Java Coding Exercise: File Handling with OOP Concepts

Problem Description:

You are required to create a Java program that manages a student database using a text file (`students.txt`). The program should demonstrate Object-Oriented Programming (OOP) concepts such as classes, objects, encapsulation, and methods. The tasks will involve creating a `Student` class to represent student information, writing and reading student data from a file, and using methods to manipulate this data.

OOP Concepts to Apply:

1. **Class and Objects**: Define a `Student` class with appropriate attributes.
2. **Encapsulation**: Use private fields in the `Student` class and provide public getter and setter methods.
3. **Constructor**: Implement constructors for initializing `Student` objects.
4. **File Handling**: Use file reading and writing to manage student data.
5. **Method Overloading**: Create multiple methods with the same name but different parameters in the `StudentManager` class.
6. **Exception Handling**: Properly handle exceptions related to file operations.
7. **Static Methods**: Implement static utility methods for common tasks.

Tasks:

1. **Define a `Student` Class:**
   - Create a `Student` class with the following private fields: `String name`, `int enrollmentYear`, and `String major`.
   - Implement public getters and setters for each field.
   - Provide two constructors:
     - A default constructor.
     - A parameterized constructor to initialize all fields.
2. **Create a `StudentManager` Class:**
   - This class will manage the list of students and perform file operations.
   - The `StudentManager` class should have the following methods:
     - `void addStudent(Student student)`: Adds a student to the list and writes it to the `students.txt` file.
     - `List<Student> readStudentsFromFile()`: Reads student data from the `students.txt` file and returns a list of `Student` objects.
     - `void displayAllStudents()`: Reads and displays all students from the file.
     - `void searchStudentByName(String name)`: Searches for a student by name and displays the result.
     - `void updateStudentMajor(String name, String newMajor)`: Updates the major of a student by name and saves the updated list to the file.
     - `void deleteStudent(String name)`: Deletes a student by name and updates the file.

3. **Implement File Operations:**
    - Use `FileWriter` to write student data to the `students.txt` file.
    - Use `Scanner` to read data from the file and parse it into `Student` objects.
4. **Demonstrate the Functionality in the `main` Method:**
    - Create instances of `Student` and add them to the `StudentManager`.
    - Demonstrate reading from the file and displaying student data.
    - Implement searching, updating, and deleting student operations.
5. **Bonus Task** (Optional):
    - Implement method overloading in the `StudentManager` class by creating overloaded versions of the `addStudent` method:
        - `void addStudent(String name, int enrollmentYear, String major)`: Creates a new `Student` object from parameters and adds it to the file.
        - `void addStudent(List<Student> students)`: Takes a list of students and writes them all to the file.

Expected Output:

- The program should be able to write, read, search, update, and delete student data from the `students.txt` file.
- Display all relevant information on the console for each operation.
- Handle errors gracefully, such as when searching for a non-existent student or trying to delete a student who is not in the file.

Hints:

- Use `ArrayList<Student>` to manage a dynamic list of students.
- Utilize OOP principles to encapsulate student-related data and operations.
- Implement file reading and writing operations in a modular and reusable way.