleado sigue siendo “Juan” con id = “1234”**
Company	**removeEmployee**	setupEmployees4	Id = “1234”	**El primer empleado ahora es “Federico” con id = “2598”, a su izquierda “Camila” con id = “0000” y a su derecha Federico con id = “2598”**
Company	**removeEmployee**	setupEmployees4	Id = “1111”	**El empleado a la derecha de “Camila” con id “0000” ahora es null.**
Company	**removeEmployee**	setupEmployees4	Id = “0000”	**El empleado a la izquierda del primero ahora es “Ana” con id = “1111”**

Objetivo de la prueba: Determinar si el método addContract añade correctamente un contrato a una lista doblemente enlazada de contratos

Clase	Método	Escenario	Valores de Entrada	Resultado
Company	**addContract**	setupContracts1	Objeto contrato con atributos Name = “cX Id = “C2222”	**El contrato se añade como primer contrato**
Company	**addContract**	setupContracts2	Objeto contrato con atributos Name = “cX” Id = “C2222”	**El contrato se añade después**

				del primer contrato.
Company	**addContract**	setupContracts3	Objeto contrato con atributos Name = “cX” Id = “C2222”	**El contrato se añade después del segundo contrato.**
Company	**addContract**	setupContracts4	Objeto contrato con atributos Name = “cX” Id = “C2222”	**El contrato se añade de último en la lista.**

Objetivo de la prueba: Determinar si el método findContract encuentra correctamente un contrato dado su id, y retorna null si no lo encuentra.

Clase	Método	Escenario	Valores de Entrada	Resultado
Company	**findContract**	setupContracts1	Id = “C2222”	**Retorna nulo**
Company	**findContract**	setupContracts2	Id = “C2222”	**Retorna nulo**
Company	**findContract**	setupContracts2	Id = “C1234”	**Retorna una referencia al contrato “c1” con id “C1234”**
Company	**findContract**	setupContracts3	Id = “C2222”	**Retorna nulo**
Company	**findContract**	setupContracts3	Id = “C2598”	**Retorna una referencia al contrato “c2” con id “C2598”**
Company	**findContract**	setupContracts4	Id = “C2222”	**Retorna nulo**
Company	**findContract**	setupContracts4	Id = “C0000”	**Retorna una referencia al contrato “c3” con id “C0000”**

Objetivo de la prueba: Determinar si el método removeContract remueve correctamente a un contrato de una lista doblemente enlazada de contratos y el empleado asociado es desvinculado.

Clase	Método	Escenario	Valores de Entrada	Resultado
Company	**removeContract**	setupContracts1	Id = “2222”	**El primer contrato sigue siendo null, no se ha desvinculado a ningún empleado.**
Company	**removeContract**	setupContracts2	Id = “2222”	**El primer contrato sigue siendo “c1” con id = “C1234”, no se ha desvinculado a ningún empleado.**
Company	**removeContract**	setupContracts2	Id = “1234”	**El primer contrato ahora es null y el empleado ligado al contrato borrado no está vinculado a ningún contrato.**
Company	**removeContract**	setupContracts3	Id = “2222”	**El primer contrato sigue siendo “c1” con id = “C1234”, no se ha desvinculado a ningún empleado.**
Company	**removeContract**	setupContracts3	Id = “1234”	**El primer contrato ahora es “c2” con id = “C2598” y el empleado ligado al contrato borrado no está vinculado a ningún contrato.**
Company	**removeContract**	setupContracts4	Id = “2222”	**El primer contrato sigue siendo “c1” con id = “C1234”, no se ha desvinculado a ningún empleado.**
Company	**removeContract**	setupContracts4	Id = “1111”	**El primer contrato sigue siendo “c1” con id = “C1234” y el tercer contrato tiene por siguiente contrato null. Ell empleado ligado al contrato borrado no está vinculado a ningún contrato.**

Objetivo de la prueba: Determinar si el método addBranchOffice añade correctamente una oficina a una lista doblemente enlazada de oficinas.

