### **Lewis Ochieng’ Ombaka - Programming Assignment Unit 7**

### **Overview of the Application**

The Student Management System provides functionalities for managing student records, course enrollments, and grade assignments, along with dynamic updates to the interface and robust error handling. The application's design focuses on being intuitive, user-friendly, and easily navigable via menus and tables.

### **Key Functionalities**

1. **Student Management**:
   * **Add Students**: Administrators can add new students by entering their names. These names are stored in an ArrayList.
   * **View Students**: A dynamically updating table (JTable) displays all the added students along with their enrolled courses and assigned grades.
   * **Update Student Information**: Although not implemented in this iteration, placeholders for future enhancements can allow modifying student names or other data.
2. **Course Enrollment**:
   * Students can be enrolled in specific courses.
   * When a course is specified, the system checks whether the student exists and adds the course to the student’s record. Courses are also stored in a HashMap for managing enrollments.
3. **Grade Management**:
   * Administrators can assign grades to students for specific courses they are enrolled in.
   * Grades are stored in a HashMap within each student object, mapping courses to corresponding grades.
4. **Dynamic Interface Updates**:
   * The application uses a DefaultTableModel to dynamically update the JTable that displays student information, ensuring the interface reflects all changes, such as adding students, enrolling in courses, and assigning grades.
5. **Error Handling**:
   * Dialog boxes notify users of invalid inputs, such as empty student names or non-enrolled courses.
   * Error conditions like "Student not found" or "Invalid grade input" are handled gracefully with user-friendly messages.

### **Structure of the Code**

1. **Main Application (**StudentManagementSystem **Class)**:
   * **Attributes**:
     + A JFrame serves as the main window.
     + An ArrayList stores all students.
     + A HashMap tracks course enrollments.
     + A DefaultTableModel powers the student table for dynamic updates.
   * **Methods**:
     + initializeGUI(): Sets up the main GUI layout, including menus, table, and event listeners.
     + addStudent(): Adds a new student to the system.
     + refreshStudentTable(): Updates the table to reflect the latest student data.
     + enrollInCourse(): Enrolls a student in a course.
     + assignGrade(): Assigns grades to students.
     + viewStudentGrades(): Displays a student's grades.
2. **Student Class**:
   * **Attributes**:
     + name: Stores the student's name.
     + courses: Tracks enrolled courses as a list.
     + grades: Maps courses to grades.
   * **Methods**:
     + getName(), getCourses(), and getGrades(): Provide access to the student's data.
     + addCourse(String course): Adds a course to the student's list of courses.
     + addGrade(String course, String grade): Assigns a grade for a specific course.

