Alexander Laudino

CSC 161-04

Dr. Farrett

Lab Assignment 10 – Employee Class

**Pseudo-code**

This program creates three Employee objects to hold the following data:

Name ID Number Department Position

Susan Meyer 47899 Accounting Vice President

Mark Jones 39119 IT Programmer

Joy Rogers 81774 Manufacturing Engineer

The program prompts the user to validate the data prior to writing it to a file and then asks the user to confirm the data is correct by reading the data from the file and displaying the results to the screen.

EmployeeDemo

Begin

1. Create empty ArrayList of Employee objects as employees
2. Employee.setInput() prompts user to add data automatically or enter it manually, autoAddData variable set to false if manually
3. If Employee.getInput() is true, equals autoAddData is true
4. Add new Employee objects with parameters included to employees ArrayList
5. Print “Number of employees to add: “ + employees.size()
6. Print “Preparing to write data to file.”
7. Else
8. Employee.inputData(employees) prompts user to enter data manually
9. If employees contains elements
10. Boolean dataNotValidated = true
11. Do-while loop to validate data
12. If employees.size() > 0
13. Boolean dataValidation = Employee.confirmData(employees) calls method to confirm
14. data, returns true if data is validated, false if data not validated
15. If dataValidation = true
16. dataNotValidated = false, end do-while loop
17. Print “Data validated”
18. Else
19. employees.clear() to remove all elements from ArrayList
20. Print “Please re-enter data.”
21. Employee.inputData(employees) to manually add data
22. Else employee ArrayList is empty
23. Print “No employees to be added.”
24. Close Scanner object
25. Exit program
26. End do-while loop when dataNotValidated == false
27. Try opening file
28. Employee.openFile()
29. Print “File opened.”
30. Catch IOException
31. Print “An error occurred while trying to open the file.”
32. Try writing to file
33. Employee.writeToFile(employees)
34. Print “Data written to file.”
35. Catch IOException
36. Print “An error occurred while writing to the file.”
37. Try reading file
38. Employee.readFile()
39. Print “Data verified”
40. Catch Exception
41. Print “An error occurred while reading the file.”
42. Finally
43. Print “Employees added to the database.”
44. Close Scanner object
45. Exit program
46. If employees does not contain any elements
47. Print “No employees added to database.”
48. Close Scanner object
49. Exit program

End

These are the variables, constructors, and methods that the program depends on to create new Employee objects, take user input, verify data, opening, writing, and reading a file, handling exceptions, and mutating and accessing the objects attribute fields.

Employee

// ----- VARIABLES -----

public static final Scanner input = new Scanner(System.in); For methods to use for getting user input

private static boolean autoAddData = true; For adding data automatically

private static String fileName; Name of file

private String name; Employee Name

private int idNumber; Employee ID Number

private String department; Employee Department

private String position; Employee Position

// ----- CONSTRUCTORS -----

Begin Employee(String name, int idNum, String dept, String pos)

1. this.name = name;
2. this.idNumber = idNum;
3. this.department = dept;
4. this.position = pos;

End

Begin Employee(String name, int idNum)

1. this.name = name;
2. this.idNumber = idNum;
3. this.department = “”;
4. this.position = “”;

End

Begin Employee()

1. this.name = “”;
2. this.idNumber = 0;
3. this.department = “”;
4. this.position = “”;

End

// ----- METHODS -----

Begin setInput()

1. int num = 0; used to hold user input
2. continueInput = true; for ending user input
3. Do-while loop for user input validation
4. Try
5. Print “To input data automatically, enter 1. To input data manually, enter 2: ”
6. num = input.nextInt(); prompts user for input
7. If (num == 1 || num == 2), continueInput = false; end do-while loop
8. Else
9. Print “Try again. (Incorrect input: 1 or 2 is required)”
10. Continue do-while loop
11. Catch InputMismatchException if user input is not an integer
12. Print “Try again. (Incorrect input: an integer is required.)”
13. input.nextLine(); discard input
14. End do-while loop when continueInput == false
15. If num == 2, autoAddData = false
16. input.nextLine(); discard input

End

Begin getInput()

1. return autoAddData

End

Begin inputData(ArrayList<Employee> employees)

1. continueInput = true; for ending user input
2. numOfNewEmp = 0; counter for number of employees added
3. Do-while loop for adding new employee
4. Print “Add new employee? (Y/N): “
5. newEmp = input.nextLine(); prompt user for input
6. If newEmp == Y
7. Print “Enter employee name: “
8. name = input.nextLine(); prompt user for input
9. Print “Enter employee ID Number: “
10. idNum = input.nextInt(); prompt user for input
11. input.nextLine(); // discard input
12. Print “Enter employee department: “
13. dept = input.nextLine(); prompt user for input
14. Print “Enter employee position: “
15. pos = input.nextLine(); prompt user for input
16. employees.add( new Employee(name, idNum, dept, pos) ); creates new employee object
17. and adds it to employees ArrayList
18. numOfNewEmp++;
19. Else if newEmp == N && numOfNewEmp > 0
20. Print “Number of employees to add: “ + numOfNewEmp
21. Print “Preparing to write data to file.”
22. continueInput = false; end do-while loop
23. Else if newEmp == N && numOfNewEmp == 0
24. Print “Continue without adding any employees? (Y/N): “
25. noEmp = input.nextLine(); prompt user for input
26. If noEmp == Y; continueInput = false; ends do-while loop
27. Else if noEmp == N; continue loop
28. Else; print “Try again. (Incorrect input: Please enter Y or N)”
29. Else; print “Try again. (Incorrect input: Please enter Y or N)”; continue loop
30. End do-while loop when continueInput == false

