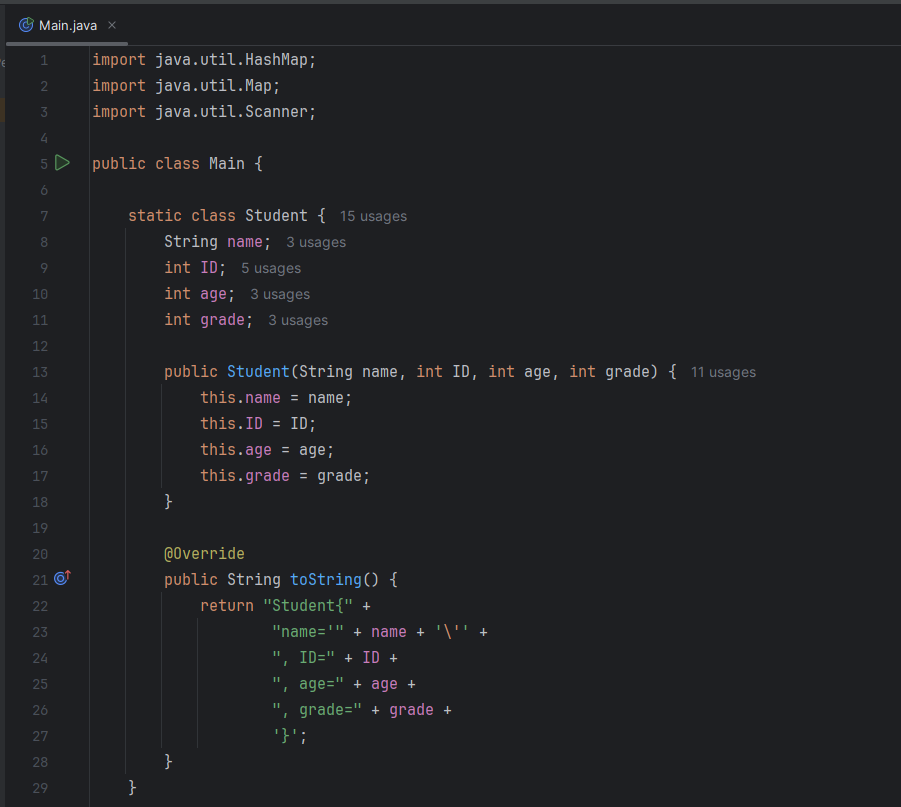
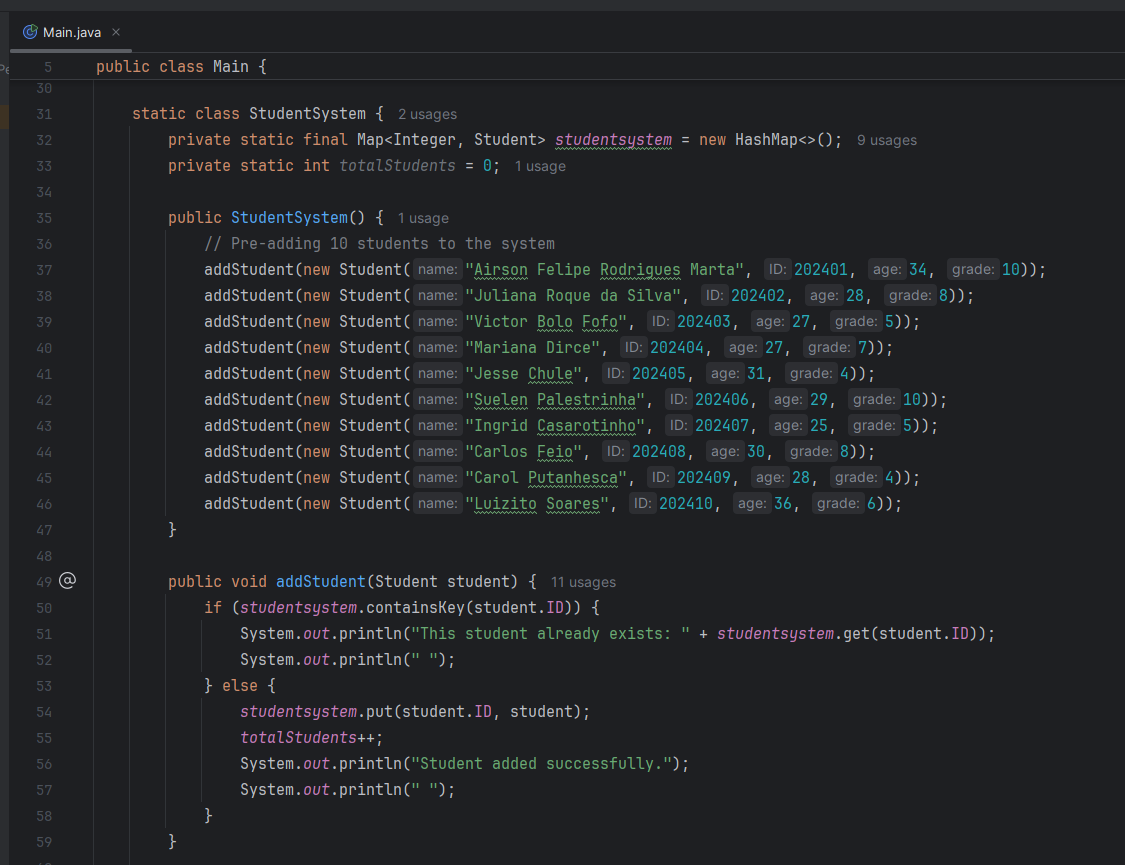