Clase	Método	Escenario	Valores de Entrada	Resultado
Company	**addBranchOffice**	setupBranchOffices1	Objeto oficina con atributos Name = “oX Id = “O2222”	**La oficina se añade como primera oficina.**

Company	**addBranchOffice**	setupBranchOffices2	Objeto contrato con atributos Name = “oX Id = “O2222”	**La oficina se añade después de la primera oficina.**
Company	**addBranchOffice**	setupBranchOffices3	Objeto contrato con atributos Name = “oX Id = “O2222”	**La oficina se añade después de la segunda.**
Company	**addBranchOffice**	setupBranchOffices4	Objeto contrato con atributos Name = “oX Id = “O2222”	**La oficina se añade de última en la lista.**

Objetivo de la prueba: Determinar si el método findBranchOffice encuentra correctamente una oficina dado su id, y retorna null si no lo encuentra.

Clase	Método	Escenario	Valores de Entrada	Resultado
Company	**findBranchOffice**	setupBranchOffices1	Id = “O2222”	**Retorna null**
Company	**findBranchOffice**	setupBranchOffices2	Id = “O2222”	**Retorna null**
Company	**findBranchOffice**	setupBranchOffices2	Id = “O1234”	**Retorna una referencia a la oficina “o1” con id “O1234”**
Company	**findBranchOffice**	setupBranchOffices3	Id = “O2222”	**Retorna null**
Company	**findBranchOffice**	setupBranchOffices3	Id = “O2598”	**Retorna una referencia a la oficina “o2” con id “O2598”**
Company	**findBranchOffice**	setupBranchOffices4	Id = “O2222”	**Retorna null**
Company	**findBranchOffice**	setupBranchOffices4	Id = “O0000”	**Retorna una referencia a la oficina “o3” con id “O0000”**

Objetivo de la prueba: Determinar si el método removeBranchOffice remueve correctamente a una oficina de una lista doblemente enlazada de oficinas.

Clase	Método	Escenario	Valores de Entrada	Resultado
Company	**removeBranchOffice**	setupBranchOffice1	Id = “O2222”	**La primera oficina sigue siendo null.**
Company	**removeBranchOffice**	setupBranchOffice2	Id = “O2222”	**La primera oficina sigue siendo “o1” con id = “O1234”.**
Company	**removeBranchOffice**	setupBranchOffice2	Id = “O1234”	**La primera oficina ahora es null.**
Company	**removeBranchOffice**	setupBranchOffice3	Id = “O2222”	**La primera oficina sigue siendo “o1” con id = “O1234”.**
Company	**removeBranchOffice**	setupBranchOffice3	Id = “O1234”	**La primera oficina ahora es “o2” con id = “O2598”.**
Company	**removeBranchOffice**	setupBranchOffice4	Id = “O2222”	**La primera oficina sigue siendo “o1” con id = “O1234”, no se ha modificado la lista.**
Company	**removeBranchOffice**	setupBranchOffice4	Id = “O1111”	**La primera oficina sigue siendo “o1” con id = “O1234” y la tercera oficina tiene por siguiente oficina null.**

Objetivo de la prueba: Determinar si el método insertionSortBranchOffices ordena apropiadamente usando como criterio el id y la ciudad.

Clase	Método	Escenario	Valores de Entrada	Resultado
Company	**insertionSortBranchOffices**	setupBranchOffices1	**ninguno**	**No hay cambios.**
Company	**insertionSortBranchOffices**	setupBranchOffices2	**ninguno**	**No hay cambios.**
Company	**insertionSortBranchOffices**	setupBranchOffices3	**ninguno**	**No hay cambios.**
Company	**insertionSortBranchOffices**	setupBranchOffices5	**ninguno**	**Se ordena intercambiando los objetos.**
Company	**insertionSortBranchOffices**	setupBranchOffices4	**ninguno**	**Se ordena dejando el orden siguiente para las oficinas: “O0000”-”O1111”-”O1234”-**

				”O2598”

