# Final Exam Question: Java OOP-Based CRUD Operations Using Streams and ArrayLists

### Problem Description

You are tasked with creating a library management system using Object-Oriented Programming (OOP) principles in Java. The system should manage a collection of books, allowing users to add, remove, update, and search for books in the library. Each book will be represented as an object and stored in an **`ArrayList<Book>`**. You must use Java Stream operations to implement the search and filtering methods instead of traditional loops and conditionals.

### Requirements

1. **Book Class**
   - Create a class named **`Book`** with the following attributes:
     - **`isbn`** (String): The unique ISBN identifier for the book.
     - **`title`** (String): The title of the book.
     - **`author`** (String): The author of the book.
     - **`price`** (double): The price of the book.
   - Implement the following methods in the **`Book`** class:
     - **`public Book(String isbn, String title, String author, double price)`**: Constructor to initialize all attributes.
     - **`public String getIsbn()`**: Returns the **`isbn`** of the book.
     - **`public String getTitle()`**: Returns the **`title`** of the book.
     - **`public String getAuthor()`**: Returns the **`author`** of the book.
     - **`public double getPrice()`**: Returns the **`price`** of the book.
     - **`public void setPrice(double price)`**: Sets the **`price`** of the book.
     - **`@Override public String toString()`**: Returns a string representation of the book, including all attributes.

2. **Library Management**
   - Create a class named `` `Library` `` that manages a collection of `` `Book` `` objects using an `` `ArrayList<Book>` ``.
   - Implement the following methods in the `` `Library` `` class using Java Streams:
     - `` `public void addBook(Book book) throws DuplicateBookException` ``: Adds a new book to the library. Ensure that a book with the same `` `isbn` `` does not already exist in the library.
     - `` `public void removeBook(String isbn) throws BookNotFoundException` ``: Removes a book from the library based on its `` `isbn` ``. If the book does not exist, throw a custom exception `` `BookNotFoundException` ``.
     - `` `public void updateBookPrice(String isbn, double newPrice) throws BookNotFoundException` ``: Updates the price of a book based on its `` `isbn` ``. If the book does not exist, throw a custom exception `` `BookNotFoundException` ``.
     - `` `public Optional<Book> searchBookByIsbn(String isbn)` ``: Searches for a book by its `` `isbn` `` and returns it as an `` `Optional<Book>` ``. If the book is not found, return an empty `` `Optional` ``.
     - `` `public List<Book> searchBooksByTitle(String title)` ``: Returns a list of books whose titles contain the given keyword, using a case-insensitive search.
     - `` `public List<String> getAllAuthors()` ``: Returns a list of all distinct authors in the library.
     - `` `public double calculateTotalPrice()` ``: Returns the total price of all books in the library.
3. **Custom Exception Classes**
   - Create two custom exception classes:
     - `` `BookNotFoundException` ``: Thrown when a book with a specified `` `isbn` `` is not found in the library.
     - `` `DuplicateBookException` ``: Thrown when trying to add a book with an `` `isbn` `` that already exists in the library.
   - Each exception should have at least one constructor that takes a `` `String` `` message and passes it to the superclass `` `Exception` ``.
4. **Main Application**
   - Create a `` `Main` `` class with a `` `main` `` method to test all the functionalities of the `` `Library` `` class.
   - Demonstrate adding, removing, updating, and searching for books.
   - Demonstrate handling of the custom exceptions by providing invalid inputs (e.g., searching for a book that doesn't exist).

**Example Scenario**

1. Add a book with `isbn = "978-0134685991"`, `title = "Effective Java"`, `author = "Joshua Bloch"`, `price = 45.99`.
2. Add another book with `isbn = "978-0201633610"`, `title = "Design Patterns"`, `author = "Erich Gamma"`, `price = 54.99`.
3. Attempt to add a book with `isbn = "978-0134685991"` again (should throw `DuplicateBookException`).
4. Update the price of the book with `isbn = "978-0134685991"` to 49.99.
5. Remove the book with `isbn = "978-1491950357"` (should throw `BookNotFoundException`).
6. Search for a book by `isbn = "978-0201633610"` and display its details.
7. Search for books with the title containing "Effective" and display the results.
8. Retrieve a list of all distinct authors in the library.
9. Calculate and display the total price of all books in the library.

**Grading Criteria**

- Correctness of the implementation (50%)
- Proper use of OOP principles (20%)
- Implementation and use of Java Streams (15%)
- Implementation and use of custom exception classes (10%)
- Code readability and comments (5%)