Disaster Relief System

Created by: Deven Powell (UCID: 30173192)

This disaster relief system allows for the creation, storage, and access for various locations, disaster victims, and inquirers. It's main purpose is to provide a user-interface for relief workers to keep track of where individuals are housed, and to aid in the reunification of family members.

File Structure and Dependencies

All *.java files that are required for the program to run are included in the edu.ucalgary.oop package. For the test files to work, Junit and Hamcrest-Core will need to be installed outside of the edu folder. In order for the database access to work, PostgreSQL and the JDBC Driver are required. In order to enter genders for disaster victims, a file named `GenderOptions.txt` must be added outside of the edu folder.

Database Access

In PostgreSQL, login with user 'oop' and password 'ucalgary' (or enter the query "CREATE USER oop PASSWORD 'ucalgary';", if this user has not been created). The included file "project.sql" has minor changes from the one provided in D2L, namely that the "id" was removed from the INSERT statement as it caused issues with SERIAL. Copy the first three lines of the file into the psql shell, and execute the statement (if there is an error along the lines of "user does not have permission to create database," the query "ALTER USER oop CREATEDB;" executed as admin will fix it). Now, copy the rest of the file into the shell and execute (data in the tables may be changed).

Compilation

This program can be compiled and run in the command line using javac. It is intended to be run on Java 11, and using any other Java version may cause unpredictable behaviour. The command will slightly differ depending on the operating system. The "main" method is located in `UserInterface.java`. The command line tag `central` launches the program in "central" mode. It can also be run using `location` instead of `central` but the program will automatically switch to central mode since no locations will be available initially.

Unix-based (Mac, Linux):

1. `cd` to the directory in which the edu.ucalgary.oop package is located
2. To compile: `javac -cp ".:{path to JDBC driver}" edu/ucalgary/oop/UserInterface.java`

3. To run: `java -cp ".:{path to JDBC driver}" edu.ucalgary.oop.UserInterface central`

Windows (PowerShell, Command Prompt):

1. `cd` to the directory in which the edu.ucalgary.oop package is located
2. To compile: `javac -cp ".;{path to JDBC driver}" edu/ucalgary/oop/UserInterface.java`
3. To run: `java -cp ".;{path to JDBC driver}" edu.ucalgary.oop.UserInterface central`

Features

This section will describe how some of the project requirements have been implemented, and how they can be tested.

Central and location-based modes:
Central mode is able to access all locations and the database. It allows for the creation of locations, adding disaster victims to any location, and updating the database.

Location mode is only able to access a selected location. They can add disaster victims to only the location they are setup in, and cannot access the database.

Family relationship consistency:
Consider three distinct disaster victims *Peace*, *Sam*, and *Diamond*. If *Sam* and *Peace* are siblings, and *Sam* and *Diamond* are siblings, *Sam* and *Diamond* will also be listed as siblings. In order to test this, create a location and add three disaster victims to it. Set the relationship between the disaster victims with the same format as described above. When prompted to "enter the relationship" enter `siblings` into the console. The relationships can be seen in all three disaster victims by selecting a victim by their ID then selecting the option to "display all known disaster victim data."

Supply consistency:
Supplies can only be allocated to a disaster victim if enough of that supply exists at the location. When supply is allocated to a disaster victim, that amount of supply is removed from the location's storage. When a supply is removed from a disaster victim, it is considered used, and is not allocated back to the location for hygenic purposes.

Multiple interactions with inquirer:
An inquirer is considered the same as another inquirer when they have the same first name, last name, and phone number. Inquiries with the same inquirer are all shown under one inquirer, instead of multiple. This can be tested by selecting "display all inquiries" after creating two inquires with an inquirer with the same name and phone number.