Question 1: Library Book Reservation System using JDBC

Problem Statement:
Create a JDBC-based menu-driven Java program to manage book reservations in a library. Users can add new books, view all books, search by genre, update availability status, and delete a book record.

Menu Options:

1. Add New Book
2. Display All Books
3. Display Books by Genre
4. Update Book Availability
5. Delete Book by ID
6. Exit

Table Schema:

```
CREATE TABLE Book (
   bookId INT PRIMARY KEY,
   title VARCHAR(150),
   genre VARCHAR(50),
   isAvailable BOOLEAN
);
```

Input Format:

- **For Insert:**
    - Book ID (int)
    - Title (String)
    - Genre (String)
    - Is Available (true/false)
- **For Display by Genre:**
    - Genre name (String)
- **For Update Availability:**
    - Book ID (int)
    - New Availability (true/false)
- **For Delete:**
    - Book ID (int)

Output Format:

- Messages such as:
    - "Database connected successfully."
    - "Book added successfully."

- "Book availability updated."
- "Book deleted successfully."
- "Book not found."
- "Connection closed successfully."

- Display books in tabular format:
  ID | Title | Genre | Available

Question 2: Vehicle Service Center Management using JDBC

Problem Statement:
Develop a JDBC-based menu-driven Java application to manage vehicle service records. Allow filtering by vehicle type and updating service cost.

Menu Options:

1. Add New Service Record
2. Display All Service Records
3. Display Records by Vehicle Type
4. Update Service Cost
5. Delete Record by ID
6. Exit

Table Schema:

```
CREATE TABLE ServiceRecord (
   serviceId INT PRIMARY KEY,
   ownerName VARCHAR(100),
   vehicleType VARCHAR(50),
   serviceCost DOUBLE
);
```

Input Format:

- **For Insert:**
  - Service ID (int)
  - Owner Name (String)
  - Vehicle Type (String, e.g., "Car", "Bike", "Truck")
  - Service Cost (double)
- **For Display by Vehicle Type:**
  - Vehicle type (String)
- **For Update Cost:**
  - Service ID (int)

- New Service Cost (double)

- **For Delete:**
  - Service ID (int)

Output Format:

- Messages such as:
  - "Database connected successfully."
  - "Service record added successfully."
  - "Service cost updated."
  - "Service record deleted."
  - "Record not found."
  - "Connection closed successfully."
- Display service records in tabular format:
  ID | Owner Name | Vehicle Type | Service Cost

Question 3: String List Operations using ArrayList

Problem Statement:
Write a menu-driven Java program to perform the following operations on an ArrayList of strings with some advanced string manipulations.

Menu Options:

1. Add a string to the list
2. Insert a string at a specific index
3. Check if a string exists (case-insensitive)
4. Remove a string by index
5. Display all strings in UPPERCASE
6. Display all strings in lowercase
7. Count how many strings have length greater than a given number
8. Display strings that contain only digits
9. Reverse each string in the list and print
10. Clear the entire list
11. Exit

Notes:

- Use ArrayList<String> to store and manage the strings.
- Ensure input validation for operations like insert at index and remove by index.
- Only one loop or method should be used to print all strings.

Question 4: String Filter & Modify Operations using ArrayList

Problem Statement:
Write a menu-driven Java program to manage an ArrayList of strings that allows the user to perform filtering and string transformation operations.

Menu Options:

1. Add a new string
2. Remove all strings shorter than a given length
3. Display all strings starting with a vowel
4. Replace a specific string with another
5. Convert all strings to Title Case (First Letter Capital)
6. Find and print the longest string(s)
7. Find and print the shortest string(s)
8. Remove duplicate strings
9. Display all strings in reverse order of insertion
10. Display all strings with their lengths
11. Exit

Notes:

- Focus on filtering and transforming string content.
- Handle case sensitivity where required.
- Use efficient string operations like replace, substring, toLowerCase, etc.

Question 5: CRUD Operations on a List of Library Members

Problem Statement:
Write a menu-driven Java program to perform CRUD operations on a list of Member objects for a library system. The Member class should contain:

- memberId (int): Unique ID of the member
- name (String): Member's name
- membershipType (String): e.g., "Regular", "Premium"
- booksIssued (int): Number of books currently issued

Also override equals(), hashCode(), and toString() methods in the Member class.

Menu-Driven Operations:

1. Add a Member
2. Display All Members
3. Update Member (by memberId)
4. Delete Member (by memberId)
5. Display members with more than 3 books issued

6. Sort members by name in ascending order
7. Count members by membership type (e.g., how many Premium, how many Regular)
8. Exit

Additional Notes:

- Use ArrayList<Member> for data storage
- Implement methods for each operation
- Perform case-insensitive comparisons where necessary
- Keep prompting the user until Exit is selected

Question 6: CRUD Operations on a List of Movie Records

Problem Statement:
Write a menu-driven Java program to manage a list of Movie records. Each Movie object should have:

- movieId (int): Unique movie ID
- title (String): Movie title
- genre (String): Movie genre (e.g., Action, Comedy)
- rating (double): Rating between 0.0 and 10.0
- duration (int): Duration in minutes

Override equals(), hashCode(), and toString() methods in the Movie class.

Menu-Driven Operations:

1. Add a Movie
2. Display All Movies
3. Update Movie Rating (by movieId)
4. Delete Movie (by movieId)
5. Sort Movies by rating (descending)
6. Display all movies longer than 2 hours
7. Calculate average rating of all movies
8. Exit

Additional Notes:

- Use ArrayList<Movie>
- Ensure ratings are within valid range
- Display durations in a friendly format (e.g., "2h 10m") if desired
- Keep running until the user selects exit