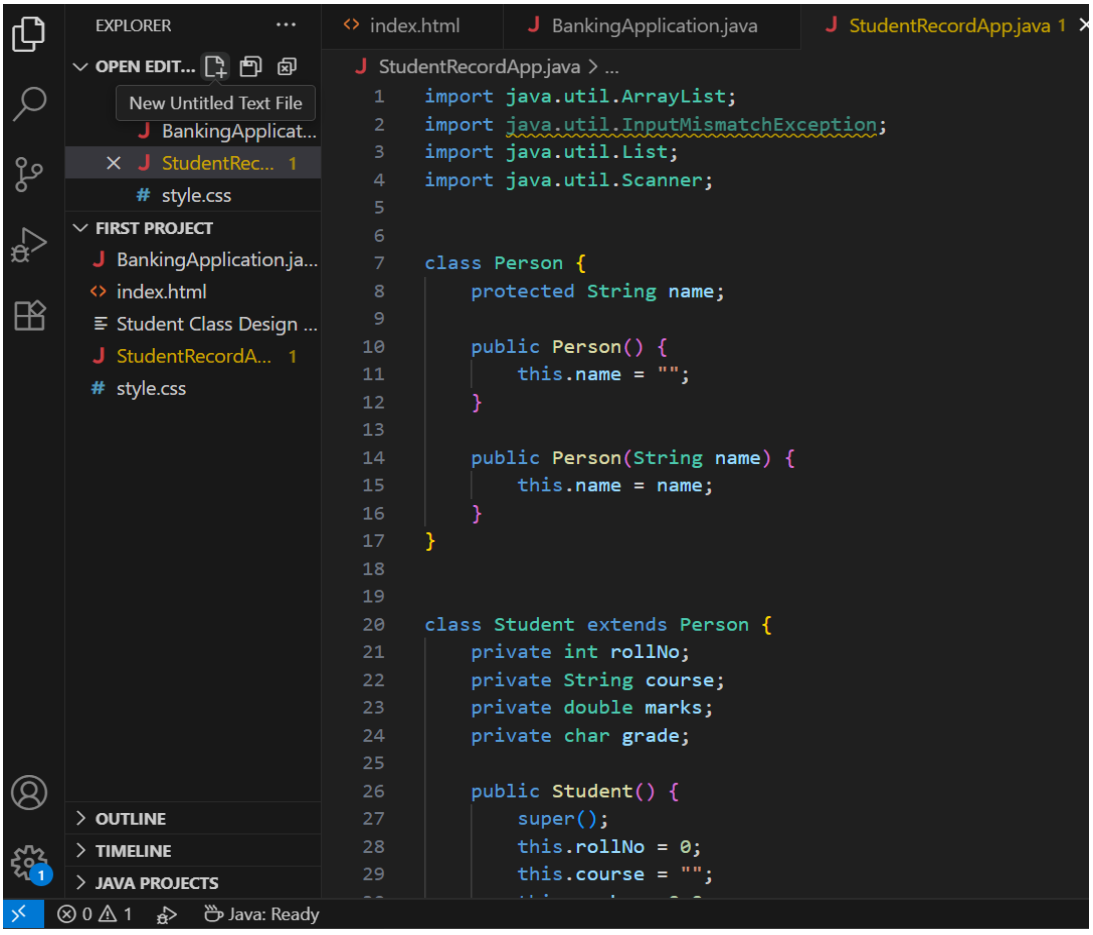
JAVA LAB ASSIGNMENT NUMBER: 01

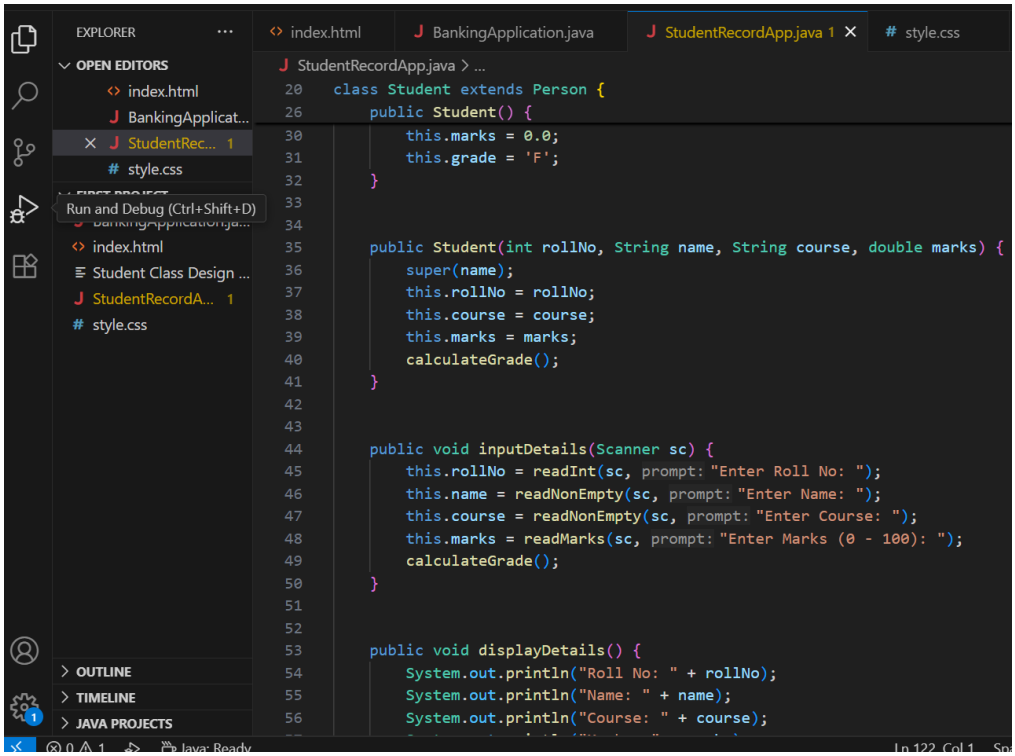
Name: Aman

