CICS-160 Spring 2023

Assignment 4: Using lists in Java

Due on Sunday May 7

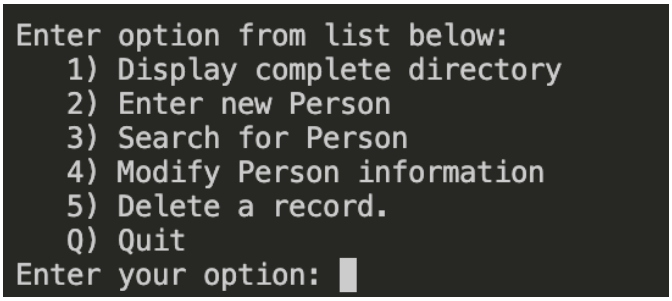
Learning Goals:

This assignment is designed to for you to demonstrate the ability to use lists in Java. We will do that by implementing the system you started to design for assignment #2.

Overview

For this project, you will be implementing a system that will present a user with a menu-driven system for adding information about people. This will be the system you designed for assignment 2, which is not exactly the same as the one you implemented in assignment 3 (similar, but not identical). If you want, you can use, as a starting point, any idea/concept/code we have used or talked about in class for those classes. Your completed project will have the following classes: Person, Employee, Students, Persons, and MainProgram.

The behavior of the system, when completed, will be as follows. The user will be presented with a menu of options from which to choose, each one of them implementing an operation on a list. The operations are: entering a new record; listing all records; displaying a particular record; modifying a particular record; and deleting records. A possible menu might look as displayed in the following image. Your code will start by validating the input entered by the user, asking them to enter another option if the value they enter does not match any of the options presented. All throughout the design, whenever letters are to be entered by the user, the program should behave correctly regardless of that input being uppercase or lowercase.



The operations indicated in the menu should be performed by the following methods, using the indicated method names (otherwise the autograder will fail to find them).

All records, be them of type Person, Student, or Employee, will be stored inside an object of type Persons. Class Persons has two attributes: a list of objects of type Person; and a count of how many records are in the list. The list inside Persons can be either an ArrayList or a LinkedList, your choice.

1. create toString() methods for all classes so that they respond as needed when they are printed. toString() is the Java equivalent to Python's __str__(). The format of the printing can be any that correctly displays all of the attributes of all of the objects stored inside in the Persons object.

2. Implement a function MainProgram.enterNewPerson(Persons P, Scanner k). MainProgram.enterNewPerson() begins by asking the user for the information to be stored inside a Person object. After the basic data for a Person object is entered, the system asks the user if the person being entered is a student. If it is, data specific to a student is asked for. If it is not a student, the system asks if the person being entered is an employee. If the user answers positively, data specific to an employee is asked for. Finally, all collected data is stored in the Persons object that MainProgram.enterNewPerson() received as input by calling it add() method.

3. Calls Persons.search(s), which returns an object of type Persons that contains only those Person objects that have s as their name. It then prints the content of the Persons object that search(s) returns.

4. Asks for the name in the record we are trying to modify. Then calls Persons.search(s), and displays the matching Person objects, one at a time, asking the user if they want to modify that record. If the user answers 'y' or 'Y', gets input from the user and then calls methods of class Person in order to

do we need to ask which attributes user wants to modify

modify the object. Assume that only attributes of class Person need to be modified (i.e. you do not need to provide for the modification of attributes added in Employee or in Student).

5. Asks for the index of the record to delete. It then displays that index, and asks the user if they want to delete it. If the user answers 'y' or "Y", Persons.delete(i) is run, which deletes that object from the Persons object. If the index entered is invalid, prints an error message and deletes nothing.

Once the selected operation has been performed, the menu is re-displayed, and the process continues until the user selects to quit.

Your tasks

Based on the operations listed above, implement the following list of methods for classes Person, Student, Employee, MainProgram, and Persons.

Person: (13 points total)

Person(String newName, String address, String phone)	5 points
setName(String name)	2 point
getName()→ string	2 point
getPhone()→ string	2 points
toString()→ string	2 points

Student: (11 points total)

Student(String newName, String address, String phone, int year)	5 points
toString()→ string	3 points
getGraduationYear()→ int	1 point
setGraduationYear(int year)	2 point

Employee: (11 points total)

Employee(String newName, String address, String phone, String department)	5 points
toString()→ string	3 points
getDepartment()→ string	1 point
setDepartment(String department)	2 point

Persons: (25 points total)

Persons()	1 point
search(String name)→Persons	15 points
add(Person/Student/Employee p)	2 points
getSize() → int	2 points
delete(int i)	5 points

IMPORTANT NOTE: YOUR PERSONS CLASS MUST HAVE A METHOD CALLED getInternalList() WHICH RETURNS THE LIST OF PERSON OBJECTS THAT IT STORES. THIS METHOD IS NOT CALLED BY THE REST OF YOUR CODE, BUT IT IS NECESSARY FOR THE AUTOGRADER TO WORK CORRECTLY.

Things that will be checked manually: (40 points total)

Program has the complete functionality described above	
correct menu operation	10 points
correctly adds records	5 points
correctly searches for records	5 points
correctly modifies records	5 points
correctly deletes records	5 points
Class Employee and class Student inherit from Person.	5 points
Employee and Person do not implement things they can inherit from Person.	5 points.

Submit your work in five files: Person.java, Student.java, Employee.java, MainProgram.java, and Persons.java.

Extra credit:

Code includes tests for Persons.delete()	5 points.
Code includes tests for Persons.insert()	5 points