Final Group Project Report

Employee Management System (EMS)

35

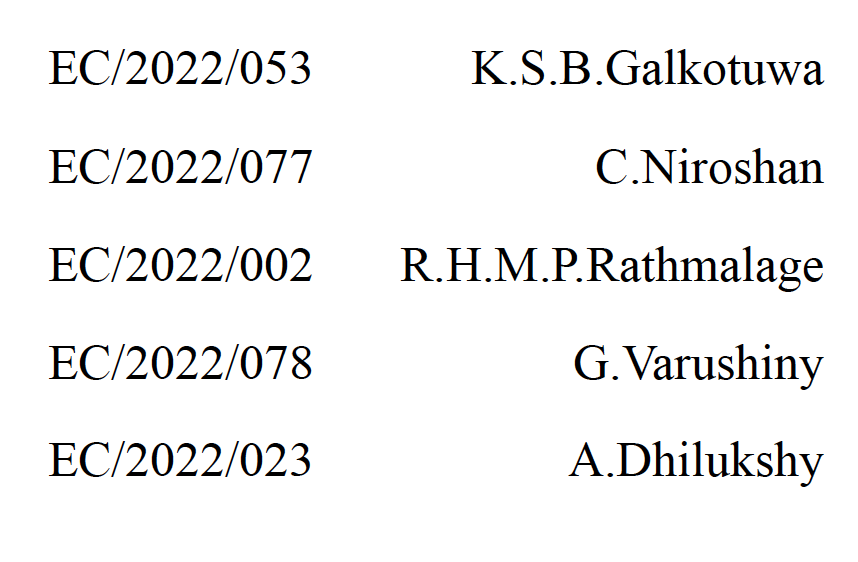


Table of Contents

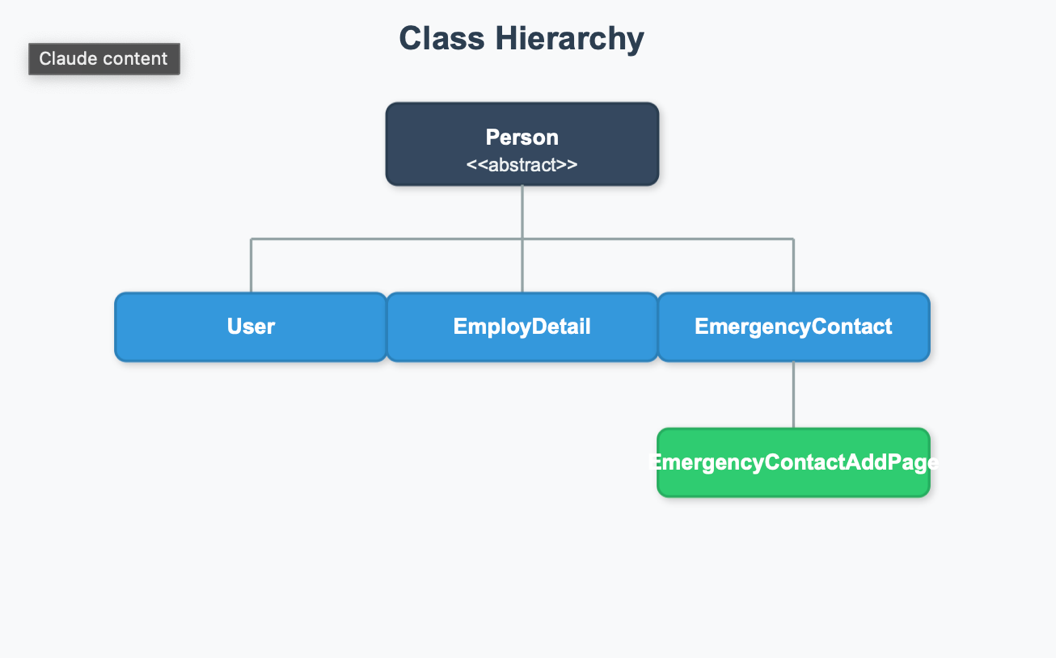
[Final Group Project Report 1](#_Toc188475091)

[Employee Management System (EMS) 1](#_Toc188475092)

[Introduction to the System 3](#_Toc188475093)

[Architecture 3](#_Toc188475094)

[ Core Classes Hierarchy 3](#_Toc188475095)

[ 4](#_Toc188475096)

[ Interface Implementations 4](#_Toc188475097)

[1. FileFunctions (Interface) 4](#_Toc188475098)

[4](#_Toc188475099)

[2. Calcultions <<Interface>> 5](#_Toc188475100)

[Key Components 6](#_Toc188475101)

[1. Authentication System 6](#_Toc188475102)

[ Implements user registration and login functionality 6](#_Toc188475103)

[ Stores user credentials in a text file (userdata.txt) 6](#_Toc188475104)

[ Provides basic security through username/password validation 6](#_Toc188475105)

[2. Employee Management 6](#_Toc188475106)

[ Adding Employees (Employee\_Add) 6](#_Toc188475107)

[- Collects comprehensive employee details 6](#_Toc188475108)

[- Generates HTML files for employee records 6](#_Toc188475109)

[- Creates separate emergency contact records 6](#_Toc188475110)

[- Calculates and stores salary information 6](#_Toc188475111)

[ Viewing Employees (Employee\_Show) 6](#_Toc188475112)

[- Opens HTML files using desktop browser 6](#_Toc188475113)

[- Provides formatted view of employee details 6](#_Toc188475114)

[ Updating Employees (Employee\_Update) 6](#_Toc188475115)

[- Supports in-place updates of HTML files 6](#_Toc188475116)

[- Maintains file structure during updates 6](#_Toc188475117)

