**JavaFX Project**

**🎓E-Class System**

# Supervisor

## Dr. Subecz Zoltan

# Project Members:

## SULEYMAN Jumaniyazov      EG6X6K

## ALBERTO Balazs Sandor       NKNHIK

****

**John Von Neumann University**

**Academic Year: 2024-2025**

**Introduction with Thanks**

We are immensely grateful to the divine presence that has granted me the fortitude, vitality, and determination to accomplish this endeavor.

Our heartfelt gratitude extends to my beloved family members whose unwavering support and various forms of assistance have propelled me to this juncture of our educational journey.

We extend our sincere appreciation to Dr. Subecz Zoltan, our esteemed supervisor, whose invaluable guidance, insightful feedback, and unwavering encouragement have been instrumental in bringing this project to fruition.

We express profound gratitude to the esteemed members of the evaluation committee for graciously agreeing to assess our work.

Lastly, We extend heartfelt thanks to all the educators who have imparted their knowledge and wisdom, as well as to all individuals who have supported and assisted us throughout my academic and professional pursuits.

**INSTRUCTION**

1. **Database menu**: a CRUD application with the chosen database

**Read submenu**: Displays the database data in a table. It uses data from at least 3 tables in the database. If the database contains too much data, it is sufficient to display only a part of it.

**Read2 submenu**: Similar to the previous one, but a Form should first appear on the page, which you can use to filter the database data. Use the following elements in the form: text input field, drop-down list, radio button, check box.

**Write submenu**: A form should appear on the page, with the help of which you can add a new record to one of the tables of the database.

**Change submenu**: A form should appear on the page, which you can use to change an existing record in one of the tables of the database. The ID of the record should be selectable from a drop-down list.

**Delete submenu**: To delete a record of a table. The ID of the record should be selectable from the drop-down list.

1. **SoapClient menu**: Create a SOAP client for the Hungarian National Bank web service.

**Download submenu**: Downloads all data to the Bank.txt file.

**Download2 submenu**: A form appears on the page (e.g. input fields, drop-down list, radio button, check box). Downloads the selected data to the Bank.txt file

**Graph submenu**: Similar to the Download2 submenu, select a dataset and display the data on a chart.

1. **Parallel submenu**: Show parallel programming on one page.

e.g. after clicking a button, a changing text should appear in one Label every 1 second and in the other Label every 2 seconds.

1. **Forex menu**: using Oanda API

**Account Information submenu**: Print the account information in a table

**Current prices submenu**: You can choose a currency pair from a drop-down list. Print the current price by clicking a button.

**Historical prices submenu**: You can choose a currency pair, start and end date from drop-down lists. It lists the prices between the two dates in a table, which is also shown on a graph.

**Position opening submenu**: You can choose a currency pair, quantity and direction (buy, sell) from the drop-down lists and open the position.

**Position closing submenu**: The position ID can be entered in the input field. Close that position.

**Opened positions submenu**: Lists opened positions in a table.

1. Use the GitHub (github.com) version tracking system.

(**Mandatory element! The source will be checked based on it**)

Upload not only the finished application in one step, but also the sub-states in at least 5 steps per person.

1. Use the project work method on GitHub. **(5 points)**

**Choose your own name on the GIT, based on which the uploader can be identified.**

1. Create an **executable JAR file (Required item! the operation will be checked based on it)**

**Project Overview**

The EClass System is a JavaFX-based desktop application designed to manage educational data, featuring multiple functionalities including database management, exchange rate tracking, and parallel processing.

**Team Collaboration**

**Team Members**

* 🧑‍💻Suleyman Jumaniyazov
* 🧑‍💻Alberto Magarin

**Work Distribution**

* **[Student 1** Suleyman Jumaniyazov**]** **🧑‍💻**
* Implemented database management system
* Developed CRUD operations for Students and Subjects
* Created data import and export utilities
* Implemented statistics and reporting features
* **[Student 2** Alberto Magarin**]** **🧑‍💻**
* Developed MNB Exchange Rates service
* Implemented Parallel Processing demo
* Created data visualization components
* Handled UI design and user interaction

**Technical Architecture**

**Technologies Used**

* Java 17
* JavaFX
* SQLite
* Apache POI
* SLF4J Logging

**Key Features**

1. **Database Management 📂**
   * ➕🔍✏️❌CRUD operations for Students, Subjects, and Marks
   * Data import from text files
   * Advanced filtering and searching
2. **MNB Exchange Rates 💱**
   * 🌐Real-time currency rate download
   * 📈 Dynamic Line chart visualization
   * 🗓️Flexible date and currency selection
   * 💹 Multiple currency support
3. **Parallel Processing 🔀**
   * Demonstrates concurrent thread execution ⏱️
   * Two independent counters with different update intervals 🕰️
   * Thread-safe implementation 🧵

