

VICTORIA UNIVERSITY

OBJECT ORIENTED PROGRAMMING

GROUP I

KIDULA MARK

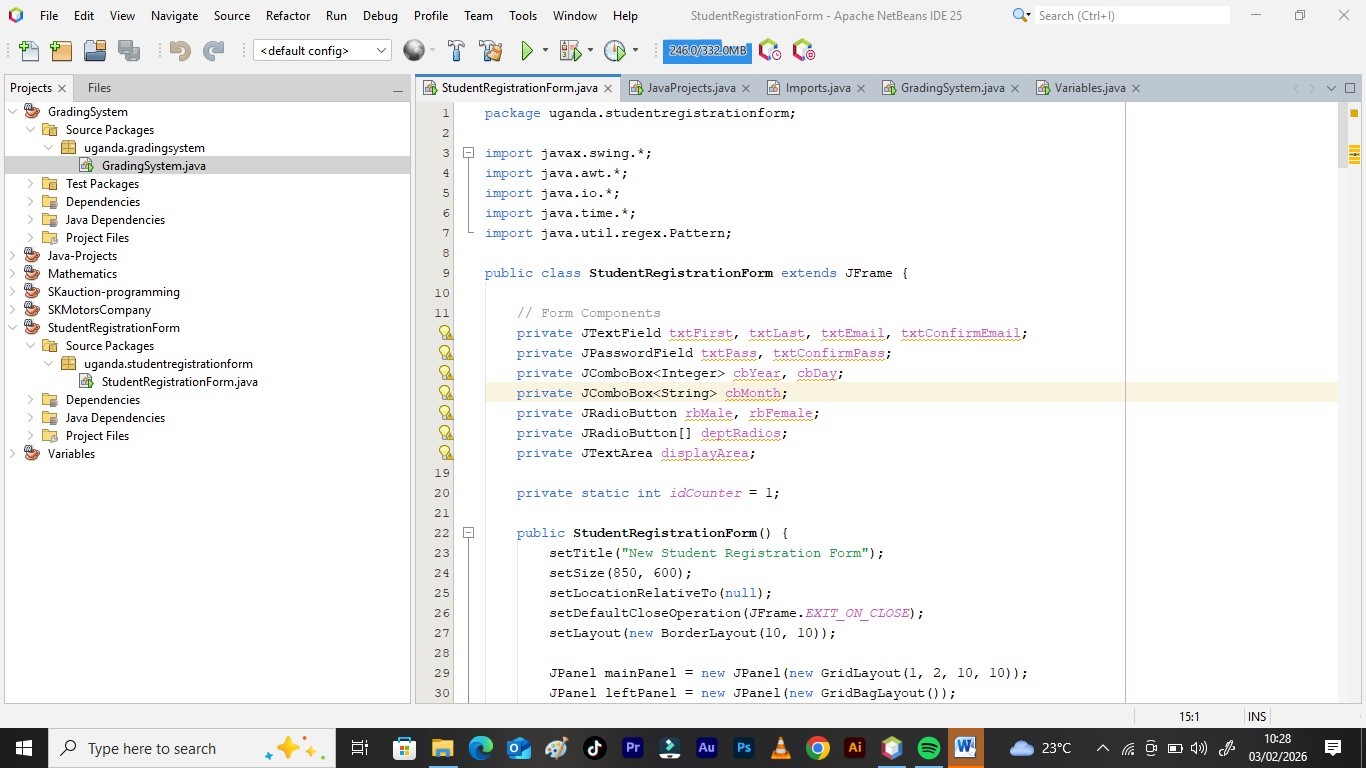
WADABA ALEX

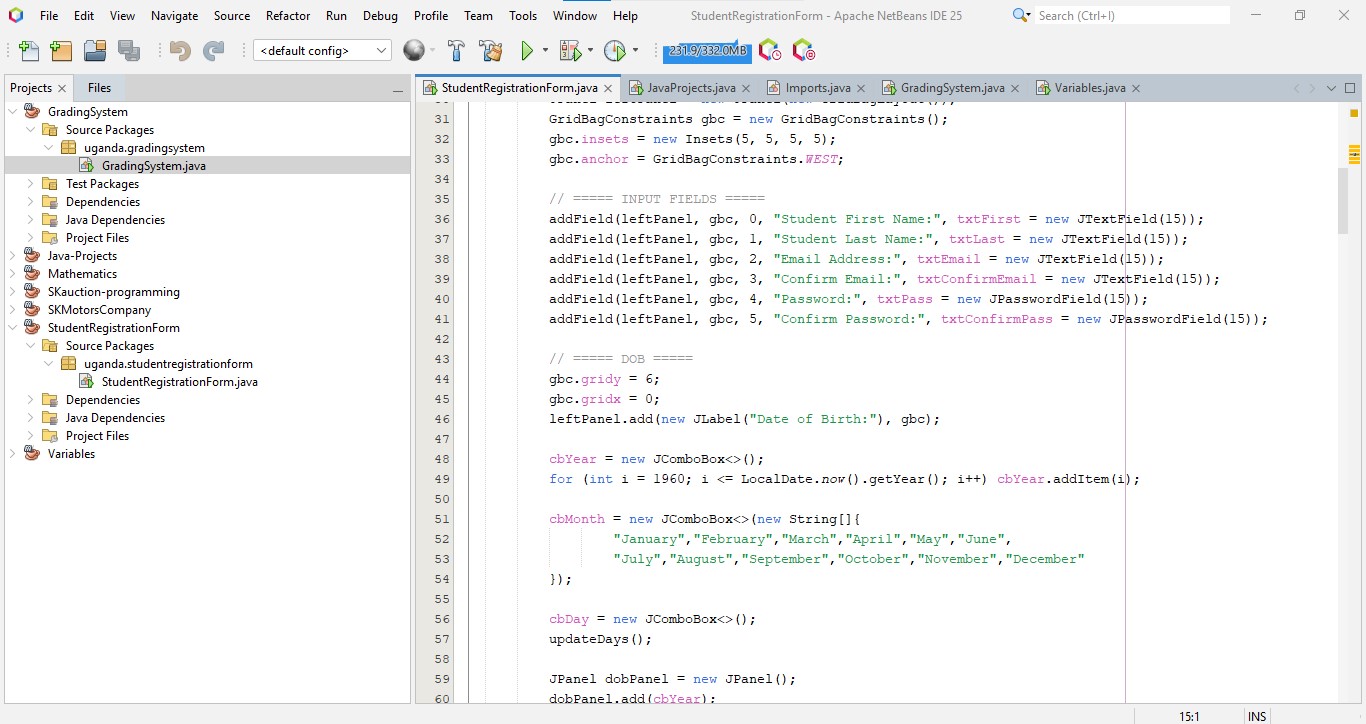
NANNONO SUMAYIYA VU-BIT-2503-1511-EVE

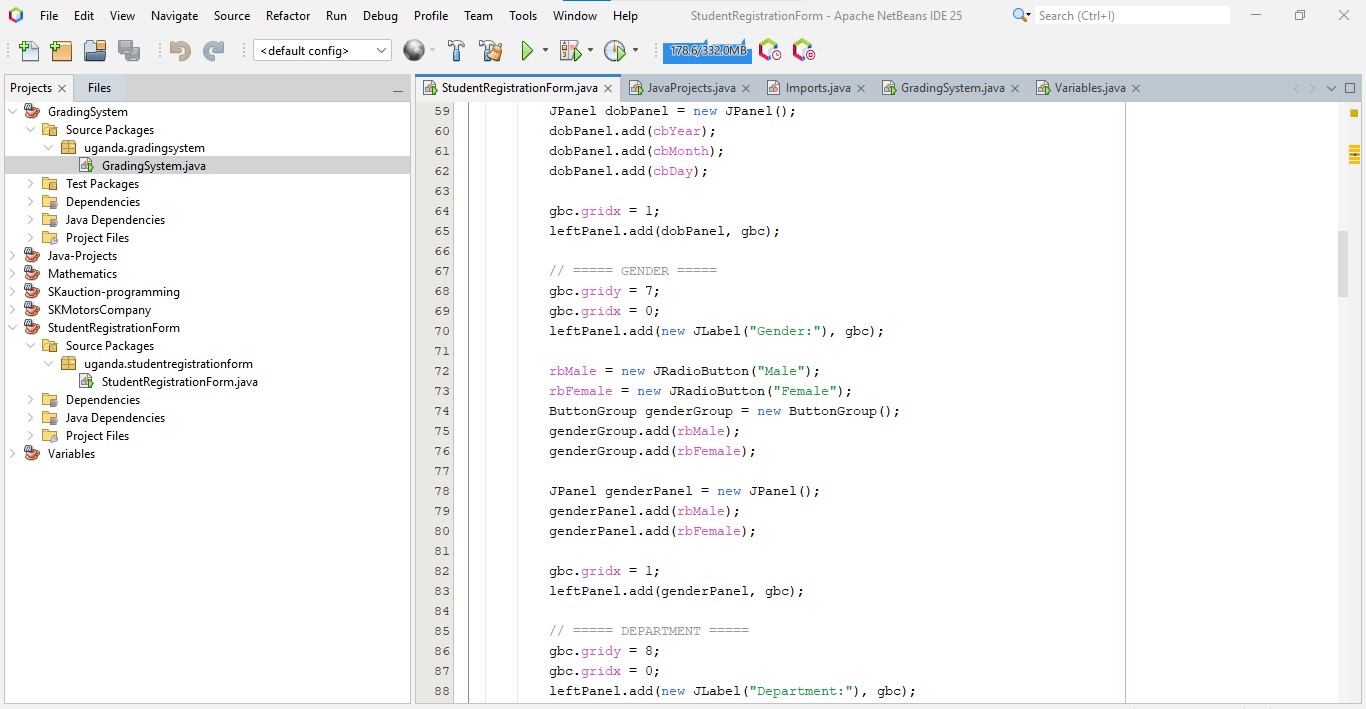
KAYIIRA TONNY

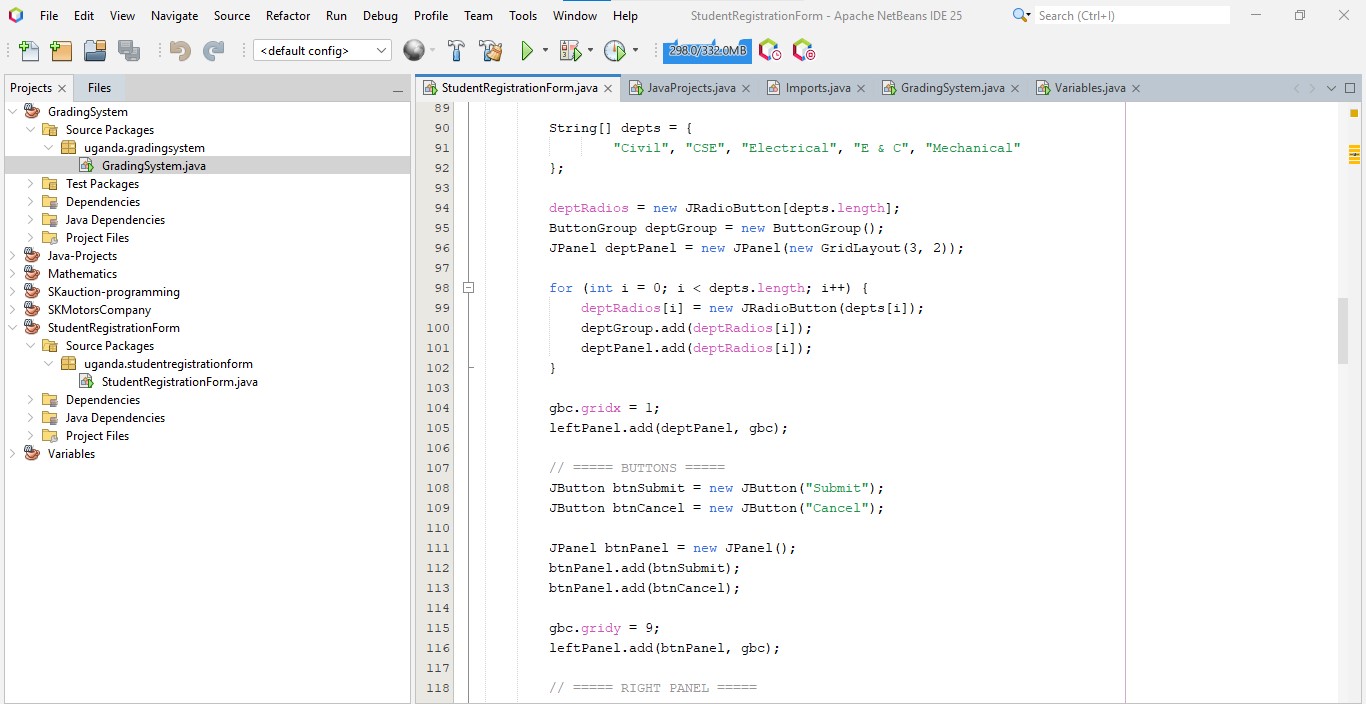
KASOZI UTHUMAN

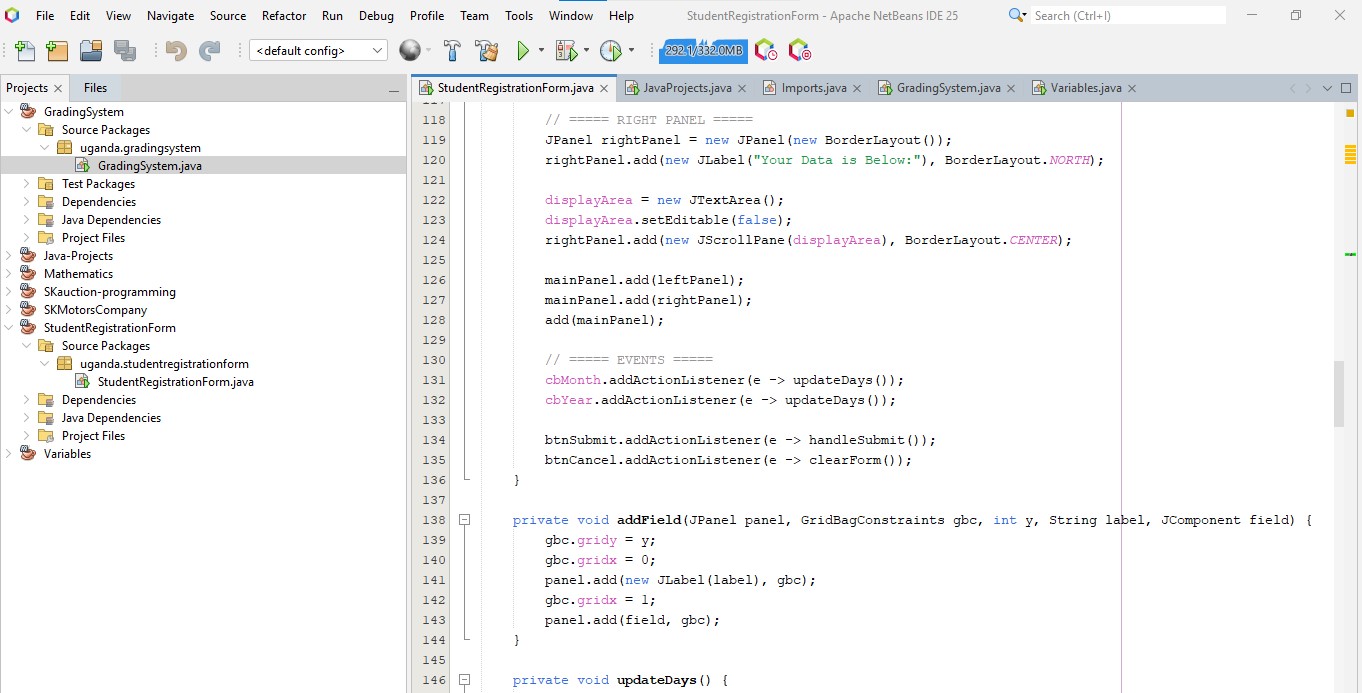
**QUESTION ONE A**

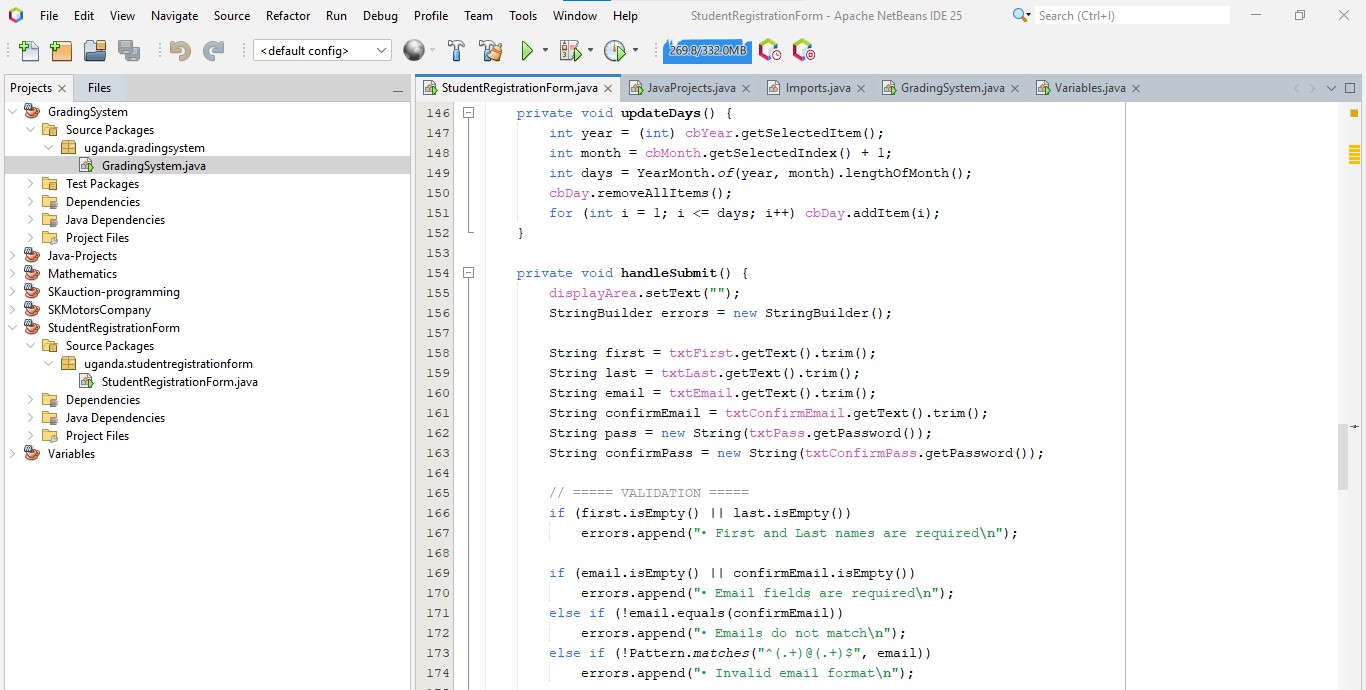


