**COLLEGE OF BUSINESS EDUCATION**

****

**DODOMA CAMPUS**

**REPORT OF EMERGENCY SERVICES DIRECTORY**

***Course* : BIT**

***Lecturer*:Madam ATUPELE MWAITETE**

***Subject:* PROGRAMMING IN JAVA**

***Nature of Work:* INDIVIDUAL ASSIGNMENT**

***Student name:* KELVIN J. HASUNGA**

***Reg No:*  03.9322.01.02.2023**

**Table Contents**

[**1. Introduction** 3](#_Toc188922958)

[**2. Code Overview** 3](#_Toc188922959)

[**3. Functional Breakdown** 3](#_Toc188922960)

[**4. User Interface Design** 4](#_Toc188922961)

[**5. Features and Functionalities** 4](#_Toc188922962)

[**6. Emerency Services Data Structure** 4](#_Toc188922963)

[**7. Search Implementation** 5](#_Toc188922964)

[**8. Improvements and Suggestions** 6](#_Toc188922965)

[**9. Summary** 8](#_Toc188922966)

[**REFERENCES** 9](#_Toc188922967)

**1. Introduction**

The provided Java code implements an "Emergency Services Directory," a GUI-based application designed to list and search emergency services across different regions. This report details the code’s structure, functionality, and potential enhancements while highlighting its real-world applicability.

**2. Code Overview**

The application uses the Java Swing library to create a user-friendly interface. Core components include:

* JFrame for the main window.
* JTable for displaying service data.
* JTextField and JButton for search functionality.
* A predefined dataset of emergency services, covering various categories and regions in Tanzania.

The code is structured around the EmergencyServicesDirectory class, which extends JFrame and encapsulates all UI elements and logic.

**3. Functional Breakdown**

**3.1. Class Structure**

The EmergencyServicesDirectory class consists of:

* A constructor for initializing and setting up the GUI components.
* Methods for loading data (loadServices) and filtering results (searchServices).

**3.2. Main Method**

The main method invokes the Swing application on the Event Dispatch Thread (EDT) using SwingUtilities.invokeLater, ensuring thread safety.

**4. User Interface Design**

The application’s layout follows a straightforward structure:

**4.1. Search Panel**

* **Components:** A JLabel, JTextField, and JButton are arranged in a BorderLayout.
* **Functionality:** Users can input search terms and trigger a search with the button.

**4.2. Table Panel**

* **Components:** A JTable embedded within a JScrollPane displays emergency services data.
* **Configuration:** The table uses a DefaultTableModel with predefined column names: Name, Category, Region, and Contact.

**5. Features and Functionalities**

**5.1. Data Loading**

The loadServices method populates the table with hardcoded data representing emergency services. Each service has attributes like name, category, region, and contact.

**5.2. Search Functionality**

The searchServices method filters the dataset based on user input. It:

* Clears the existing table data.
* Iterates over the predefined dataset, adding rows that match the search query in any of the columns.

**5.3. Dynamic Updates**

The table dynamically updates its content in response to search actions, ensuring real-time feedback.