[ Removing Employees (Employee\_Remove) 6](#_Toc188475118)

[- Removes employee HTML files 6](#_Toc188475119)

[- Clean deletion of records 6](#_Toc188475120)

[ Salary Management 6](#_Toc188475121)

[- Implements monthly and yearly salary calculations 6](#_Toc188475122)

[- Accounts for varying days in months 6](#_Toc188475123)

[- Generates detailed salary summaries in HTML format 6](#_Toc188475124)

[Implementation Details 8](#_Toc188475125)

[1. Data Validation 8](#_Toc188475126)

[ The system implements robust validation across multiple areas: 8](#_Toc188475127)

[- Email format validation using regex 8](#_Toc188475128)

[- Contact number format validation (9 digits) 8](#_Toc188475129)

[- Required field validation 8](#_Toc188475130)

[- Salary validation (must be greater than zero) 8](#_Toc188475131)

[2. HTML Generation 8](#_Toc188475132)

[ The system creates three types of HTML files for each employee: 8](#_Toc188475133)

[- Main employee details 8](#_Toc188475134)

[- Emergency contact information 8](#_Toc188475135)

[- Salary information 8](#_Toc188475136)

[ Each HTML file includes: 8](#_Toc188475137)

[- Consistent styling through external CSS 8](#_Toc188475138)

[- Interactive elements via JavaScript 8](#_Toc188475139)

[- Responsive design elements 8](#_Toc188475140)

[- Cross-linking between related files 8](#_Toc188475141)

[3. Exception Handling 8](#_Toc188475142)

[ The system implements comprehensive error handling for: 8](#_Toc188475143)

[- File operations 8](#_Toc188475144)

[- Invalid user input 8](#_Toc188475145)

[- Data validation 8](#_Toc188475146)

[- Authentication failures 8](#_Toc188475147)

[4. Directory Management 8](#_Toc188475148)

[- Automatic creation of required directories 8](#_Toc188475149)

[- Structured file organization 8](#_Toc188475150)

[- Consistent naming conventions 8](#_Toc188475151)

[Classes, Attributes, and Methods 9](#_Toc188475152)

[Use Cases of OOP Concepts 12](#_Toc188475153)

[Demonstration of the System 13](#_Toc188475154)

[Technical Strengths 14](#_Toc188475155)

[1. Modularity: Clear separation of concerns through interface implementation 14](#_Toc188475156)

[2. Extensibility: Abstract classes and interfaces allow for easy system expansion 14](#_Toc188475157)

[3. Data Persistence: HTML-based storage provides both data storage and presentation 14](#_Toc188475158)

[4. Input Validation: Comprehensive validation across all user inputs 14](#_Toc188475159)

[5. Error Handling: Robust exception handling throughout the system 14](#_Toc188475160)

[Areas for Improvement 14](#_Toc188475161)

[1. Security: Basic text-file based authentication could be enhanced 14](#_Toc188475162)

[2. Database Integration: Could benefit from proper database implementation 14](#_Toc188475163)

[3. GUI Interface: Currently console-based, could be enhanced with a graphical interface 14](#_Toc188475164)

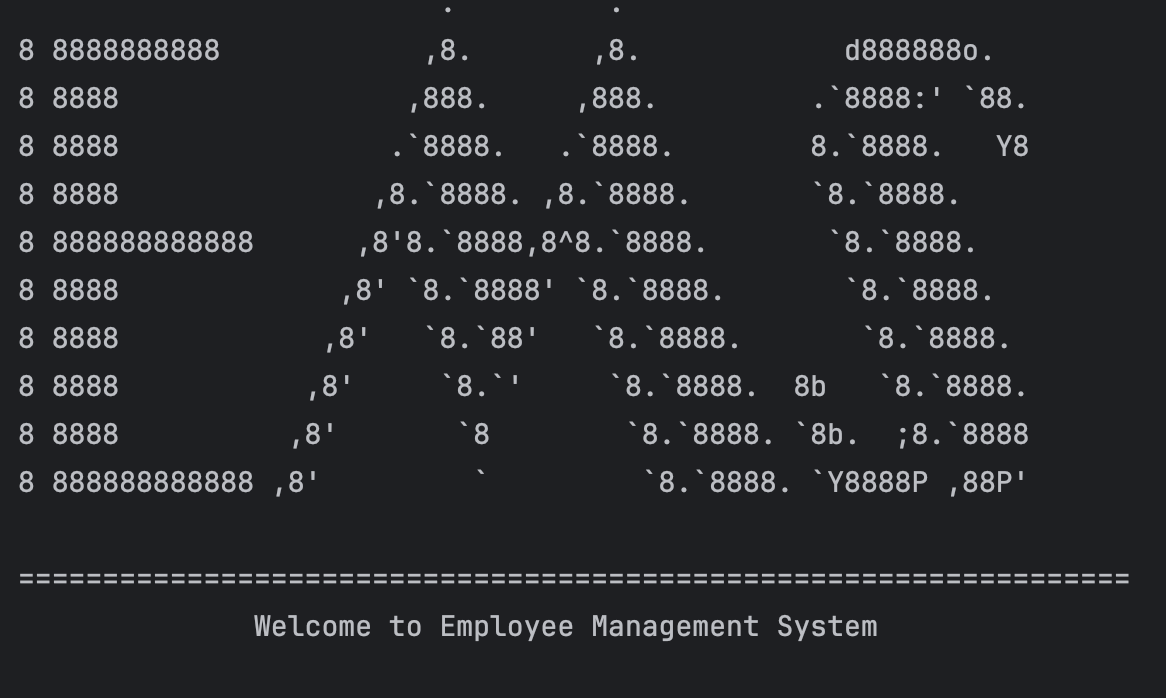
[4. Data Encryption: Sensitive data is stored in plain text 14](#_Toc188475165)

[5. Backup System: No built-in backup functionality 14](#_Toc188475166)

[Modifications from the Initial Plan 14](#_Toc188475167)

[Teamwork 15](#_Toc188475168)

[Conclusion 16](#_Toc188475169)



### Introduction to the System

The **Employee Management System (EMS)** is a user-friendly and efficient solution designed to help small and medium-sized businesses manage their employee records with ease. In today’s fast-paced world, keeping track of employee information is crucial for any organization, but traditional methods like paper-based systems or plain-text files can be time-consuming, error-prone, and difficult to maintain. The EMS solves these problems by using modern technologies and **Object-Oriented Programming (OOP)** principles to create a simple yet powerful platform for managing employee data.