End

Begin confirmData(ArrayList<Employee> employees)

1. Print “Please confirm data entered is correct.”
2. Print formatted table heading “Name IDNumber Department Position”
3. For loop to print horizontal line heading divider
4. For employee object in employees Arraylist
5. Print formatted “employee.getName(), employee.getIDNumber, employee.getDepartment,
6. employee.getPosition()”
7. confirmWrite = “null”; for user input
8. continueInput = true; for ending user input
9. Do-while loop to prompt user to confirm writing data to file
10. Print “Write data to file? (Y/N): “
11. confirmWrite = input.next(); prompt user for input
12. If confirmWrite == Y || confirmWrite == N
13. continueInput = false; to end do-while loop
14. Else print “Try again. (Incorrect input: Enter Y or N)”
15. End do-while loop when continueInput == false
16. input.nextLine(); discard input
17. if confirmWrite == Y; return true
18. else; return false

End

Begin getFileName()

1. Print “Enter name of text file to store data: “
2. file = input.nextLine(); prompt user for input
3. setFile(file); call setFile method to assign string value of file to fileName attribute

End

Begin setFile(String file)

1. fileName = file

End

Begin getFile()

1. return fileName

End

Begin openFile()

1. getFileName(); to get filename
2. File file = new File(getFile()); create new File object with fileName value as argument
3. If file.exists()
4. Print “File already exists. Do you want to overwrite the existing file? (Y/N): “
5. overwrite = input.nextLine(); prompt user for input
6. If overwrite == Y
7. file.createNewFile(); create new file over existing file
8. Print “Overwriting file: “ + file.getName()
9. Else; openFile(); calls self recursively until file is created
10. Else if file.createNewFile() == true; creates new file and returns true if file is created
11. Print “File created: “ + file.getName()

End

Begin writeToFile(ArrayList<Employee> employees)

1. Print “Writing data to file.”
2. PrintWriter writer = new PrintWriter(getFile()); creates new Printerwriter object
3. Write formatted table heading “Name, ID Number, Department, Position”
4. writer.flush()
5. for loop to write horizontal line table heading divider
6. writer.flush()
7. for employee in employees ArrayList
8. write formatted data ( employee.getName(), employee.getIDNumber(),
9. employee.getDepartment(), employee.getPosition() )
10. writer.flush()
11. writer.close()

End

Begin readFile()

1. continueInput = true; for ending user input
2. Do-while loop to prompt user to verify data
3. Print “Verify data written to file? (Y/N): “
4. dataVerification = input.next(); prompt user for input
5. If dataVerification == Y
6. Print “Reading data from file.”
7. FileReader reader = new FileReader(getFile()); create new FileReader object
8. char[] a = new char[1000]; create character array to hold FileReader input
9. reader.read(a); reads data from file and stores in character array
10. For character in character array
11. Print character
12. reader.close()
13. continueInput = false; to end do-while loop
14. Else if dataVerification == N
15. Print “Data not verified.”
16. continueInput = false; to end do-while loop
17. Else print “Try again. (Incorrect input: Enter Y or N)”
18. End-do while loop when continueInput == false

End

Begin setName(String name)

1. this.name = name

End

Begin getName()

1. return name

End

Begin setIDNumber(int idNum)

1. this.idNumber = idNum

End

Begin getIDNumber()

1. return idNumber

End

Begin setDepartment(String dept)

1. this.department = dept

End

Begin getDepartment()

1. return department

End

Begin setPosition(String pos)

1. this.position = pos

End

Begin getPosition()

1. return position

End

**UML**

|  |
| --- |
| EmployeeDemo |
| +employees : ArrayList |
|  |

|  |
| --- |
| Employee |
| +input : Scanner  -autoAddData : Boolean  -fileName : String  -name : String  -idNumber : int  -department : String  -position : String |
| +Employee(name : String, idNum : int, dept : String, pos : String)  +Employee(name: String, idNum : int)  +Employee()  +setInput() : void  +getInput() : Boolean  +inputData(employees : ArrayList<Employee>) : void  +confirmData(employees : ArrayList<Employee>) : boolean  +getFileName() : void  -setFile(file : String) : void  -getFile() : String  +openFile() : void  +writeToFile(employees : ArrayList<Employee>) :void  +readFile() : void  -setName(name : String)  +getName() : String  -setIDNumber(idNum : int)  +getIDNumber() : int  -setDepartment(dept : String)  +getDepartment() : String  -setPosition(pos : String)  +getPosition() : String |

**Requirements**

Exception handling was designed into the program to handle any exceptions that would be thrown from file input and output.