Compiled by Rohit Ahuja (9893075987)

Assignment 1: Library Book Management System

Scenario:

Emma is a software developer working for a library, "City Library Management." The library wants to maintain a digital record of all the books available for lending. Emma has been tasked with creating a book database management system that can store and manage book information for the library.

The database should allow the library staff to:

- Insert new book details as the library acquires new books.
- Update existing book details like the number of copies available.
- Delete outdated book records if a book is no longer in the library.
- View all book details for inventory management.

Emma needs to develop a Java application using JDBC and MySQL for this purpose. The application should support CRUD (Create, Read, Update, Delete) operations, and the library wants the system to handle SQLExceptions appropriately.

Table Name: `book`

Table Structure:

- `bookId`: INT (Primary Key)
- `bookTitle`: VARCHAR(100) (Not NULL)
- `author`: VARCHAR(50)
- `genre`: VARCHAR(30)
- `copiesAvailable`: INT

NOTE:

- Utilize try/catch and finally block to handle the SQLException.
- The table name is case-sensitive and must match the one specified above.
- The table is created and some tuples have been already inserted into the table.
- `bookId` range is between 1001 to 1005.

Input Format:

The input begins with a number representing the desired CRUD operation:

1. `1` - Insert (followed by all attributes for the new book)

2. `2` - Update (update the number of copies available based on `bookId`)
3. `3` - Delete (remove a book based on `bookId`)
4. `4` - Show all details (display all existing book records)

Output Format:

The output should perform all the CRUD operations based on a menu-driven approach.

Based on the operation selected, the output should display appropriate messages, such as "Book added successfully," "Book updated successfully," "Book deleted successfully," followed by the updated list of books.

Sample Test Cases:

Testcase 1 Input:

```
1
1006
The Great Gatsby
F. Scott Fitzgerald
Fiction
10
```

Testcase 1 Output:

```
Book added successfully.
Book ID: 1001, Book Title: To Kill a Mockingbird, Author: Harper Lee, Genre: Fiction, Copies Available: 15
Book ID: 1002, Book Title: 1984, Author: George Orwell, Genre: Dystopian, Copies Available: 20
Book ID: 1003, Book Title: The Catcher in the Rye, Author: J.D. Salinger, Genre: Fiction, Copies Available: 12
Book ID: 1004, Book Title: Pride and Prejudice, Author: Jane Austen, Genre: Romance, Copies Available: 8
Book ID: 1005, Book Title: Moby Dick, Author: Herman Melville, Genre: Adventure, Copies Available: 5
Book ID: 1006, Book Title: The Great Gatsby, Author: F. Scott Fitzgerald, Genre: Fiction, Copies Available: 10
```

Testcase 2 Input:

```
2
1001
20
```

Testcase 2 Output:

Book updated successfully.

Book ID: 1001, Book Title: To Kill a Mockingbird, Author: Harper Lee, Genre: Fiction, Copies Available: 20

Assignment 2: Employee Record Management System

Scenario:

Sophia is a software developer working for an organization, "Tech Solutions Pvt Ltd." The HR team wants to maintain an employee record management system to store and update employee details. Sophia is assigned to create a database management system for managing the employee records.

The database should allow the HR team to:

- Insert new employee details as they hire new employees.
- Update existing employee details like salary as they receive promotions.
- Delete outdated employee records if an employee leaves the company.
- View all employee details for payroll processing and reporting.

Sophia needs to develop a Java application using JDBC and MySQL for this purpose. The application should support CRUD (Create, Read, Update, Delete) operations, and the organization wants the system to handle SQLExceptions appropriately.

Table Name: `employee`

Table Structure:

- `employeeId`: INT (Primary Key)
- `employeeName`: VARCHAR(50) (Not NULL)
- `department`: VARCHAR(30)
- `salary`: DOUBLE
- `dateOfJoining`: DATE

NOTE:

- Utilize try/catch and finally block to handle the SQLException.
- The table name is case-sensitive and must match the one specified above.
- The table is created and some tuples have been already inserted into the table.
- `employeeId` range is between 101 to 105.