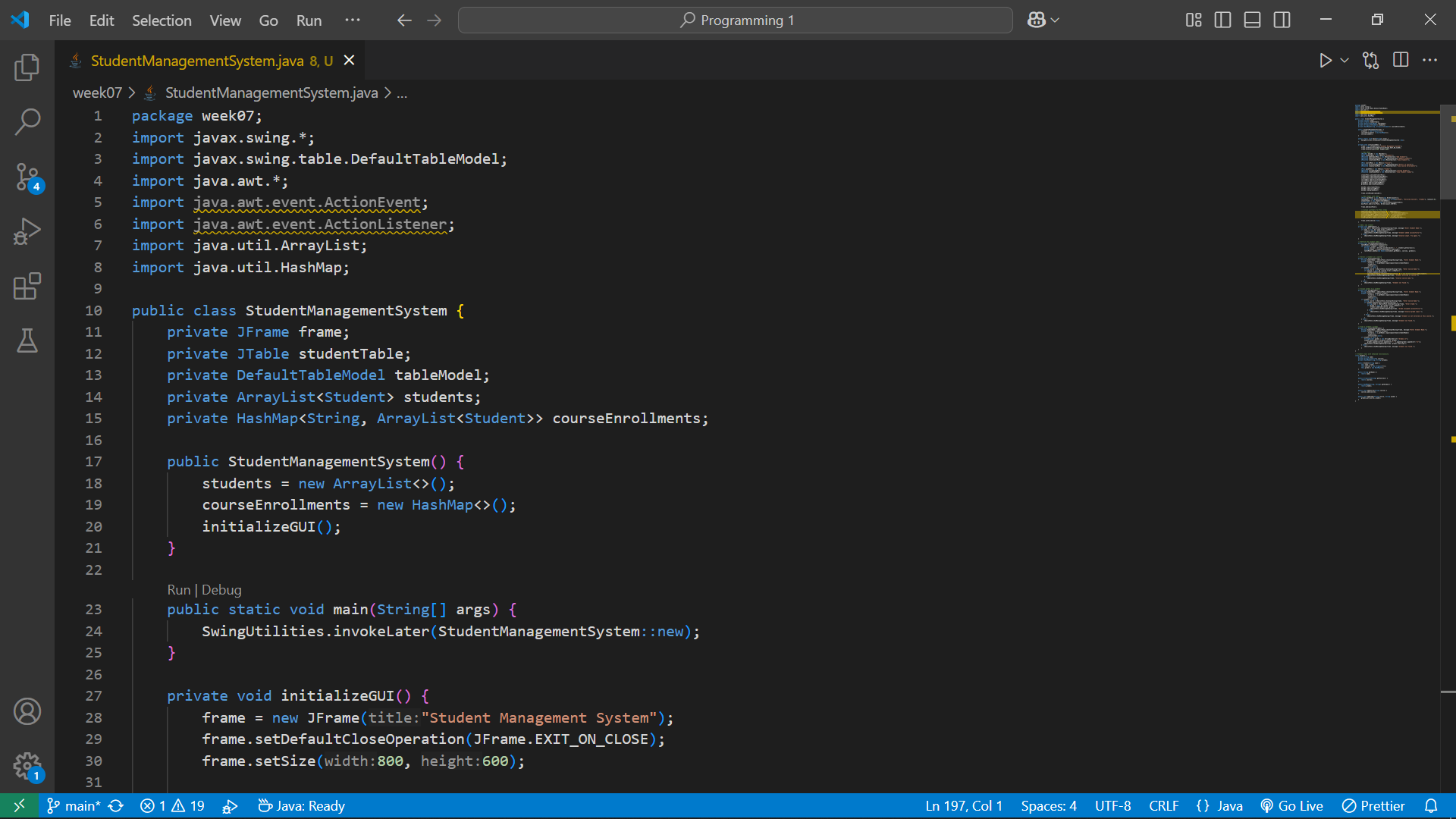
### **Design Choices**

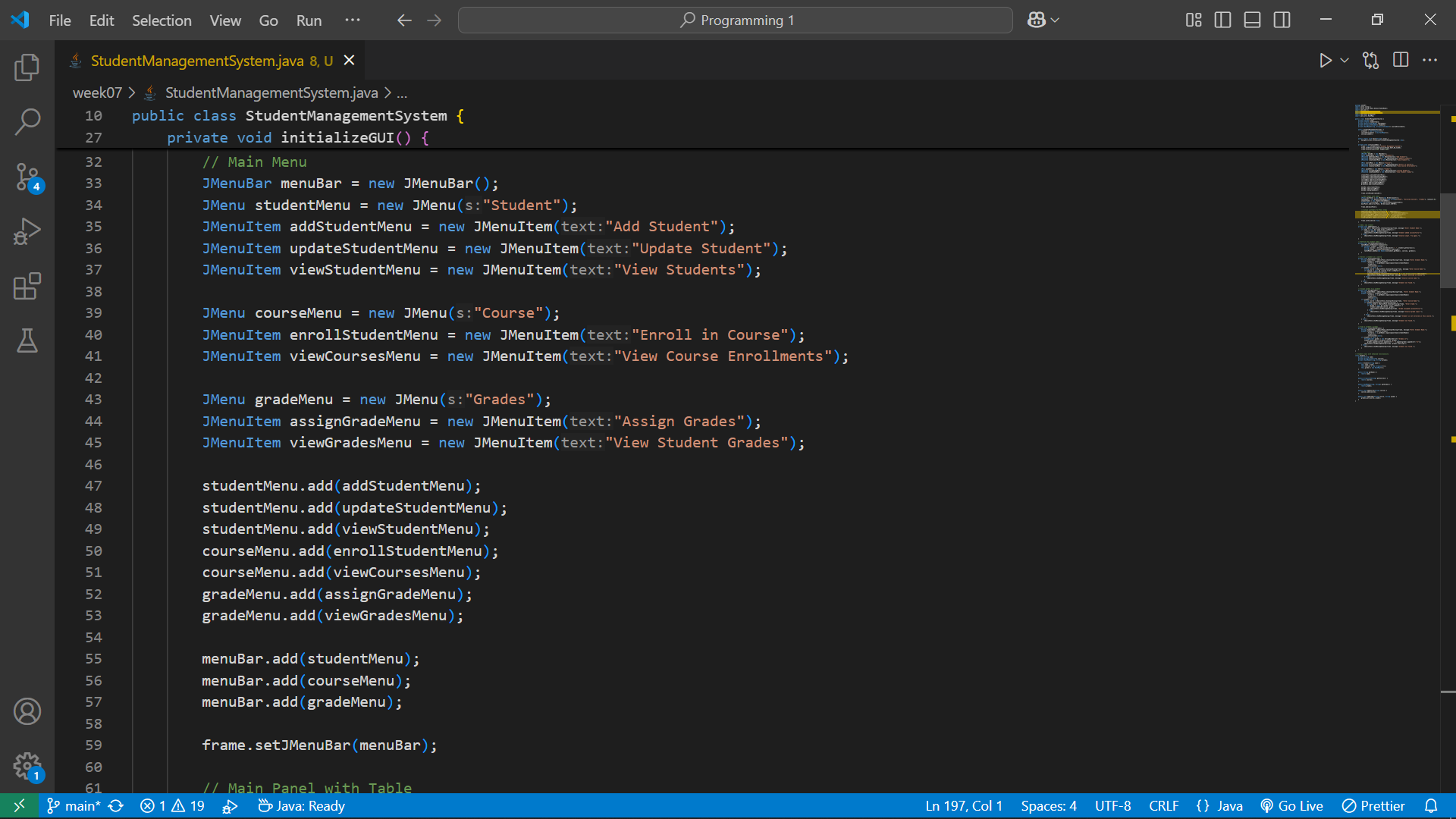
1. **Swing Framework**:
   * Swing provides robust components and flexibility for creating interactive desktop applications.
   * Using JTable allows dynamic updates for visualizing student data efficiently.
2. **Menus and Tables**:
   * A JMenuBar organizes functionalities into logical sections: Student, Course, and Grades.
   * JTable enhances user experience by presenting data in a structured format.
3. **Dynamic Updates**:
   * Employing DefaultTableModel ensures real-time updates without requiring manual refreshes.
4. **Error Handling**:
   * Validation of inputs prevents data inconsistency and enhances user experience.
   * Messages guide administrators in resolving errors, making the application user-friendly.