At its heart, the EMS uses **HTML-based file storage** to store employee records in a neat and organized way. This means that all the information is not only easy to access but also looks professional and can be viewed in any web browser. This makes it a great tool for businesses that want to modernize their HR processes without needing complex software. The system is built using **Java**, a popular programming language known for its reliability and flexibility, making it a perfect choice for creating a system that works across different platforms.

The EMS has a **menu-driven console interface**, which means users can easily navigate through the system using simple text commands. With this interface, users can add new employees, view their details, update information, or remove records when needed. The system also includes some advanced features like **emergency contact management** and **salary calculation**. For example, the salary calculation feature automatically calculates monthly and yearly salaries, taking into account the number of days in each month, and presents the information in a clear and easy-to-understand HTML format.

One of the best things about the EMS is that it follows **OOP principles**, which make the system modular, scalable, and easy to maintain. Each part of the system is designed to handle a specific task, such as adding employees or calculating salaries. This modular design not only makes the system easier to understand but also allows for future improvements without much hassle.

The EMS is especially useful for small businesses or startups that need a simple and cost-effective way to manage employee records without investing in expensive database systems. By storing data in HTML files, the system is both **portable** and **easy to use**, as the files can be opened on any device with a web browser. Additionally, the system includes strong **data validation** and **error handling** features to ensure that the information entered is accurate and that the system runs smoothly even if something goes wrong.

In conclusion, the **Employee Management System** is a modern, efficient, and practical solution for managing employee records. It combines the power of Java programming with the simplicity of HTML-based storage to create a platform that is both user-friendly and effective. Whether you’re a small business owner or someone just starting to learn about computer science, the EMS is a great example of how technology can simplify everyday tasks and improve productivity. It’s a project that not only solves real-world problems but also demonstrates the importance of good design and planning in software development.

We would like to extend our heartfelt thanks to **Dr. B.M.T. Kumarika** for her invaluable guidance and support throughout the development of this project. Her expertise and dedication in conducting the **BECS 12243 module** have been instrumental in helping us understand the core concepts of Object-Oriented Programming and software development. This project would not have been possible without her mentorship and encouragement. Thank you, Dr. Kumarika, for inspiring us to push our limits and create something meaningful!

### Architecture

* Core Classes Hierarchy
* Interface Implementations

1. FileFunctions (Interface)

1. Calculations <<Interface>>

### Key Components

1. Authentication System

* Implements user registration and login functionality
* Stores user credentials in a text file (userdata.txt)
* Provides basic security through username/password validation

2. Employee Management

* Adding Employees (Employee\_Add)
  + Collects comprehensive employee details
  + Generates HTML files for employee records
  + Creates separate emergency contact records
  + Calculates and stores salary information
* Viewing Employees (Employee\_Show)
  + Opens HTML files using desktop browser
  + Provides formatted view of employee details
* Updating Employees (Employee\_Update)
  + Supports in-place updates of HTML files
  + Maintains file structure during updates
* Removing Employees (Employee\_Remove)
  + Removes employee HTML files
  + Clean deletion of records
* Salary Management
  + Implements monthly and yearly salary calculations
  + Accounts for varying days in months
  + Generates detailed salary summaries in HTML format

### File Structure

userdata.txt – store username and password in comma separate format

### Implementation Details

1. Data Validation

* The system implements robust validation across multiple areas:
  + Email format validation using regex
  + Contact number format validation (9 digits)
  + Required field validation
  + Salary validation (must be greater than zero)

2. HTML Generation

* The system creates three types of HTML files for each employee:
  + Main employee details
  + Emergency contact information
  + Salary information
* Each HTML file includes:
  + Consistent styling through external CSS
  + Interactive elements via JavaScript
  + Responsive design elements
  + Cross-linking between related files

3. Exception Handling

* The system implements comprehensive error handling for:
  + File operations
  + Invalid user input
  + Data validation
  + Authentication failures

4. Directory Management

* + Automatic creation of required directories
  + Structured file organization
  + Consistent naming conventions

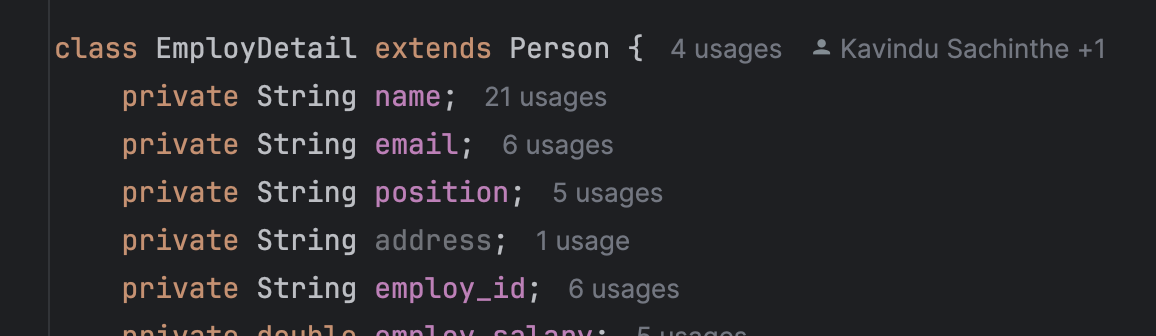
### Classes, Attributes, and Methods

The system is composed of several classes, each responsible for specific functionalities. Below is a breakdown of the key classes, their attributes, and methods