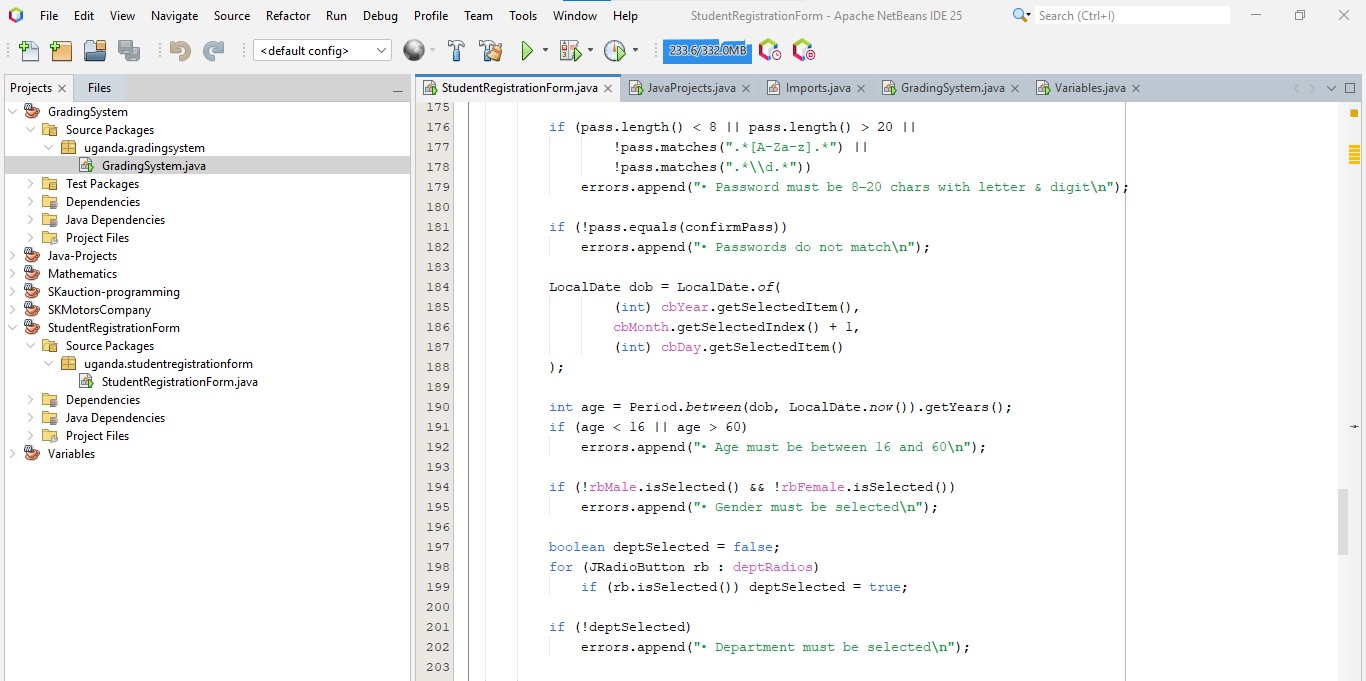


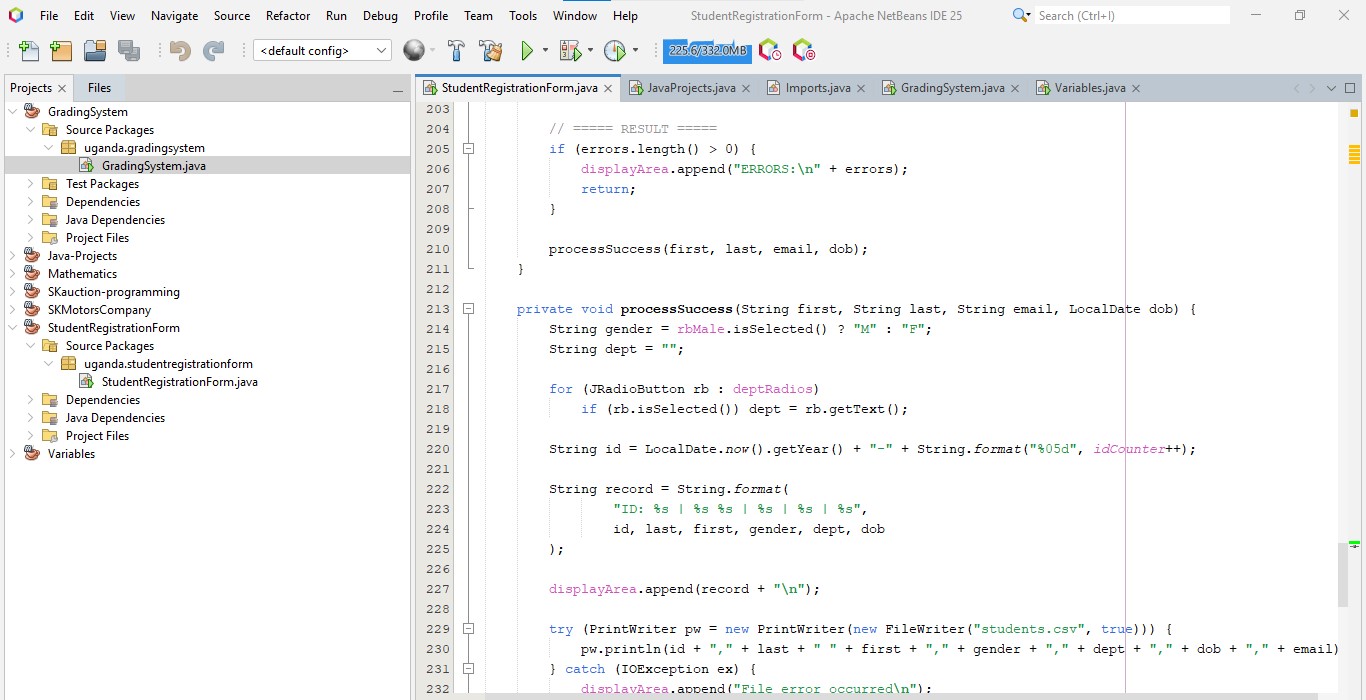


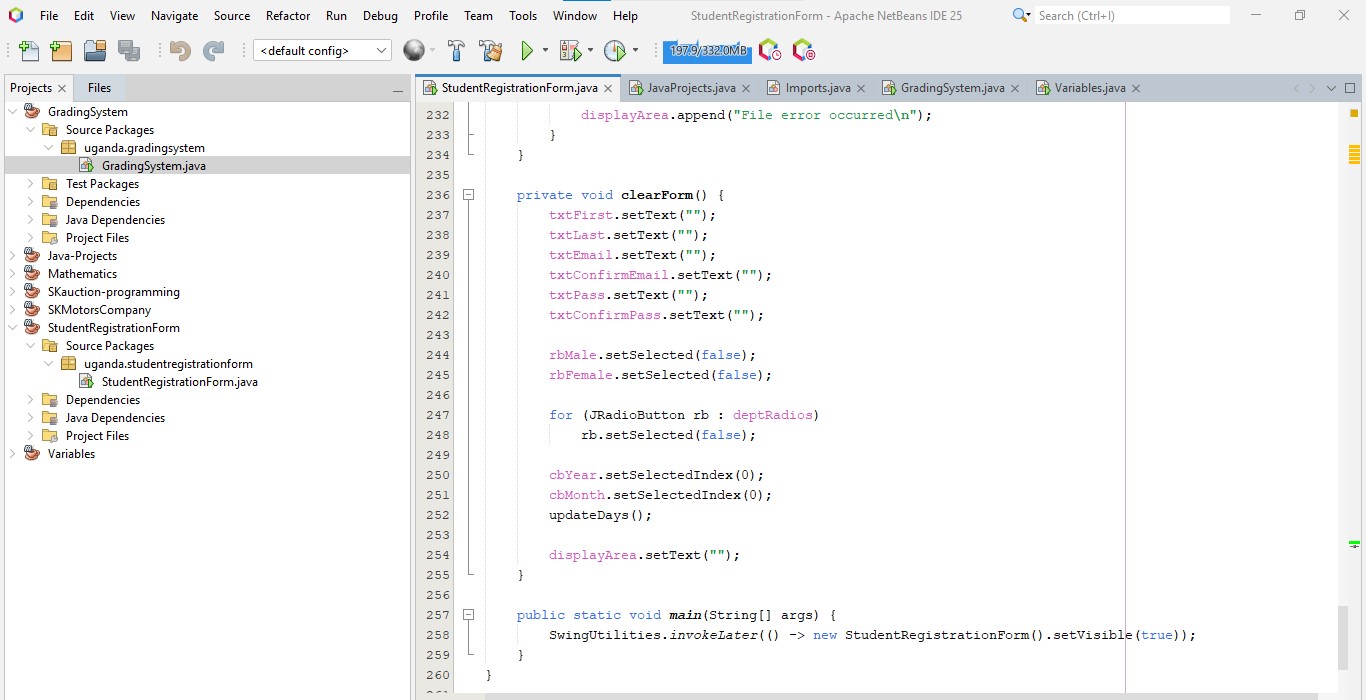


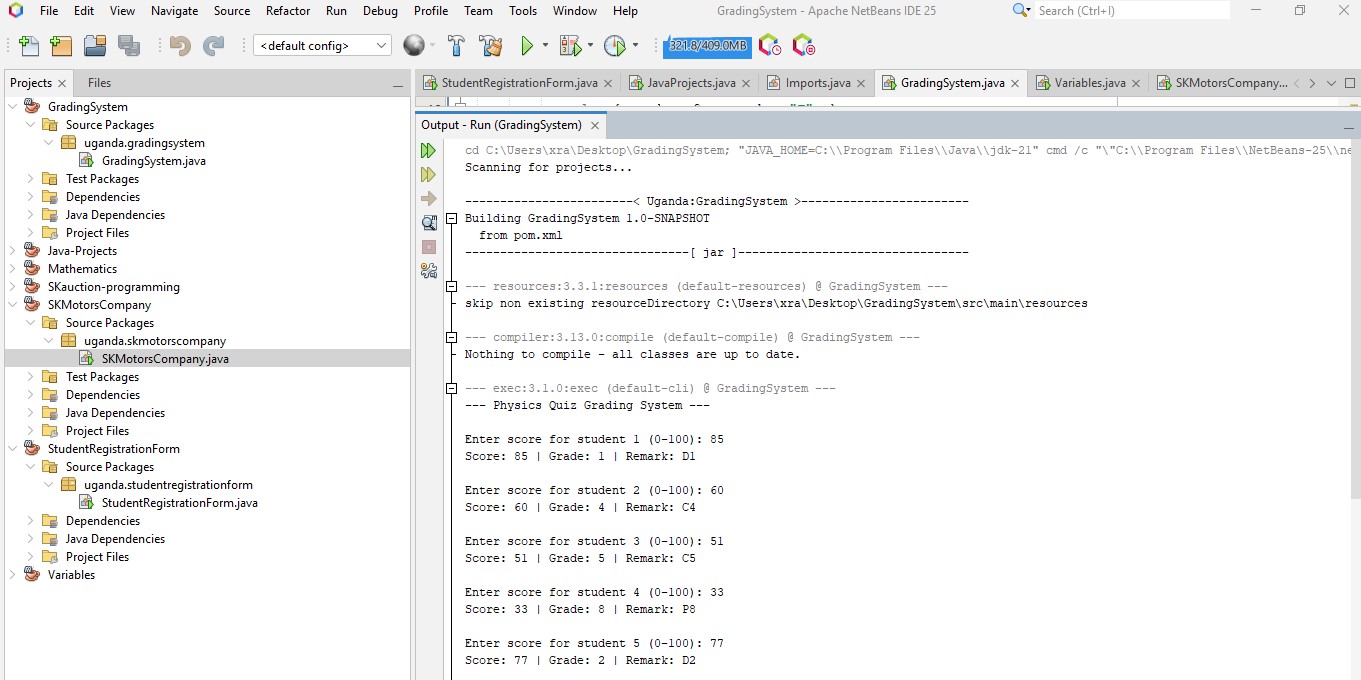


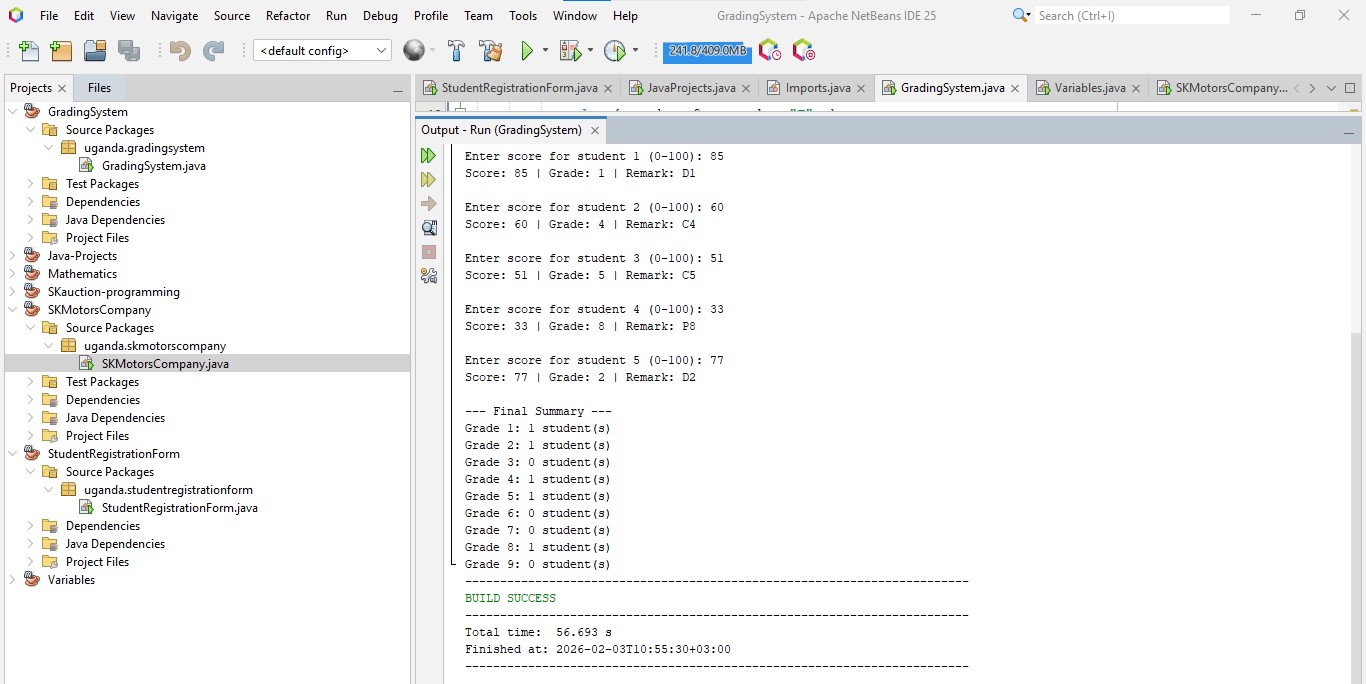












https://github.com/nannonosumayiya-coder/GRADING-APPLLICATION

**PROJECT OVERVIEW**

The program automates the process of evaluating student scores, assigning specific academic grades and remarks, and generating a statistical summary of performance.