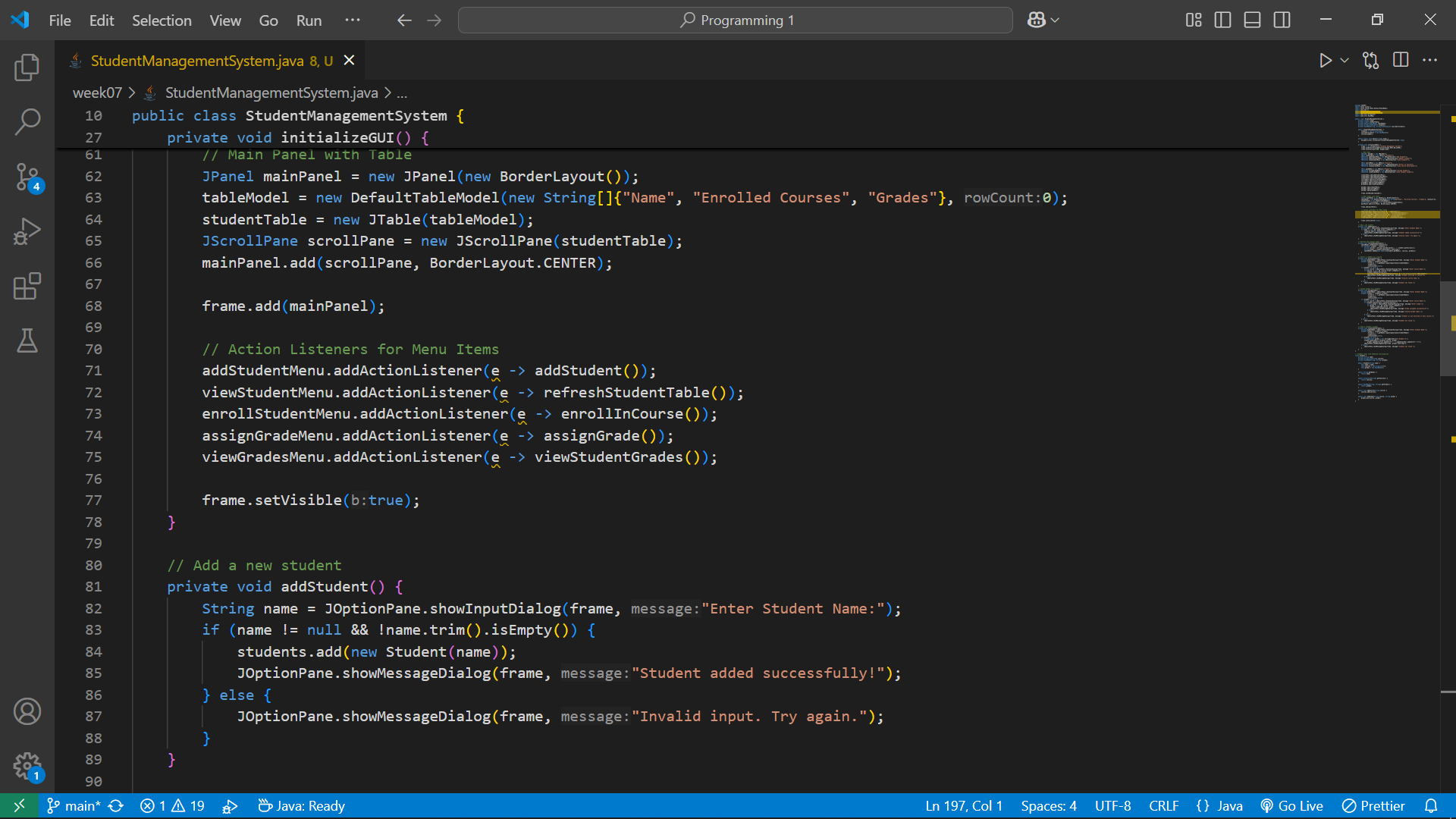
### **Instructions for Running the Program**

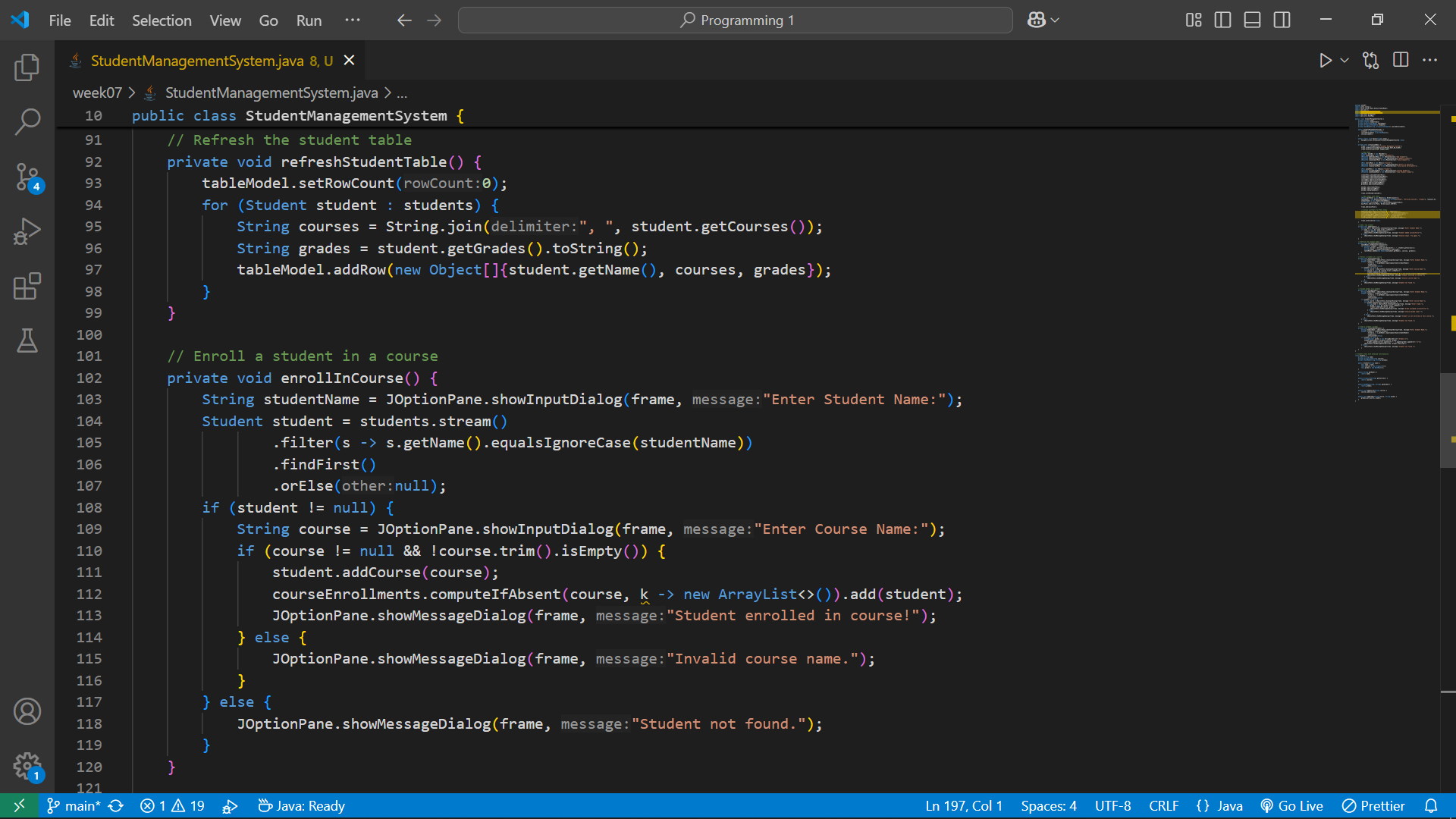
1. Compile the code using a Java IDE or terminal.
2. Run the StudentManagementSystem class's main method.
3. Navigate through the menus to:
   * Add students.
   * View the student list in the table.
   * Enroll students in courses.
   * Assign grades.
   * View grades.
4. Follow on-screen instructions and dialog prompts for user interactions.

