INFORME PROYECTO DPOO:

TuChamba

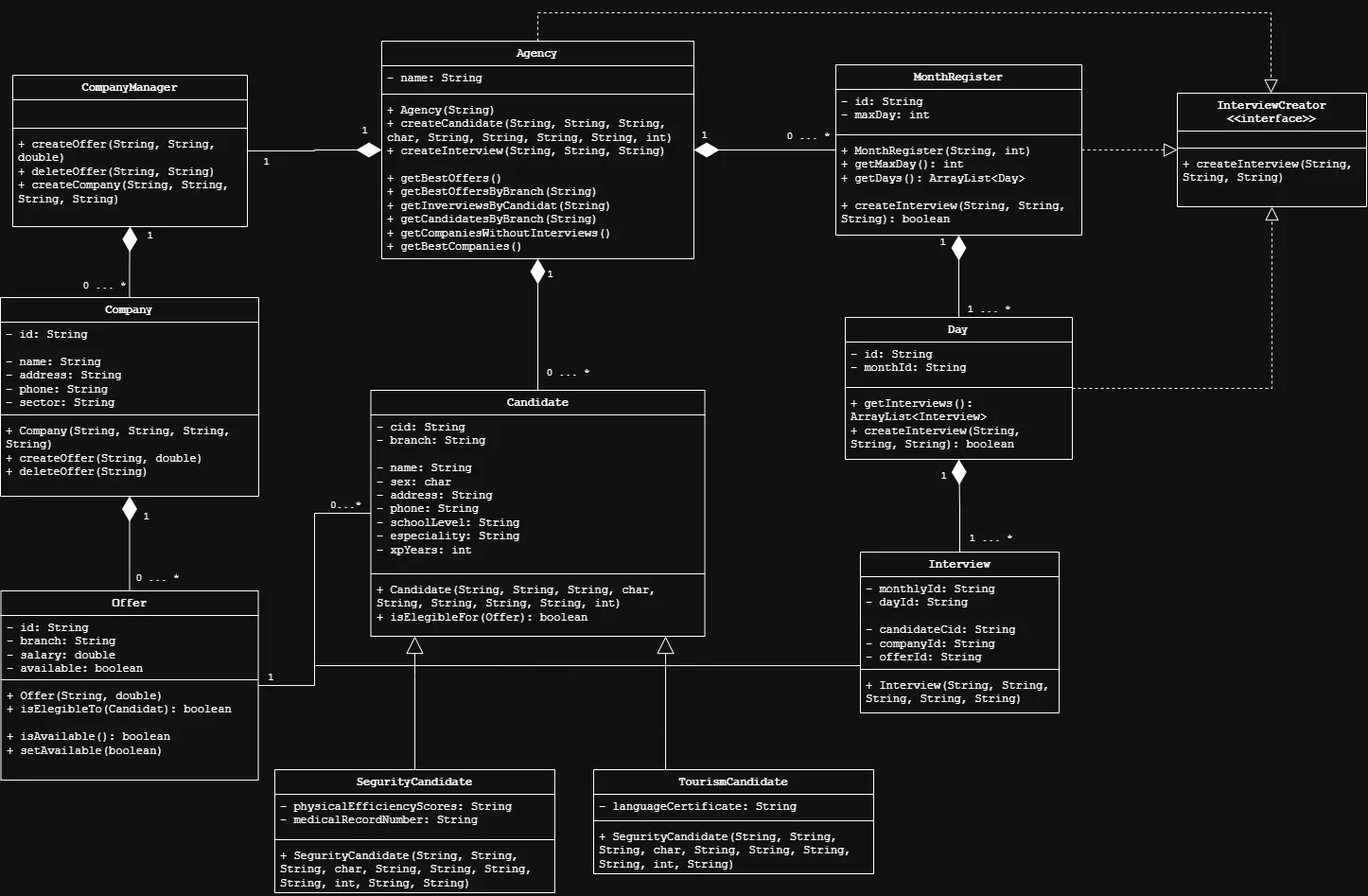
Integrantes:

- Rodny Roberto Estrada León

- Martín Alejandro García Babastro

Grupo: 11

1.Diagrama de clases:



2.Explicación de las Clases y Relaciones

Clases Principales:

2.1. Company

- Atributos: `id`, `name`, `address`, `phone`, `sector`.

- Métodos: `Company(String, String, String, String)`, `createOffer(String, double)`, `deleteOffer(String)`.

- Relaciones: Tiene una relación 1 a \* con la clase `Offer`.

2.2 \*\*Offer\*\*

- Atributos: `id`, `branch`, `salary`, `available`.

- Métodos: `Offer(String, double)`, `isElegibleTo(Candidate): boolean`, `isAvailable(): boolean`, `setAvailable(boolean)`.

- Relaciones: Relación 0 a \* con `Candidate`.

3. Candidate

- Atributos: `cid`, `branch`, `name`, `sex`, `address`, `phone`, `schoolLevel`, `especiality`, `xpYears`.

- Métodos: `Candidate(String, String, String, char, String, String, String, String, int)`, `isElegibleFor(Offer): boolean`.

- Relaciones: Relación 0 a \* con `Offer`.

4. Agency

- Atributos: `name`.