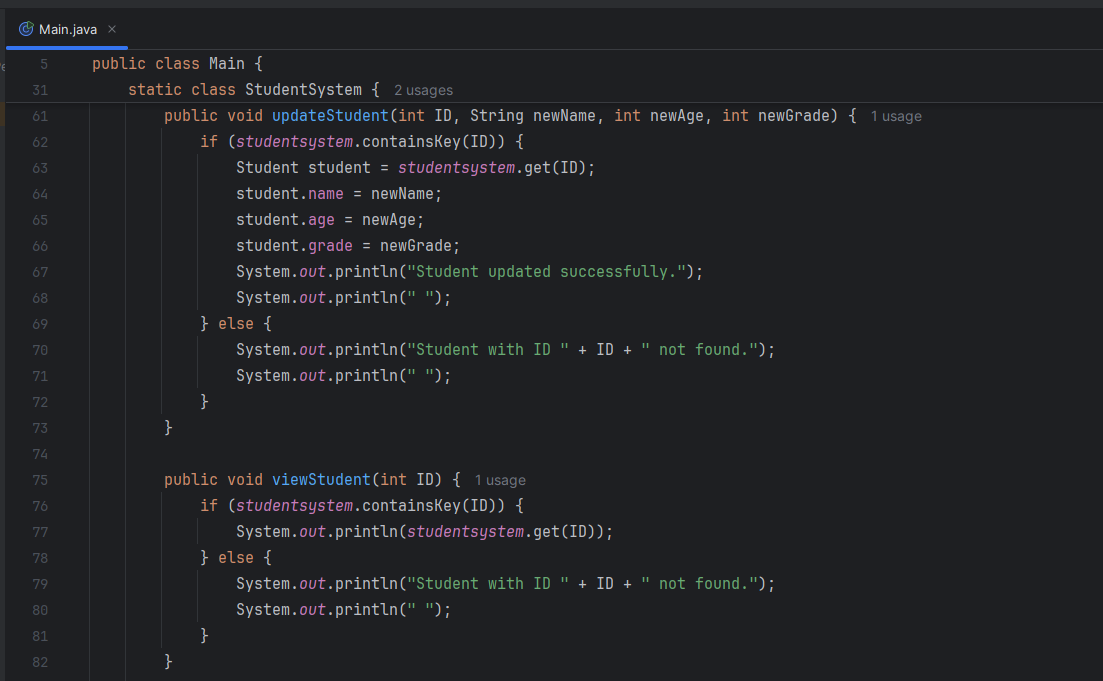
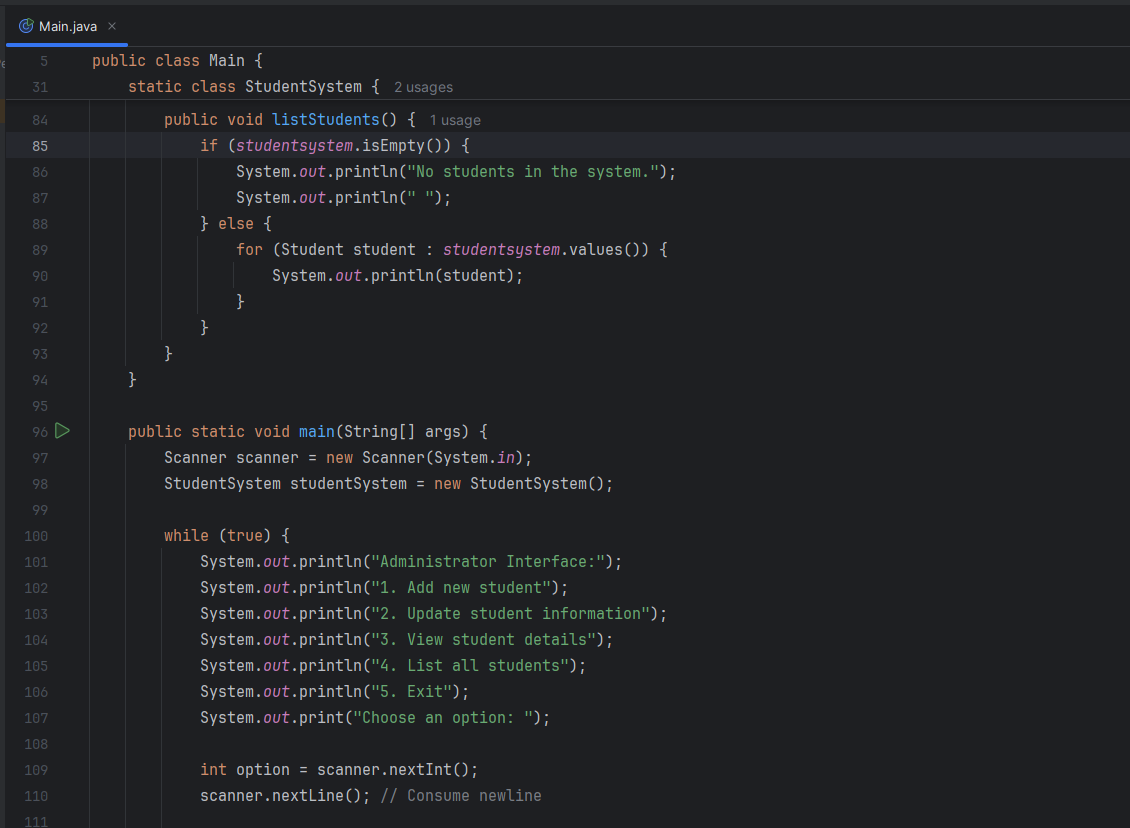
**Programming Assignment Unit 3**