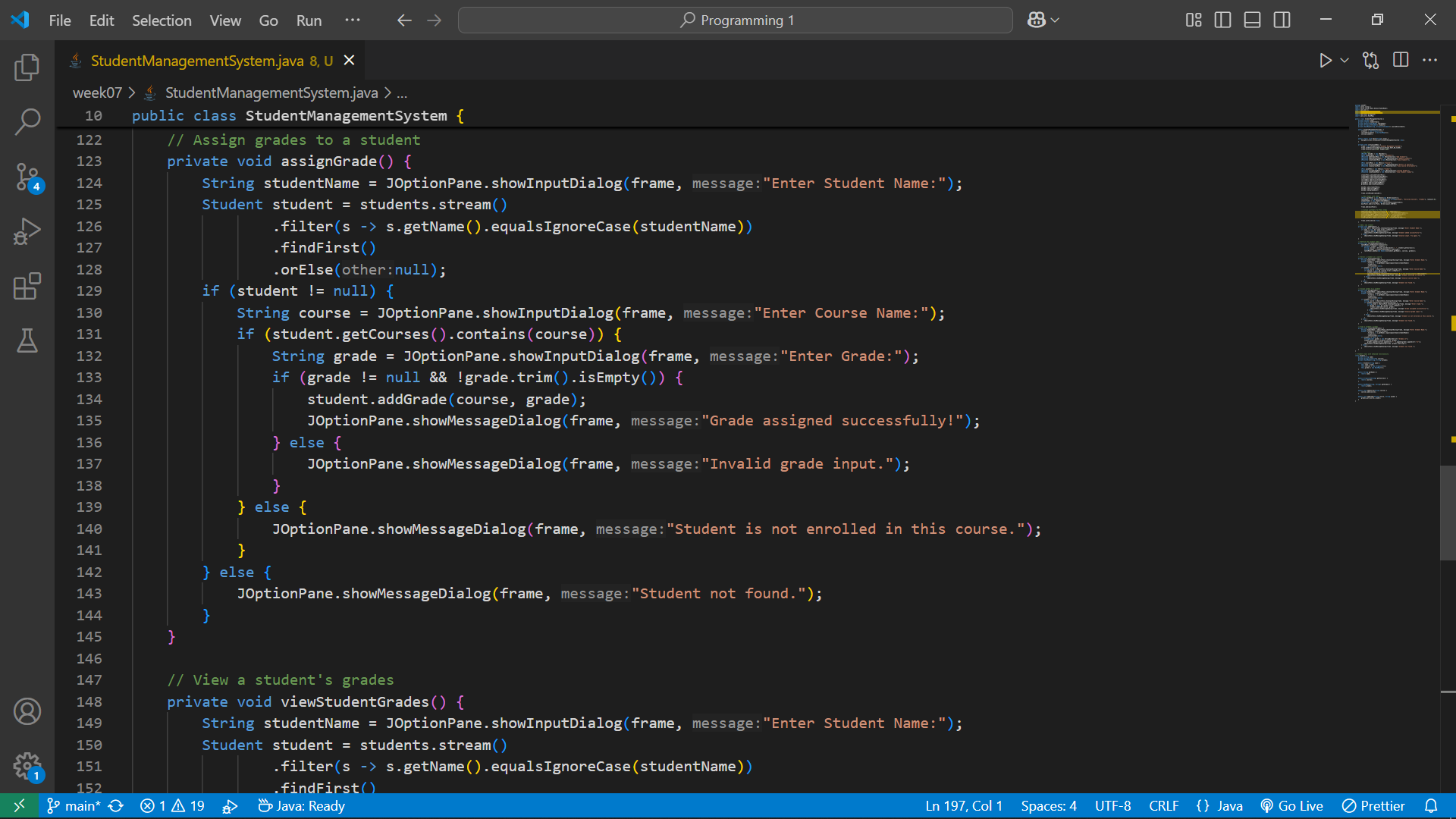
Codes

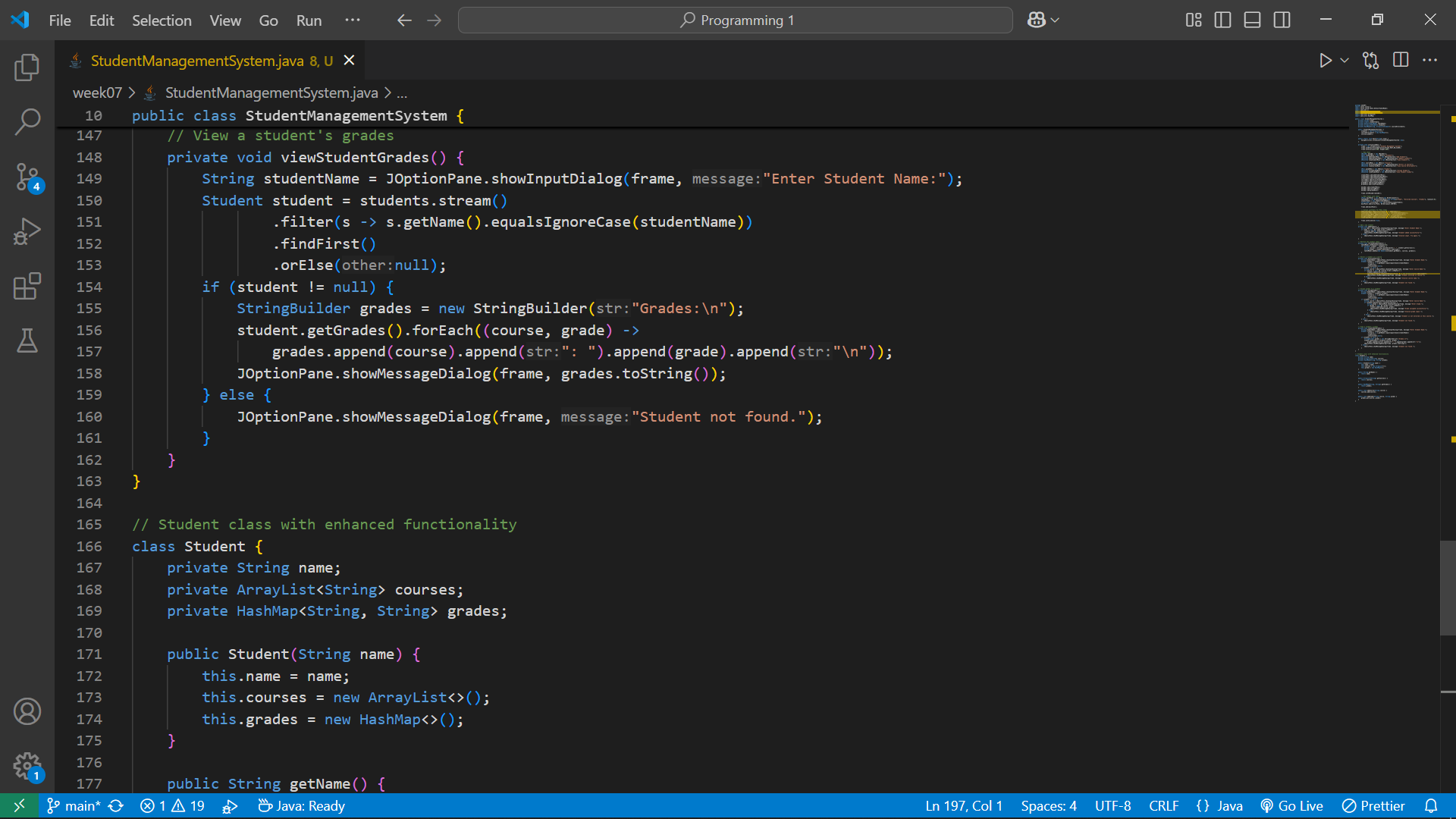