**6. Emerency Services Data Structure**

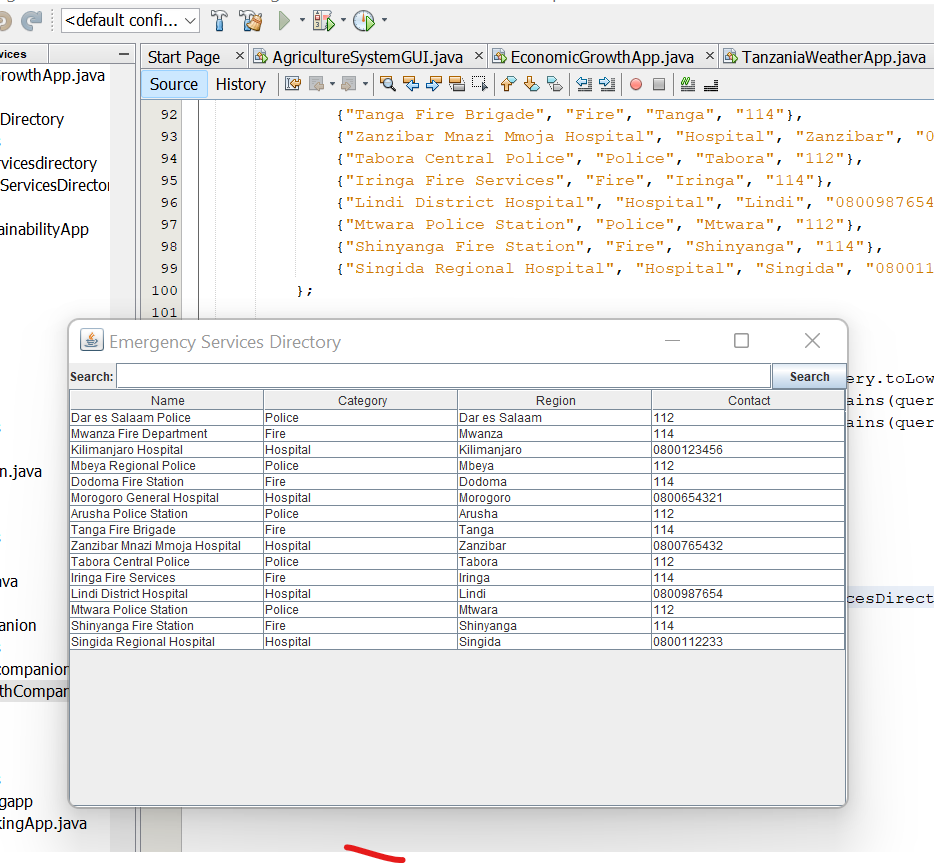
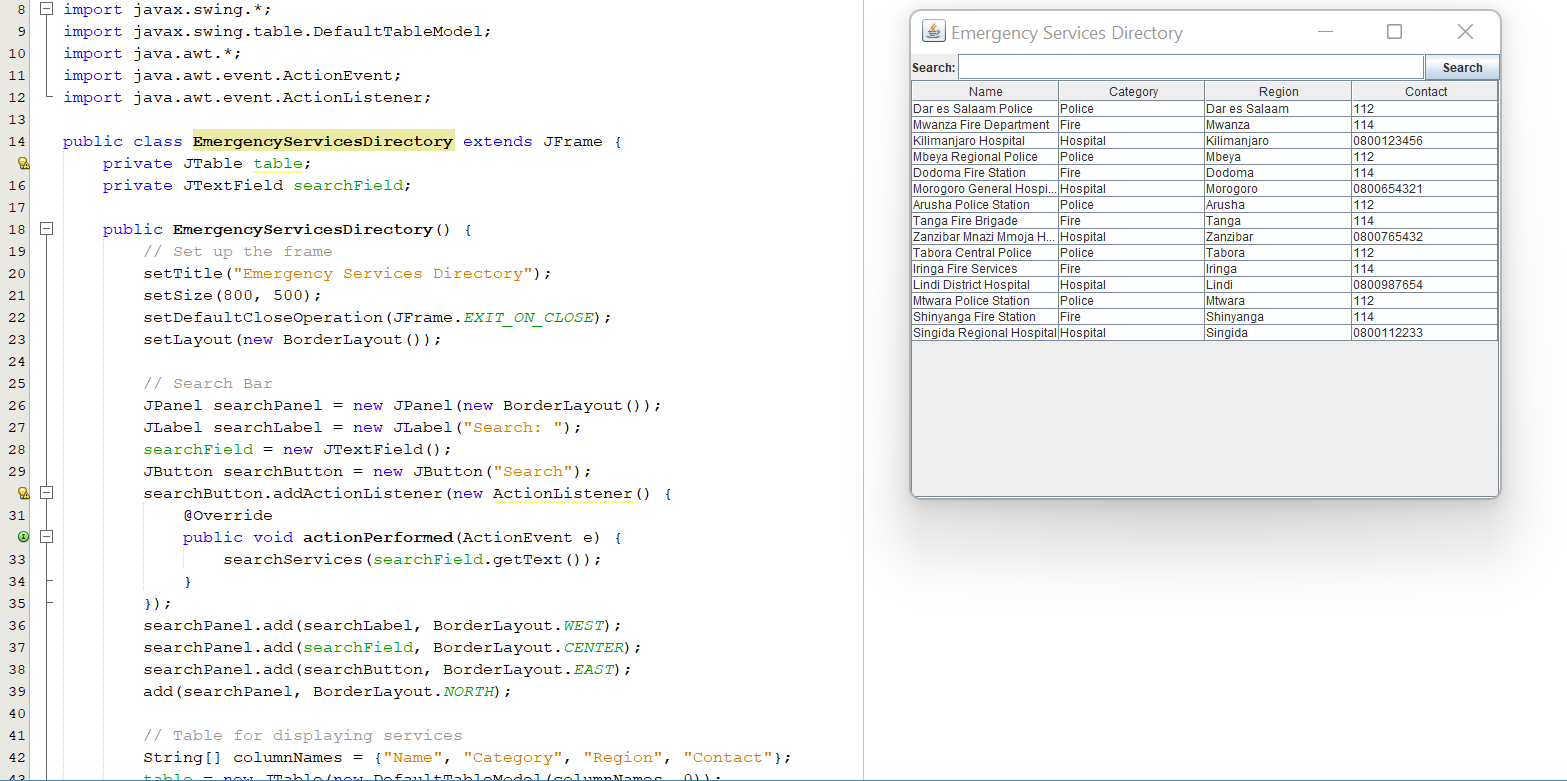
**6.1. Hardcoded Dataset**