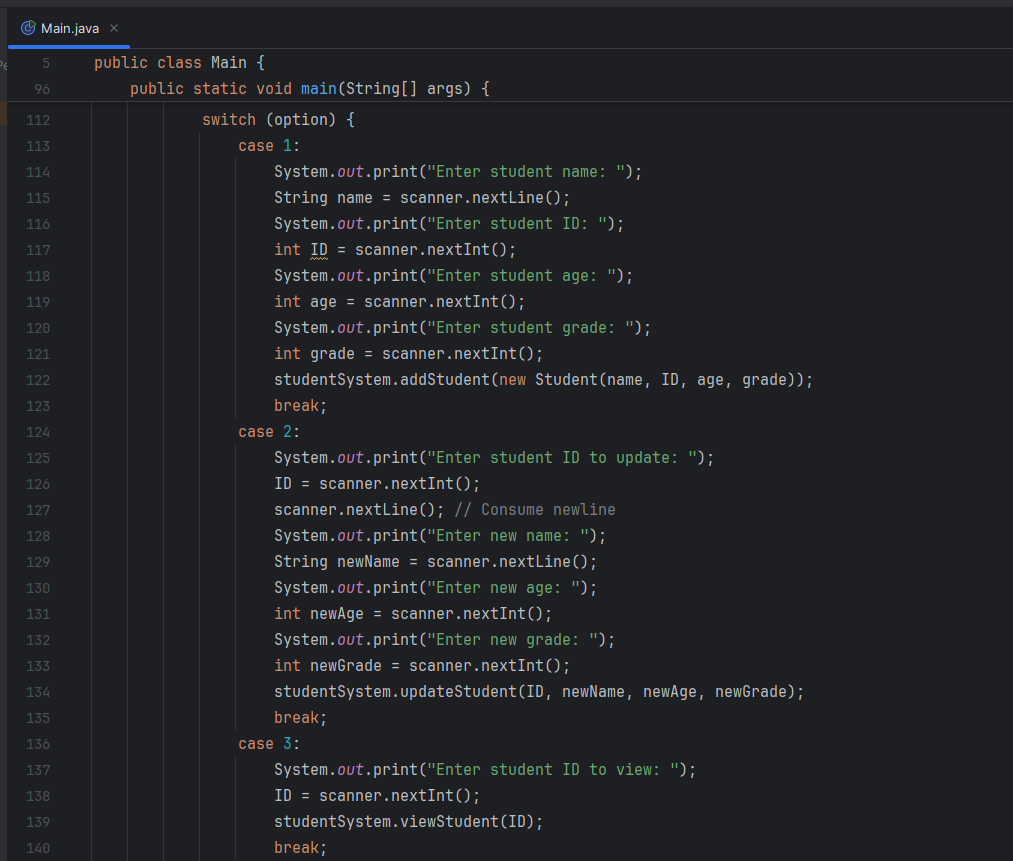
**My complete code:**

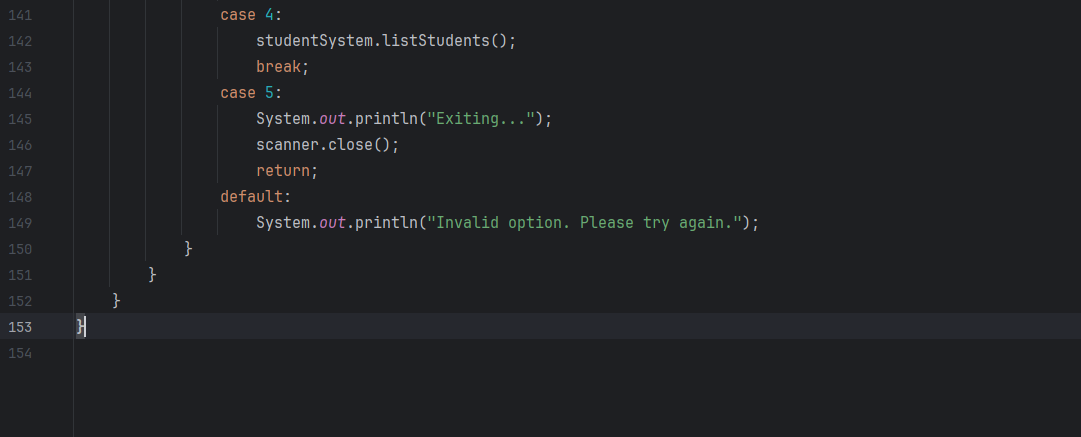
****

****

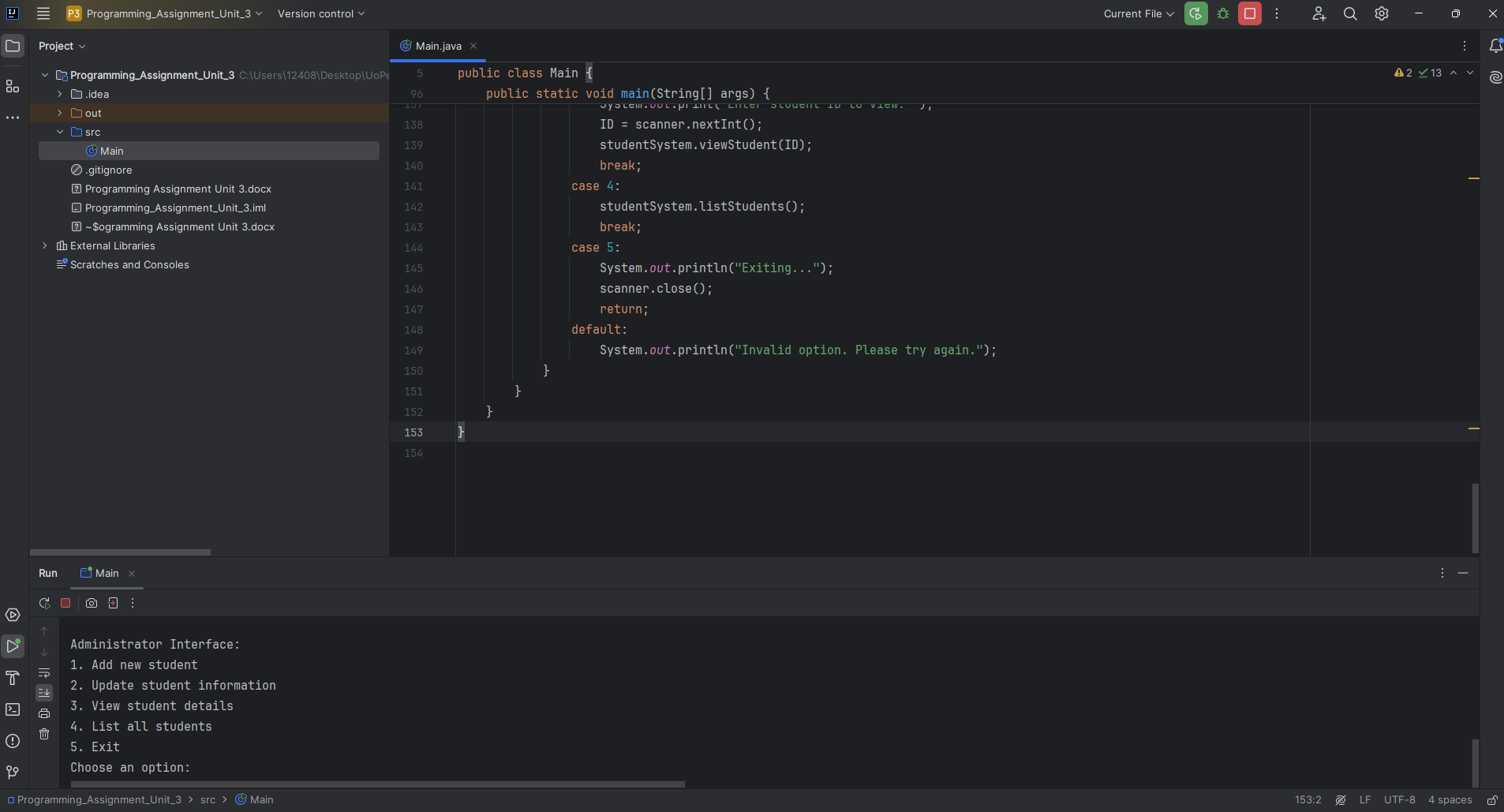