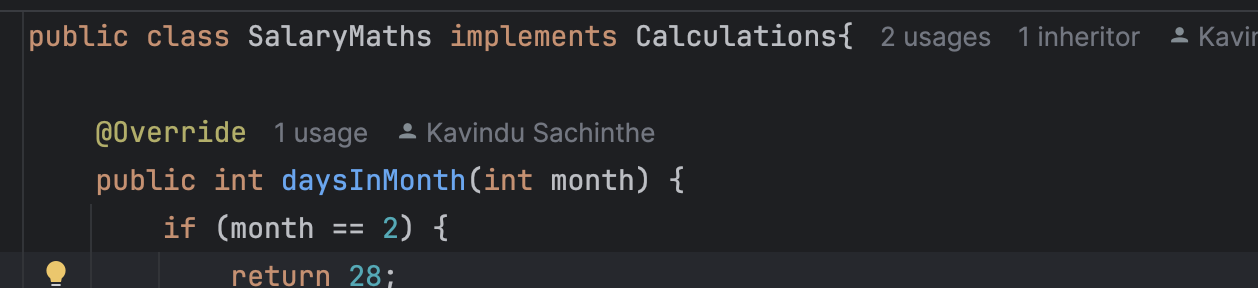
1. EntryPoint
   * Attributes:
     + USER\_DATA\_FILE
     + USER\_DATA\_DIR
     + EMERGENCY\_DATA\_DIR
     + SALARY\_DATA\_DIR
   * Methods:
     + main(): Entry point of the application, handles user registration and login.
     + createDirectories(): Creates necessary directories for storing employee data.
     + createUserDataFile(): Creates a file to store user credentials.
     + login(): Handles user login and navigation through the system.
     + register(): Handles user registration.
     + authenticateUser(): Validates user credentials during login.
2. MainMenu
   * Methods:
     + menu(): Displays the main menu of the system using ASCII art.
3. EmployDetail
   * Attributes:
     + Name
     + Email
     + Position
     + Address
     + employ\_id
     + employ\_salary
     + employ\_contact
     + emergencyContact\_name
     + emergencyContact\_contact
     + emergencyContact\_address
     + emergencyContact\_email
   * Methods:
     + getInfo(): Collects employee details from the user.
     + toString(): Returns a string representation of the employee details.
4. Employee\_Add
   * Methods:
     + createFileBaseHTML(): Creates an HTML file for the employee and writes their details into it.
     + listUserdataFiles(), removeFile(), viewFile(), updateFile(): Placeholder methods for file operations.
5. Employee\_Show
   * Methods:
     + viewFile(): Opens and displays the HTML file of a specific employee.
6. Employee\_Remove
   * Methods:
     + removeFile(): Deletes the HTML file of a specific employee.
7. Employee\_Update
   * Methods:
     + updateFile(): Updates specific details in an employee's HTML file.
8. EmergencyContact
   * Attributes:
     + Inherits from Person class: name, contactNo, email, address
   * Methods:
     + toString(): Returns a string representation of the emergency contact details.
9. EmergencyContactAddPage
   * Methods:
     + createFileBaseHTML(): Creates an HTML file for the emergency contact details.
10. SalaryCalculator
    * Methods:
      + generateSalarySummary(): Generates a yearly salary summary in HTML format for an employee.
      + getMonthName(): Returns the name of the month based on the month number.
11. SalaryMaths
    * Methods:
      + daysInMonth(): Returns the number of days in a given month.
12. FileFunctions
    * Attributes:
      + USER\_DATA\_FILE
      + USER\_DATA\_DIR
      + EMERGENCY\_DATA\_DIR
      + SALARY\_DATA\_DIR
    * Methods:
      + createFileBaseHTML(), listUserdataFiles(), removeFile(), viewFile(), updateFile(): Interface methods for file operations.
13. Person
    * Attributes:
      + Name
      + contactNo
      + email
      + address
    * Methods:
      + getInfo(), getAddress(), setAddress(), getContactNo(), setContactNo(), getEmail(), setEmail(), getName(), setName()
14. User
    * Attributes:
      + Name
      + password
    * Methods:
      + getName(), getPassword(), toString()
15. CodeExit
    * Methods:
      + out(): Displays an exit message and terminates the program.

### Use Cases of OOP Concepts

1. Inheritance:
   * The EmergencyContact class inherits from the Person class, reusing attributes like name, contactNo, email, and address.
   * The SalaryCalculator class inherits from the SalaryMaths class, reusing the daysInMonth() method.