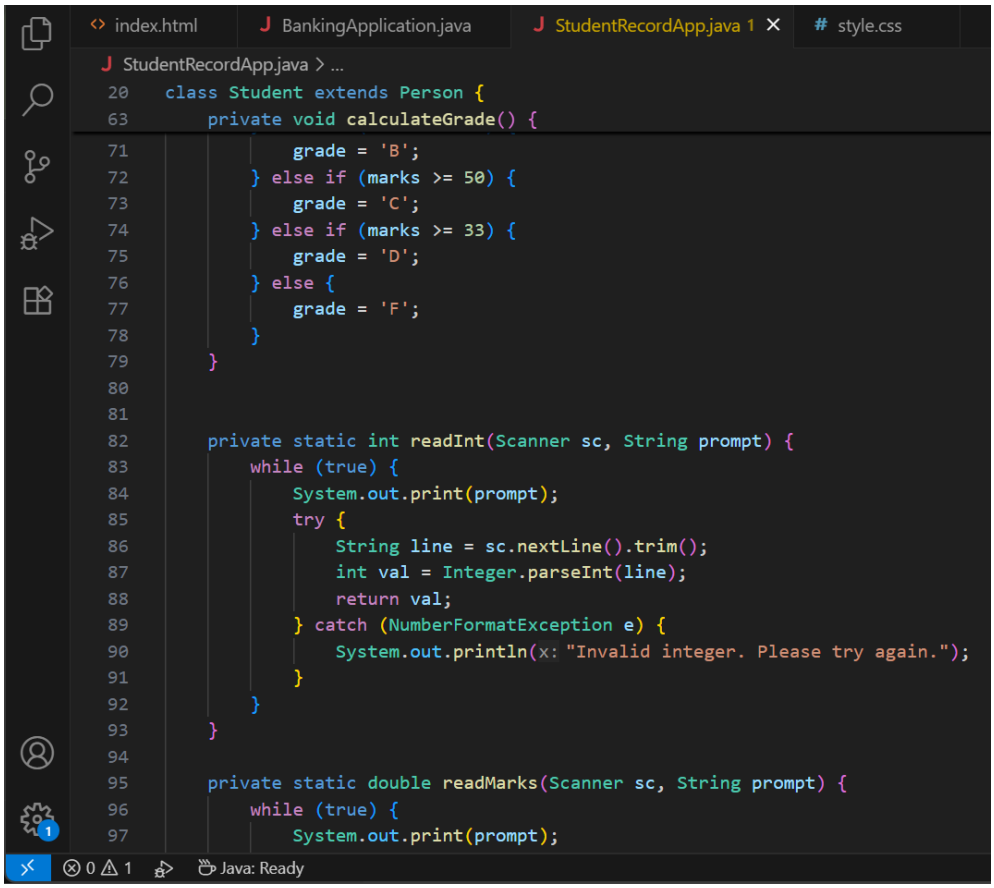
Roll no: 2401201115

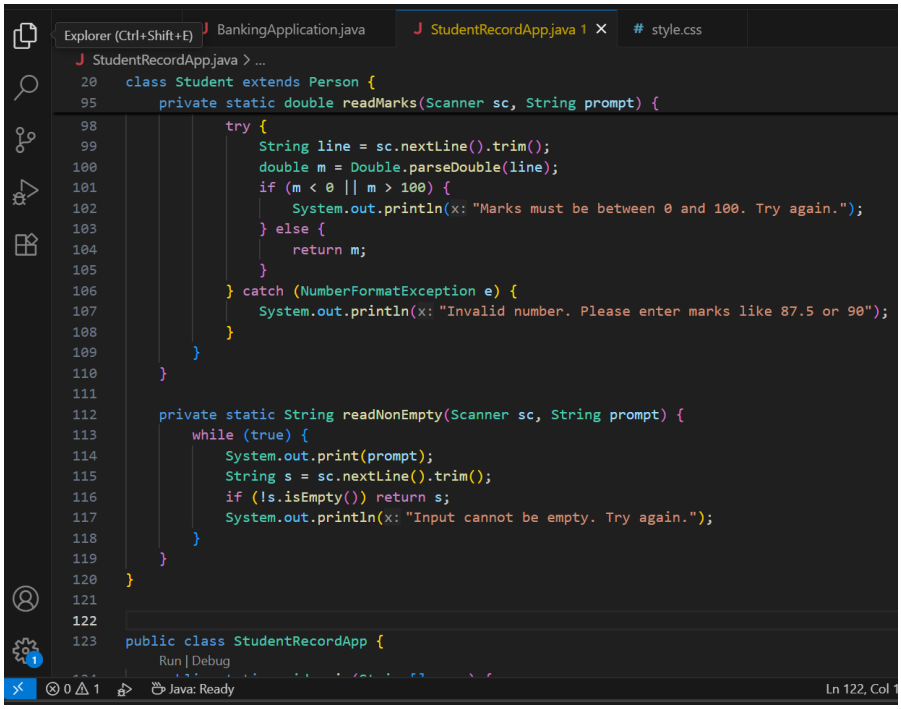
Course: BCA (AI&DS)

