**University Mail Server (UMS) Project Documentation**

**1. Project Overview**

The University Mail Server is a Java-based system that simulates a university email system. It implements different types of users (students, TAs, lecturers) with specific message sending restrictions. The project demonstrates core Object-Oriented Programming concepts including Encapsulation, Polymorphism, Inheritance, and Abstraction.

**2. Project Structure**

The project is organized into three main packages:

- `eg.fue.cs.jobs`: Contains all job-related classes

- `eg.fue.cs.messages`: Contains all message-related classes

- `eg.fue.cs.users`: Contains all user-related classes

**3. Core Components and Their Purposes**

***3.1 Job System***

- Base Job Class: Represents a generic job with basic attributes (name, description, faculty, department)

- Job Subclasses:

- Student: Represents a student job with level information

- TeacherAssistant: Represents a TA job with graduation year

- Lecturer: Represents a lecturer job with years of experience

***3.2 Message System***

- Base Message Class: Handles basic message attributes (subject, body, sender, receiver)

- Message Types:

- PrivateMessage: Can only be sent between users of the same type

- PublicMessage: Can be sent to anyone without restrictions

***3.3 User System***

- Base User Class: Manages user information and message handling

- User Types:

- FullTimeUser: Can send messages to any FullTimeUser or PartTimeUser

- PartTimeUser: Can send messages to any FullTimeUser and PartTimeUsers of same specialty

**4. Key Features and Implementation**

***4.1 Encapsulation***

- All instance variables are declared as private

- Access to variables is controlled through getters and setters

- Super class variables are private with protected access methods

- Ensures data integrity and controlled access

***4.2 Inheritance***

- Job hierarchy: Base Job class with specialized job types

- Message hierarchy: Base Message class with specialized message types

- User hierarchy: Base User class with specialized user types

- Each subclass inherits and extends base functionality

***4.3 Polymorphism***

- Overridden methods in subclasses for specific behaviors

- Different message sending rules for different user types

- Dynamic method binding for message handling

- Flexible and extensible design

***4.4 Abstraction***

- Abstract message sending rules

- Abstract user types and roles

- Abstract job types and responsibilities

- Clear separation of concerns

***5. Message Sending Rules***

**5.1 User-Specific Rules**

***1. Student Restrictions:***

- Limited to sending messages only to other students

- Cannot communicate with TAs or lecturers

***2. Teacher Assistant Rules:***

- Can communicate with students

- Can communicate with TAs and lecturers of same specialty

- Restricted from communicating with different specialties

***3. Lecturer Rules:***

- Full communication privileges

- Can send messages to any user type

***4. PartTimeUser Rules:***

- Can communicate with any FullTimeUser

- Limited to communicating with PartTimeUsers of same specialty

***5. FullTimeUser Rules:***

- Can communicate with any FullTimeUser

- Can communicate with any PartTimeUser

***5.2 Message Type Rules***

***1. Private Messages:***

- Restricted to same user type communication

- Example: TA to TA, Student to Student

***2. Public Messages:***

- No restrictions on sender or receiver

- Universal communication capability

**6. Testing and Validation**

The project includes comprehensive testing through the Main class that demonstrates:

- Creation of different user types

- Message sending between various user combinations

- Message receiving and inbox management

- Message deletion functionality

- Validation of all sending rules

**7. Design Patterns and Best Practices**

- Use of ArrayList for dynamic message storage

- Proper encapsulation of data

- Clear inheritance hierarchy

- Consistent method naming and implementation

- Default constructors for all classes

- Proper initialization of all instance variables

**8. Project Benefits**

***1. Educational Value:***

- Demonstrates OOP concepts in practice

- Shows real-world application of inheritance and polymorphism

- Illustrates proper encapsulation techniques

***2. Practical Application:***

- Simulates real-world email system

- Implements role-based access control

- Demonstrates message handling and management

***3. Extensibility:***

- Easy to add new user types

- Simple to implement new message types

- Flexible for future modifications