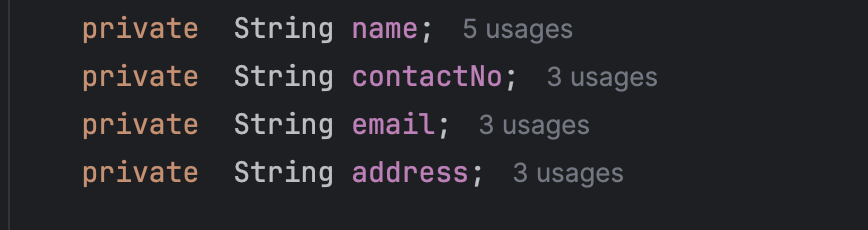


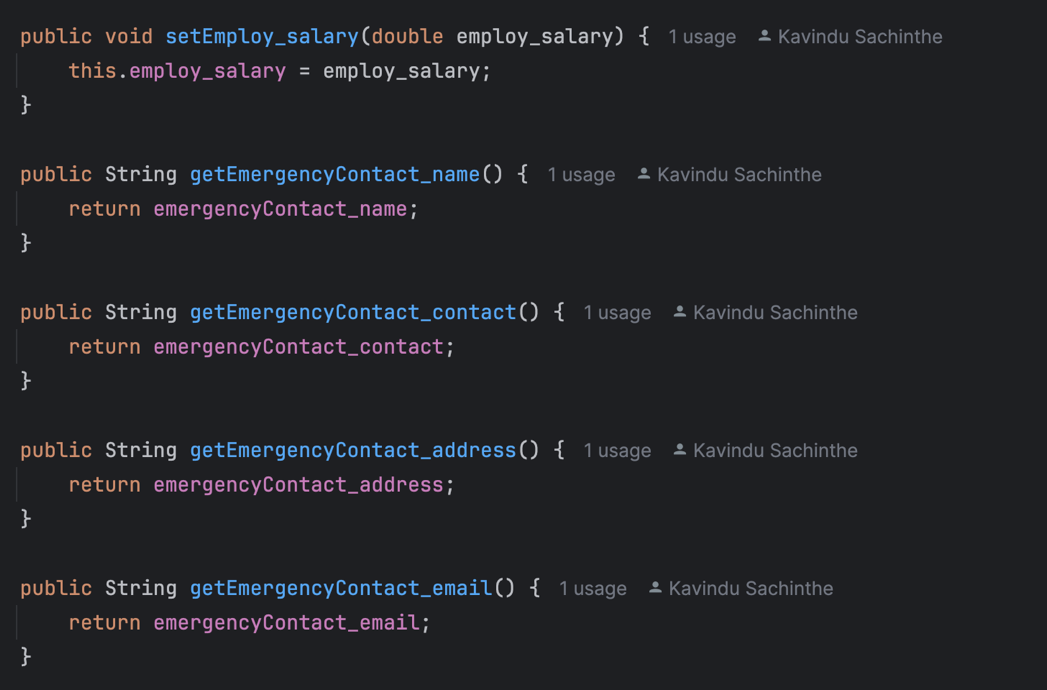


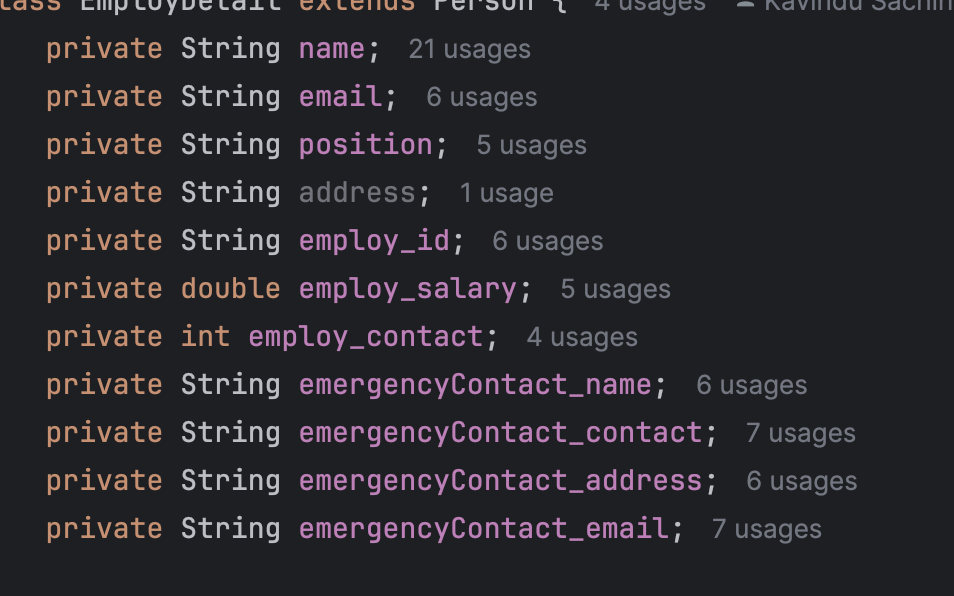
1. Polymorphism:
   * The toString() method is overridden in multiple classes (EmployDetail, EmergencyContact, User) to provide customized string representations of objects.





