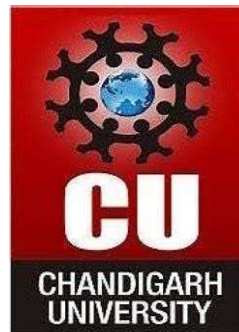


UNIVERSITY INSTITUTE OF ENGINEERING

Department of Computer Science & Engineering

(BE-CSE 5th Sem)



Project Based Learning Java

Subject Code: 23CSH-304

Assignment: 3

Submitted to:

ER. Praveen Kumar [E13782]

Submitted by:

Name: Rajat Puri

UID: 23BCS10183

Section: 23BCS-DC-902

Group: A

1. **Develop a Java application that reads data from a CSV file containing student records, processes the data to calculate the average grade for each student, and stores the results in a database using JDBC.**

Solution:

Table Structure:

```
CREATE TABLE student_avg (  
    id INT PRIMARY KEY,  
    name VARCHAR(100),  
    average DOUBLE  
);
```

Program:

```
import java.io.*;  
import java.sql.*;  
import java.util.*;  
  
public class StudentCSVtoDB {  
    public static void main(String[] args) {  
        String csvFile = "students.csv";  
        String line;  
        String jdbcURL = "jdbc:mysql://localhost:3306/schooldb";  
        String username = "root";  
        String password = "yourpassword";  
  
        try (BufferedReader br = new BufferedReader(new FileReader(csvFile));  
            Connection conn = DriverManager.getConnection(jdbcURL, username, password)) {  
  
            String insertQuery = "INSERT INTO student_avg (id, name, average) VALUES (?, ?, ?)";  
            PreparedStatement pstmt = conn.prepareStatement(insertQuery);  
  
            br.readLine(); // Skip header line  
            while ((line = br.readLine()) != null) {  
                String[] data = line.split(",");  
                int id = Integer.parseInt(data[0]);  
                String name = data[1];  
                double math = Double.parseDouble(data[2]);  
                double science = Double.parseDouble(data[3]);  
                double english = Double.parseDouble(data[4]);  
                double avg = (math + science + english) / 3.0;  
  
                pstmt.setInt(1, id);  
                pstmt.setString(2, name);  
                pstmt.setDouble(3, avg);  
            }  
        }  
    }  
}
```

```
        pstmt.executeUpdate();
    }

    System.out.println("Data inserted successfully!");

} catch (Exception e) {
    e.printStackTrace();
}
}
```

Output: Data inserted successfully into the database

2. Sort a list of employees by salary in descending order using a lambda expression.

Solution:

Program:

```
import java.util.*;
```

```
class Employee {  
    int id;  
    String name;  
    double salary;
```

```
    Employee(int id, String name, double salary) {  
        this.id = id;  
        this.name = name;  
        this.salary = salary;  
    }  
}
```

```
    public String toString() {  
        return id + " - " + name + " - " + salary;  
    }  
}
```

```
public class EmployeeSort {  
    public static void main(String[] args) {  
        List<Employee> employees = Arrays.asList(  
            new Employee(1, "Rajat", 75000),  
            new Employee(2, "Ananya", 90000),  
            new Employee(3, "Arjun", 65000)  
        );  
  
        // Sort by salary descending  
        employees.sort((e1, e2) -> Double.compare(e2.salary, e1.salary));  
  
        System.out.println("Sorted Employees by Salary (Descending):");  
        employees.forEach(System.out::println);  
    }  
}
```

Output:

```
Sorted Employees by Salary (Descending):  
2 - Ananya - 90000.0  
1 - Rajat - 75000.0  
3 - Arjun - 65000.0
```

3. Create a Java program that demonstrates thread synchronization using multiple threads accessing a shared resource, such as a bank account balance, while preventing race conditions and ensuring data integrity.

Solution:

Program:

```
class BankAccount {
    private int balance = 1000;

    // synchronized method to prevent race conditions
    public synchronized void withdraw(int amount, String threadName) {
        if (balance >= amount) {
            System.out.println(threadName + " is withdrawing " + amount);
            try { Thread.sleep(100); } catch (InterruptedException e) { e.printStackTrace(); }
            balance -= amount;
            System.out.println(threadName + " completed withdrawal. Remaining balance: " + balance);
        } else {
            System.out.println(threadName + " - Insufficient balance!");
        }
    }
}

class WithdrawalThread extends Thread {
    private BankAccount account;
    private int amount;

    WithdrawalThread(BankAccount account, int amount, String name) {
        super(name);
        this.account = account;
        this.amount = amount;
    }

    public void run() {
        account.withdraw(amount, getName());
    }
}

public class BankSynchronizationDemo {
    public static void main(String[] args) {
        BankAccount account = new BankAccount();
        WithdrawalThread t1 = new WithdrawalThread(account, 700, "Thread-1");
        WithdrawalThread t2 = new WithdrawalThread(account, 500, "Thread-2");

        t1.start();
        t2.start();
    }
}
```



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

Output:

Thread-1 is withdrawing ₹700

Thread-1 completed withdrawal. Remaining balance: ₹300

Thread-2 - Insufficient balance!