****

****

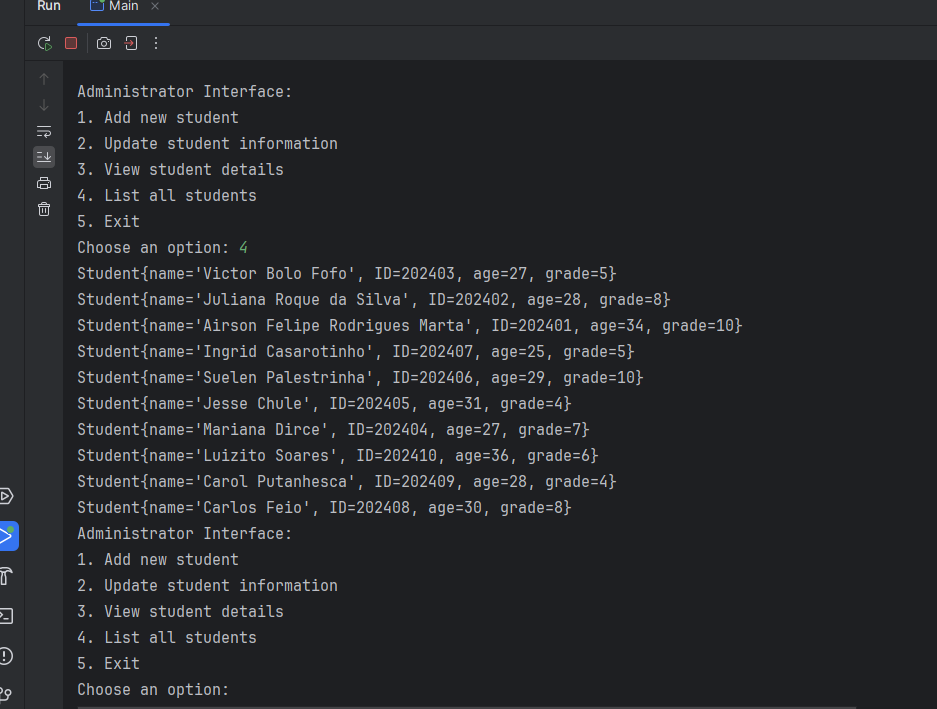
****

****

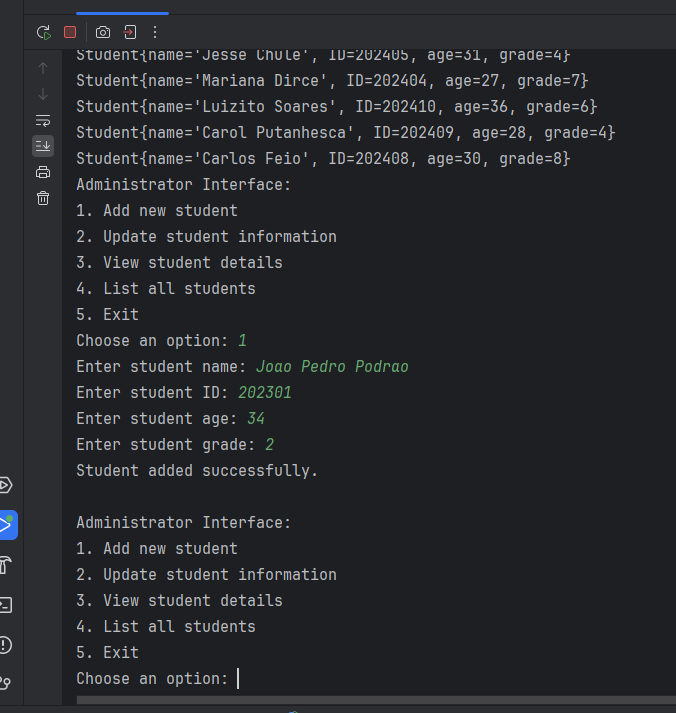