Input Format:

The input begins with a number representing the desired CRUD operation:

1. `1` - Insert (followed by all attributes for the new employee)
2. `2` - Update (update the salary based on `employeeId`)
3. `3` - Delete (remove an employee based on `employeeId`)
4. `4` - Show all details (display all existing employee records)

Output Format:

The output should perform all the CRUD operations based on a menu-driven approach.

Based on the operation selected, the output should display appropriate messages, such as "Employee added successfully," "Employee updated successfully," "Employee deleted successfully," followed by the updated list of employees.

Sample Test Cases:

Testcase 1 Input:

```
1
106
Alice Johnson
IT
55000
2022-01-15
```

Testcase 1 Output:

```
Employee added successfully.
Employee ID: 101, Employee Name: John Doe, Department: HR, Salary: 50000, Date of Joining: 2020-06-10
Employee ID: 102, Employee Name: Jane Smith, Department: Finance, Salary: 60000, Date of Joining: 2019-08-25
Employee ID: 103, Employee Name: Bob Brown, Department: IT, Salary: 65000, Date of Joining: 2021-04-12
Employee ID: 104, Employee Name: Clara Lee, Department: Marketing, Salary: 55000, Date of Joining: 2018-11-30
Employee ID: 105, Employee Name: David Miller, Department: Sales, Salary: 52000, Date of Joining: 2017-02-05
Employee ID: 106, Employee Name: Alice Johnson, Department: IT, Salary: 55000, Date of Joining: 2022-01-15
```

Testcase 2 Input:

2

103

70000

Testcase 2 Output:

Employee updated successfully.

Employee ID: 103, Employee Name: Bob Brown, Department: IT, Salary: 70000, Date of Joining: 2021-04-12

Assignment 3: Student Data Management System

Scenario:

You're assigned to develop a Student Data Management System for a college using JDBC and MySQL. The application should support CRUD (Create, Read, Update, Delete) operations on student data stored in a database. The system should be able to manage student details such as student ID, name, department, year of study, and grade.

Table Name: `student`

Table Structure:

- `student_id`: INT (Primary Key)
- `student_name`: VARCHAR(50) (NOT NULL)
- `department`: VARCHAR(50)
- `year_of_study`: INT
- `grade`: VARCHAR(2)

Operations:

1. Insert Student Details: Add a new student record with all the details including student_id, student_name, department, year_of_study, and grade.
2. Update Student Details: Update the department, year_of_study, and grade for a student based on their student_id.
3. Delete Student Details: Remove a student record based on their student_id.
4. Show All Student Details: Display all student details, sorted by department in ascending order.

NOTE:

- Utilize `try/catch` and `finally` blocks to handle `SQLException`.

- The table name is case-sensitive and must match the one specified above.

- The table is already created and contains some records.

- `student_id` should be between 101 and 105.

Input Format:

The input consists of a number representing the CRUD operation:

1 - Insert (followed by all attributes for the new student)

2 - Update (update department, year_of_study, and grade based on `student_id`)

3 - Delete (remove a student based on `student_id`)

4 - Show all details (display all existing student records)

Output Format:

Based on the operation selected, display appropriate messages such as "Student added successfully," "Student updated successfully," "Student deleted successfully," followed by the updated list of student records.

Sample Test Cases:

- Test Case 1:

Input:

```
1

106

Alice Johnson

Computer Science

2

A
```

Output:

```
Student added successfully.

Student ID: 101, Student Name: John Doe, Department: Computer Science, Year of Study: 3, Grade: B

Student ID: 102, Student Name: Jane Smith, Department: Mechanical, Year of Study: 2, Grade: A

Student ID: 103, Student Name: Mike Davis, Department: Electronics, Year of Study: 1, Grade: B

Student ID: 106, Student Name: Alice Johnson, Department: Computer Science, Year of Study: 2, Grade: A
```

- Test Case 2:

Input:

```
2
101
Electronics
4
A
```

Output:

```
  Student updated successfully.
Student ID: 101, Student Name: John Doe, Department: Electronics, Year of Study: 4, Grade: A
```

- Test Case 3:

Input:

```
3
101
```

Output:

```
Student deleted successfully.
```

- Test Case 4:

Input:

```
4
```

Output:

```
  Student ID: 102, Student Name: Jane Smith, Department: Mechanical, Year of Study: 2, Grade: A
```

Assignment 4: Product Inventory Management

Scenario:

You are developing a Product Inventory Management System using JDBC and MySQL for an e-commerce company. The application should support CRUD (Create, Read, Update, Delete) operations on product data stored in a database. The system should be able to manage product details such as product ID, product name, category, quantity, and price.