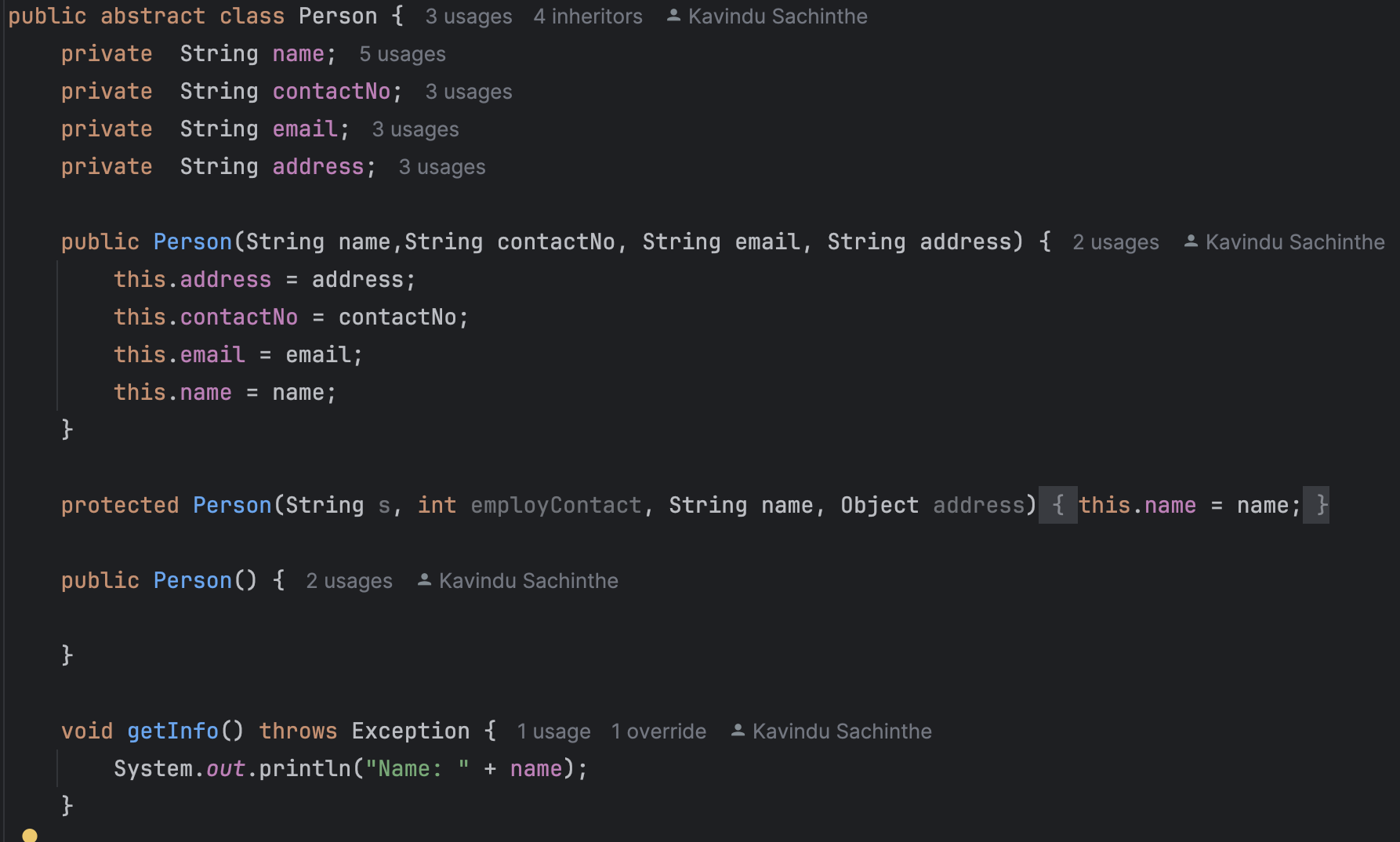
1. Encapsulation:
   * All attributes in the Person class are private, and access to them is controlled through public getter and setter methods.