**Database Structure 💾**

**Tables**

1. **students**
   * id (INTEGER PRIMARY KEY) 🆔
   * sname (TEXT)
   * class (TEXT)
   * boy (INTEGER)
2. **subjects**
   * id (INTEGER PRIMARY KEY) 🆔
   * sname (TEXT)
   * category (TEXT)
3. **marks**
   * id (INTEGER PRIMARY KEY) 🆔
   * studentid (INTEGER)
   * mdate (TEXT)
   * mark (INTEGER)
   * type (TEXT)
   * subjectid (INTEGER)

**Project Setup and Deployment**

**Prerequisites**

* Java Development Kit 17
* Maven
* SQLite

**Building the Project**

bash

mvn clean package

**Running the Application**

bash

java -jar target/eClassSystemJavaFX-1.0-SNAPSHOT.jar

**GitHub Repository**

* Repository Link: [Insert GitHub Repository URL]

**🧩 Development Workflow**

**1. 🎬 Project Initialization**

* Create project structure
* Set up Maven dependencies
* Configure JavaFX environment

**2. 🗃️ Database Setup**

* Design SQLite schema
* Implement data import mechanisms
* Create CRUD service classes

**3. 🖥️ UI Development**

* Design FXML layouts
* Implement JavaFX controllers
* Create interactive components

**4. 🔌 Service Integration**

* Develop MNB Exchange Rates service
* Implement parallel processing demo
* Add error handling and logging

**5. 🧪 Testing and Refinement**

* Unit testing
* Integration testing
* User experience optimization

**🚧 Challenges and Solutions**

**1. 🧵 Thread Synchronization**

**Challenge:** Ensuring thread-safe updates in parallel processing **Solution:**

* Used **AtomicInteger** for counter management
* Implemented **Platform.runLater()** for UI updates

**2. 📡 Real-time Data Fetching**

**Challenge:** Retrieving dynamic exchange rates **Solution:**

* Asynchronous thread-based data retrieval
* Implemented robust error handling
* Fallback mechanism for unavailable data

**🔮 Future Roadmap**

* 🔐 User authentication system
* 📈 Advanced analytics dashboard
* 🌐 Cloud synchronization
* 📱 Mobile companion app

**💡 Key Learnings**

* Modular application design
* Concurrent programming techniques
* JavaFX UI development
* Database management best practices

**📂 Root Project Structure**

eClassSystemJavaFX/

│

├── src/

│ ├── main/

│ │ ├── java/ # Java source code

│ │ │ └── com/

│ │ │ └── example/

│ │ │ └── eclasssystem/

│ │ │ ├── Main.java # Application entry point

│ │ │ ├── services/ # Core business logic

│ │ │ ├── controller/ # UI controllers

│ │ │ ├── model/ # Data models

│ │ │ └── util/ # Utility classes

│ │ │

│ │ ├── resources/ # Non-code resources

│ │ │ ├── fxml/ # JavaFX layout files

│ │ │ ├── database/ # SQLite database

│ │ │ ├── txt/ # Initial data files

│ │ │ └── css/ # Styling resources

│ │ │

│ │ └── module-info.java # Java module configuration

│ │

│ └── test/ # Unit and integration tests

│

├── pom.xml # Maven project configuration

└── DOWNLOAD.zip # Distributable package

**🔍 Key Architectural Components**

**1. 🚪 Entry Point: Main.java**

A computer screen shot of a program

Description automatically generated

A screenshot of a computer program

Description automatically generated

A screenshot of a computer program

Description automatically generated

A screen shot of a computer

Description automatically generated

**A computer screen shot of a program

Description automatically generated**

**A screenshot of a computer program

Description automatically generated**

**A screen shot of a computer

Description automatically generated**

1. **🎨 Controller Design Pattern**

Architectural Highlights:

Reactive UI updates

Separation of data loading and presentation

Dynamic filtering and searching

**A computer screen shot of text

Description automatically generated**

1. **🔀 Parallel Processing Demonstration**

Concurrency Techniques:

ExecutorService for thread management

AtomicBoolean for thread synchronization

Platform.runLater() for safe UI updates

**A computer screen shot of a program code

Description automatically generated**