Table Name: `product`

Table Structure:

- `product_id`: INT (Primary Key)
- `product_name`: VARCHAR(100) (NOT NULL)
- `category`: VARCHAR(50)
- `quantity`: INT
- `price`: DECIMAL(10, 2)

Operations:

1. Insert Product Details: Add a new product with all the details including product_id, product_name, category, quantity, and price.
2. Update Product Details: Update the product name, category, quantity, and price based on product_id.
3. Delete Product Details: Remove a product based on product_id.
4. Show All Product Details: Display all product details, sorted by category in ascending order.

NOTE:

- Utilize `try/catch` and `finally` blocks to handle `SQLException`.
- The table name is case-sensitive and must match the one specified above.
- The table is already created and contains some records.
- `product_id` should be between 101 and 105.

Input Format:

The input consists of a number representing the CRUD operation:

1 - Insert (followed by all attributes for the new product)

2 - Update (update product name, category, quantity, and price based on `product_id`)

3 - Delete (remove a product based on `product_id`)

4 - Show all details (display all existing product records)

Output Format:

Based on the operation selected, display appropriate messages such as "Product added successfully," "Product updated successfully," "Product deleted successfully," followed by the updated list of product records.

Sample Test Cases:

- Test Case 1:

```
Input:
1
106
Samsung Galaxy S21
Electronics
50
999.99
```

```
Output:
  Product added successfully.
Product ID: 103, Product Name: HP Pavilion Laptop, Category: Computers, Quantity: 30, Price: 1200.00
Product ID: 104, Product Name: Sony Headphones, Category: Electronics, Quantity: 100, Price: 150.00
Product ID: 106, Product Name: Samsung Galaxy S21, Category: Electronics, Quantity: 50, Price: 999.99
```

- Test Case 2:

```
Input:
2
103
HP Spectre x360
Computers
40
1500.00
```

Output:

```
Product updated successfully.
Product ID: 103, Product Name: HP Spectre x360, Category: Computers, Quantity: 40, Price: 1500.00
```

- Test Case 3:

Input:

```
3
104
```

Output:

```
 Product deleted successfully.
```

- Test Case 4:

Input:

```
4
```

Output:

```
 Product ID: 103, Product Name: HP Spectre x360, Category: Computers, Quantity: 40, Price: 1500.00
```

Assignment 5: `Book Data Management`

Scenario: You are tasked with developing a Java application to manage book information using JDBC and MySQL. The application should allow users to perform CRUD (Create, Read, Update, Delete) operations on book records stored in a database.

Table Name: `book`

Table Structure:

- `bookId`: `INT` (Primary Key)
- `bookTitle`: `VARCHAR(100)` NOT NULL
- `author`: `VARCHAR(100)` NOT NULL
- `genre`: `VARCHAR(50)`
- `price`: `DOUBLE`

Operations:

1. Insert Book Details: Add a new book entry with all details, including `bookId`, `bookTitle`, `author`, `genre`, and `price`.

2. Update Book Details: Update the `price` and `genre` for a book based on its `bookId`.

3. Delete Book Details: Remove a book entry based on its `bookId`.

4. Show All Book Details: Display all book records, sorted alphabetically by `bookTitle`.

NOTE:

- Utilize `try/catch` and `finally` blocks to handle the `SQLException`.

- The table name is case-sensitive and must match the one specified above.

- The table is created, and some tuples have already been inserted into the table.

- `bookId` range is between 1001 to 1005.

Input Format:

- The input consists of a number representing the CRUD operations as follows:

  - `1` - Insert: Followed by all attributes for the new book (`bookId`, `bookTitle`, `author`, `genre`, `price`).

  - `2` - Update: Followed by the `bookId`, new `price`, and new `genre`.

  - `3` - Delete: Followed by the `bookId` of the book to be removed.

  - `4` - Show All: No additional input is required.

Output Format:

The output should perform the selected CRUD operation based on the input. For each operation, display appropriate messages and show updated book details as follows:

- Insert: Display "Book added successfully." followed by the updated list of books sorted by `bookTitle` in alphabetical order.