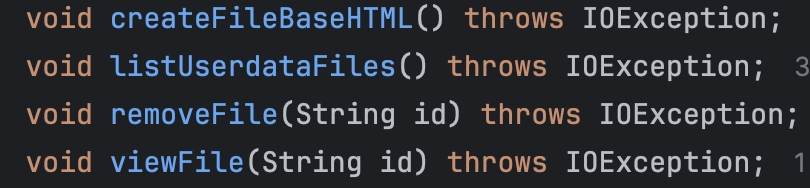




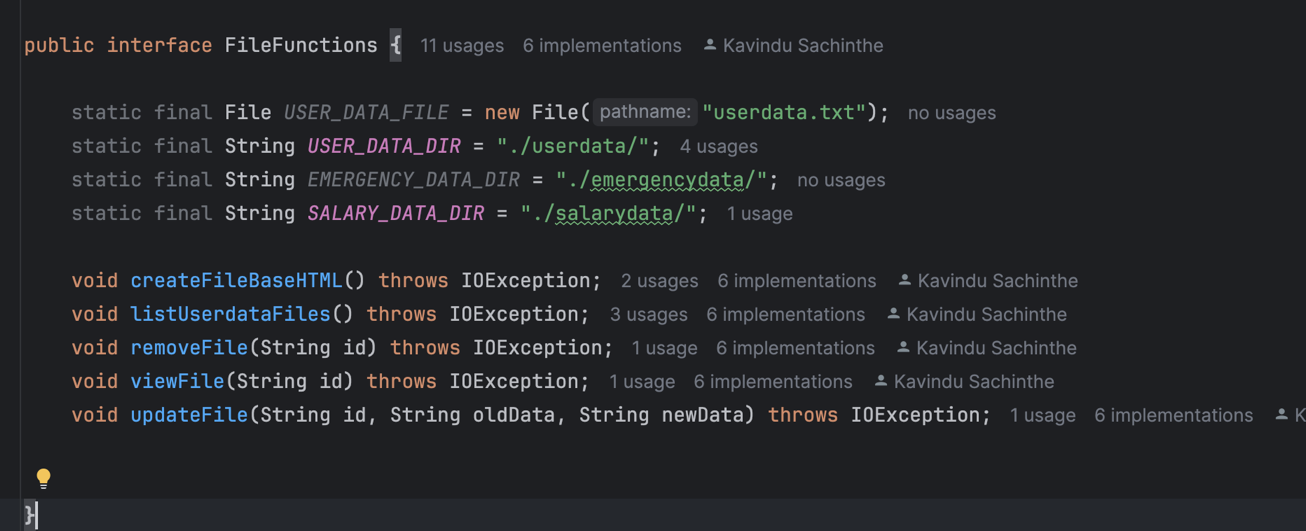


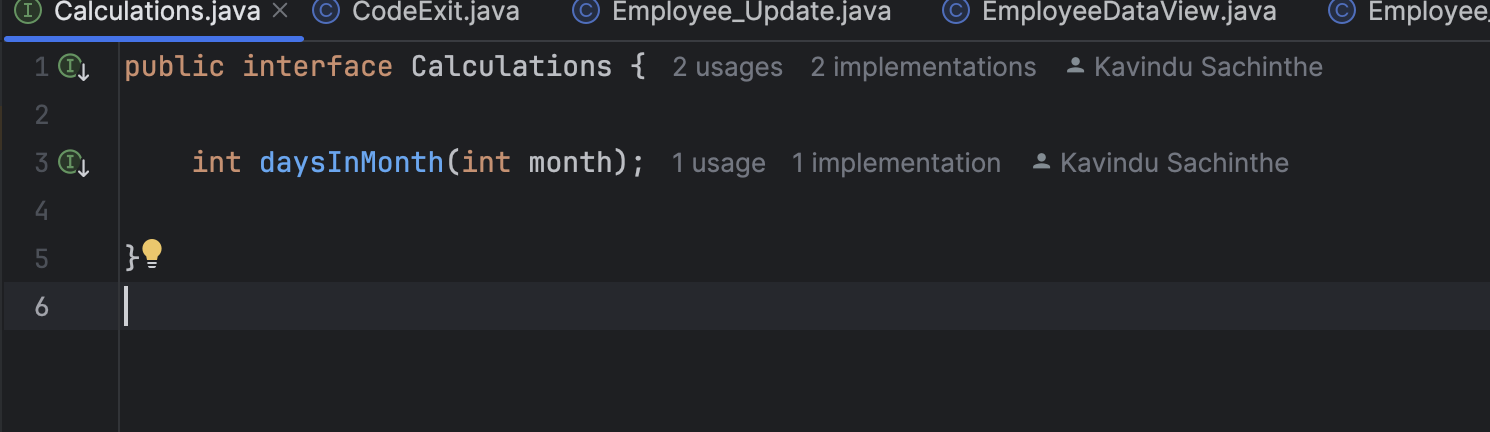
1. Abstraction:
   * The FileFunctions interface defines a contract for file operations, which is implemented by classes like Employee\_Add, Employee\_Remove, and Employee\_Show.

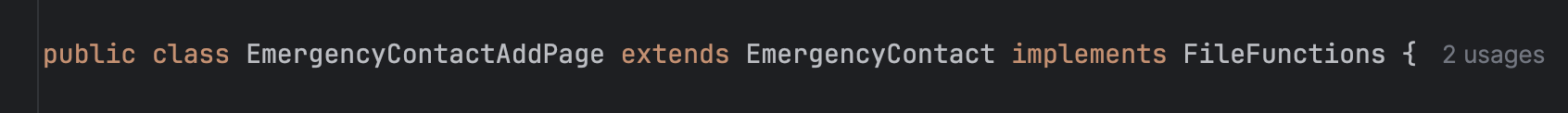




1. Interface:
   * The FileFunctions interface is implemented by multiple classes to ensure consistent file handling across the system.
   * The Calculations interface is implemented by multiple classes to calculate calculate months in the system.





