**Name:** Arunkumar Sriharish Muthumalai

**Student number:** s3938156

**Lecturer:** Minh Vu Thanh

**Github link:**

**Title**

**Application Description:** The Claim Process Manager Interface is a blueprint for managing insurance claim processes. It outlines methods for adding, updating, deleting, retrieving, and saving claims.

The Claim Class symbolizes a single insurance claim. It houses attributes like claim ID, claim date, insured person details, card number, exam date, list of documents, claim amount, status, and receiver banking information. This class also includes a constructor for initializing a claim object and methods to fetch the claim ID and provide a string representation of the claim.

The Insurance Claim Manager Class is an implementation of the Claim Process Manager interface. It oversees a list of claims and offers methods to execute various operations on claims, such as adding, updating, deleting, retrieving, and saving to a file. It also features a method to generate random numbers for claim IDs and a method to add default claims during initialization.

The Main Class serves as the gateway to the application. It houses the main method where user interaction takes place. Users can select from a menu of options to execute actions like adding, updating, deleting, retrieving, and saving claims.

This application’s key features and functionalities encompass:

Add a new insurance claim with details provided by the user.

Updating an existing insurance claim with revised details.

Deleting an insurance claim using its ID.

Fetching a specific insurance claim using its ID.

Fetching all insurance claims.

Saving all insurance claims to a file.

Moreover, the application carries out basic validation, such as verifying the format of claim IDs and correctly parsing dates.

In summary, this application offers a user-friendly platform for efficient management of insurance claims. It aids insurance administrators in streamlining claim processing and maintaining well-organized records of claims.

**Application Flow (Diagram):**

A[Start] --> B{Choose option}

B --> C{Add claim}

B --> D{Update claim}

B --> E{Delete claim}

B --> F{Get claim by ID}

B --> G{Get all claims}

B --> H{Save claims to file}

B --> I{Exit}

C --> J{Get details} --> K{Validate claim ID} --> L{Create claim object} --> M{Add claim} --> N{Print success message} --> A

D --> O{Enter claim ID} --> P{Find claim by ID} --> Q{Found?}[no] --> A | Q{Found?}[yes] --> R{Enter updated details} --> S{Create updated claim object} --> T{Update claim} --> N --> A

E --> O --> P --> Q{Found?}[no] --> A | Q{Found?}[yes] --> U{Delete claim} --> N --> A

F --> O --> P --> Q{Found?}[no] --> A | Q{Found?}[yes] --> V{Print claim details} --> A

G --> W{Get all claims} --> X{Print all claims} --> A

H --> Y{Enter filename} --> Z{Save claims to file} --> N --> A

I --> []

**API list (With brief description):**

Interface:

Claim Process Manager:

add Claim (Claim claim): Adds a new claim to the system.

update Claim (String claimed, claim updated Claim): Updates an existing claim with the specified ID.

delete Claim (String claimed): Deletes a claim with the specified ID.

get Claim (String claimed): Retrieves a claim by its ID.

getAllClaims (): Returns a list of all claims in the system.

saveClaimsToFile (String filename): Saves all claims to a file with the given name.

Classes:

Claim:

Constructor (String id, Date claim Date, String insured Person, String card Number, Date exam Date, List<String> documents, double claim Amount, String status, String receiver Banking Info): Creates a new Claim object with the provided details.

getId(): Returns the claim ID.

to String (): Provides a string representation of the claim's information.

Insurance Claim Manager:

Constructor: Initializes the manager and adds default claims.

add Default Claims (): Adds default claims to the system.

Generate Random Numbers (int length): Generates a string of random numbers with the specified length.

Implements all methods from the Claim Process Manager interface: Provides the actual implementation for managing claims.

**Any drawback and Future Work:**

* Drawback:

Limited Data Persistence: At present, claims are stored in an Array List, leading to data loss upon program termination. A more durable solution would be to use a database or a structured format file for persistent storage.

Insufficient User Input Validation: The application lacks comprehensive validation for user input during claim addition and updating. For example, it doesn’t verify if dates are correctly formatted or if claim amounts are positive. This could compromise data integrity.

Restricted Search Functionality: Claims can only be fetched by ID. Introducing search functionalities based on other parameters (like insured person, date range, claim status) would make retrieval more convenient.

Absence of Security Measures: The code lacks access control or user authentication features, posing a security risk if the application handles sensitive claim data in a production environment.

Default Claims: Embedding default claims in the code may not be suitable in real-world scenarios. A separate mechanism for loading initial data would be more appropriate.

* Future work:

Persistent Storage Implementation: Incorporate a database or a structured format file to persistently store claim data, ensuring data preservation post program termination.

Advanced User Input Validation: Introduce rigorous validation for user input during claim addition and updating to avoid data integrity issues. This includes checking date formats, claim amounts, and other relevant fields.

Expanded Search Functionality: Enable claim search based on various criteria like insured person, date range, claim status, etc., to facilitate easier retrieval of specific claims.

Security Enhancements: Implement access control and user authentication to prevent unauthorized access to claim data, especially crucial if the application is deployed in a production environment.

Error Handling: Enhance error handling across the code to provide more informative messages to the user during claim operations.

Logging: Consider incorporating logging to monitor user actions and system events for audit purposes.

User Interface: Develop a more intuitive user interface (like a Graphical User Interface) to offer users a more user-friendly interaction with the claim management system.