**Output:**

****

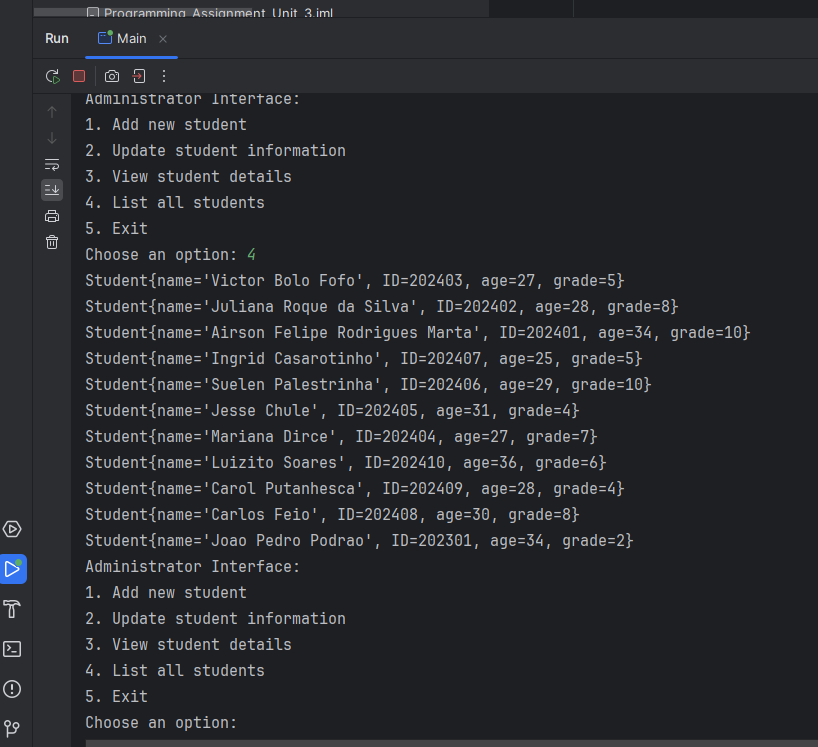
**Choosing the option 4, List all students (I pre-add 10 students to show how the program works):**

