Java Coding Question: CRUD Operations on ArrayList with OOP Concepts and Exception Handling

Problem Statement:

You are required to design and implement a simple student management system in Java. The system should manage student records, allowing for detailed CRUD (Create, Read, Update, Delete) operations on a list of students. The system should handle various exceptions, such as attempting to update or delete a student who does not exist.

Requirements:

1. **Create a `` `Student` `` class** with the following attributes:
   - `` `int id` `` (unique identifier for each student)
   - `` `String name` ``
   - `` `int age` ``
   - `` `String course` ``
2. **Create a `` `StudentNotFoundException` `` class:**
   - This custom exception should be thrown when attempting to update or delete a student that does not exist in the list.
3. **Create a `` `StudentManagement` `` class** that manages a list of `` `Student` `` objects. This class should provide the following methods:**
   - `` `public void addStudent(Student student)` ``: Adds a new student to the list. If a student with the same ID already exists, an exception should be thrown.
   - `` `public Student getStudentById(int id) throws StudentNotFoundException` ``: Retrieves a student's details by their ID. Throws `` `StudentNotFoundException` `` if the student is not found.
   - `` `public void updateStudent(int id, String name, int age, String course) throws StudentNotFoundException` ``: Updates the details of an existing student. If the student does not exist, throw `` `StudentNotFoundException` ``.
   - `` `public void deleteStudent(int id) throws StudentNotFoundException` ``: Deletes a student by their ID. If the student does not exist, throw `` `StudentNotFoundException` ``.
   - `` `public List<Student> listAllStudents()` ``: Returns a list of all students in the system.
   - `` `public List<Student> searchStudentsByCourse(String course)` ``: Returns a list of students enrolled in a particular course.
4. **Implement the `` `StudentManagement` `` methods with the following logic:**
   - The `` `addStudent` `` method should check for duplicate IDs and throw an exception if an ID conflict occurs.
   - The `` `getStudentById` ``, `` `updateStudent` ``, and `` `deleteStudent` `` methods should validate the existence of the student by ID and throw `` `StudentNotFoundException` `` if the student is not found.
   - The `` `listAllStudents` `` method should return the complete list of students.
   - The `` `searchStudentsByCourse` `` method should return a list of students enrolled in a specific course. If no students are found for the course, return an empty list.

5. **Write a `main` method** to test your implementation. The `main` method should demonstrate the following:
    - Adding multiple students to the list.
    - Attempting to add a student with a duplicate ID and handling the exception.
    - Retrieving a student by ID and handling the `StudentNotFoundException` when an invalid ID is used.
    - Updating a student's details and handling the `StudentNotFoundException` when an invalid ID is used.
    - Deleting a student and handling the `StudentNotFoundException` when an invalid ID is used.
    - Listing all students.
    - Searching for students by course and displaying the results.

Example:

```java
public class Main {
    public static void main(String[] args) {
        StudentManagement management = new StudentManagement();

        try {
            management.addStudent(new Student(1, "Alice", 20, "Computer Science"));
            management.addStudent(new Student(2, "Bob", 22, "Mathematics"));
            management.addStudent(new Student(1, "Charlie", 19, "Physics")); // Should thr
        } catch (Exception e) {
            System.out.println(e.getMessage());
        }

        try {
            Student student = management.getStudentById(1);
            System.out.println(student);

            management.updateStudent(1, "Alice Johnson", 21, "Data Science");
            System.out.println(management.getStudentById(1));

            management.deleteStudent(3); // Should throw exception
        } catch (StudentNotFoundException e) {
            System.out.println(e.getMessage());
        }

        management.listAllStudents().forEach(System.out::println);

        List<Student> csStudents = management.searchStudentsByCourse("Computer Science");
        System.out.println("Students enrolled in Computer Science:");
        csStudents.forEach(System.out::println);
    }
}
```

Instructions:

1. Implement the `Student`, `StudentManagement`, and custom exception classes as described above.
2. Ensure that your methods handle exceptions appropriately, providing meaningful error messages.
3. Test your implementation using the `main` method to validate all scenarios.

Evaluation Criteria:

- Correctness of exception handling and custom exception implementation.
- Proper use of OOP concepts such as encapsulation, class design, and polymorphism (if applicable).
- Clarity and structure of code.
- Comprehensive testing of all possible CRUD scenarios.