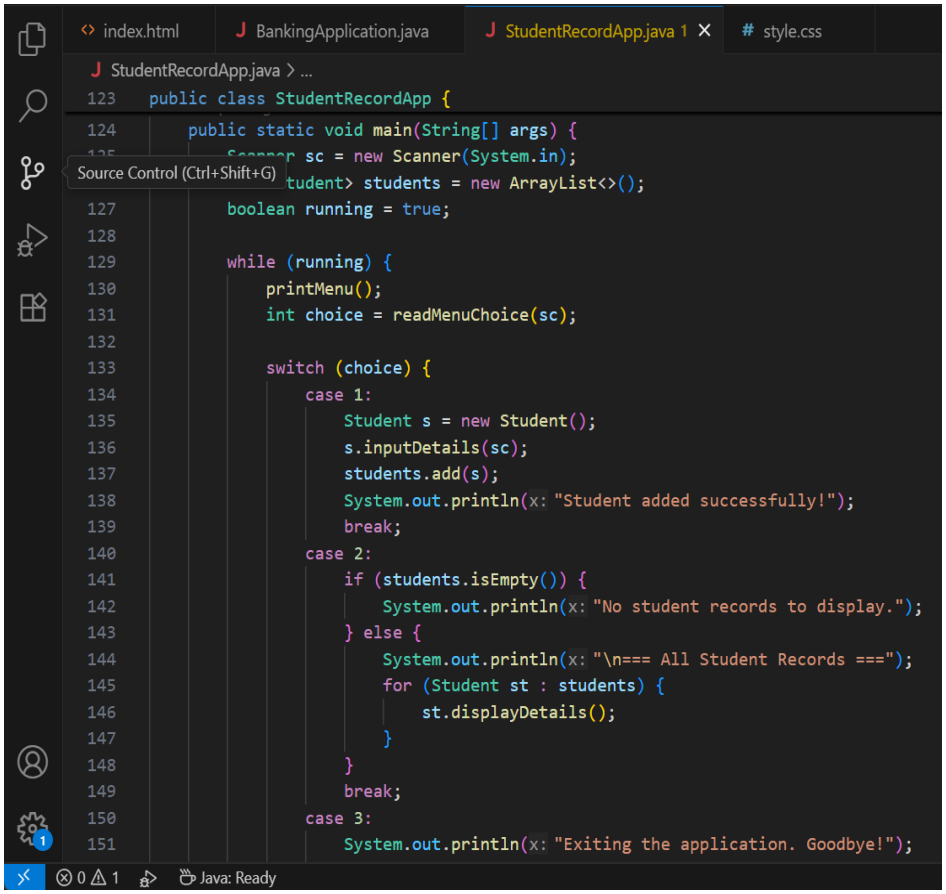
INPUT:

