COP3503 Project 3 – Work Order Generator

Objectives:

- Show an understanding of how to
    - Create class files that serve as templates for creating objects
    - Setup inheritance and implement interfaces
    - Read UML diagrams and implement the classes and methods outlined in the diagram

Submission Requirements:

- Submit your project via the Canvas course
    - Upload a zip file containing your 8 class files specified below
    - The zip file should be named Project3_N#.zip
        - For Example: Project3_N00123456.zip
- Project requires 1 zip file to be submitted

Design Specification Requirements: 100 points

Use the Project 3 UML Diagram as an outline for the project.

1. Display the project title followed by a blank line
2. 20 points: Display "Loading Employee Data" to console
    a. Use the logger method in FileHandler to write the String "Loading Employee Data" to the log file
    b. Then use the readEmployee method in the FileHandler class to load the data from the employee file
        i. Ignore the header in the csv file
    c. Employee objects must be created for each row in the employee file and then loaded into the Employee ArrayList in the Project3 class file
3. 25 points: Display "Loading Ticket Data" to console
    a. Use the logger method in FileHandler to write the String "Loading Ticket Data" to the log file
    b. Then use the readTicket method in the FileHandler class to load the data from the ticket file
        i. Ignore the header in the csv file
    c. Ticket objects along with their respective Customer objects need to be created for each row in the ticket file then load the Ticket objects into the Ticket ArrayList in the Project3 class file

4. 25 points: Display “Creating Work Orders” to console
    a. Use the logger method in FileHandler to write the String “Creating Work Orders” to the log file
    b. Use the createWorkOrders method to iterate of the list of Tickets and Employees to create WorkOrders for each Ticket
        i. Call the constructor method in the WorkOrder class to create work orders for each ticket in the list
            1. The WorkOrder objects created need to be loaded into the ArrayList in the Project 3 class file
            2. The createdAt variable for the WorkOrder objects must be assigned the current date and time for when the work order was created.
        ii. The number of employees is less than the number of tickets. You will need to dynamically determine how many tickets need to be assigned to each employee
            1. For example, if we have 100 employees and 1000 tickets each employee should be assigned 10 tickets
            2. You do not need to account for having less tickets than employees
5. 30 points: Display “Writing Work Order Data to File” to console
    a. Use the logger method in FileHandler to write the String “Writing Work Order Data to File” to the log file
    b. Create a new csv file called workorder_data.csv and write out the data for each WorkOrder object in the workOrderList to the file
        i. Use the witeDate method in the FileHandler class to iterate of the list of WorkOrders
        ii. Create a header for the csv file denoting what each row is storing
        iii. Use the getFileData method from the WorkOrder class to get the String containing all the data for the work order then write that String to the csv file
            1. The returned String should be comma separated
            2. The getFileData method in WorkOrder should rely on the getFileData methods of the other classes (Employee, Ticket, & Customer)
    c. Use the FileHandler logger method to write out data to the log file for each WorkOrder object created
        i. Use the getInfo method from the WorkOrder class to get the String containing all the data for the work order then write that String to the log file
        ii. The getInfo method in WorkOrder should rely on the getInfo methods of the other classes (Employee, Ticket, & Customer)
6. Display “Work Orders created. Program Exiting” to the console
    a. Use the logger method in FileHandler to write the String “Work Orders created. Program Exiting” to the log file

Note: Refer to the sample output in the **Example Output** section below.

Minimum Requirements: - 100 points

- All 8 class files need to be created and all methods and variables shown in the UML diagram must be present