- Update: Display "Book updated successfully." followed by the updated list of books sorted by `bookTitle` in alphabetical order.

- Delete: Display "Book deleted successfully." followed by the updated list of books sorted by `bookTitle` in alphabetical order.

- Show All: Display all book records sorted by `bookTitle`.

Sample Input and Output:

Input 1:

```
1
1006
The Great Gatsby
F. Scott Fitzgerald
Fiction
10.99
```

Output 1:

```
Book added successfully.
ID: 1006, Title: The Great Gatsby, Author: F. Scott Fitzgerald, Genre: Fiction, Price: 10.99
ID: 1001, Title: Brave New World, Author: Aldous Huxley, Genre: Science Fiction, Price: 9.50
ID: 1002, Title: Moby Dick, Author: Herman Melville, Genre: Adventure, Price: 12.75
```

Input 2:

```
2
1001
11.50
Dystopian
```

Output 2:

```
Book updated successfully.
ID: 1001, Title: Brave New World, Author: Aldous Huxley, Genre: Dystopian, Price: 11.50
ID: 1002, Title: Moby Dick, Author: Herman Melville, Genre: Adventure, Price: 12.75
```

Assignment 6: `Customer Data Management`

Scenario: You are tasked with developing a Java application to manage customer information using JDBC and MySQL. The application should allow users to perform CRUD (Create, Read, Update, Delete) operations on customer records stored in a database.

Table Name: `customer`

Table Structure:

- `customerId`: `INT` (Primary Key)
- `customerName`: `VARCHAR(100)` NOT NULL
- `email`: `VARCHAR(100)` UNIQUE NOT NULL
- `phoneNumber`: `VARCHAR(15)`
- `city`: `VARCHAR(50)`

Operations:

1. Insert Customer Details: Add a new customer entry with all details, including `customerId`, `customerName`, `email`, `phoneNumber`, and `city`.
2. Update Customer Details: Update the `email` and `city` for a customer based on their `customerId`.
3. Delete Customer Details: Remove a customer entry based on their `customerId`.
4. Show All Customer Details: Display all customer records, sorted alphabetically by `customerName`.

NOTE:

- Utilize `try/catch` and `finally` blocks to handle the `SQLException`.
- The table name is case-sensitive and must match the one specified above.
- The table is created, and some tuples have already been inserted into the table.
- `customerId` range is between 1001 to 1005.

Input Format:

- The input consists of a number representing the CRUD operations as follows:
  - `1` - Insert: Followed by all attributes for the new customer (`customerId`, `customerName`, `email`, `phoneNumber`, `city`).
  - `2` - Update: Followed by the `customerId`, new `email`, and new `city`.
  - `3` - Delete: Followed by the `customerId` of the customer to be removed.
  - `4` - Show All: No additional input is required.

Output Format:

The output should perform the selected CRUD operation based on the input. For each operation, display appropriate messages and show updated customer details as follows:

- Insert: Display "Customer added successfully." followed by the updated list of customers sorted by `customerName` in alphabetical order.
- Update: Display "Customer updated successfully." followed by the updated list of customers sorted by `customerName` in alphabetical order.

- Delete: Display "Customer deleted successfully." followed by the updated list of customers sorted by `customerName` in alphabetical order.

- Show All: Display all customer records sorted by `customerName`.

Sample Input and Output:

Input 1:

```
1

1006

John Doe

johndoe@example.com

123-456-7890

Los Angeles
```

Output 1:

```
Customer added successfully.

ID: 1001, Name: Alice Smith, Email: alice@example.com, Phone: 987-654-3210, City: New York

ID: 1006, Name: John Doe, Email: johndoe@example.com, Phone: 123-456-7890, City: Los Angeles

ID: 1002, Name: Bob Johnson, Email: bob@example.com, Phone: 456-789-0123, City: San Francisco
```

Input 2:

```
2

1002

bob.johnson@newmail.com

San Diego
```

Output 2:

```
Customer updated successfully.

ID: 1001, Name: Alice Smith, Email: alice@example.com, Phone: 987-654-3210, City: New York

ID: 1002, Name: Bob Johnson, Email: bob.johnson@newmail.com, Phone: 456-789-0123, City: San Diego
```