CSCI 202: Object-Oriented Programming

Lab 4 - Due Thursday March 21, 2024 at 11:59pm

Resources:

- Java API index: https://docs.oracle.com/en/java/javase/17/docs/api/index-files/index-1.html

Part 1: Working with Files (Lab4_1)

Class File represents a file or path on your computer's file system. Reading from a text file can be done using class Scanner. Writing to a text file can be done using class PrintWriter.

In this part you will create a Java program that stores a user's name in a text file called *name.txt*. When the program is complete, it will do the following:

On startup, the program will read the user's name from the file *name.txt* and uses this to display a personal greeting. The user is then given a chance to change their name. If they change their name, the contents of *name.txt* must be overwritten with their new name. If the user does not change their name, then they are given a chance to unregister their name. Unregistering their name deletes *name.txt*.

If *name.txt* does not exist when the program starts, then the program must display a message telling the user that their name is not registered. In this case, the program must gives the user a chance to register their name. If they register their name, then the program must create *name.txt* and write their name to it.

Your Task:

1. Download and unzip Lab4_1.zip from canvas onto your local computer. Then open up the unzipped project with IntelliJ.
2. In IntelliJ open up Lab4_1.java (this is in package lab4_1 under src).
3. Method `registerName()` is used to register or change the user's name. In here do the following:
   a. Prompt the user for their name:

      ```
      System.out.print("Enter your name: ");
      String name = input.nextLine();
      ```

   b. Create a `PrintWriter` object for writing to file *name.txt*:

      ```
      PrintWriter out = new PrintWriter(file);
      ```

      The static data field `file` is the `File` object for *name.txt*. Calling `new PrintWriter(file)` will create *name.txt* if it does not exist.

   c. Write `name` to *name.txt*. This will overwrite anything in *name.txt*.

      ```
      out.println(name);
      ```

   d. Finally, close the `PrintWriter`:

      ```
      out.close();
      ```

   Calling `new PrintWriter(file)` will throw a `FileNotFoundException` if it cannot create or open the file. Instead of handling this exception here, we declare it to be thrown in `registerName()` and handle it in `main()`.
4. In `main()` write the following:

```
try
{
    Scanner in = new Scanner(file); // may throw a FileNotFoundException
    String name = in.nextLine();
    System.out.println("Hello " + name + ".");
    in.close(); // we are done reading from in, so close it

    // Section 1:
```

```
}
catch (FileNotFoundException e)
{
    // Section 2:
}
```

If *name.txt* does not exist, then `new Scanner(file)` will throw a `FileNotFoundException`. In this case the program jumps to section 2. Otherwise the program displays a personal message, closes `Scanner in`, and continues at section 1.

5. Section 1 (the user's name is already registered): Add code to section 1 to do the following:

   a. Prompt the user to check if they want to change their name. If so, then call `registerName()` to change their name. Write a try-catch block around the call to `registerName()`, handling the case when it throws a `FileNotFoundException`.

   b. If the user does not change their name, then prompt them if they want to unregister their name. If so, then delete *name.txt* by doing:

      ```
      boolean isDeleted = file.delete();
      ```

      This call `file.delete()` returns a `boolean`, which represents if the file was successfully deleted or not (`true` = success, `false` = fail). If the file was successfully deleted, then display a message stating that the user's name has been unregistered.

      `file.delete()` also throws a `SecurityException` if *name.txt* cannot be deleted. Add a try-catch block around the call to `file.delete()` to handle this exception.

6. Section 2 (the user's name is not registered): Add code to section 2 to prompt the user if they want to register their name. If so, then register their name by calling `registerName()`. Add a try-catch block around call to `registerName()` to handle the case when it throws a `FileNotFoundException`.

Hint: The user's response to a yes/no question can be read in with:

```
String response = input.nextLine().toLowerCase();
```

and the response can be tested with the condition

```
response.equals("y")
```

Sample runs:

```
Your name is not registered.
Would you like to register your name (y/n)? y
Enter your name: Daniel
Your name has been registered.
```

```
Hello Daniel.
Would you like to change your name (y/n)? y
Enter your name: Belteshazzar
Your name has been changed.
```

```
Hello Belteshazzar.
Would you like to change your name (y/n)? n
Would you like to unregister your name? (y/n)? y
Your name has been unregistered.
```

Part 2: Copy Hw3 as Lab4_2 and modify this to use exceptions

Copy your project for Hw3 into a project called Lab4_2

In package `juicer` of project Lab4_2 create a new custom exception class called `IllegalMassException`. This class must be an unchecked exception. An instance of this class is meant to be thrown to indicate that a mass of zero or less was encountered. Give this class a private non-static data field of type `double` for storing the invalid mass amount. Create the public constructor `IllegalMassException(double mass)` that initializes the `mass` data field to the parameter `mass`. Finally, add a public get method for the `mass` data field.

In Fruit.java (in package `juicer` of project Lab4_2) do the following:

- Modify constructor `protected Fruit(double theMass)` to throw an `IllegalMassException` if `theMass` is less than or equal to zero. Do this instead of setting `mass` to 1 for this case. Pass `theMass` to this constructor call (i.e., `throw new IllegalMassException(theMass);`).
- Modify the set method `setMass(double value)` to throw an `IllegalMassException` if `value` is less than or equal to zero. Do this instead of setting `mass` to 1 for this case. Pass `theMass` to this constructor call (i.e., `throw new IllegalMassException(theMass);`).

Modify `TestFruit` (in package `test` of project Lab4_2) by adding code that tests your modified version of this project. Write code to test and make sure your modifications to the `Fruit` class throw an instance of `IllegalMassException` when they are supposed to.

When you are done zip up your Lab4_1 and Lab4_2 projects in files called Lab4_1.zip and Lab4_2.zip respectively. Then in canvas click on Assignments, go to Lab 4, and then upload these .zip files here as one submission.