Introduction to Java

CS9053

Tuesday 6 PM – 8:30 PM

Prof. Dean Christakos

Oct. 5th, 2022

Due: October 12th, 2022 11:55 PM

Part I: Exceptions

1. Go back to the Employee Hierarchy code from Assignment 4. As long as you got the basics right from the first assignment, you should be all set with the code you’ve already written. I’ve put in the basic employee hierarchy, but you should paste in your own code. Looking at the code, a few simple rules should be obvious:
   * For hourly employees, hourly rate must be greater than minimum wage (7.25 in the USA) and the number of weekly hours must be greater than or equal to zero
   * For salaried employees, the annual salary must be greater than 15,080 (52 weeks of work at $7.25/hr for 40 hours of work per week.
   * For managers, the executive rank must be greater than or equal to zero

Given this, if one tries to create a Employee object (or its subclasses), then we should raise a EmployeeException.

1. Create a EmployeeException class
2. Ensure that the constructor accepts a String argument which is passed up to the Exception superclass
3. Throw a EmployeeException when any of the three “illegal” conditions are raised.
4. The EmployeeException should have a message explaining the error. (eg, “weeklyHours cannot be negative”)
5. Take the following code, ListOfNumbers.java:

import java.io.\*;

import java.util.List;

import java.util.ArrayList;

public class ListOfNumbers {

private List list;

private String inFile;

public ListOfNumbers () {

// create an ArrayList of Pairs of Integers

}

public List getList() {

return this.list;

}

public void createList() {

for (int i = 0 ; i< 100 ; i++) {

Integer number1 = (int) (Math.*random*()\*10000);

Integer number2 = (int) (Math.*random*()\*10000);

// fill the existing list with Pair objects

// of two numbers.

}

}

public ListOfNumbers (String inFile) {

this();

this.inFile = inFile;

}

public void readList() {

}

public void writeList() {

PrintWriter out = null;

try {

System.*out*.println("Entering try statement");

out = new PrintWriter(new FileWriter("outFile.txt"));

for (int i = 0; i < list.size(); i++)

out.println(list.get(i).getKey() + " " + list.get(i).getValue());

} catch (IndexOutOfBoundsException e) {

System.*err*.println("Caught IndexOutOfBoundsException: " +

e.getMessage());

} catch (IOException e) {

System.*err*.println("Caught IOException: " + e.getMessage());

} finally {

if (out != null) {

System.*out*.println("Closing PrintWriter");

out.close();

} else {

System.*out*.println("PrintWriter not open");

}

}

}

}

You’re going to do a couple of things:

1. You can see the class “Pair”. Now, this takes two Objects, a key and a value. Like ArrayList, it’s parameterized. So you can have a Pair with a key of a String and a value of an Integer, like Pair<String, Integer>, or a Pair of integers where they key is an Integer and the value is an integer, such as Pair<Integer, Integer>. You would access each item of the pair with getKey() and getValue().

For example, I could create a pair of 5 and 6 like so:

Pair<Integer, Integer> p = new Pair<Integer, Integer>(5,6);

Here, p.getKey() would be 5 and p.getValue() would be 6.

What you’re going to do first is have the field pairList be an ArrayList of Pairs, properly parameterized (there should be no warnings associated with ArrayList in the code).

Next, you’re going to implement createList. Currently in createList, you can see that it generates two random integers between 0 and 9999. You’re going to take each pair of integers and put them in a Pair object, and then add that Pair object to the ArrayList called pairList.

So at this point pairList should have 100 Pair objects, where each object contains a Key and a Value of random integers. Once you’ve done this, the method writeList should compile correctly without errors (you shouldn’t have to modify that code directly for the errors to go away).

1. Add a readList method to ListOfNumbers.java. This method should re-initialize the pairList field with a new, empty ArrayList, read in int values from a file, print each value, put the pair of numbers in each line in a Pair object, and append them to the end of pairList. You should catch all appropriate errors. You will read from the text file numberfile.txt.

There’s a trick when reading in data that you want to split up. If you read in a line, it will contain two numbers separated by a space, and you will have a String that looks like “5 6”. Call it line, which is a String object. If you execute the method line.split(), it will return an array of Strings such that if you have String[] nums = line.split(), then nums[0] will be the String “5” and nums[1] will be the String “6”. Convert those Strings to Integers and use those integers in the constructor to your Pair object, and add the Pair object to the ArrayList.

The writeList method writes out the contents of the ArrayList to outFile.txt.

1. Modify the following cat method so that it will compile.

public static void cat(String fileName) {

RandomAccessFile input = null;

String line = null;

try {

input = new RandomAccessFile(file, "r");

while ((line = input.readLine()) != null) {

System.out.println(line);

}

return;

} finally {

if (input != null) {

input.close();

}

}

}

1. In the assignment project is a file called employees.txt. This has a list of employees types, followed by a set of comma-separated values corresponding to the arguments you would use.

You’re going to read each line for each employee and call createEmployee, which will create one of the available employees, HourlyEmployee, IndividualContributor, or Manager and returns an Employee, which you will add to the ArrayList called employeeList. Here, you can use the split() method with the argument split(“,”) to split a single string into an array of strings, split by the comma.\*

\*yes, you are very correct for realizing that the flaw in the code design here is that Ids are automatically generated but that managerId is set with the assumption of a know already-existing Id, requiring managers in the text file to be ordered such that their Ids can be assumed based on that order. You are very clever.

The file has some unavailable employee types. If the employee is unavailable, createEmployee should throw a EmployeeException. You should catch a EmployeeException and continue reading the file.

By the end, the size of employeeList should be 17

Summary:

* Create a EmployeeException class
* Implement createEmployee to return the appropriate employee depending on the input string
* createEmployiee should throw an EmployeeException if it is not an HourlyEmployee, IndividualContributor, or Manager
* A loop should read in the employees.txt file line-by-line
* If the file cannot be read, you should break out of the loop
* If you get a EmployeeException, you should continue reading the file