The **Grading System** application processes results for a fixed batch of five students. It utilizes a controlled loop to capture inputs and uses a multi-tier conditional structure to categorize performance based on standard grading scales.

### Key Features

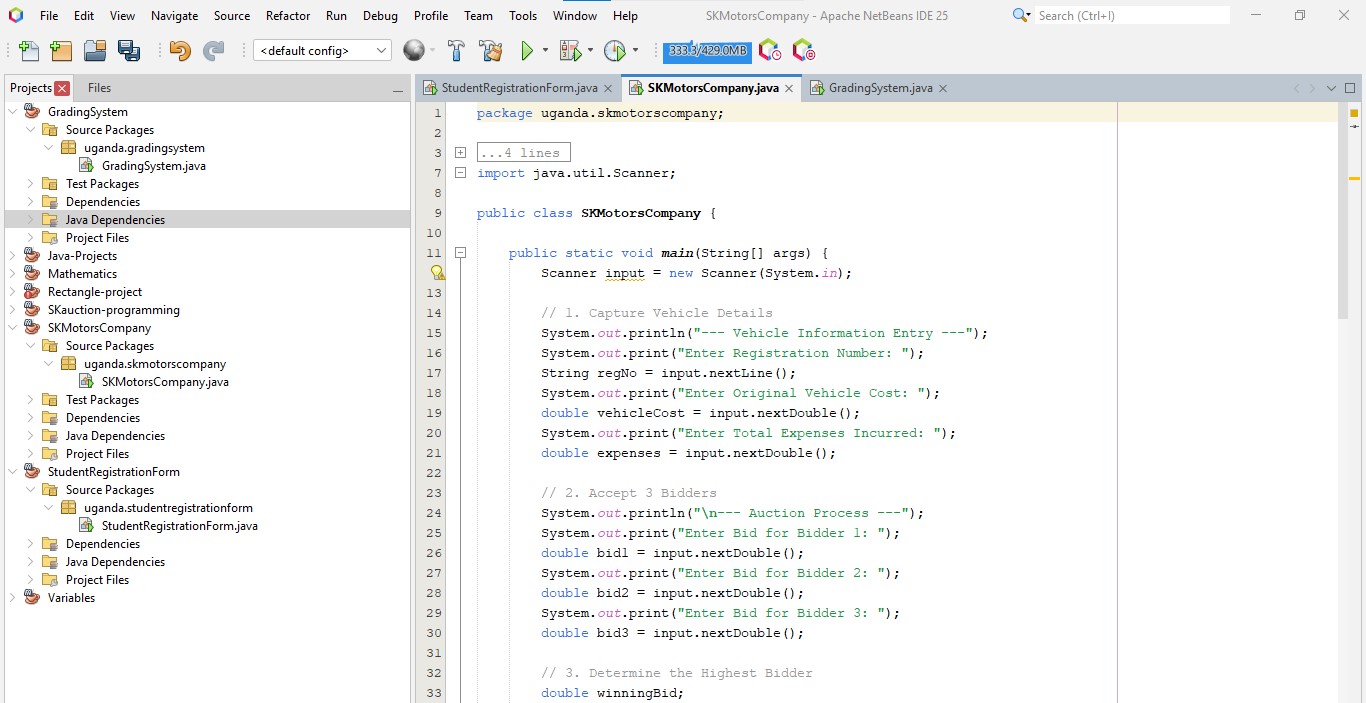
* **Iterative Processing:** Uses a while loop to process exactly five student records sequentially.
* **Input Validation:** Ensures that all entered scores fall within the logical range of **0 to 100**; invalid entries trigger an error message and a retry.
* **Grading Logic:** Implements a detailed if-else-if ladder to assign grades (1–9) and remarks (e.g., D1, C3, P7, F).
* **Summary Statistics:** Utilizes an array (grade Counts) to keep a running tally of how many students achieved each specific grade.