- Métodos: `Agency(String)`, `createCandidate(...)`, `createInterview(...)`, `getBestOffers()`, `getBestOffersByBranch(String)`, `getInterviewsByCandidate(String)`, `getCandidatesByBranch(String)`, `getCompaniesWithoutInterviews()`, `getBestCompanies()`.

- Relaciones: Relación 1 a \* con `MonthRegister`.

5. MonthRegister

- Atributos: `id`, `maxDay`.

- Métodos: `MonthRegister(String, int)`, `getMaxDay(): int`, `getDays(): ArrayList<Day>`, `createInterview(String, String, String): boolean`.

- Relaciones: Relación 1 a \* con `Day`.

6. Day

- Atributos: `id`, `monthId`.

- Métodos: `getInterviews(): ArrayList<Interview>`, `createInterview(String, String, String): boolean`.

- Relaciones: Relación 1 a \* con `Interview`.

7. Interview

- Atributos: `monthlyId`, `dayId`, `candidateCid`, `companyId`, `offerId`.

- Métodos: `Interview(String, String, String, String, String)`.

3. División en Capas y Paquetes

La solución está organizada en capas y paquetes para garantizar modularidad, claridad y separación de responsabilidades. A continuación, se detalla cada capa y sus respectivos paquetes:

3.1 Capa de Presentación (`gui`)

Esta capa contiene las clases relacionadas con la interfaz gráfica del usuario. Está subdividida en paquetes según las funcionalidades específicas:

- \*\*about\*\*: Contiene la pantalla de información general del sistema (`AboutScreen.java`).

- \*\*candidate\*\*: Maneja la gestión de candidatos, incluyendo formularios, pantallas de administración y programación de entrevistas (`CandidateFormDialog.java`, `CandidateManagerScreen.java`, `ScheduleInterviewDialog.java`).

- \*\*company\*\*: Gestiona las empresas y sus ofertas, con formularios y pantallas de administración (`CompanyFormDialog.java`, `CompanyManagerScreen.java`, `OfferFormDialog.java`).

- \*\*components\*\*: Contiene componentes reutilizables como botones personalizados, tablas y campos de búsqueda (`FilterDialog.java`, `MButton.java`, `MTable.java`, `SearchField.java`).

- \*\*reports\*\*: Incluye las clases para generar y visualizar reportes (`ReportPanel.java`, `ReportsHomeScreen.java`).

- \*\*HomeScreen.java\*\* y \*\*Login.java\*\*: Pantallas principales y de inicio de sesión.

3.2 Capa de Negocio (`logic`)

Encapsula la lógica del sistema y las clases principales que representan las entidades del dominio:

- \*\*Agency.java\*\*: Clase principal que coordina las operaciones del sistema.

- \*\*candidate\*\*: Contiene las clases relacionadas con los candidatos (`Candidate.java`, `SegurityCandidate.java`, `TourismCandidate.java`).

- \*\*company\*\*: Maneja las empresas y sus ofertas (`Company.java`, `CompanyManager.java`, `Offer.java`).

- \*\*interview\*\*: Gestiona las entrevistas y registros mensuales (`Day.java`, `Interview.java`, `MonthRegister.java`).

- \*\*GlobalAgency.java\*\*: Clase que centraliza la gestión de la agencia global.

3.3 Capa de Ejecución (`runner`)

Contiene la clase principal que inicia la aplicación, incluyendo el registro de las ventanas para poder navegar entre ellas. (`Runner.java`).

3.4 Capa de Utilidades (`utils`)

Proporciona clases auxiliares y constantes para facilitar la implementación:

- constants: Define enumeraciones como ramas, sectores y especialidades (`Branch.java`, `Sector.java`, `Specialty.java`).

- Generator.java: Clase para generar identificadores únicos.

- Id.java: Maneja la lógica de identificación.

- Navigation.java: Facilita la navegación entre pantallas.

4. Responsabilidades de los Integrantes

Rodny Roberto Estrada León

- Diseño del Diagrama de Clases: Responsable de crear y ajustar el diagrama de clases según las observaciones del corte.

- Implementación de la Capa de Negocio: Desarrollo de las clases principales como `Agency`, `Candidate`, `Company`, y `Offer`, asegurando la correcta lógica de negocio y relaciones entre las entidades. También se encargó de implementar el CRUD de las entidades.

- Pantallas de Candidate y Company: Desarrollo de las interfaces gráficas relacionadas con la gestión de candidatos y empresas, incluyendo formularios y pantallas de administración.

- Pantalla de About: Creación de la pantalla de información general del sistema.

Martín Alejandro García Babastro

- Desarrollo de la Capa de Presentación: Implementación de las pantallas principales como `HomeScreen`, `Login`, y los componentes reutilizables como `MButton`, `MTable`, y `SearchField`.

- Generación de Reportes: Desarrollo de las clases para la generación y visualización de reportes (`ReportPanel`, `ReportsHomeScreen`).

Reportes:

* Mejores Ofertas: Se recorrerá la lista de todas las ofertas existentes, agregando las ofertas con un salario mayor a $10000 a una nueva lista que retornara el método.

- Integración de Componentes Reutilizables: Diseño e implementación de componentes para garantizar consistencia y reutilización en la interfaz.

- Aspectos Generales de la Aplicación: Trabajo en el resto de los aspectos de la aplicación, incluyendo la navegación entre pantallas y la integración de las capas.