Due Date: 2/9/2024, 11:59pm

CMP_SC 3330 – Object-oriented Programming
Homework 2

You are tasked with creating a program to manage information about students. The program should read student data from a file, initialize student objects, and perform various operations based on user input.

Class Definition:

Student Class:

- Create a ***Student*** class with private fields for the following attributes: *id* (**int**), *name* (**String**), *grade* (**double**).
- Implement a default constructor and a parameterized constructor that initializes the fields.
- Include getter and setter methods for each field.
- Implement a ***toString()*** method to display the student information.

StudentManager Class:

- Create a ***StudentManager*** class that will handle the main logic.
- Use an array (***Student[]***) to store student objects.

Program Requirements:

Read from File and Initialize Objects:

- Implement a method in ***StudentManager*** to read student data from a file using ***FileInputStream*** and ***Scanner***.
- The file will contain lines with the following format: <id> <name> <grade>. Each line represents a student.
- Use the data read from the file to initialize Student objects.
- Store the created objects in the array ***(Student[]***) in the ***StudentManager*** class.
- **Method signature:** `public boolean readFromFile(String fileName)`
    - Return ***true*** if the read file and initialization are successful.
    - Return ***false*** if cannot read/find the file or initialize objects.

Display Students:

- Implement a method to display the details of all students in the ***StudentManager*** class. (**Hint:** use the implemented ***toString()*** method from the ***Student*** class.)
- Check if the Student[] array is empty or not. If empty, display a message to inform the user.
- **Method signature:** `public void displayStudents()`

Search Students:

- Implement a method to search for a student by ID. (**Hint:** implement and use the ***equals()*** method from the ***Student*** class)
- Display the details of the student if found (use ***toString()*** method again), or a message if not found.
- **Method signature:**

  ```
  public boolean searchStudentById(int id)
  ```

    - Return ***true*** if student ID was found.
    - Return ***false*** if student ID was not found.

Update Student Information:

- Implement a method to update a student's grade by ID.
- Use the search method you implemented to check whether the student ID exists.
- **Method signature:**

  ```
  public boolean updateStudentGradeById(int id, double grade)
  ```

    - Return ***true*** if the student was found and updated successfully.
    - Return ***false*** if student ID was not found.

Sample Usage:

```
public class Main {
  public static void main(String[] args) {
    // Instantiate StudentManager, perform operations based on the requirements.
    StudentManager studentManager = new StudentManager();

    // Read student data from a file and initialize Student objects.
    boolean fileReadStatus = studentManager.readFromFile("studentData.txt");

    // Display all students.
    studentManager.displayStudents();

    // Search for a student by ID.
    boolean studentFound = studentManager.searchStudentById(101);

    // Update the grade of a student by ID.
    boolean studentGradeUpdateStatus = studentManager.updateStudentGradeById(102, 95);

    // Display all students after the update.
    studentManager.displayStudents();
  }
}
```

Submission Guidelines:

- Each team is required to create a GitHub repository for the project.
- The repository should include all the required Java files (Main.java, Student.java, and StudentManager.java) and any other necessary files to run the program.
- Team members are expected to contribute equally to the project.
- Each team member should make meaningful contributions, and commit messages must be descriptive and related to the changes made. Your grades will be affected by your commits.
- The GitHub repository should demonstrate good version control practices, with commits logically organized and documenting the evolution of the code.
- Make sure to include a README.md file providing clear instructions on how to run the program, any dependencies, and a brief explanation of the project.
- Verify that the repository is accessible and properly organized, allowing anyone to clone and run the program without additional configuration.
- Your program must use the classes with described methods, given prototypes and signatures exactly. You are allowed to implement additional helper methods and classes.
- Late submission between 0hrs < late <= 24hrs will lose half of the grade. After 24 hours, submissions will receive a grade of 0 for the assignment.
- **Not following the submission guidelines will result in a penalty on your grades.**

Note:

- Ensure that your program handles cases where the file is not found or if there are any issues during file reading.
- Make use of the concepts you've learned, such as constructors, getter/setter methods, static fields/methods, and the toString() method.
- Test your program with different scenarios, including cases where the student is not found and the update is unsuccessful.