Human And Nature Resources

Group Members : Abolfazl Aslani Abrbekooh & Zahra Jandaghi Jafari

Version : 1.0.0

Different Part of the Project

First We Analyze Different Sections of the project

The Project folder Structure looks like this

1.app
----HumanResources.java
----NatureResources.java
----Resources.java
----Welcome.java

2.database
----lands.txt

----user.txt
3.menu
----HumanResourcesMenu.java
----Intro.java
----NatureResourcesMenu.java
----StatisticsDataMenu.java
4.randomdata
----has 9 text files
5.statistics
----AgeCounter.java
----DeathCounter.java
----GenderBlockRetriever.java
6.utils
----1.editors
--------NatureEditor.java
--------PersonEditor.java
----2.Retrievers
--------NatureRetriever.java
--------PersonRetriever.java
----DataAppender.java
----DataCollector.java
----DateOfDeathAdder.java

----DuplicateSsn.java

----SystemClear.java

1.app

Class `HumanResources`:

- It holds personal information of individuals such as name, last name, social security number (SSN), age, etc.
- It has `submit` and `edit` methods. The `submit` method saves the person's information to a text file, and the `edit` method allows editing the person's information.

Class `NatureResources`:

- It holds information about natural resources such as animal types, plant types, country, etc.
- It has `submit` and `edit` methods. The `submit` method saves the resource information to a text file, and the `edit` method allows editing the resource information.

Class `Resources`:

- It has a property named `city` that stores the name of a city.
- It provides general methods for creating, searching, and performing other operations related to individuals and natural resources.

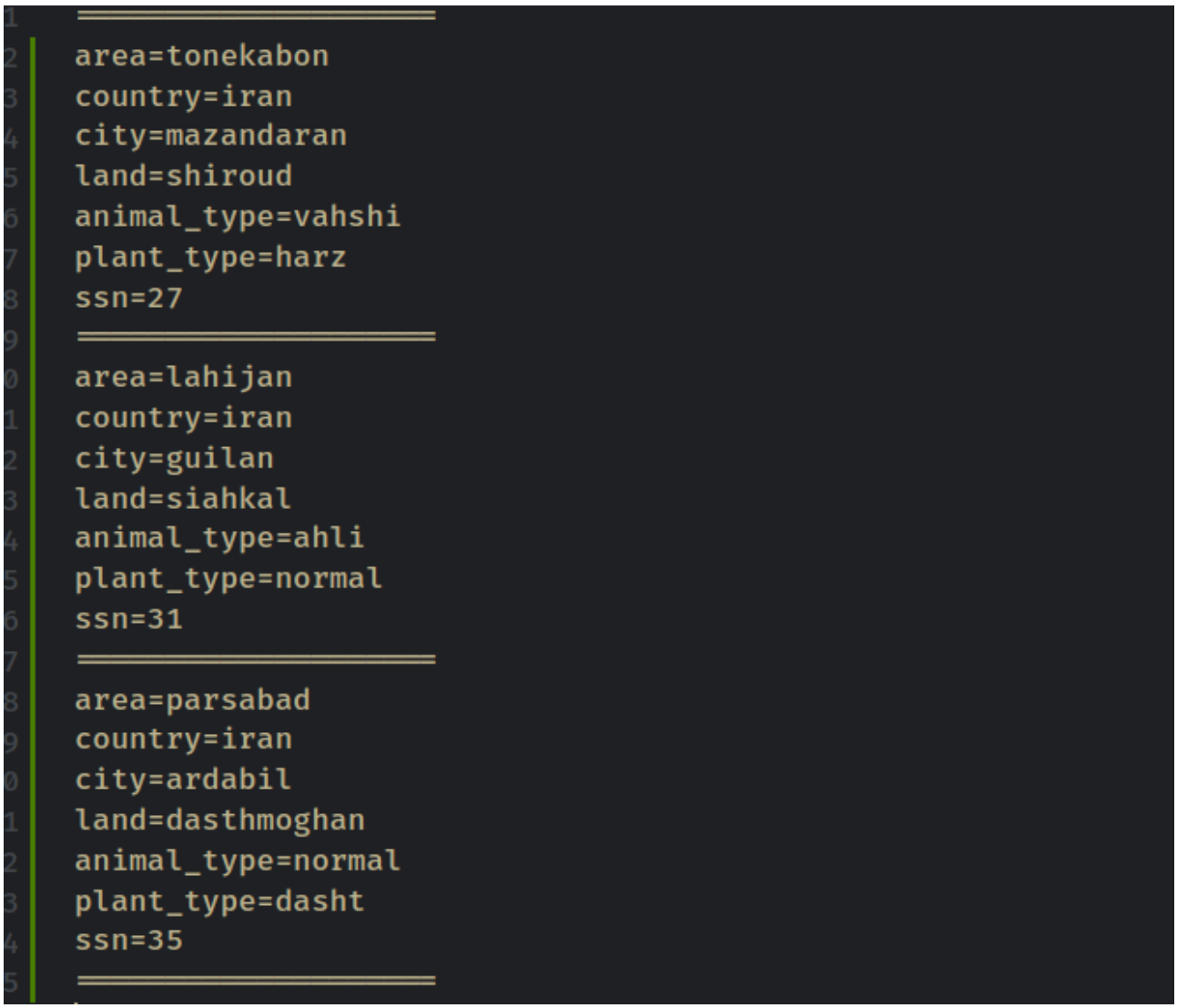
Class `Welcome`:

- This class is the main entry point of the program and contains the `main` method, which starts the execution of the program.

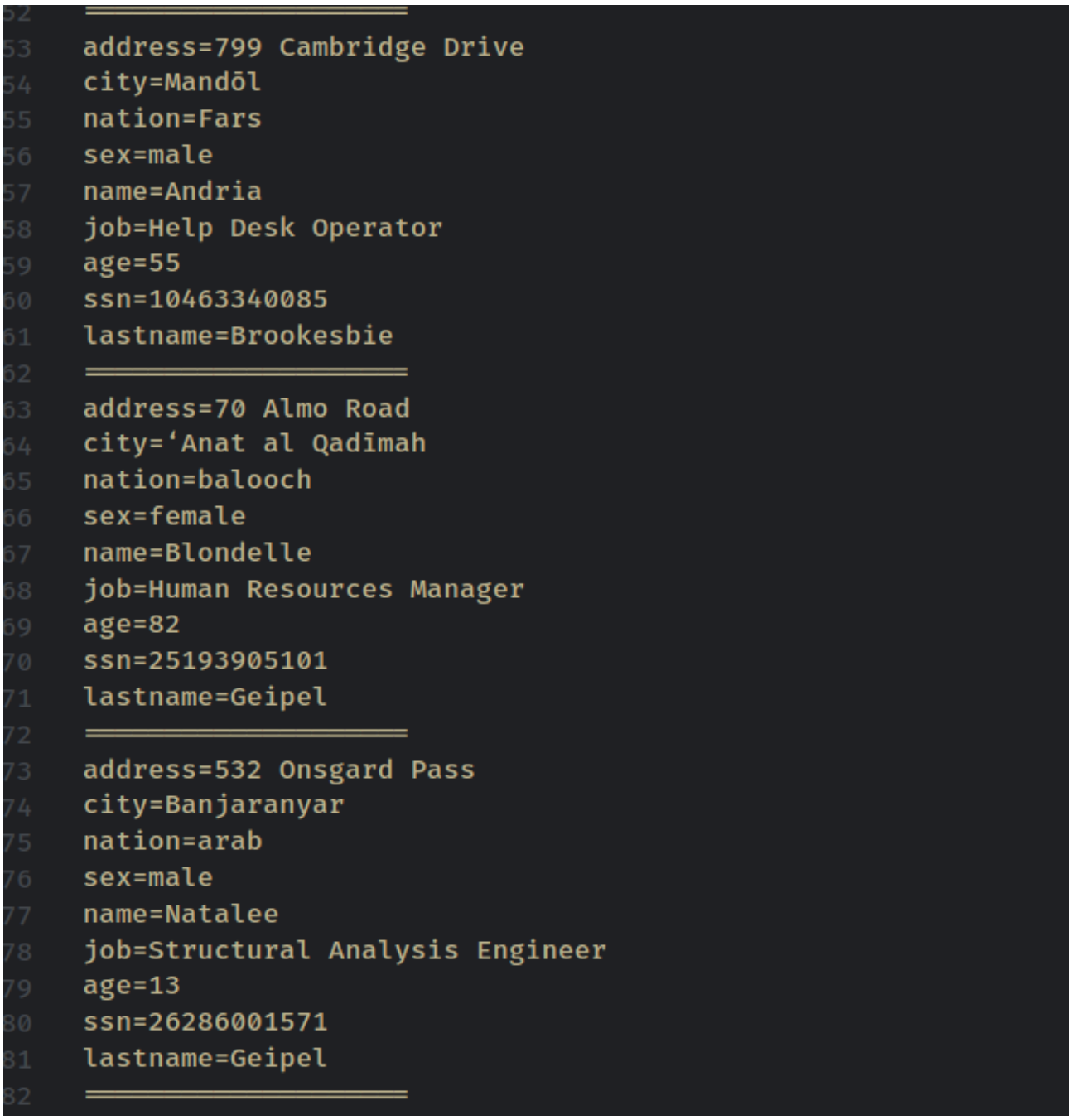
]

2.database

This folder has 2 text files which act as a database to store data.