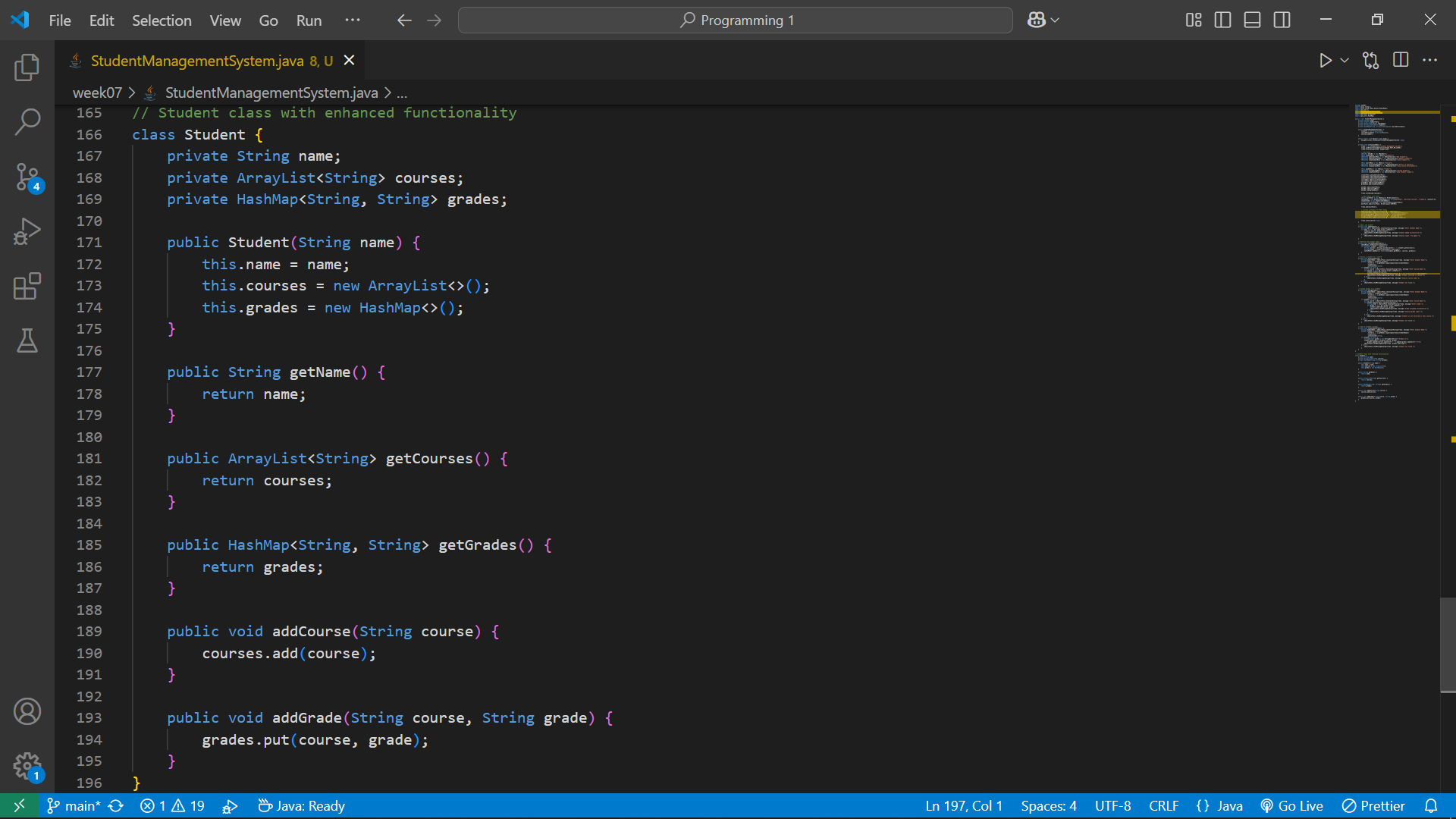




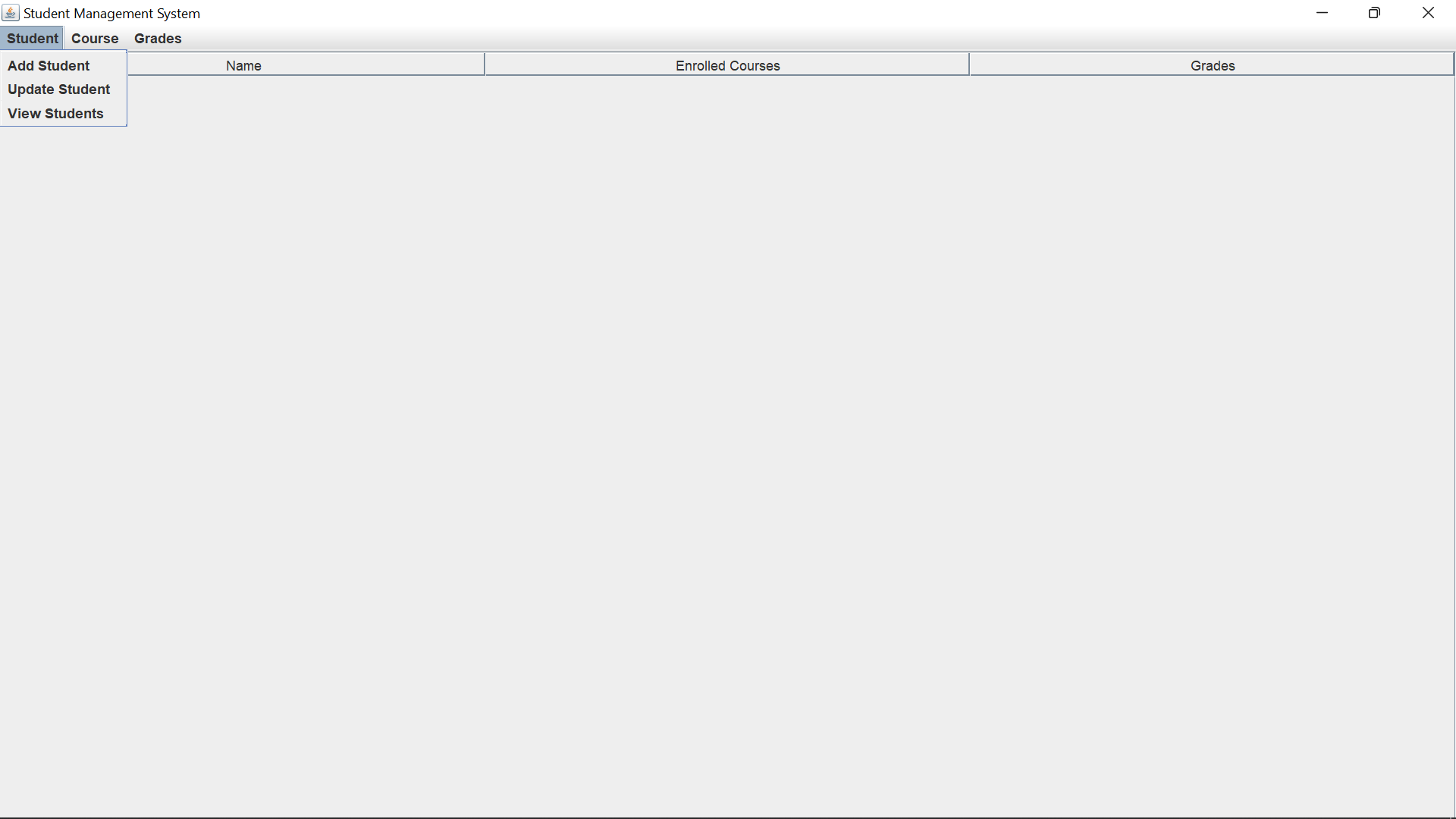


