WEEK1 - TASK

STUDENT INFORMATION SYSTEM

Project Overview

The Student Information System is a Java-based console application designed to store, manage, and display student records efficiently. The system allows users to add, view, search, update, and delete student information using a menu-driven interface.

Project Objectives

The main objectives of this project are:

- To understand object-oriented programming concepts in Java
- To implement CRUD (Create, Read, Update, Delete) operations
- To apply input validation and error handling
- To use data structures (ArrayList) for data persistence
- To build a user-friendly, menu-driven console application

Scope of the Project

- Store student details such as ID, name, age, grade, and contact information
- Validate user input to prevent invalid data entry
- Provide search functionality using student ID or name
- Display student records in formatted tabular form

Setup Instructions

Software Requirements

- Operating System: Windows / macOS / Linux
- Java JDK: Version 17 or higher
- IDE**:** Eclipse IDE for Java Developers

Installing Java JDK

1. Visit: https://adoptium.net
2. Download JDK 17
3. Install the JDK
4. Verify installation:
5. java --version

Installing Eclipse IDE

1. Visit: https://www.eclipse.org/downloads/
2. Download Eclipse IDE for Java Developers
3. Install and launch Eclipse

Running the Project in Eclipse

1. Open Eclipse
2. Create a New Java Project
3. Name the project: StudentInformationSystem
4. Inside the src folder, create the following classes:
    - Student.java
    - StudentManager.java
    - ValidationUtils.java
    - StudentInformationSystem.java
5. Paste the respective code into each file
6. Right-click StudentInformationSystem.java
7. Select Run As → Java Application

Code Structure

File Hierarchy

StudentInformationSystem

```
│
├── src
│   ├── Student.java
│   ├── StudentManager.java
│   ├── ValidationUtils.java
│   └── StudentInformationSystem.java
│
└── JRE System Library
```

Description of Files

Student.java

- Defines the **Student class**
- Contains attributes: studentId, name, age, grade, contact
- Uses encapsulation (private variables + getters/setters)

StudentManager.java

- Handles **business logic**
- Manages student records using ArrayList
- Performs add, search, and delete operations

ValidationUtils.java

- Contains reusable **input validation methods**
- Prevents application crashes due to invalid input

StudentInformationSystem.java

- Main class

- Contains menu-driven interface
- Coordinates user interaction and system operations

Technical Details

Programming Language

- Java (Object-Oriented Programming)

Data Structures Used

- ArrayList<Student>
  Used for dynamic storage of student records.

Algorithms Used

- **Linear Search:**
  Used to find students by ID or name.
- **Input Validation Loop:**
  Ensures valid numeric input using try-catch.

Architecture

- **Layered Architecture**
  - Presentation Layer: Menu system
  - Business Logic Layer: StudentManager
  - Data Model Layer: Student class
  - Utility Layer: ValidationUtils

User Manual

Main Menu Options

1. Add Student
2. View All Students
3. Search Student
4. Update Student

5. Delete Student

6. Exit

How to Use the System

Add Student

- Enter student ID, name, age, grade, and contact
- Input is validated before saving