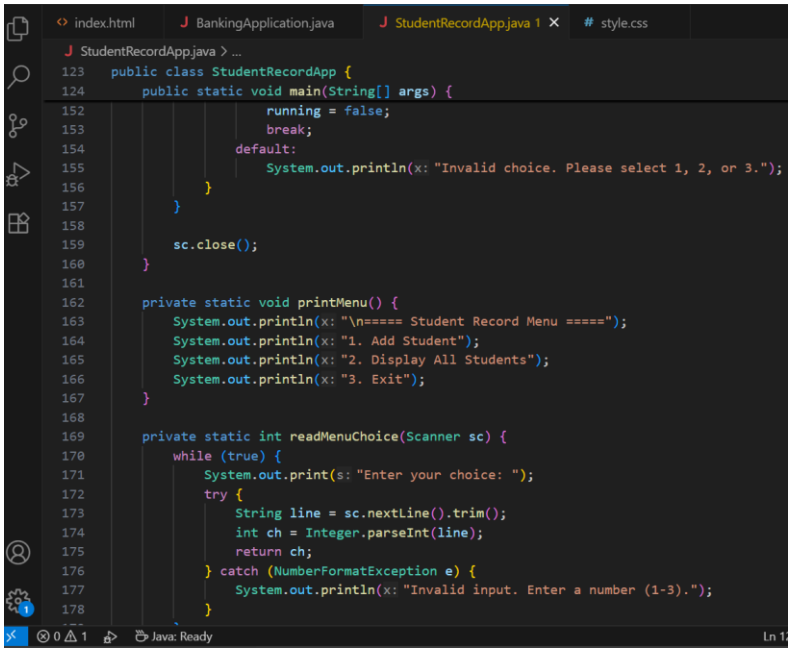



EXPLORER
index.html
BankingApplication.java
StudentRecordApp.java 1
style.css
OPEN EDITORS
index.html
BankingApplicat...
StudentRec... 1
style.css
Run and Debug (Ctrl+Shift+D)
index.html
Student Class Design ...
StudentRecordA... 1
style.css
OUTLINE
TIMELINE
JAVA PROJECTS
StudentRecordApp.java > ...
20 class Student extends Person {
26 public Student() {
30 this.marks = 0.0;
31 this.grade = 'F';
32 }
35 public Student(int rollNo, String name, String course, double marks) {
36 super(name);
37 this.rollNo = rollNo;
38 this.course = course;
39 this.marks = marks;
40 calculateGrade();
41 }
44 public void inputDetails(Scanner sc) {
45 this.rollNo = readInt(sc, prompt: "Enter Roll No: ");
46 this.name = readNonEmpty(sc, prompt: "Enter Name: ");
47 this.course = readNonEmpty(sc, prompt: "Enter Course: ");
48 this.marks = readMarks(sc, prompt: "Enter Marks (0 - 100): ");
49 calculateGrade();
50 }
53 public void displayDetails() {
54 System.out.println("Roll No: " + rollNo);
55 System.out.println("Name: " + name);
56 System.out.println("Course: " + course);