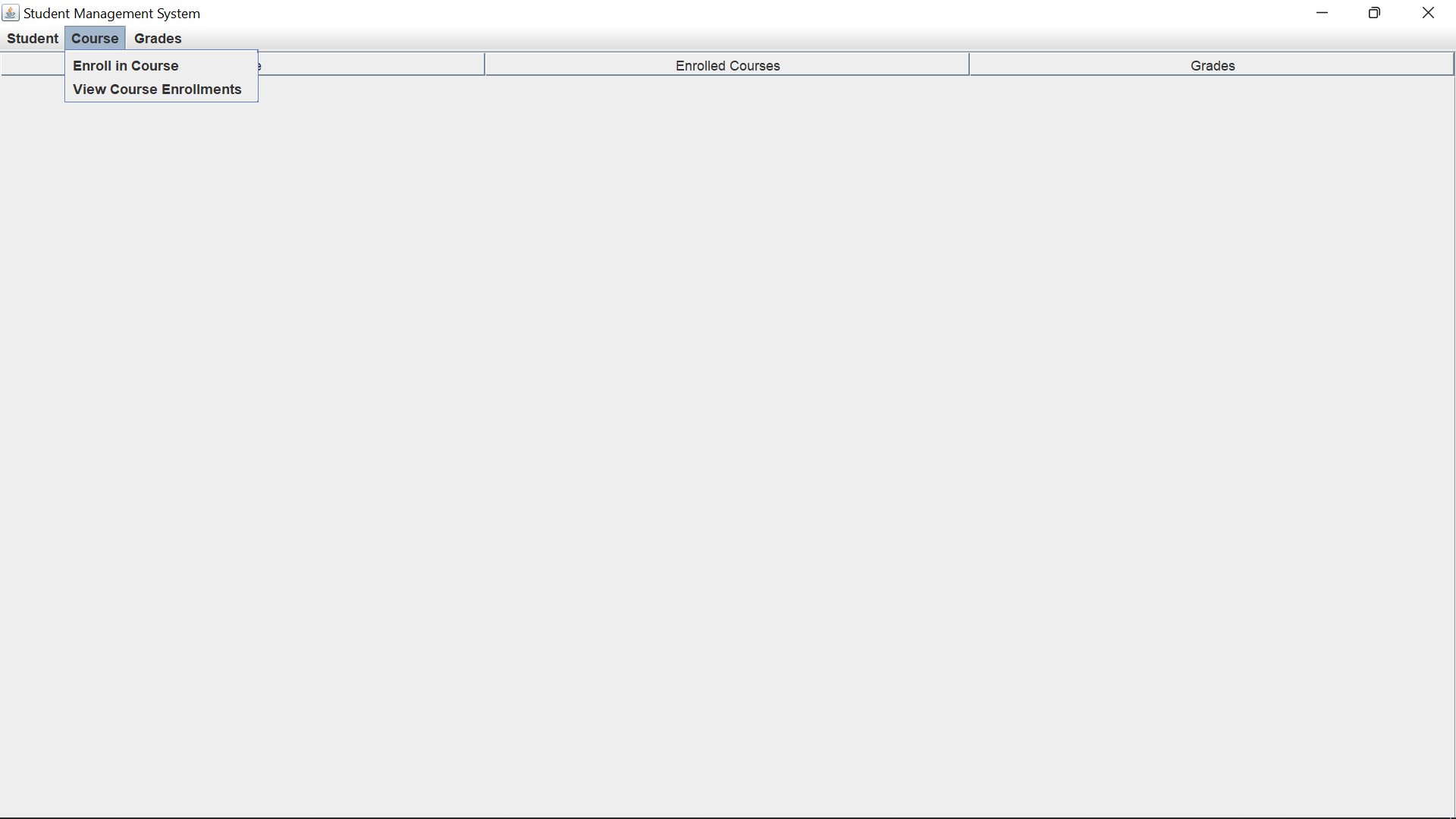


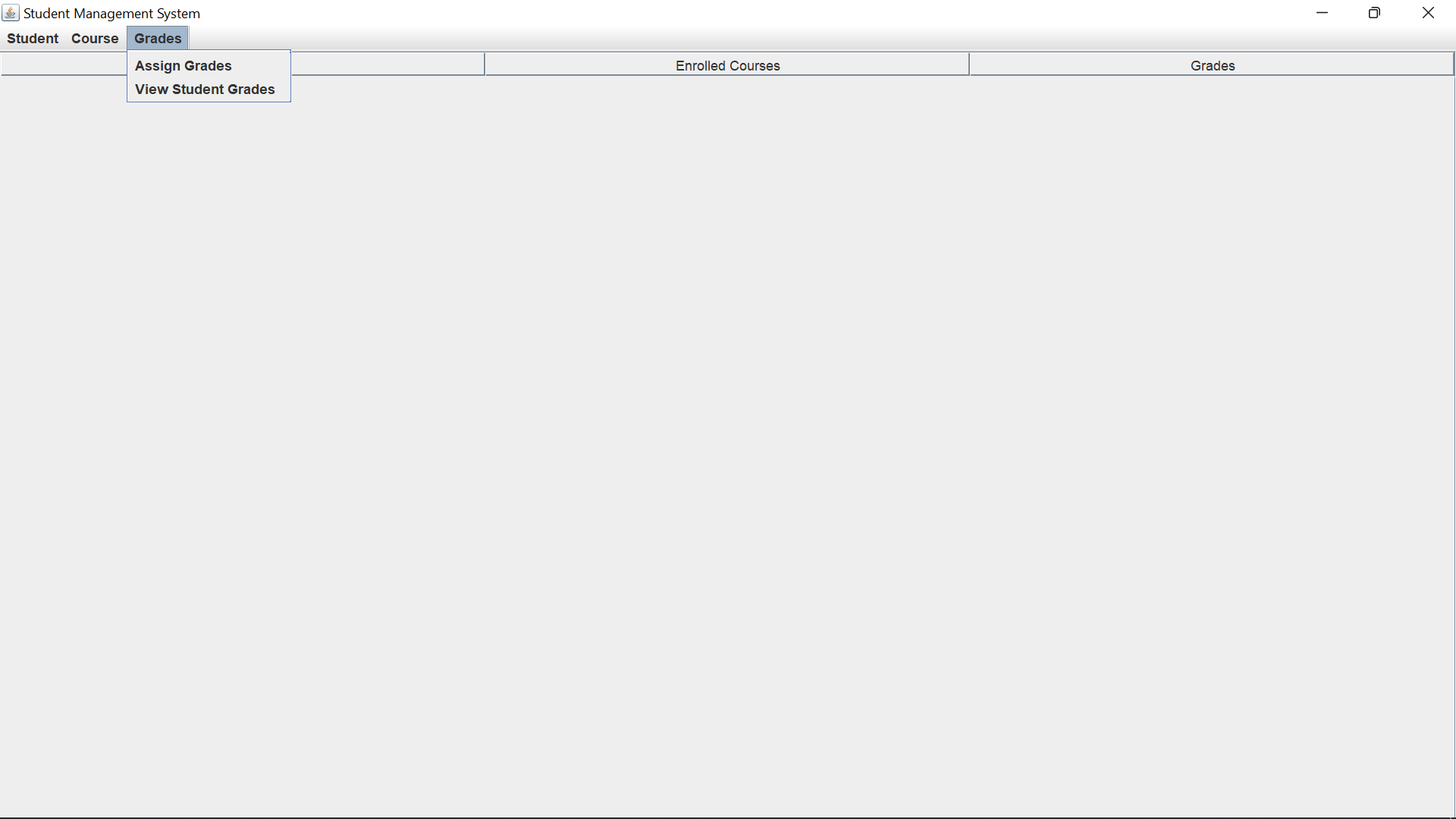




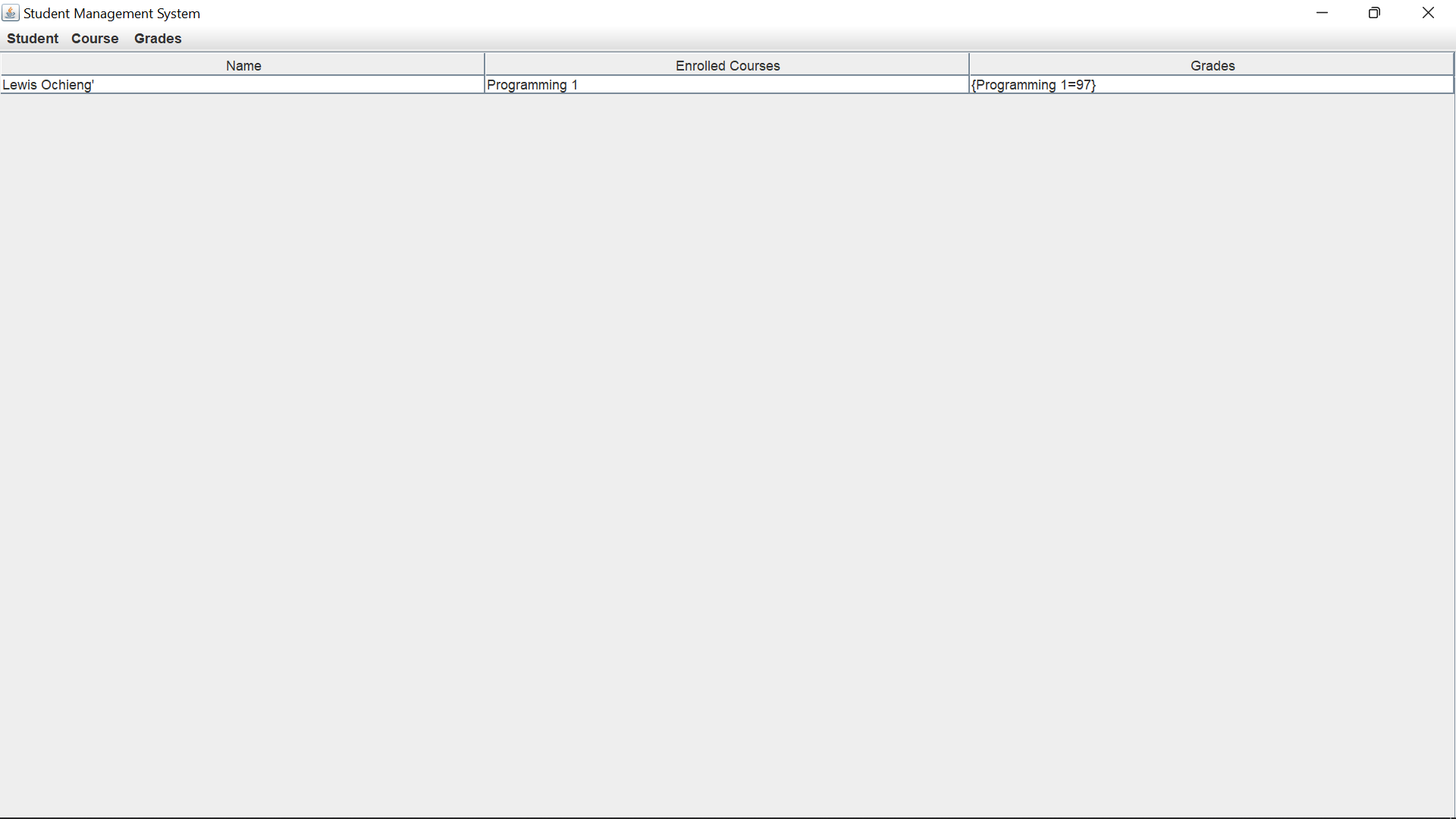
Output







This is how it looks after a student has been added, enrolled to a course and received a grade.



Error Handling trying to find a student that is not yet enrolled