Objetivo de la prueba: Determinar si el método selectionSortBranchOffices ordena apropiadamente usando como criterio el id.				
Clase	**Método**	**Escenario**	**Valores de Entrada**	**Resultado**
Company	selectionSortBranchOffices	setupBranchOffices1	**ninguno**	**No hay cambios.**
Company	selectionSortBranchOffices	setupBranchOffices2	**ninguno**	**No hay cambios.**
Company	selectionSortBranchOffices	setupBranchOffices3	**ninguno**	**No hay cambios.**
Company	selectionSortBranchOffices	setupBranchOffices5	**ninguno**	**Se ordena intercambiando los objetos.**
Company	selectionSortBranchOffices	setupBranchOffices4	**ninguno**	**Se ordena dejando el orden siguiente para las oficinas: “O0000”-”O1111”-”O1234”-”O2598”**

Objetivo de la prueba: Determinar si el método bubbleSortContracts ordena apropiadamente usando como criterio el id de los contratos.				
Clase	**Método**	**Escenario**	**Valores de Entrada**	**Resultado**
Company	bubbleSortContracts	setupContracts1	**ninguno**	**No hay cambios.**
Company	bubbleSortContracts	setupContracts2	**ninguno**	**No hay cambios.**
Company	bubbleSortContracts	setupContracts3	**ninguno**	**No hay cambios.**
Company	bubbleSortContracts	setupContracts5	**ninguno**	**Se ordena intercambiando los objetos.**
Company	bubbleSortContracts	setupContracts4	**ninguno**	**Se ordena dejando el orden siguiente para los contratos: “C0000”-”C1111”-”C1234”-”C2598”**

Objetivo de la prueba: Determinar si el método addCompany agrega correctamente a una compañía nueva a un árbol binario de búsqueda de compañías.

Clase	Método	Escenario	Valores de Entrada	Resultado
Holding	**addCompany**	setupCompany1	Objeto compañia con atributos Name = “Scribe” Id = “N2222”	**La compañía se agrega como primera compañía**
Holding	**addCompany**	setupCompany2	Objeto compañia con atributos Name = “Scribe” Id = “N2222”	**La compañía se agrega a la derecha de la primera compañía**
Holding	**addCompany**	setupCompany3	Objeto compañia con atributos Name = “Scribe” Id = “N2222”	**La compañía se agrega a la izquierda de la compañía “Carvajal”.**
Holding	**addCompany**	setupCompany4	Objeto compañia con atributos Name = “Scribe Id = “N2222”	**La compañía se agrega a la izquierda de la compañía “Carvajal”.**
Holding	**addCompany**	setupCompany4	Objeto compañia con atributos Name = “Camilo” Id = “N1125”	**La compañía se agrega a la derecha de la La compañía “Reprograf”**

Objetivo de la prueba: Determinar si el método removeCompany remueve correctamente a una compañía de un árbol binario de búsqueda de compañías.

Clase	Método	Escenario	Valores de Entrada	Resultado
Holding	**removeCompany**	setupCompany1	NIT = “N2222”	**La primera compañía sigue siendo null.**
Holding	**removeCompany**	setupCompany2	NIT = “N2222”	**La primera compañía sigue siendo “Norma” con NIT = “N1234”**