The dataset is represented as a two-dimensional Object array. Each inner array corresponds to an emergency service entry.

**6.2. Fields**

* **Name:** Name of the service provider (e.g., "Dar es Salaam Police").
* **Category:** Type of service (e.g., Police, Fire, Hospital).
* **Region:** Geographical region (e.g., Dodoma, Arusha).
* **Contact:** Emergency contact number.

**6.3. Data Sample output**

**7. Search Implementation**

**7.1. Query Handling**

The searchServices method converts the query to lowercase for case-insensitive comparison. It checks if the query is contained in any field of a service record.

**7.2. Filtering Logic**

For each service record:

* If the query matches the Name, Category, or Region, the record is added to the table.
* Otherwise, it is excluded.

**7.3. Efficiency**

This implementation is simple but may not scale efficiently for large datasets. For improved performance, a database or indexing structure could be utilized.

**8. Improvements and Suggestions**

**8.1. Data Source**

* **Current:** Hardcoded data.
* **Suggestion:** Use a database or external file (e.g., CSV or JSON) for dynamic and scalable data management.

**8.2. Advanced Search**

* Add support for advanced filters, such as selecting specific categories or regions.
* Implement partial and fuzzy matching for better user experience.

**8.3. UI Enhancements**

* **Theme:** Apply modern UI themes using libraries like FlatLaf or JavaFX.
* **Sorting:** Enable column-based sorting for easier navigation.

**8.4. Localization**

Provide support for multiple languages to cater to diverse users.

**8.5. Scalability**

For large datasets:

* Use pagination to limit the number of displayed rows.
* Implement asynchronous loading to maintain responsiveness.

**8.6. Error Handling**

* Handle invalid search queries gracefully.
* Notify users if no results are found.

**9. Summary**

The "Emergency Services Directory" application is a functional and intuitive tool for managing and searching emergency service information. While the current implementation is suitable for small datasets, enhancements like dynamic data sources, advanced search capabilities, and improved UI design can significantly enhance its utility and user experience.

# **REFERENCES**

**Swing Library (javax.swing)** Java Platform, Standard Edition. (2025). javax.swing package. Oracle. Retrieved from <https://docs.oracle.com/en/java/javase/17/docs/api/java.desktop/javax/swing/package-summary.html>

**DefaultTableModel Class (javax.swing.table)** Oracle. (2025). DefaultTableModel class. Oracle. Retrieved from <https://docs.oracle.com/en/java/javase/17/docs/api/java.desktop/javax/swing/table/DefaultTableModel.html>

**ActionEvent and ActionListener (java.awt.event)** Oracle. (2025). java.awt.event package. Oracle. Retrieved from <https://docs.oracle.com/en/java/javase/17/docs/api/java.desktop/java/awt/event/package-summary.html>

**BorderLayout Class (java.awt)** Oracle. (2025). BorderLayout class. Oracle. Retrieved from <https://docs.oracle.com/en/java/javase/17/docs/api/java.desktop/java/awt/BorderLayout.html>