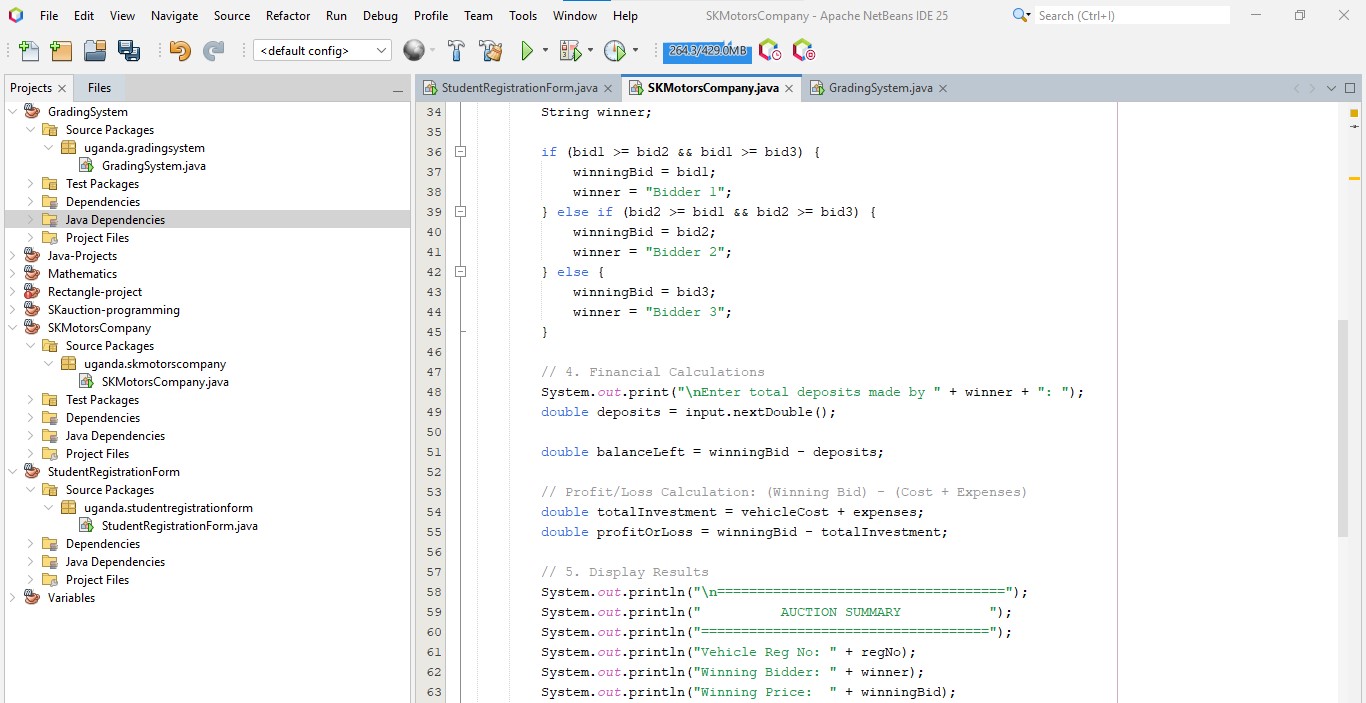
### Grading Scale Reference

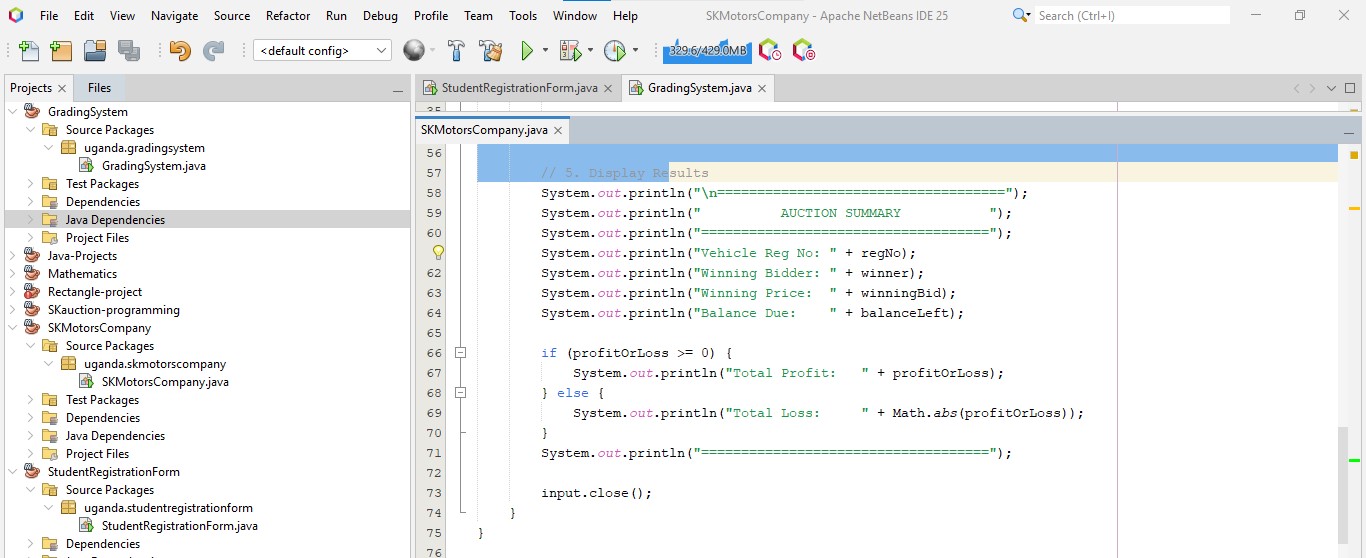
The system evaluates scores based on the following threshold table:

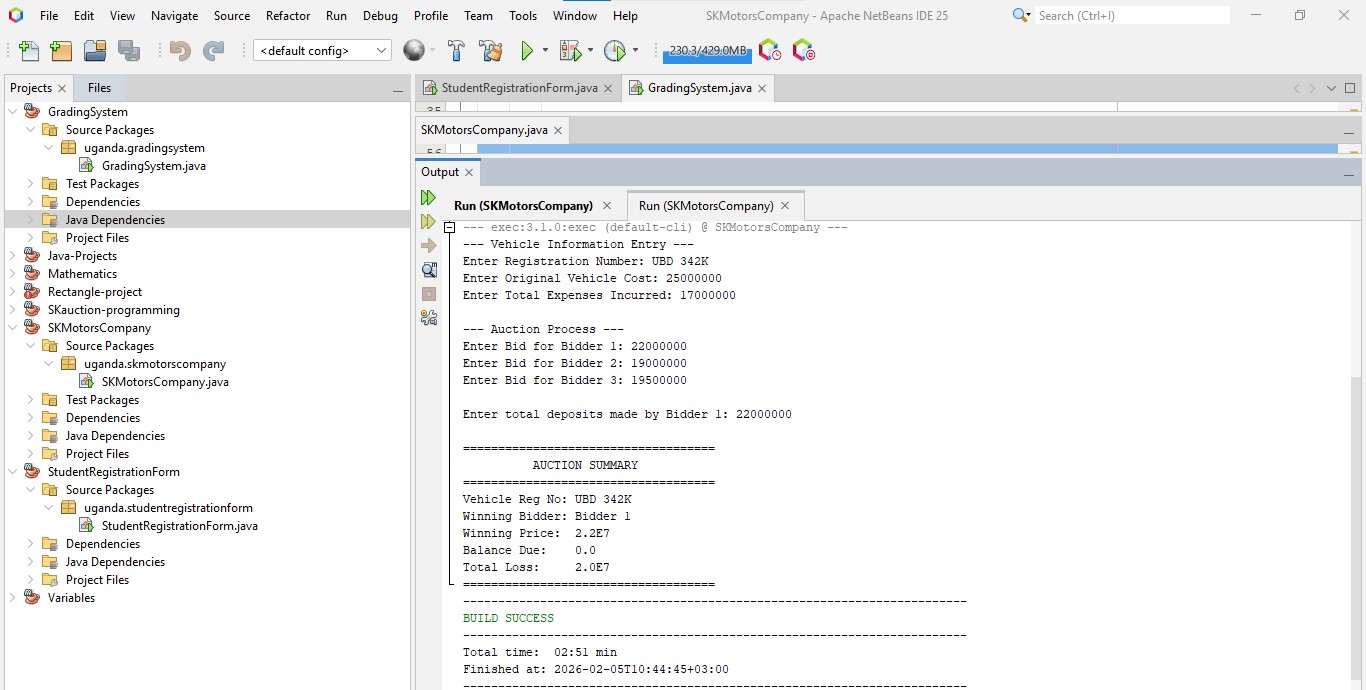
| **Score Range** | **Grade** | **Remark** |
| --- | --- | --- |
| 80 - 100 | 1 | D1 (Distinction) |
| 75 - 79 | 2 | D2 (Distinction) |
| 66 - 74 | 3 | C3 (Credit) |
| 60 - 65 | 4 | C4 (Credit) |
| 50 - 59 | 5 | C5 (Credit) |
| 45 - 49 | 6 | C6 (Credit) |
| 35 - 44 | 7 | P7 (Pass) |
| 30 - 34 | 8 | P8 (Pass) |
| 0 - 29 | 9 | F (Fail) |

**QUESTION ONE B**

****

****





**PROJECT OVERVIEW**

The **SK Motors Company** application streamlines the auctioning of vehicles by automating the comparison of bids and calculating the remaining balances for winning bidders. It provides a clear summary of the transaction, including the net financial outcome for the company based on the initial investment.

**Key Features**

* **Vehicle Data Entry:** Captures the unique registration number, original purchase cost, and additional expenses incurred.
* **Bidding Logic:** Accepts three separate bids and automatically identifies the highest offer.
* **Financial Tracking:** \* Calculates the **Balance Due** after subtracting deposits from the winning bid.
* Computes **Profit or Loss** by comparing the winning bid against the sum of the vehicle cost and expenses.

**Detailed Summary:** Generates a formatted report for the final transaction.

**How It Works**

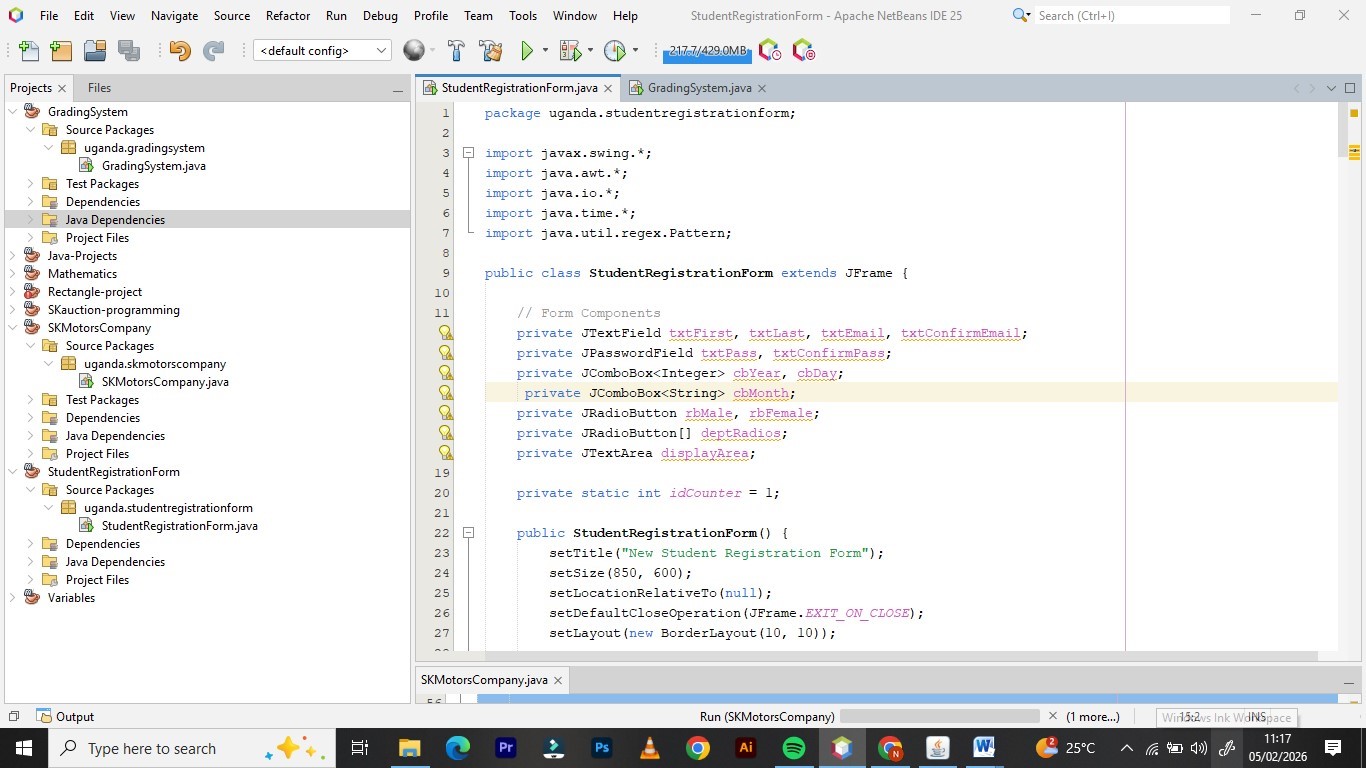
The program follows a linear execution flow:

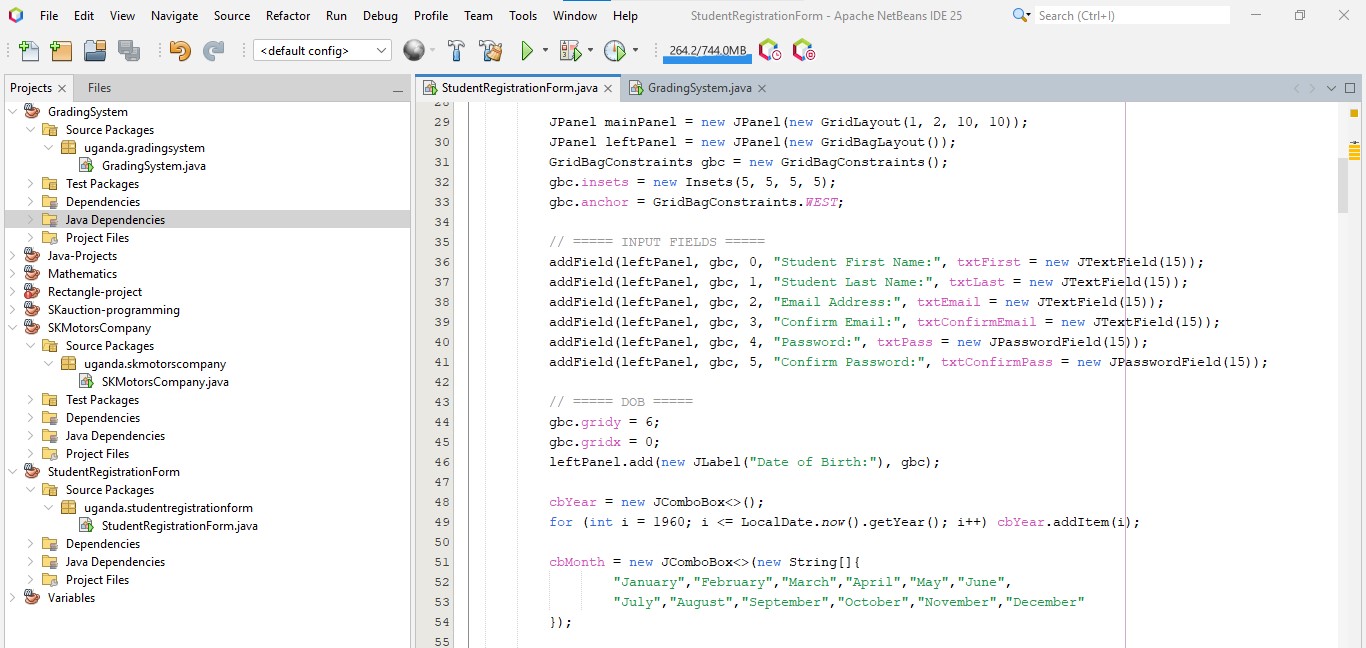
* **Input Phase:** The user enters the vehicle's "pedigree" (Cost and Reg No).
* **Auction Phase:** Three bidders submit their amounts.
* **Evaluation Phase:** The program uses a conditional logic block to determine the winningBid:
  + $WinningBid = \max(bid\_1, bid\_2, bid\_3)$
* **Closing Phase:** The user enters the deposits made by the winner, and the system outputs the final AUCTION SUMMARY.

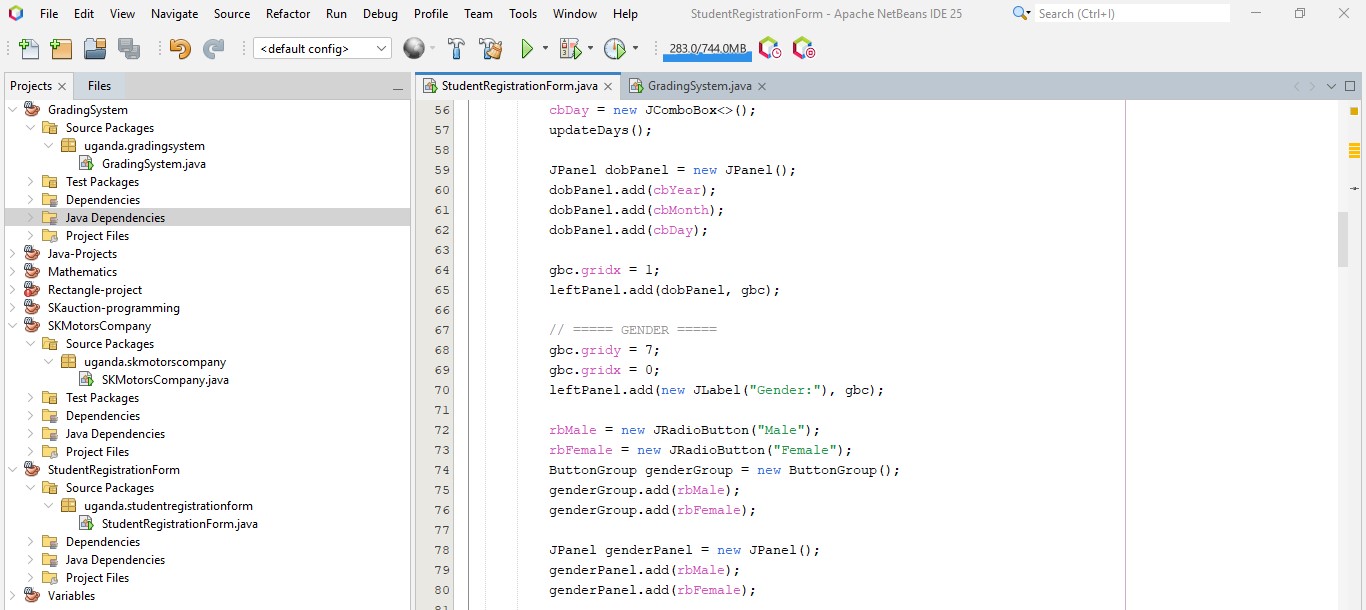
**Technical Specifications**

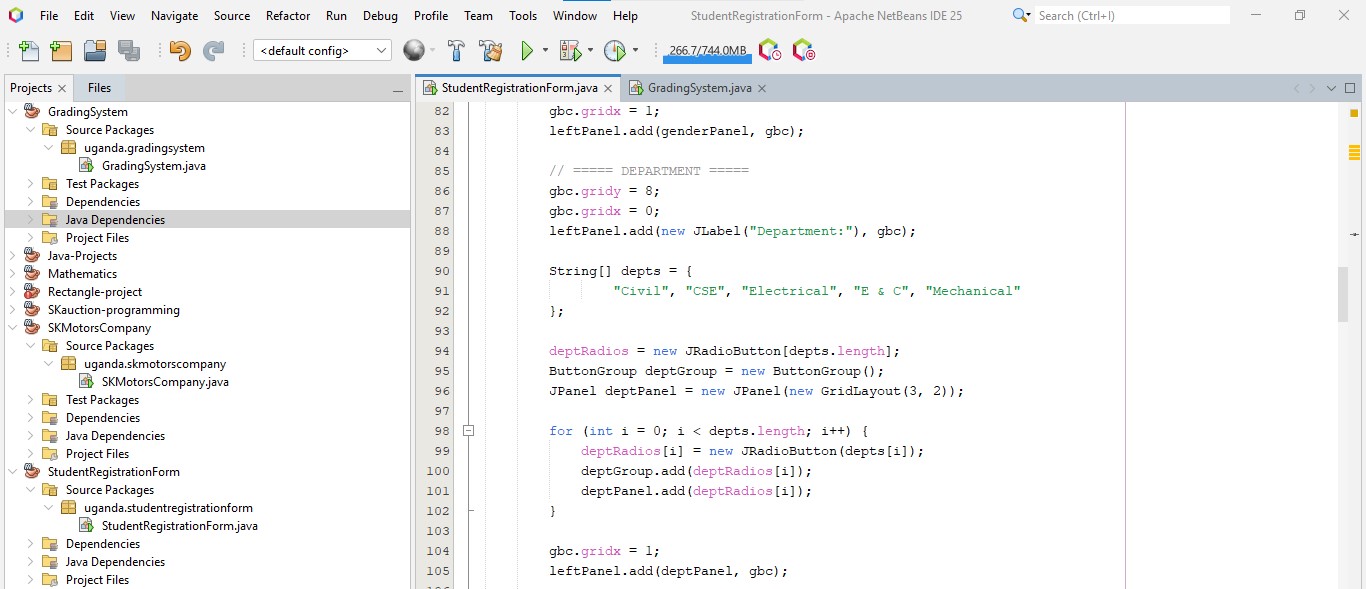
* **Language:** Java
* **Input Method:** java.util.Scanner
* **Logic Structures:** If-else decision trees for bid comparison and profit/loss branching.
* **Math Operations:** Absolute value conversion (Math.abs) for displaying loss figures cleanly.