Snapshot 2.1 lands.txt



Snapshots 2.2 user.txt

3.menu

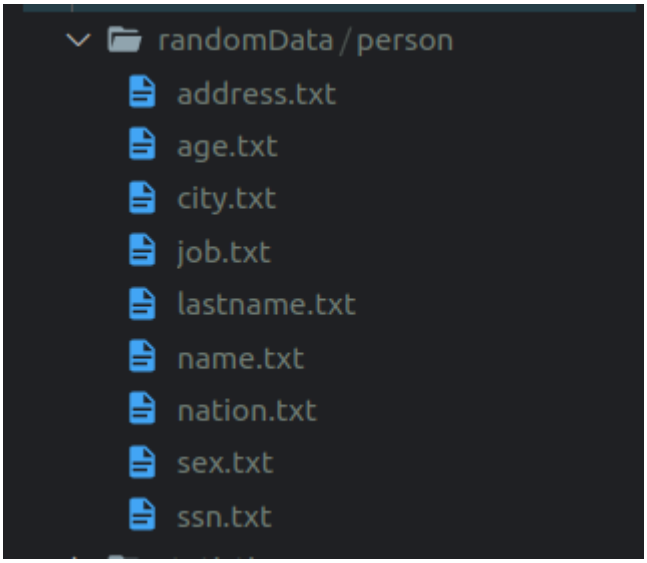
1. `HumanResourceMenu`:
    - This class handles the menu options related to human resources.
    - It contains methods for creating a new human resource, editing an existing human resource, finding a human resource, submitting a death record, and creating random human resources.
    - Each method prompts the user to input various details for a human resource (e.g., SSN, name, lastname, city, address, job, age, sex, nation) using `Scanner` for input.
    - The methods make use of other classes like `SystemClear`, `DuplicateSsn`, and `HumanResources` to perform various operations such as clearing the command prompt, validating SSN or code uniqueness, submitting or editing human resources, finding human resources, etc.
2. `Intro`:
    - This class represents the main menu of the program.
    - It prompts the user to select an option from the main menu.
    - The available options are: managing resources (human or nature), creating random resources, or accessing statistic reports.
    - Depending on the user's choice, it further prompts for sub-options related to managing human resources, nature resources, or statistic reports.
    - The `Intro` class interacts with other classes like `HumanResourceMenu`, `NatureResourceMenu`, and `StatisticsDataMenu` to handle user's choices and execute the corresponding operations.
3. `NatureResourceMenu`:
    - This class is similar to `HumanResourceMenu` but deals with nature resources instead.
    - It contains methods for creating a new nature resource, editing an existing nature resource, and finding a nature resource.
    - The methods prompt the user to input various details for a nature resource (e.g., code, animal type, plant type, city, land name, country, area) using `Scanner` for input.
    - The methods make use of other classes like `SystemClear`, `DuplicateSsn`, and `NatureResources` to perform operations related to nature resources.

Overall, these classes provide a menu-driven interface to manage human and nature resources. The program allows users to perform operations like creating, editing, finding, and submitting death records for human resources, as well as creating and editing nature resources.

4.randomData

This folder contains 9 txt files with random data

this data are used for creating random resources



Snapshot 4.1

5.statistics

The "statistics" folder contains three Java classes related to statistical analysis of data stored in a text file named "user.txt":

1. AgeCounter: This class has a static method named "counter" that takes an integer parameter "age" as input. It reads the "user.txt" file line by line and counts the number of blocks (sections delimited by "==========") that have ages higher or lower than the given input age. The counts are printed to the console.
2. DeathCounter: This class has a main method that calls a static method named "countBlocksWithDateOfDeath." It reads the "user.txt" file line by line and counts the number of blocks that contain the string "dateofdeath=". The count is printed to the console.
3. GenderBlockRetriever: This class has a static method named "retrieveBlocksByGender" that takes a string parameter "gender" as input. It reads the "user.txt" file line by line and retrieves blocks that contain the specified gender (identified by the string "sex=" + gender). The retrieved blocks are stored in a list and returned as the result.