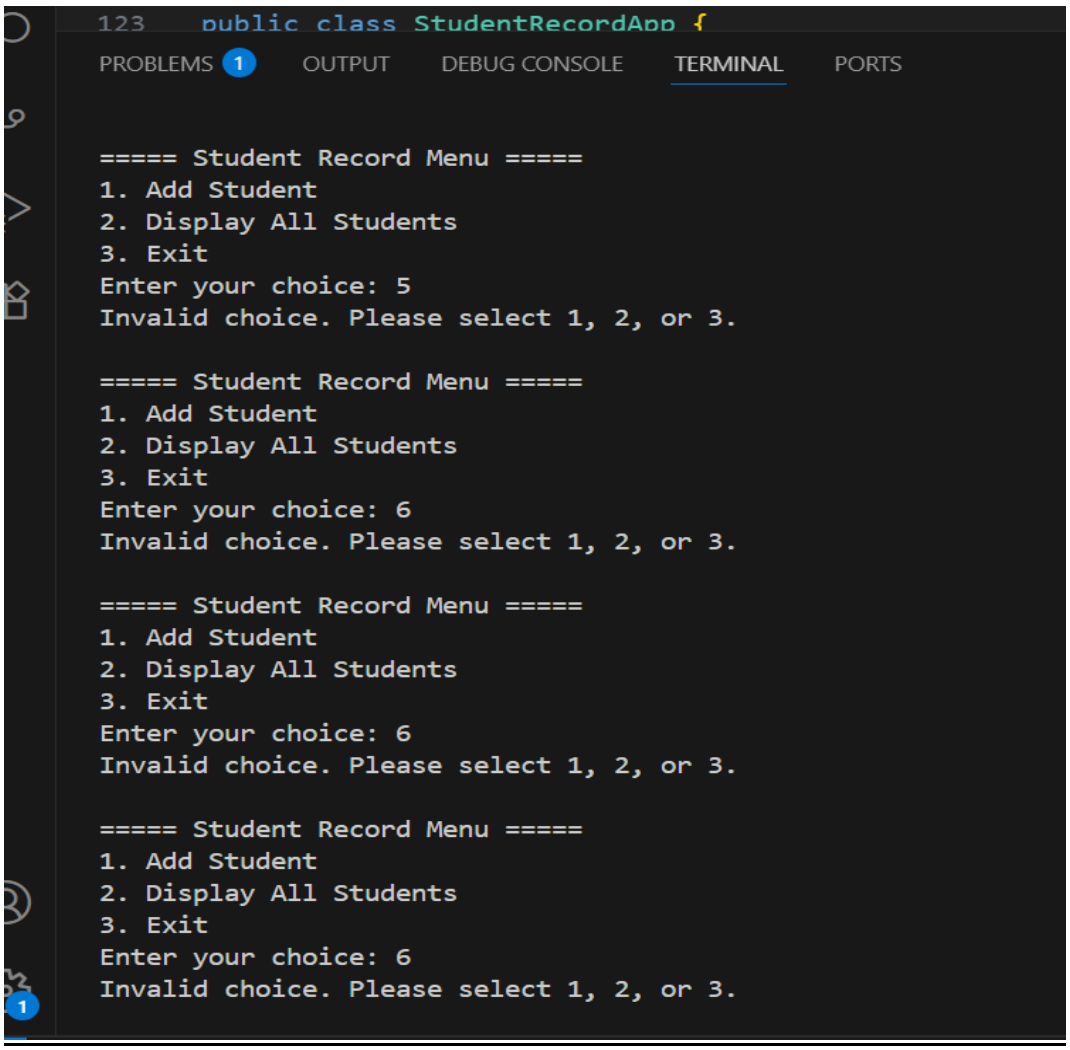

index.html
BankingApplication.java
StudentRecordApp.java 1
style.css
StudentRecordApp.java > ...
20 class Student extends Person {
63 private void calculateGrade() {
71 grade = 'B';
72 } else if (marks >= 50) {
73 grade = 'C';
74 } else if (marks >= 33) {
75 grade = 'D';
76 } else {
77 grade = 'F';
78 }
79 }
82 private static int readInt(Scanner sc, String prompt) {
83 while (true) {
84 System.out.print(prompt);
85 try {
86 String line = sc.nextLine().trim();
87 int val = Integer.parseInt(line);
88 return val;
89 } catch (NumberFormatException e) {
90 System.out.println(x: "Invalid integer. Please try again.");
91 }
92 }
93 }
95 private static double readMarks(Scanner sc, String prompt) {
96 while (true) {
97 System.out.print(prompt);
⊗ 0 ⚠ 1
Java: Ready


Explorer (Ctrl+Shift+E)
BankingApplication.java
StudentRecordApp.java 1
style.css
StudentRecordApp.java > ...
class Student extends Person {
private static double readMarks(Scanner sc, String prompt) {
try {
String line = sc.nextLine().trim();
double m = Double.parseDouble(line);
if (m < 0 || m > 100) {
System.out.println(x: "Marks must be between 0 and 100. Try again.");
} else {
return m;
}
} catch (NumberFormatException e) {
System.out.println(x: "Invalid number. Please enter marks like 87.5 or 90");
}
}
}
private static String readNonEmpty(Scanner sc, String prompt) {
while (true) {
System.out.print(prompt);
String s = sc.nextLine().trim();
if (!s.isEmpty()) return s;
System.out.println(x: "Input cannot be empty. Try again.");
}
}
}
public class StudentRecordApp {
Run | Debug
Java: Ready
Ln 122, Col 1


index.html
BankingApplication.java