View All Students

- Displays all students in formatted table form

Search Student

- Search by student ID or name

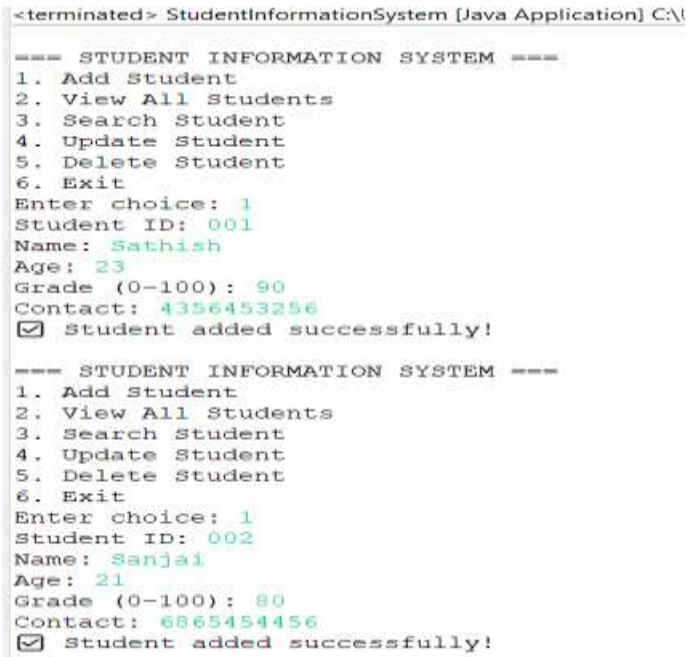
Update Student

- Modify existing student details

Delete Student

- Remove student record using student ID

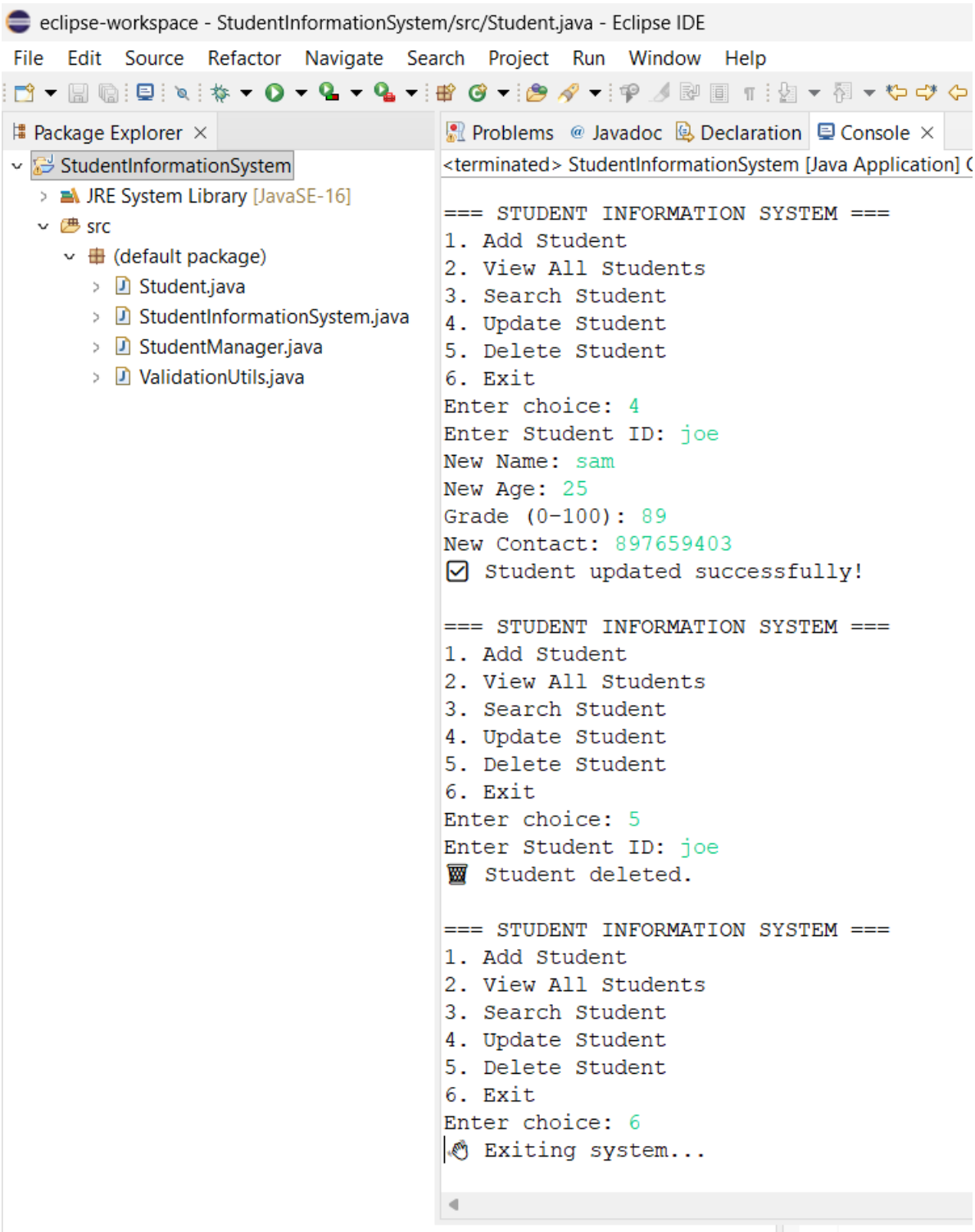
Visual Documentation



```
=== STUDENT INFORMATION SYSTEM ===
1. Add Student
2. View All Students
3. Search Student
4. Update Student
5. Delete Student
6. Exit
Enter choice: 1
Student ID: Joe
Name: b
Age: 56
Grade (0-100): 70
Contact: 55676585854
☑ Student added successfully!

=== STUDENT INFORMATION SYSTEM ===
1. Add Student
2. View All Students
3. Search Student
4. Update Student
5. Delete Student
6. Exit
Enter choice: 2
ID          Name                  Age    Grade    Contact
------------------------------------------------------------
001         Sathish               23     90.00    4356453256
002         Sanjai                21     80.00    6865454456
Joe         b                     56     70.00    55676585854

=== STUDENT INFORMATION SYSTEM ===
1. Add Student
2. View All Students
3. Search Student
4. Update Student
5. Delete Student
6. Exit
```

```
<terminated> StudentInformationSystem [Java Application] C:\Users\DE
4. Update Student
5. Delete Student
6. Exit
Enter choice: 3
Enter Student ID or Name: sathish
✔ Student Found
ID: 001
Name: Sathish
Age: 23
Grade: 90.0
Contact: 4356453256

=== STUDENT INFORMATION SYSTEM ===
1. Add Student
2. View All Students
3. Search Student
4. Update Student
5. Delete Student
6. Exit
Enter choice: 4
Enter Student ID: joe
New Name: sam
New Age: 25
Grade (0-100): 89
New Contact: 897659403
☑ Student updated successfully!

=== STUDENT INFORMATION SYSTEM ===
1. Add Student
2. View All Students
3. Search Student
4. Update Student
5. Delete Student
6. Exit
Enter choice: 5
Enter Student ID: joe
🗑 Student deleted.
```

```
=== STUDENT INFORMATION SYSTEM ===
1. Add Student
2. View All Students
3. Search Student
4. Update Student
5. Delete Student
6. Exit
Enter choice: 6
👋 Exiting system...
```