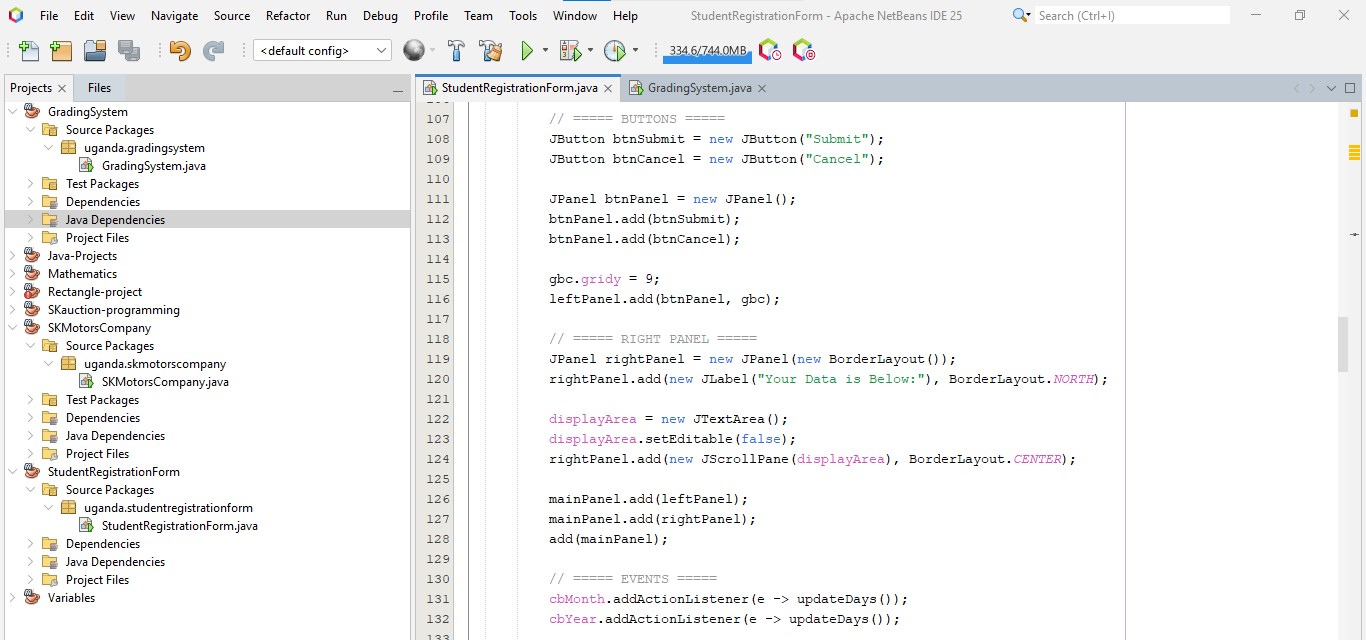
**QUESTION TWO**

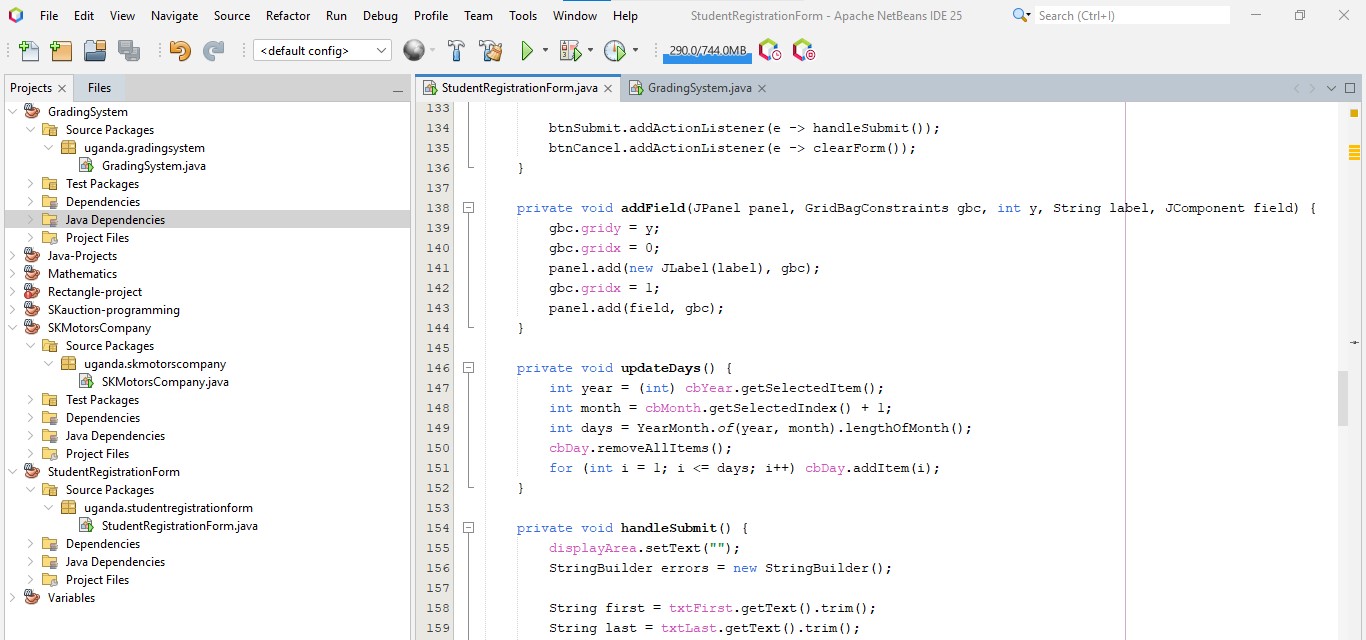
****

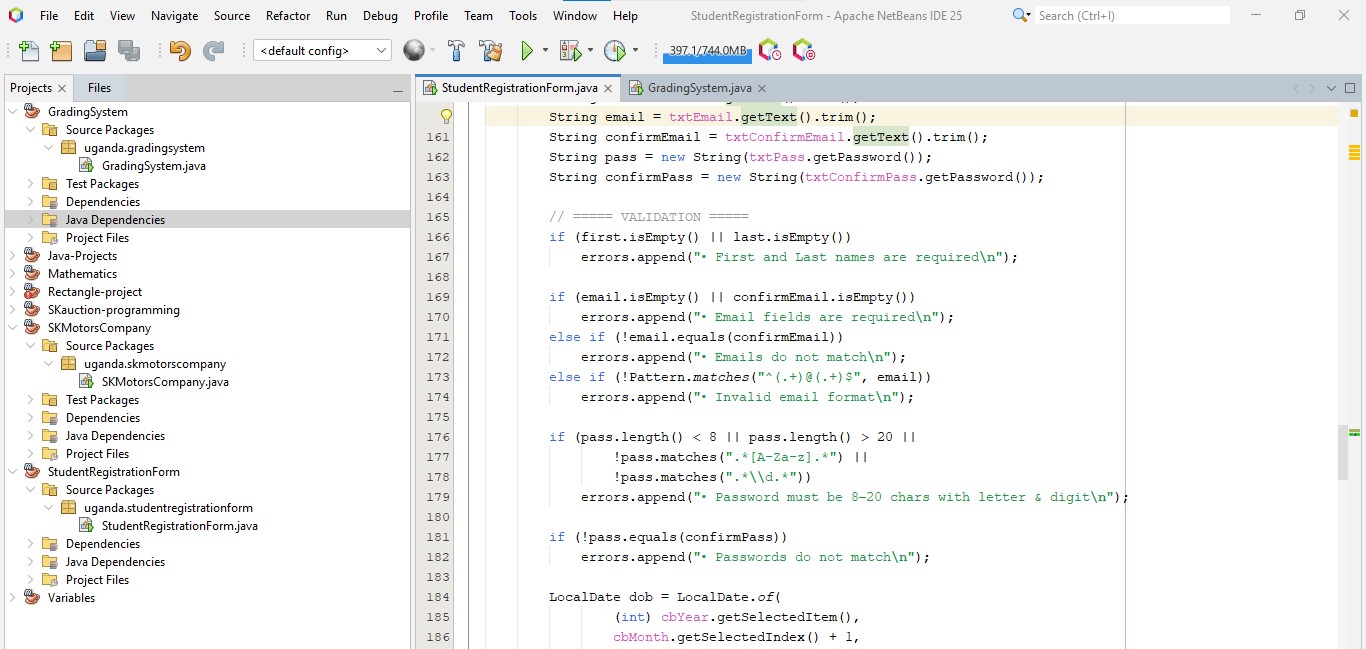


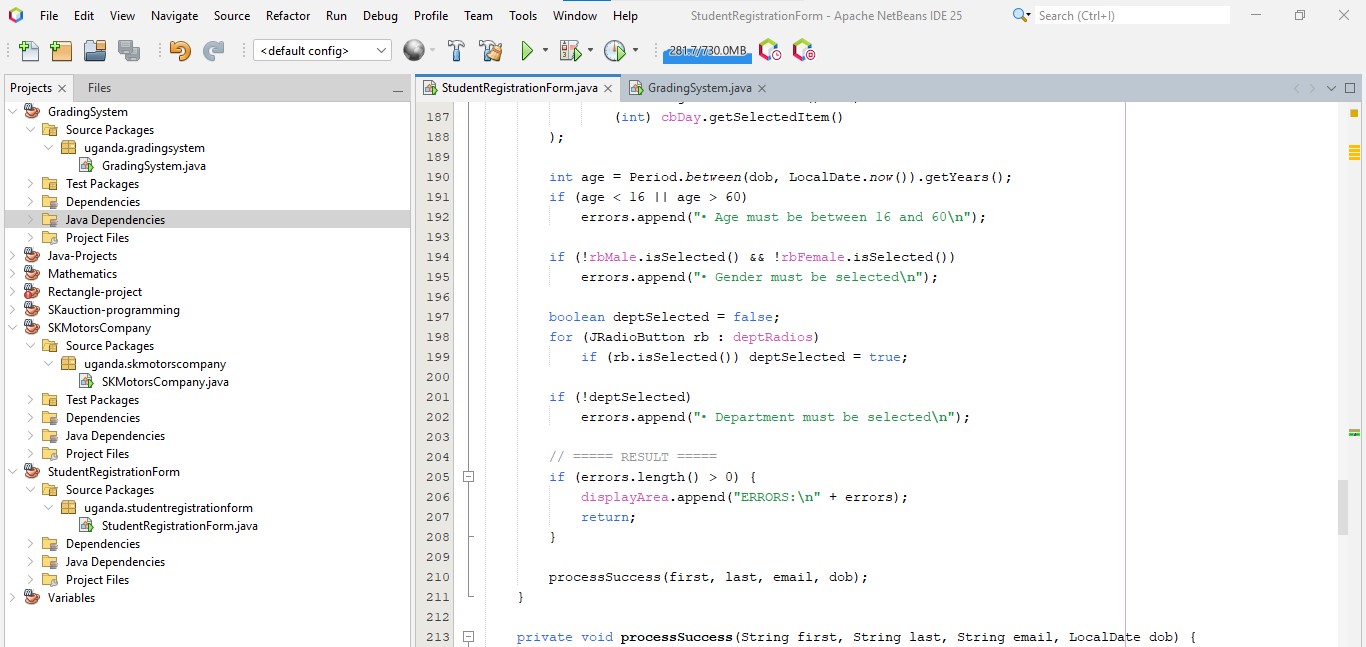


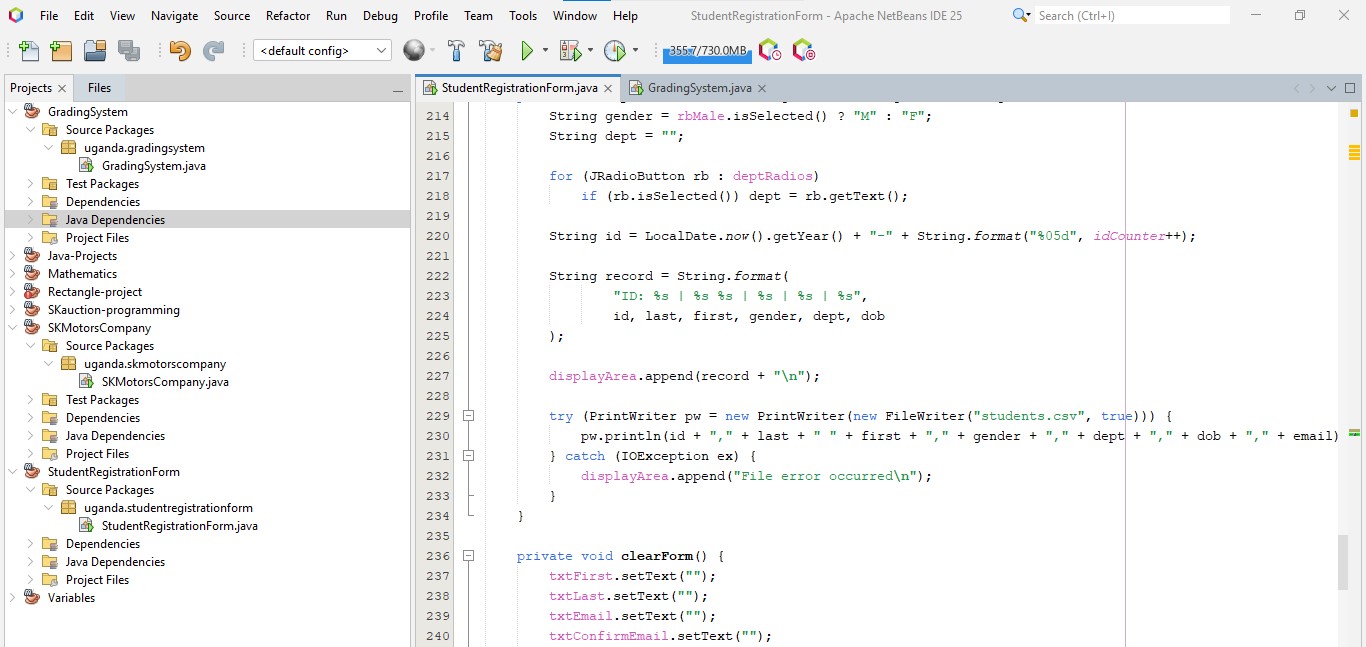


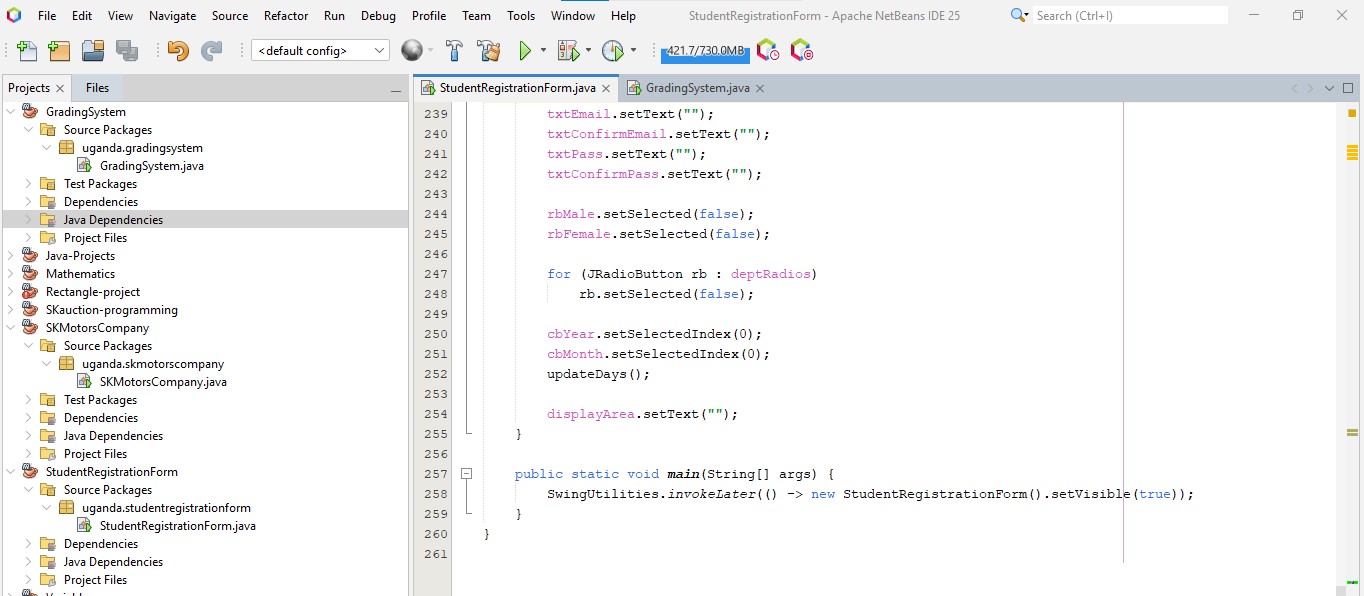


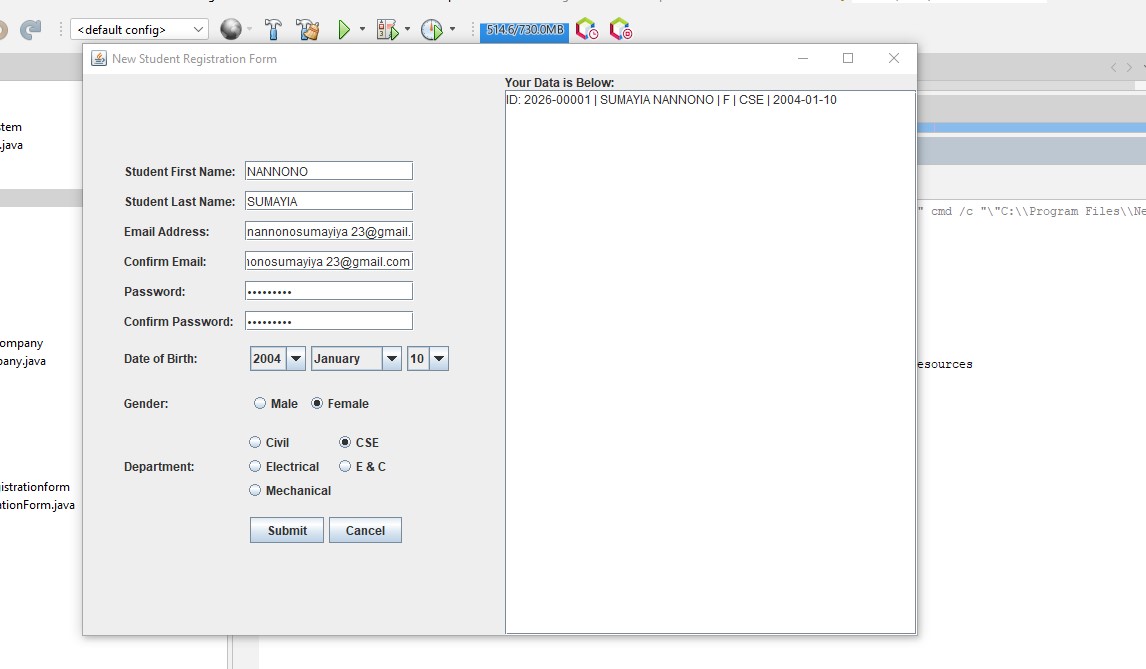












### PROJECT OVERVIEW

The **Student Registration Form** is a comprehensive data entry tool designed for academic institutions. It features a split-pane interface where the left side is dedicated to a structured input form and the right side acts as a live data preview area. The system is built to ensure data integrity by enforcing strict validation rules before any record is finalized.

### Key Features

* **Comprehensive Data Capture:** Collects personal information including name, email, date of birth, gender, and academic department.
* **Dynamic Date Selection:** Automatically calculates the correct number of days in a month based on the selected year and month (including leap year support).
* **Robust Input Validation:**
  + **Email Verification:** Checks for correct email formatting and ensures confirmation fields match.
  + **Password Security:** Enforces a length of 8–20 characters and requires a mix of letters and digits.
  + **Age Eligibility:** Restricts registration to individuals between the ages of 16 and 60.
* **Automated ID Generation:** Generates a unique student ID based on the current year and a sequential counter (e.g., 2026-00001).
* **Persistent Storage:** Successfully submitted records are appended to a local students.csv file for external spreadsheet analysis.

### Technical Specifications

* **GUI Library:** javax.swing and java.awt.
* **Layout Management:** Uses GridBagLayout for precise alignment of form fields and BorderLayout for the overall frame structure.
* **Data Validation:** Utilizes Regular Expressions (Regex) for email and password pattern matching.
* **File I/O:** Implements PrintWriter and FileWriter to handle comma-separated values (CSV) logging.

### How to Run

1. **Environment Setup:** Ensure you have the Java Development Kit (JDK) installed.
2. **Compilation:** javac StudentRegistrationForm.java
3. **Execution:** java uganda.studentregistrationform.StudentRegistrationForm
4. **Submission:** Fill out the required fields. If successful, the data will appear in the right-hand preview box and be saved to students.csv. If errors exist, they will be highlighted in the preview area.