StudentRecordApp.java 1
style.css
StudentRecordApp.java > ...
public class StudentRecordApp {
public static void main(String[] args) {
Source Control (Ctrl+Shift+G)
tudent> students = new ArrayList<>();
boolean running = true;
while (running) {
printMenu();
int choice = readMenuChoice(sc);
switch (choice) {
case 1:
Student s = new Student();
s.inputDetails(sc);
students.add(s);
System.out.println(x: "Student added successfully!");
break;
case 2:
if (students.isEmpty()) {
System.out.println(x: "No student records to display.");
} else {
System.out.println(x: "\n=== All Student Records ===");
for (Student st : students) {
st.displayDetails();
}
}
break;
case 3:
System.out.println(x: "Exiting the application. Goodbye!");
Java: Ready


index.html
BankingApplication.java
StudentRecordApp.java 1
style.css
StudentRecordApp.java > ...
123 public class StudentRecordApp {
124 public static void main(String[] args) {
152 running = false;
153 break;
154 default:
155 System.out.println(x: "Invalid choice. Please select 1, 2, or 3.");
156 }
157 }
158
159 sc.close();
160 }
161
162 private static void printMenu() {
163 System.out.println(x: "\n===== Student Record Menu =====");
164 System.out.println(x: "1. Add Student");
165 System.out.println(x: "2. Display All Students");
166 System.out.println(x: "3. Exit");
167 }
168
169 private static int readMenuChoice(Scanner sc) {
170 while (true) {
171 System.out.print(s: "Enter your choice: ");
172 try {
173 String line = sc.nextLine().trim();
174 int ch = Integer.parseInt(line);
175 return ch;
176 } catch (NumberFormatException e) {
177 System.out.println(x: "Invalid input. Enter a number (1-3).");
178 }
Java: Ready

