Entrega Acumulada B

Tema: **18. Gestión de Votantes-Locales de Sufragio**

Nombres: **Maximiliano Valencia Saez**

1.1 Análisis de los datos a utilizar y principales funcionalidades a implementar que dan sentido a la realización del proyecto.

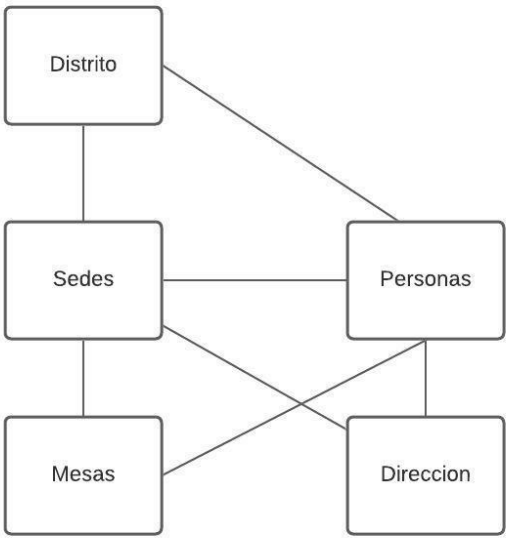
Datos principales a utilizar:

- Personas : Nombres y apellidos, Rut, dirección, partido al que pertenezca(si tiene), además la mesa a la que pertenece y la sede que quede más cercana su dirección
- Sedes: Nombre y dirección, más las mesas que estan en la sede, y las personas de la misma
- Mesas: el número de mesa, los vocales, los votantes y apoderados (todas personas)
- Dirección: calle, número y ciudad

Funcionalidades principales:

- Almacenamiento y gestión de votantes-locales de sufragio, agregar,modificar,buscar,listar y eliminar personas,mesas o sedes.
- Asignación automática por cercania geográfica entre las personas y la sede más cercana a ellas, esta funcionalidad se realiza gracias a la API geocoding de Google y una librería externa para manipulacion de objetos JSON

1.2 Diseño conceptual de clases del Dominio y su código en Java



Las clases del dominio del dominio del problema tendrán una estructura similar a la de la imagen. Esta estructura se ve con mayor detalle en el diagrama de clases UML más adelante.

En este diagrama no se incluyen las clases necesarias para cumplir con los requerimientos solamente de manera instrumental (librerias, clases de interfaz con el usuario)

1.3 Todos los atributos de todas las clases deben ser privados y poseer sus respectivos métodos de lectura y escritura (getter y setter).

Todos los atributos de las clases concretas son privados, con sus respectivos getters y setters.
La clase abstracta Persona posee 4 parámetros que son static y final por lo que se opto por dejar esos atributos públicos, ya que no pueden ser modificados.

1.4 Se deben incluir datos iniciales dentro del código.

La clase distrito posee un método cargarPrueba() con datos iniciales dentro del código.

Además, se utilizan librerías externas para manipular objetos JSON. El uso de la anterior librería es para manipular el objeto json de respuesta a la API geocoding de Google, y para el almacenamiento y persistencia de los datos del proyecto (ver archivo datos.json en la carpeta del proyecto)

2.1 Diseño conceptual y codificación de 2 (dos) niveles de anidación de colecciones de objetos.

Entiéndase "--->" como "contiene colección de" :

Distrito--->Sedes--->Mesas--->Personas
| └------>Personas
V
. Personas

2.2 Diseño conceptual y codificación de 2 (dos) clases que utilicen sobrecarga de métodos.

La clase Sede posee sobrecarga a su método agregarMesa(), de manera que hay:

- Un método sin parámetros, y otro que recibe como parámetro un int numeroMesa

La clase Distrito posee sobrecargas a sus métodos agregarPersona() y agregarSede(), de manera que hay:

- agregarSede con parámetro Sede sede, y agregarSede con parámetro arreglo de String
- agregarPersona con 7 parámetros de String, agregarPersona con 6 parámetros de String y agregarPersona con arreglo de String

2.3 Diseño conceptual y codificación de al menos una clase mapa del JCF.

La clase Distrito posee 3 mapas como atributos, un mapa<String,Persona> para las personas por su Rut, otro mapa<String,Persona> para las personas por su Sede, y otro mapa<String,Sede> para las Sedes por su nombre

También la clase Sede posee un mapa<String,Persona> para las personas por su Rut

3.1 Diseño de diagrama de clases UML

Ver UML.jpeg adjuntado con la carpeta del proyecto

3.2 Implementación de menú del sistema para funcionalidades

Dentro de la clase Control, el método run() contiene el bucle principal del programa, este lleva dentro un bloque switch que recibe las respuestas de las opciones que la clase Consola le entrega. Esta clase, Consola, contiene el menu de control por teclado del programa, contiene todas las opciones y funcionalidades posibles del programa

3.2.1 Funcionalidad básica (1): Inserción manual o agregar elemento

Los métodos agregarSede() y agregarPersona() de la clase Distrito se encargan de cumplir esta funcionalidad, se accede mediante consola.

3.2.2 Funcionalidad básica (2): Mostrar por pantalla listado de elementos

Los metodos mostrarSedes(), mostrarPersonasxSede() y mostrarSedesMesasyPersonas() de la clase Distrito se encargan de cumplir esta funcionalidad, se accede mediante consola.

A.2 Implementación de las siguientes funcionalidades en el menú:

A.2.1 Funcionalidad básica (1): Edición/modificación de elemento

A.2.2 Funcionalidad básica (2): Eliminación del elemento para las 2 colecciones anidadas.

A.3 Se debe generar un reporte en archivo txt/csv que considere mostrar datos de las 2 colecciones anidadas

Se genera reporte en el archivo reporte.csv, el método reportar() de la clase Control de encarga de ello

A.5

Funcionalidades implementadas y accesibles por consola

EP4.1:

1)seleccionar un objeto por criterio:

- se implementaron los metodos siguientes metodos en la clase Distrito.java:
- getSedeMasPersonas(), getSedeMenosPersonas() y getPersonasMasLejos(), haciendo la funcionalidad de obtener la sede con mas personas, la sede con menos personas y las personas mas lejanas a su sede asignada respectivamente.

2) subconjunto filtrado por criterio :

- se implementaron los métodos getVotantesMismaDir() haciendo la funcionalidad de obtener los votantes que tengan la misma dirección

EP4.2: Clases con sobreescritura de métodos:

Aparte de los metodos heredados de las clase abstracta Persona.java y las implementaciones de la interfaz Coordenable.java, el método toString() fue sobreescrito en la clase Persona.java (linea 44)

EP4.3:

La clase abstracta es Persona.java, de la cual es clase padre de Votante.java,Apoderade.java y Vocal.java. Esta clase es utilizada en contexto por ejemplo en el método coordenar() de la clase Distrito.java. Tambien se utiliza la clase Sede.java en el método asignarMesas().

EP4.4 :

La interfaz es Coordenable.java, esta es implementada por Persona.java(y sus clases herederas) Sede.java y Direcion.java. La interfaz es utilizada en contexto por ejemplo en el método coordenar() de la clase Distrito.java o en el método asignarSede() de la misma clase.

EP4 OP :

Se generó documentación con Javadoc

EB 1:

Integración de lo anterior

EB2:

EB3:

Se aplican principios de encapsulamiento y orientación a objetos (apreciable en el diagrama de clases)