****

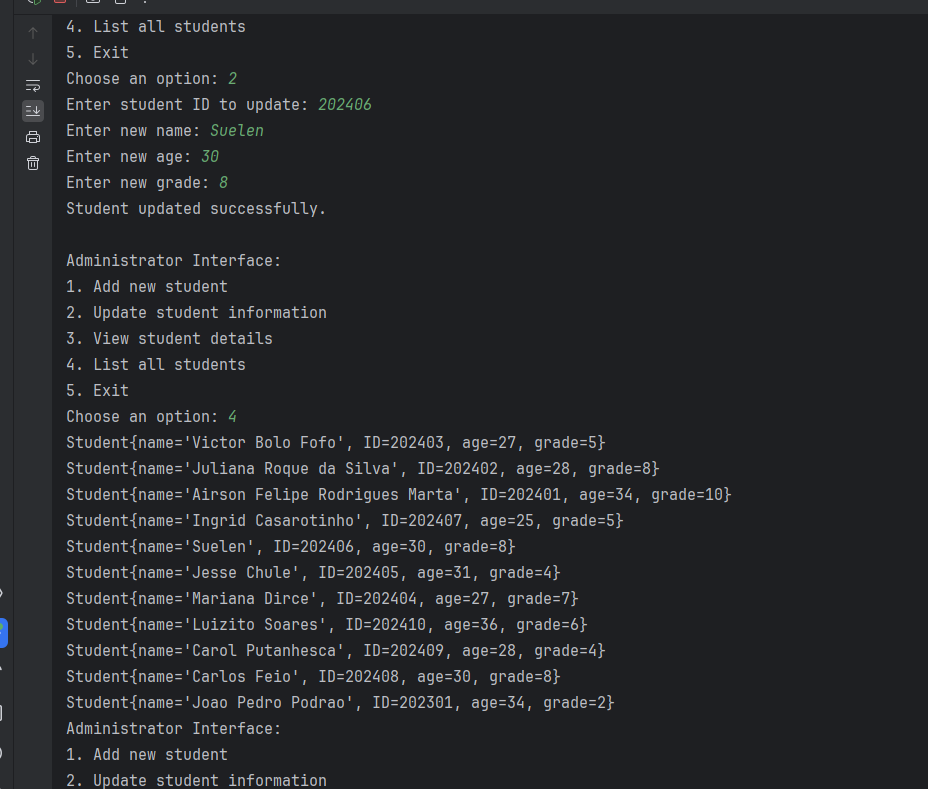
**Choosing option 1, add new student:**

****

**Now, when we choose option 4, the new student (Joao Pedro Podrao) is shown on the list:**

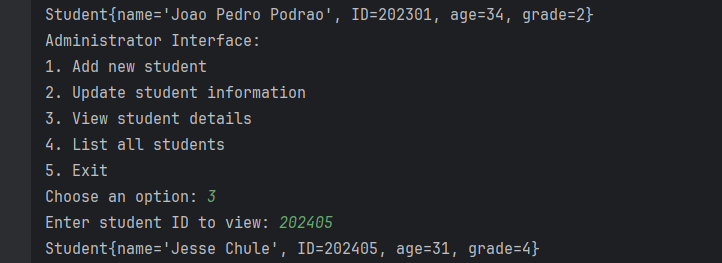
****

**Choosing option 2, update student information:**

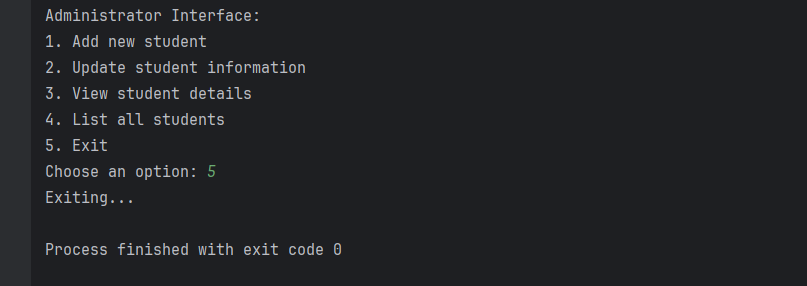
****

**The program replaced Suelen with the new name I gave her, kept the same ID (this is unique for each student), and changed her age and grade.**

**Option 3, we can see student details given the program, the student ID:**

****

**Last option, number 5, we can exit from the program:**

****

**Explaining the code:**

**Class Student**

This class represents a student and holds basic information about a student.

public class Main {  
  
 static class Student {  
 String name;  
 int ID;  
 int age;  
 int grade;  
  
 public Student(String name, int ID, int age, int grade) {  
 this.name = name;  
 this.ID = ID;  
 this.age = age;  
 this.grade = grade;  
 }  
  
 @Override  
 public String toString() {  
 return "Student{" +  
 "name='" + name + '\'' +  
 ", ID=" + ID +  
 ", age=" + age +  
 ", grade=" + grade +  
 '}';  
 }  
 }

**Elements:**

**Fields:**

**String name:** Stores the student's name.

**int ID:** Stores the student's ID.

**int age:** Stores the student's age.

**int grade:** Stores the student's grade.

**Constructor:**

public Student(String name, int ID, int age, int grade): Initializes a new Student object with the given name, ID, age, and grade.

**toString() Method:**

@Override public String toString(): Overrides the default toString() method to provide a string representation of the student object.

**Class StudentSystem**

This class manages the student records. It provides functionalities to add, update, view, and list students.

static class StudentSystem {  
 private static final Map<Integer, Student> *studentsystem* = new HashMap<>();  
 private static int *totalStudents* = 0;  
  
 public StudentSystem() {  
 // Pre-adding 10 students to the system  
