CSIT751 (Core Java)

Module IV – Collections and GUI Programming

Programming Exercises for Practice (Practical Home Assignment)

Q1. Library Book Management – Using `ArrayList`

Scenario:
A library maintains a list of book titles. The system should allow adding new books, removing books, and displaying the complete book list.

Program Structure:

- Class: `Library`
  - Use **`ArrayList<String>`** to store book titles.
  - Methods: `addBook()`, `removeBook()`, `displayBooks()`.
- Menu-driven program.

Input Format:

```
1. Add Book
2. Remove Book
3. Display All Books
4. Exit

Enter choice: 1
Enter book title: Data Structures
```

Expected Output:

```
Book added successfully.
Current Books:
Data Structures
```

Concepts: ArrayList operations (add, remove, iterate)

Q2. Student Attendance System – Using `HashSet`

Scenario:
In a classroom, attendance must not contain duplicates. Using a Set ensures each student is marked only once.

Program Structure:

- Class: `Attendance`
  - Use **`HashSet<String>`** to store student names.
  - Methods: `markAttendance()`, `displayAttendance()`.
- Prevent duplicate entries.

Input Format:

```
Enter name to mark attendance: Rohan
Enter name to mark attendance: Rohan
```

Expected Output:

```
Attendance marked for Rohan
Rohan is already marked present
```

Concepts: Set, uniqueness, no duplicates.

Q3. Bank Account Directory – Using `HashMap`

Scenario:
A bank maintains a directory mapping **account numbers** to **customer names**. It should allow adding accounts, retrieving customer name by account number, and displaying all entries.

Program Structure:

- Class: `BankDirectory`
  - Use **`HashMap<Integer, String>`**.
  - Methods: `addAccount()`, `getCustomer()`, `displayAll()`.

Input Format:

```
1. Add Account
2. Get Customer Name
3. Display All Accounts
4. Exit

Enter choice: 1
Enter Account No: 101
Enter Customer Name: Ankit
```

Expected Output:

```
Account added successfully.
Account No: 101 → Ankit
```

Concepts: Map, key-value pair, retrieval.

Q4. Product Inventory Management – Using `HashMap` and `Iterator`

Scenario:
A shop maintains an inventory where each product ID maps to its stock count. The program should add products, update stock, and display inventory using an iterator.

Program Structure:

- Class: `Inventory`
  - Use **`HashMap<Integer, Integer>`** for product ID → stock.
  - Methods: `addProduct()`, `updateStock()`, `displayInventory()`.

Input Format:

```
1. Add Product
2. Update Stock
3. Display Inventory
4. Exit

Enter choice: 1
Enter Product ID: 101
Enter Stock: 50
```

Expected Output:

```
Product 101 added with stock 50
```

Concepts: HashMap, Iterator traversal.

Q5. Online Course Enrollment – Using `LinkedHashSet`

Scenario:
Students enroll in online courses. The enrollment list must be stored **in the order of registration** but without duplicates. `LinkedHashSet` is ideal here.

Program Structure:

- Class: `CourseEnrollment`
  - Use **`LinkedHashSet<String>`**.
  - Methods: `enrollStudent()`, `displayEnrolledStudents()`.

Input Format:

```
Enroll: Aditi
Enroll: Rohan
Enroll: Aditi
```

Expected Output:

```
Enrolled: Aditi
Enrolled: Rohan
Aditi is already enrolled
Enrolled Students: [Aditi, Rohan]
```