eclipse-workspace - StudentInformationSystem/src/Student.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer
StudentInformationSystem
JRE System Library [JavaSE-16]
src
(default package)
Student.java
StudentInformationSystem.java
StudentManager.java
ValidationUtils.java
Problems Javadoc Declaration Console
<terminated> StudentInformationSystem [Java Application]
=== STUDENT INFORMATION SYSTEM ===
1. Add Student
2. View All Students
3. Search Student
4. Update Student
5. Delete Student
6. Exit
Enter choice: 4
Enter Student ID: joe
New Name: sam
New Age: 25
Grade (0-100): 89
New Contact: 897659403
Student updated successfully!
=== STUDENT INFORMATION SYSTEM ===
1. Add Student
2. View All Students
3. Search Student
4. Update Student
5. Delete Student
6. Exit
Enter choice: 5
Enter Student ID: joe
Student deleted.
=== STUDENT INFORMATION SYSTEM ===
1. Add Student
2. View All Students
3. Search Student
4. Update Student
5. Delete Student
6. Exit
Enter choice: 6
Exiting system...

Testing Evidence

Test Cases

Test Case	Input	Expected Output
Add Student	Valid data	Student added successfully
Add Student	Invalid age (-5)	Error message shown
Add Student	Invalid grade (O)	Re-prompt input
Search Student	Existing ID	Student details shown
Search Student	Invalid ID	Student not found
Delete Student	Valid ID	Student deleted
Delete Student	Invalid ID	Error message

Validation Testing

- Non-numeric input handled using try-catch
- Age must be positive
- Grade must be between 0 and 100
- Program does not crash on invalid input

Conclusion

The Student Information System successfully demonstrates the use of Java fundamentals, object-oriented principles, data structures, and error handling. The project meets all functional and technical requirements and provides a solid foundation for further enhancements such as file storage or GUI implementation.