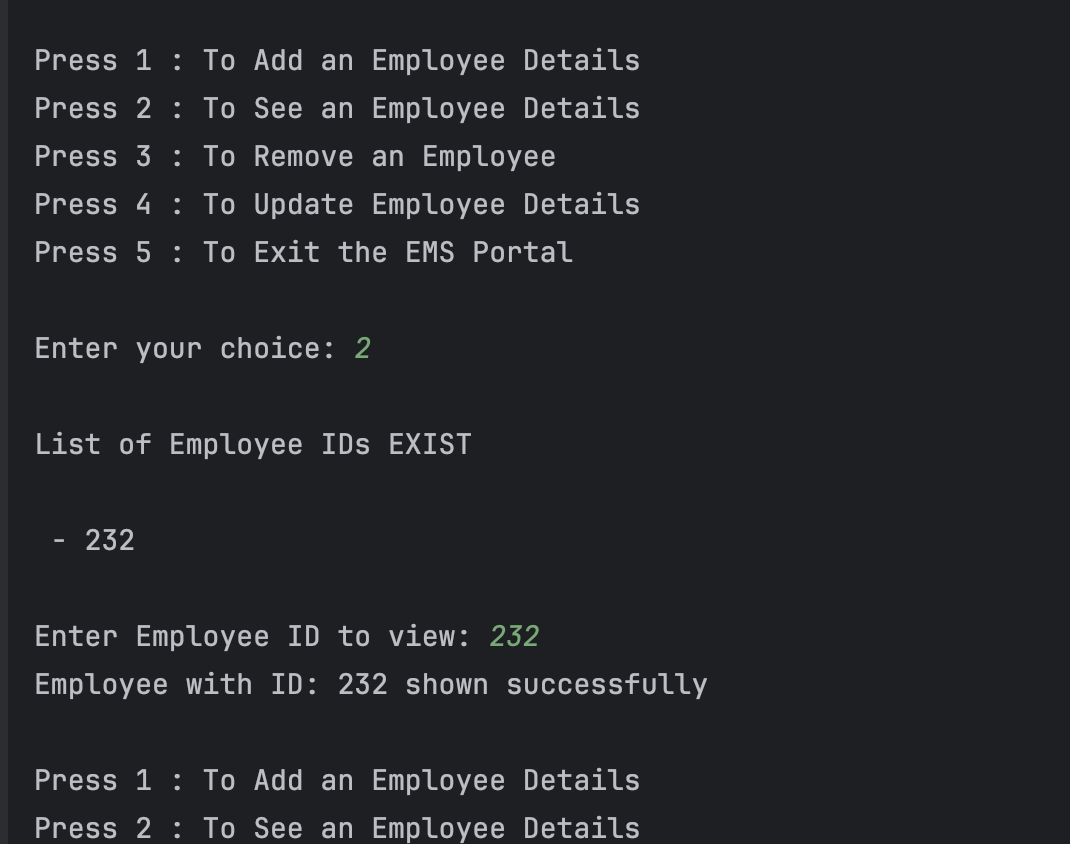






### Demonstration of the System

<https://vimeo.com/1049384123?share=copy>

1. User Registration and Login:
   * Users can register by providing a username and password. The system checks for existing users and stores the credentials in a file.
   * Upon successful login, users are presented with the main menu.
2. Adding an Employee:
   * Users can add an employee by entering details such as name, ID, email, position, contact information, and emergency contact details.
   * The system creates an HTML file for the employee and stores their details in a structured format.
3. Viewing Employee Details:
   * Users can view an employee's details by entering their ID. The system opens the corresponding HTML file in a web browser.
4. Updating Employee Details:
   * Users can update specific details of an employee by providing the employee ID, the old data, and the new data. The system modifies the HTML file accordingly.
5. Removing an Employee:
   * Users can remove an employee by entering their ID. The system deletes the corresponding HTML file.
6. Salary Calculation:
   * The system calculates the yearly salary summary for an employee, including daily and monthly salary breakdowns, and stores it in an HTML file.



### Technical Strengths

1. Modularity: Clear separation of concerns through interface implementation

2. Extensibility: Abstract classes and interfaces allow for easy system expansion

3. Data Persistence: HTML-based storage provides both data storage and presentation

4. Input Validation: Comprehensive validation across all user inputs

5. Error Handling: Robust exception handling throughout the system

### Areas for Improvement

1. Security: Basic text-file based authentication could be enhanced

2. Database Integration: Could benefit from proper database implementation

3. GUI Interface: Currently console-based, could be enhanced with a graphical interface

4. Data Encryption: Sensitive data is stored in plain text

5. Backup System: No built-in backup functionality

### Modifications from the Initial Plan

1. Addition of Emergency Contact Management:
   * Initially, the system only handled basic employee details. The emergency contact feature was added to provide a more comprehensive employee management solution.
2. Salary Calculation Module:
   * The salary calculation feature was not part of the initial plan but was added to enhance the system's functionality.
3. HTML-Based Storage:
   * The initial plan considered plain-text storage, but the team decided to use HTML files for better readability and integration with web applications.
4. Login Menu:
   * Initial Plan is to implement Open Access System, But due to security of information, Login Menu added, Wish to Add MD5 Encryption but due to Complexity and Time Feasibility It Avoided
5. Register Menu:
   * Initial Plan is to implement Open Access System, But due to security of information, Register Menu added for Adding Limited Users

### Teamwork

The project was developed collaboratively by the team members, with each member responsible for specific modules:

* EC/2022/053 (K.S.B. Galkotuwa):
  + Implemented the MainMenu and EntryPoint classes.
  + Implemented Interfaces Calculations
  + Implemented FileFunction Interface
  + Handled system coordination and integration.
  + HTML file created and handled errors and handle Exceptions
  + EmployeeDataView class created
  + Intergration Testing Implemented
  + System Testing Implemented
* EC/2022/023 (A. Dhilukshy):
  + Implemented the EmployDetail and Employee\_Add classes.
  + User Class created
  + Implemented Person super class.
  + Ensured proper validation and handling of duplicate entries.
  + Java Script File created
  + EntryPoint Created using switch-case expression
  + Unit Test Created for EmployDetail and Employee\_Add classes
* EC/2022/078 (G. Varushiny):
  + Implemented the Employee\_Show class.
  + Login Menu Added
  + Handled exceptions for missing or corrupted files.
  + Documentation Added for the code
  + User Authentication methods Added for Registration
  + Unit Test Created for Employ\_show and Login classes
* EC/2022/002 (R.H.M.P. Rathmalage):
  + Implemented the Employee\_Remove class.
  + Register menu Added
  + Validate User Inputs
  + Ensured error handling for non-existing files.
  + Unit Test Created for Employee\_Remove and RegisterMenu
  + Static Arts Created
* EC/2022/077 (C. Niroshan):
  + Implemented the Employee\_Update and CodeExit classes.
  + Handled Errors using try-catch block
  + Ensured the integrity of HTML formatting during updates.
  + CSS Stylesheet created
  + Salary Calculator HTML File Created

The team used GitHub for version control and collaboration. The repository can be accessed [here](https://github.com/kavindus0/EmployeeManagementSystemJava" \t "_blank).

<https://github.com/kavindus0/EmployeeManagementSystemJava>

### Conclusion

The **Employee Management System (EMS)** is a practical and efficient solution for managing employee records, designed with simplicity and usability in mind. By leveraging **HTML-based storage** and **Java programming**, the system provides a modern, cost-effective alternative to traditional record-keeping methods. Its **menu-driven interface** makes it accessible to users with minimal technical knowledge, while its advanced features, such as **salary calculation** and **emergency contact management**, add significant value for businesses.

The project also serves as an excellent example of how **Object-Oriented Programming (OOP)**principles can be applied to create modular, scalable, and maintainable software. The system’s design ensures that it can be easily extended or modified in the future, making it a versatile tool for businesses of all sizes.

While the EMS is already a robust solution, there is always room for improvement. Future enhancements could include the integration of a **graphical user interface (GUI)**, **database support** for better data management, and **data encryption** for improved security. These additions would further enhance the system’s functionality and make it even more user-friendly.

In conclusion, the EMS is a testament to the power of combining **technology** and **good design** to solve real-world problems. It not only simplifies employee management but also serves as a valuable learning tool for students and aspiring developers. This project has been a rewarding experience, and we are proud of what we have accomplished.