 addStudent(new Student("Airson Felipe Rodrigues Marta", 202401, 34, 10));  
 addStudent(new Student("Juliana Roque da Silva", 202402, 28, 8));  
 addStudent(new Student("Victor Bolo Fofo", 202403, 27, 5));  
 addStudent(new Student("Mariana Dirce", 202404, 27, 7));  
 addStudent(new Student("Jesse Chule", 202405, 31, 4));  
 addStudent(new Student("Suelen Palestrinha", 202406, 29, 10));  
 addStudent(new Student("Ingrid Casarotinho", 202407, 25, 5));  
 addStudent(new Student("Carlos Feio", 202408, 30, 8));  
 addStudent(new Student("Carol Putanhesca", 202409, 28, 4));  
 addStudent(new Student("Luizito Soares", 202410, 36, 6));  
 }  
  
 public void addStudent(Student student) {  
 if (*studentsystem*.containsKey(student.ID)) {  
 System.*out*.println("This student already exists: " + *studentsystem*.get(student.ID));  
 System.*out*.println(" ");  
 } else {  
 *studentsystem*.put(student.ID, student);  
 *totalStudents*++;  
 System.*out*.println("Student added successfully.");  
 System.*out*.println(" ");  
 }  
 }  
  
 public void updateStudent(int ID, String newName, int newAge, int newGrade) {  
 if (*studentsystem*.containsKey(ID)) {  
 Student student = *studentsystem*.get(ID);  
 student.name = newName;  
 student.age = newAge;  
 student.grade = newGrade;  
 System.*out*.println("Student updated successfully.");  
 System.*out*.println(" ");  
 } else {  
 System.*out*.println("Student with ID " + ID + " not found.");  
 System.*out*.println(" ");  
 }  
 }  
  
 public void viewStudent(int ID) {  
 if (*studentsystem*.containsKey(ID)) {  
 System.*out*.println(*studentsystem*.get(ID));  
 } else {  
 System.*out*.println("Student with ID " + ID + " not found.");  
 System.*out*.println(" ");  
 }  
 }  
  
 public void listStudents() {  
 if (*studentsystem*.isEmpty()) {  
 System.*out*.println("No students in the system.");  
 System.*out*.println(" ");  
 } else {  
 for (Student student : *studentsystem*.values()) {  
 System.*out*.println(student);  
 }  
 }  
 }  
}