Concepts: LinkedHashSet (ordered + unique), hashing

Q6. Student Registration Form — GUI with AWT Components

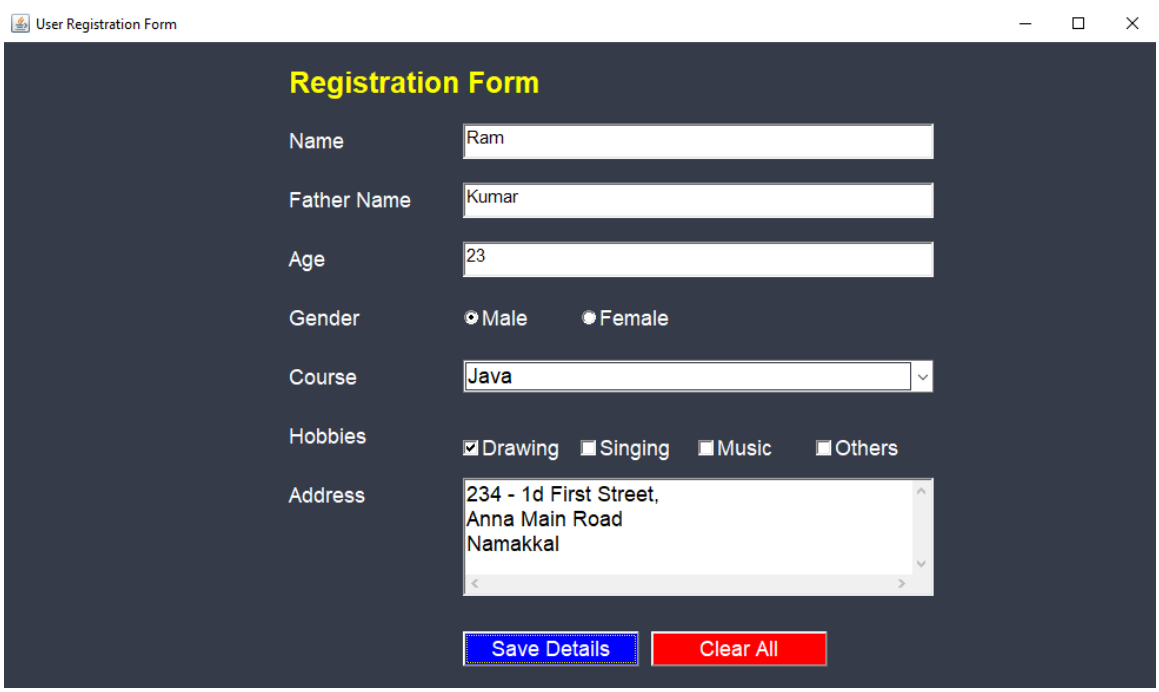
Scenario:

Create a **student registration form** using AWT components. The form should accept basic student details and display them after submission.

Task:

- Use `Frame` as container.
- Add labels and text fields for Name, Roll No, and Course.
- Add a **Submit** button.
- On click, display the entered data in a dialog box or on the console.
- Use **FlowLayout**.

Expected Input (Form View):



Expected Output (After Submission):

A pop-up message box should appear to display student successful registration along with his filled details.

Q7. Calculator Form — Event-Driven Programming

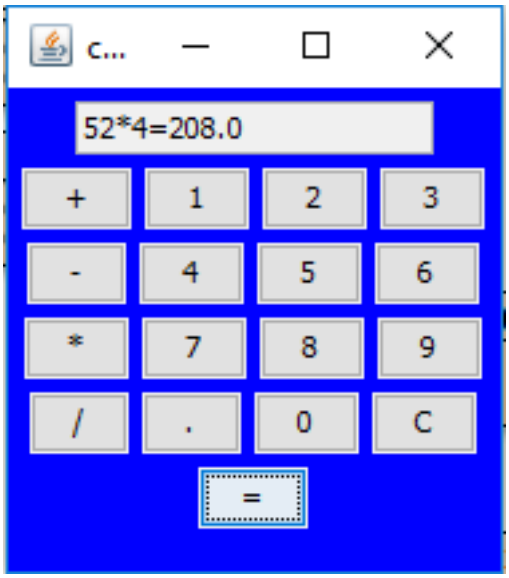
Scenario:

Create a simple calculator GUI to perform addition, subtraction, multiplication, and division between two numbers.

Task:

- Use `JFrame`, `JTextField`, and `JButton`.
- Add event handling for each operator button.
- Display the result in a text field or label.

Expected Input (Form View):



Expected Output (After Submission):

Display the correct result after applying any operation.