OUTPUT


123 public class StudentRecordApp {
PROBLEMS 1
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
===== Student Record Menu =====
1. Add Student
2. Display All Students
3. Exit
Enter your choice: 5
Invalid choice. Please select 1, 2, or 3.
===== Student Record Menu =====
1. Add Student
2. Display All Students
3. Exit
Enter your choice: 6
Invalid choice. Please select 1, 2, or 3.
===== Student Record Menu =====
1. Add Student
2. Display All Students
3. Exit
Enter your choice: 6
Invalid choice. Please select 1, 2, or 3.
===== Student Record Menu =====
1. Add Student
2. Display All Students
3. Exit
Enter your choice: 6
Invalid choice. Please select 1, 2, or 3.

Explanation —

1. Purpose of the Program

The aim of the program is to build a system that can:

- Take student details from the user
- Store multiple student records
- Display all stored records
- Calculate grade based on marks

It also teaches how to structure a program using OOP principles.

◆ 2. Class Design (OOP Concept)

Person Class (Parent Class)

- This is the base class.
- It contains only one field: `name`.
- It is used to show **inheritance**, where student objects inherit the name attribute.

Student Class (Child Class extending Person)

Student inherits `name` from Person and includes:

- `rollNo` (int)
- `course` (String)
- `marks` (double)
- `grade` (char)

This demonstrates the OOP principle **"IS-A relationship"**:
A Student **is a** Person.

◆ 3. Constructors

The Student class has two constructors:

✓ Default Constructor

Initializes the object with empty or zero values.

✓ Parameterized Constructor

Allows direct initialization of:

- roll number
- name
- course
- marks

Constructors help in object creation and initialization.

◆ 4. Methods in Student Class

✓ inputDetails()

This method:

- Takes input from the user using Scanner
- Reads roll number, name, course, and marks
- Validates marks
- Calls `calculateGrade()`

✓ calculateGrade()

This method determines the grade based on marks:

Marks Range	Grade
90–100	A
75–89	B
50–74	C
33–49	D
Below 33	F

This demonstrates the use of **if-else conditional statements**.

✓ displayDetails()

This method displays:

- Roll number
- Name
- Course

- Marks
- Grade

◆ 5. Data Storage Using ArrayList

The program uses:

```
ArrayList<Student> students = new ArrayList<>();
```

Reasons:

- It automatically grows as new students are added.
- It is easier to use compared to arrays.
- Helps store multiple objects of the Student class.

◆ 6. Menu-Driven Program

The program repeatedly shows a menu using a **while loop**:

```
1. Add Student
2. Display All Students
3. Exit
```

User chooses an option using **switch-case**:

- **1:** Creates a Student object and calls `inputDetails()`
- **2:** Calls `displayDetails()` for every student in the list
- **3:** Exits the program

This teaches:

- Loops
- Switch-case
- User input handling

◆ 7. Input Validation

The program ensures proper user input:

- Marks must be between **0 and 100**
- Name and course cannot be empty
- Incorrect input is handled properly

This makes the program robust.

◆ 8. OOP Concepts Demonstrated

✓ Classes and Objects

Student and Person are classes; each student record is an object.

✓ Inheritance

`Student extends Person` → Student inherits attributes of Person.

✓ Encapsulation

Student fields are private; accessed only through methods.

✓ Constructors

Used for object initialization.

✓ Methods

Used to input, process, and display data.

◆ 9. Control Structures

The program uses:

- **While loop** → for menu repetition
- **If-else** → for grade calculation
- **Switch-case** → for menu operations
- **For-each loop** → to display all students