**Elements:**

**Fields:**

**private static final Map<Integer, Student> system:** A map to store student records using their ID as the key.

**private static int totalStudents:** A counter to keep track of the total number of students.

**Constructor:**

public StudentSystem(): Initializes the StudentSystem by pre-adding 10 students.

**Methods:**

public void addStudent(Student student): Adds a student to the system. If the student already exists, it prints a message; otherwise, it adds the student and increments the total student count.

public void updateStudent(int ID, String newName, int newAge, int newGrade): Updates the information of a student with the given ID. If the student is not found, it prints an error message.

public void viewStudent(int ID): Prints the details of a student with the given ID. If the student is not found, it prints an error message.

public void listStudents(): Prints the details of all students in the system. If there are no students, it prints a message indicating that the system is empty.

**Main Method**

The main method serves as the entry point for the program and provides the administrator interface.

public static void main(String[] args) {  
 Scanner scanner = new Scanner(System.*in*);  
 StudentSystem studentSystem = new StudentSystem();  
  
 while (true) {  
 System.*out*.println("Administrator Interface:");  
 System.*out*.println("1. Add new student");  
 System.*out*.println("2. Update student information");  
 System.*out*.println("3. View student details");  
 System.*out*.println("4. List all students");  
 System.*out*.println("5. Exit");  
 System.*out*.print("Choose an option: ");  
  
 int option = scanner.nextInt();  
 scanner.nextLine(); // Consume newline  
  
 switch (option) {  
 case 1:  
 System.*out*.print("Enter student name: ");  
 String name = scanner.nextLine();  
 System.*out*.print("Enter student ID: ");  
 int ID = scanner.nextInt();  
 System.*out*.print("Enter student age: ");  
 int age = scanner.nextInt();  
 System.*out*.print("Enter student grade: ");  
 int grade = scanner.nextInt();  
 studentSystem.addStudent(new Student(name, ID, age, grade));  
 break;  
 case 2:  
 System.*out*.print("Enter student ID to update: ");  
 ID = scanner.nextInt();  
 scanner.nextLine(); // Consume newline  
 System.*out*.print("Enter new name: ");  
 String newName = scanner.nextLine();  
 System.*out*.print("Enter new age: ");  
 int newAge = scanner.nextInt();  
 System.*out*.print("Enter new grade: ");  
 int newGrade = scanner.nextInt();  
 studentSystem.updateStudent(ID, newName, newAge, newGrade);  
 break;  
 case 3:  
 System.*out*.print("Enter student ID to view: ");  
 ID = scanner.nextInt();  
 studentSystem.viewStudent(ID);  
 break;  
 case 4:  
 studentSystem.listStudents();  
 break;  
 case 5:  
 System.*out*.println("Exiting...");  
 scanner.close();  
 return;  
 default:  
 System.*out*.println("Invalid option. Please try again.");  
 }  
 }  
 }  
}

**Elements:**

**Scanner:**

Scanner scanner = new Scanner(System.in): Creates a Scanner object to read input from the console.

**StudentSystem Initialization:**

StudentSystem studentSystem = new StudentSystem(): Initializes the StudentSystem, which pre-adds the 10 students.

**Main Loop:**

A while (true) loop provides the administrator interface, allowing continuous interaction until the user decides to exit.

**Menu Options:**

The program displays a menu with options to add, update, view, list, and exit. Each option corresponds to a different case in the switch statement.

**Option Handling:**

Option 1 (Add new student): Prompts the user to enter student details and adds the student to the system.

Option 2 (Update student information): Prompts the user to enter the student ID and new details to update the student information.

Option 3 (View student details): Prompts the user to enter the student ID and displays the student details.

Option 4 (List all students): Lists all students in the system.

Option 5 (Exit): Exits the program.

**Error Handling:**

If the user enters an invalid option, the program prints an error message and prompts the user to try again.

**References:**

Eck, D. J. (2022). Introduction to Programming Using Java: Version 9.0, JavaFX Edition. Retrieved from http://math.hws.edu/javanotes/

Bloch, J. (2018). Effective Java (3rd ed.). Addison-Wesley Professional.

Oracle. (2023). The Java Tutorials. Oracle Corporation.

<https://pt.stackoverflow.com/>

https://www.geeksforgeeks.org/