These classes provide statistical functions for analyzing data in the "user.txt" file, such as counting blocks based on age or date of death, and retrieving blocks based on gender.

6.utils

Editors section

The "utils/editors" section contains two Java classes:

1. NatureEditor: This class provides a static method named "editNature" for editing the data related to nature stored in the "lands.txt" file. The method takes two parameters: "ssn" (social security number) and "newData" (a map of new data). It reads the lines of the "lands.txt" file, searches for the block (section) that matches the given "ssn," and updates the block's data with the new values provided in the "newData" map. The updated lines are then written back to the "lands.txt" file.
2. PersonEditor: This class provides a static method named "editPerson" for editing the data related to a person stored in the "user.txt" file. The method takes two parameters: "ssn" (social security number) and "newData" (a map of new data). It reads the lines of the "user.txt" file, searches for the block (section) that matches the given "ssn," and updates the block's data with the new values provided in the "newData" map. The updated lines are then written back to the "user.txt" file.

Both classes utilize common utility methods for reading and writing lines to a file. These methods include "readFileLines" (reads lines from a file and returns them as a list), "writeFileLines" (writes a list of lines to a file), and "updateBlockData" (updates the data within a block based on a map of new values).

These classes provide convenient editing functionality for modifying specific blocks of data in the "lands.txt" and "user.txt" files.

6.utils

Retrievers section

The "utils/Retrievers" folder contains two Java classes:

1. NatureRetriever: This class provides a static method named "retrievePersonData" for retrieving data related to nature from the "lands.txt" file. The method takes a parameter "ssn" (social security number) and returns a map containing the person's data. It reads the lines of the "lands.txt" file, searches for the block (section) that matches the given "ssn," and extracts the data from that block into a map. The map is then returned with the key-value pairs representing the data fields and their corresponding values.
2. PersonRetriever: This class provides a static method named "retrievePersonData" for retrieving data related to a person from the "user.txt" file. The method takes a parameter "ssn" (social security number) and returns a map containing the person's data. It reads the lines of the "user.txt" file, searches for the block (section) that matches the given "ssn," and extracts the data from that block into a map. The map is then returned with the key-value pairs representing the data fields and their corresponding values.

Both classes utilize common utility methods for reading lines from a file and parsing block data into key-value pairs. The utility methods include "readFileLines" (reads lines from a file and returns them as a list) and "parseBlockData" (parses a block of data into key-value pairs using the "=" symbol as a delimiter).

These classes provide convenient retrieval functionality for accessing specific blocks of data from the "lands.txt" and "user.txt" files and returning them as maps for further processing.

6.utils

The "utils" folder contains several Java classes:

1. DataAppender: This class provides a constructor that appends data from a map to a file. It takes a map, a file path, and a separator as parameters. The constructor opens the file for appending and writes the key-value pairs from the map to the file using the specified separator. It also adds a separator line to indicate the end of the map.
2. DataCollector: This class collects random data from files. It has two methods:
    - getRandomDataFromFile: This method takes a file path and returns a random line from that file.
    - collectRandomDataFromFiles: This method takes a list of file names and collects random data from corresponding files by calling the getRandomDataFromFile method. It returns a list of collected random data.

    The class also includes a main method that demonstrates the usage of these methods by collecting random data for various file names.
3. DateOfDeathAdder: This class provides a static method named "addDateOfDeath" for adding a date of death to a person's data. It takes a social security number (ssn) and a date of death as parameters. The method reads the lines from the "user.txt" file, finds the block (section) that matches the given ssn, and adds the "dateofdeath" field with the provided dateOfDeath value to that block. The modified lines are then written back to the file.
4. DuplicateSsn: This class provides methods for checking duplicate social security numbers (SSNs) in a file. It includes the following methods:
    - isDuplicateSSN: This method takes a file path and an SSN to check and returns true if the SSN is found more than once in the file.
    - readMapsFromFile: This method reads maps from a file. It parses the file lines and splits them into key-value pairs to construct maps. Each map represents a block of data separated by the line "====================".
    - checker: This method takes an SSN and checks for duplicate SSNs in the "user.txt" file by calling the isDuplicateSSN method.
    - codeChecker: This method takes an SSN and checks for duplicate SSNs in the "lands.txt" file by calling the isDuplicateSSN method.
5. SystemClear: This class provides a main method for clearing the console screen. It detects the operating system and executes the appropriate command to clear the screen.

These classes provide various utility functionalities such as appending data to files, collecting random data from files, adding a date of death to a person's data, checking for duplicate SSNs in files, and clearing the console screen.

UML Class Diagram