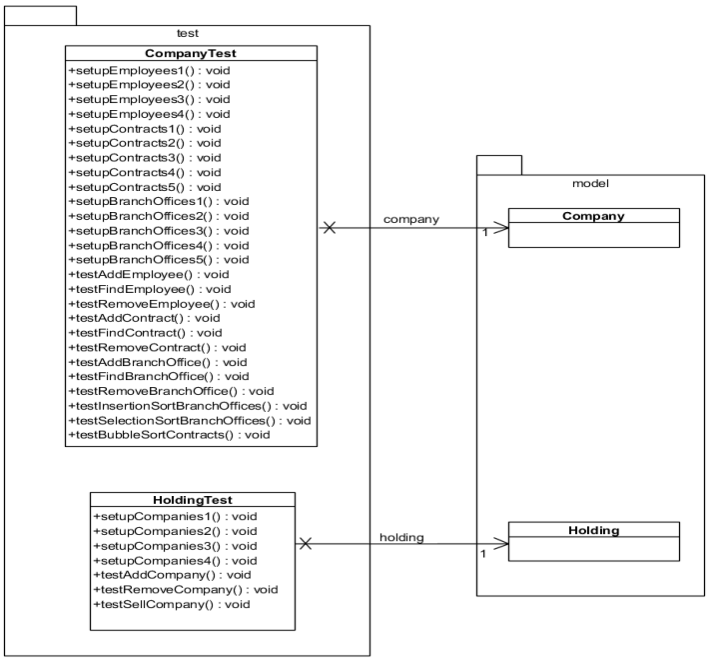
Holding	**removeCompany**	setupCompany2	NIT = “N1234”	**La primera compañía ahora es null.**
Holding	**removeCompany**	setupCompany3	NIT = “N2222”	**La primera compañía sigue siendo “Norma” con NIT = “N1234” y a su derecha “Carvajal con NIT = “N2598”**
Holding	**removeCompany**	setupCompany3	NIT = “N1234”	**La primera compañía ahora es “Carvajal” con NIT = “N2598”**
Holding	**removeCompany**	setupCompany4	NIT = “N2222”	**La primera compañía sigue siendo “Norma” con NIT = “N1234”**
Holding	**removeCompany**	setupCompany4	NIT = “N1234”	**La primera compañía ahora es “Carvajal” con NIT = “N2598”, a su izquierda “Bavaria” con NIT = “N0000” y a su derecha “Reprograf” con NIT = “N2598”**
Holding	**removeCompany**	setupCompany4	NIT= “N1111”	**La compañía a la derecha de “Bavaria” con id “N0000” ahora es null.**
Holding	**removeCompany**	setupCompany4	NIT = “N0000”	**La compañía a la izquierda de la primera ahora es “Reprograf” con NIT = “N1111”**

Objetivo de la prueba: Determinar si el método sellCompany remueve correctamente a una compañía de un árbol binario de búsqueda de compañías y además agrega su valor al holding.

Clase	Método	Escenario	Valores de Entrada	Resultado
Holding	**sellCompany**	setupCompany1	NIT = “N2222”	**La primera compañía sigue siendo null. No hay cambios en el valor del holding.**
Holding	**sellCompany**	setupCompany2	NIT = “N2222”	**La primera compañía sigue siendo “Norma” con NIT = “N1234”. No hay cambios en el valor del holding.**
Holding	**sellCompany**	setupCompany2	NIT = “N1234”	**La primera compañía ahora es null. No hay cambios en el valor del holding.**
Holding	**sellCompany**	setupCompany3	NIT = “N2222”	**La primera compañía sigue siendo “Norma” con NIT = “N1234” y a su derecha**

				“Carvajal con NIT = “N2598”. No hay cambios en el valor del holding.
Holding	**sellCompany**	setupCompany3	NIT = “N1234”	**La primera compañía ahora es “Carvajal” con NIT = “N2598”. Se suma el valor de la compañía “Norma” al holding**
Holding	**sellCompany**	setupCompany4	NIT = “N2222”	**La primera compañía sigue siendo “Norma” con NIT = “N1234”**
Holding	**sellCompany**	setupCompany4	NIT = “N1234”	**La primera compañía ahora es “Carvajal” con NIT = “N2598”, a su izquierda “Bavaria” con NIT = “N0000” y a su derecha “Reprograf” con NIT = “N2598”. Se suma el valor de la compañía “Norma” al holding.**
Holding	**sellCompany**	setupCompany4	NIT= “N1111”	**La compañía a la derecha de “Bavaria” con id “N0000” ahora es null. Se suma el valor de la compañía “Reprograf” al holding**
Holding	**sellCompany**	setupCompany4	NIT = “N0000”	**La compañía a la izquierda de la primera ahora es “Reprograf” con NIT = “N1111”. Se suma el valor de la compañía “Bavaria” al holding**

12. Diagrama de clases - Pruebas



13. Repositorio de GitHub:

github.com/kamneklogs/holding-manager-app