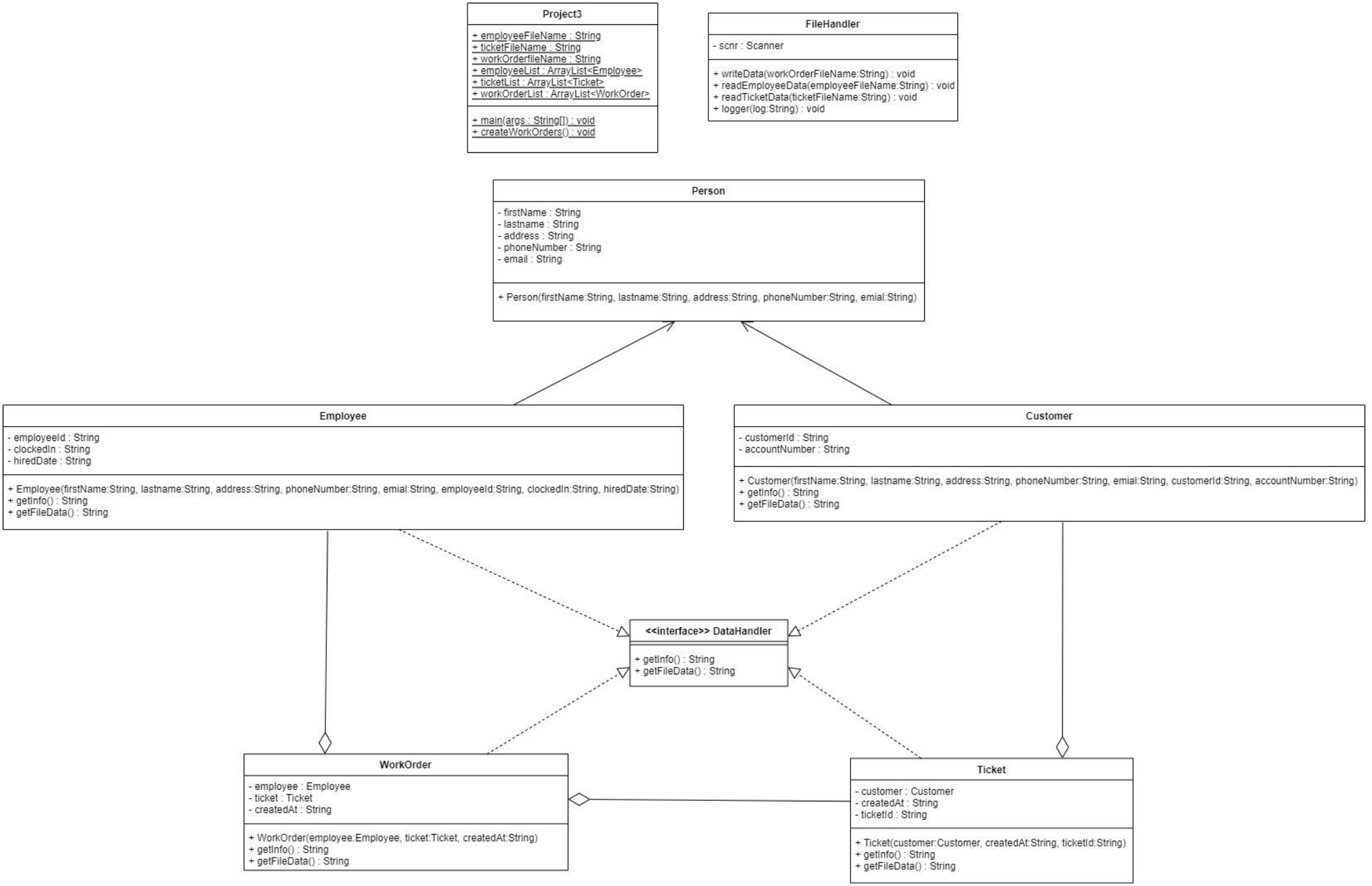
Additional Notes:

- Each class should have getter and setter methods for each class variable. This is not shown in the UML diagram but is required for the project
- Each method and variable outlined in the UML diagram must be implemented in the project. Additional methods and variables can be added if needed.
- Surround each input and output stream with a try catch
- The String class variables in Project3 need to be used to store the name of the files to be read in and created (employee_data.csv, ticket_data.csv, workorder_data.csv)
    - You can assume the files will be placed in the project's directory
- The FileHandler method logger must be used to write to the log.txt file for the project
    - This method should be set to append to the log file, not overwrite it every time it is executed
    - Each entry to the log file needs to include the date and time the entry was written to the log file
    - Refer to the example log file to determine correct format of the log file
- When outputting the work orders refer to the example workorder_data.csv for the correct format of the data and the required header
- Refer to the getInfo & getFileData output file to see an example String of what should be returned when calling those methods
- Use the @Override notation when implementing the interface methods in the classes
- Use the FileReader class wrapped with the Scanner class to read data in from file
- Use the FileWriter class wrapped with the PrintWriter class to output data to the new file
    - Use the println() method to write to the file line by line
- You do not need to verify the files exist before reading in the data
- Use the SimpleDateFormat class along with the Date class to get the current date & time
- Follow the commenting and programming guidelines outlined in the Commenting and Programming guide documents
    - Failure to do so will result in up to a 25-point reduction

Example Output:

```
Project 3 Work Order Generator

Loading Employee Data
Loading Ticket Data
Creating Work Orders
Writing Work Order Data to File
Work Orders Created. Program Exiting
```


Project3
+ employeeFileName : String
+ ticketFileName : String
+ workOrderfileName : String
+ employeeList : ArrayList<Employee>
+ ticketList : ArrayList<Ticket>
+ workOrderList : ArrayList<WorkOrder>
+ main(args : String[]) : void
+ createWorkOrders() : void
FileHandler
- scnr : Scanner
+ writeData(workOrderFileName:String) : void
+ readEmployeeData(employeeFileName:String) : void
+ readTicketData(ticketFileName:String) : void
+ logger(log:String) : void
Person
- firstName : String
- lastname : String
- address : String
- phoneNumber : String
- email : String
+ Person(firstName:String, lastname:String, address:String, phoneNumber:String, emial:String)
Employee
- employeeId : String
- clockedIn : String
- hiredDate : String
+ Employee(firstName:String, lastname:String, address:String, phoneNumber:String, emial:String, employeeId:String, clockedIn:String, hiredDate:String)
+ getInfo() : String
+ getFileData() : String
Customer
- customerId : String
- accountNumber : String
+ Customer(firstName:String, lastname:String, address:String, phoneNumber:String, emial:String, customerId:String, accountNumber:String)
+ getInfo() : String
+ getFileData() : String
<<interface>> DataHandler
+ getInfo() : String
+ getFileData() : String
WorkOrder
- employee : Employee
- ticket : Ticket
- createdAt : String
+ WorkOrder(employee:Employee, ticket:Ticket, createdAt:String)
+ getInfo() : String
+ getFileData() : String
Ticket
- customer : Customer
- createdAt : String
- ticketId : String
+ Ticket(customer:Customer, createdAt:String, ticketId:String)
+ getInfo() : String
+ getFileData() : String