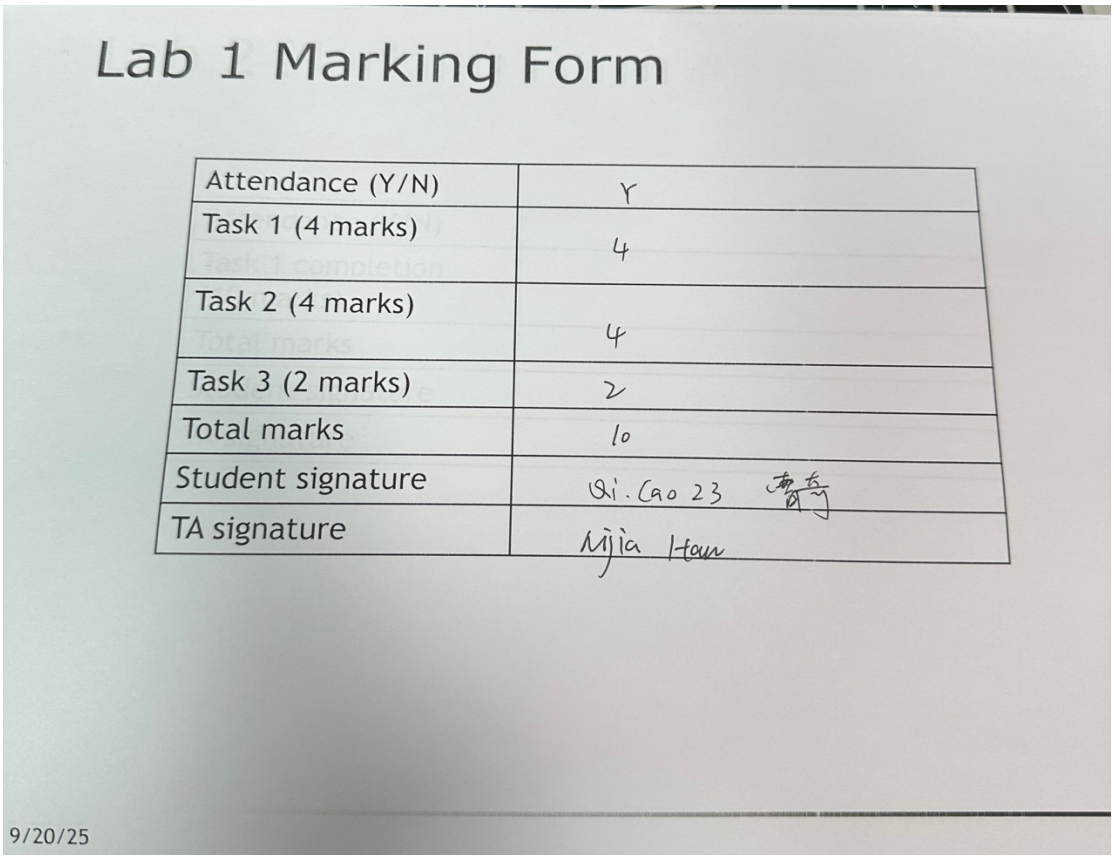
Lab 1 JDBC and Index



Lab 1 Marking Form

Attendance (Y/N)	Y
Task 1 (4 marks)	4
Task 2 (4 marks)	4
Task 3 (2 marks)	2
Total marks	10
Student signature	Qi. Gao 23
TA signature	Mijia Hou

9/20/25

Task 1

The first few rows of the successfully created Orf_Motif table are shown below.

```
mysql> SHOW TABLES;
+--------------------------+
| Tables_in_yeast_prosite |
+--------------------------+
| Orf_Motif                |
+--------------------------+
1 row in set (0.00 sec)

mysql> SELECT * FROM Orf_Motif;
+-----------+---------+------+------+------+--------------+
| orf       | acc_num | num  | pos  | len  | mmatch       |
+-----------+---------+------+------+------+--------------+
| YBL113w-a | PS00005 |    1 |   17 |    3 | SaR          |
| YBL113w-a | PS00005 |    2 |   24 |    3 | TlK          |
| YBL113w-a | PS00005 |    3 |   93 |    3 | ScR          |
| YBL113w-a | PS00005 |    4 |  133 |    3 | SyR          |
| YBL113w-a | PS00008 |    1 |  108 |    6 | GQvpSI       |
| YBL113c   | PS00001 |    1 |   59 |    4 | NSSD         |
| YBL113c   | PS00001 |    2 |   83 |    4 | NAST         |
| YBL113c   | PS00001 |    3 |   87 |    4 | NATT         |
| YBL113c   | PS00001 |    4 |   91 |    4 | NSST         |
| YBL113c   | PS00001 |    5 |   95 |    4 | NATT         |
| YBL113c   | PS00001 |    6 |  131 |    4 | NATT         |
| YBL113c   | PS00001 |    7 |  139 |    4 | NSST         |
| YBL113c   | PS00001 |    8 |  143 |    4 | NATT         |
| YBL113c   | PS00001 |    9 |  163 |    4 | NSST         |
| YBL113c   | PS00001 |   10 |  167 |    4 | NATT         |
| YBL113c   | PS00001 |   11 |  187 |    4 | NSST         |
| YBL113c   | PS00001 |   12 |  191 |    4 | NATT         |
| YBL113c   | PS00001 |   13 |  215 |    4 | NAST         |
| YBL113c   | PS00001 |   14 |  219 |    4 | NATT         |
| YBL113c   | PS00001 |   15 |  223 |    4 | NSST         |
| YBL113c   | PS00001 |   16 |  227 |    4 | NATT         |
| YBL113c   | PS00001 |   17 |  243 |    4 | NATT         |
| YBL113c   | PS00001 |   18 |  247 |    4 | NSST         |
| YBL113c   | PS00001 |   19 |  251 |    4 | NATT         |
| YBL113c   | PS00001 |   20 |  295 |    4 | NSST         |
| YBL113c   | PS00001 |   21 |  299 |    4 | NATT         |
| YBL113c   | PS00001 |   22 |  309 |    4 | NTSA         |
| YBL113c   | PS00001 |   23 |  323 |    4 | NATT         |
| YBL113c   | PS00001 |   24 |  345 |    4 | NTSA         |
| YBL113c   | PS00001 |   25 |  357 |    4 | NTSA         |
| YBL113c   | PS00001 |   26 |  367 |    4 | NSST         |
| YBL113c   | PS00001 |   27 |  371 |    4 | NATT         |
```

Task 2 and 3

The comparison between indexed and non-indexed query execution is presented below. Note that the raw data printout has been removed to improve readability.

```
Succeeded connecting to Database.
Existing index on num dropped.
Start Query without an index.
End query without an index.
End query without an index. Time=24.000 ms
Index numIndex created on Orf_Motif(num).
Start Query with an index.
End query with an index. Time=8.000 ms
```

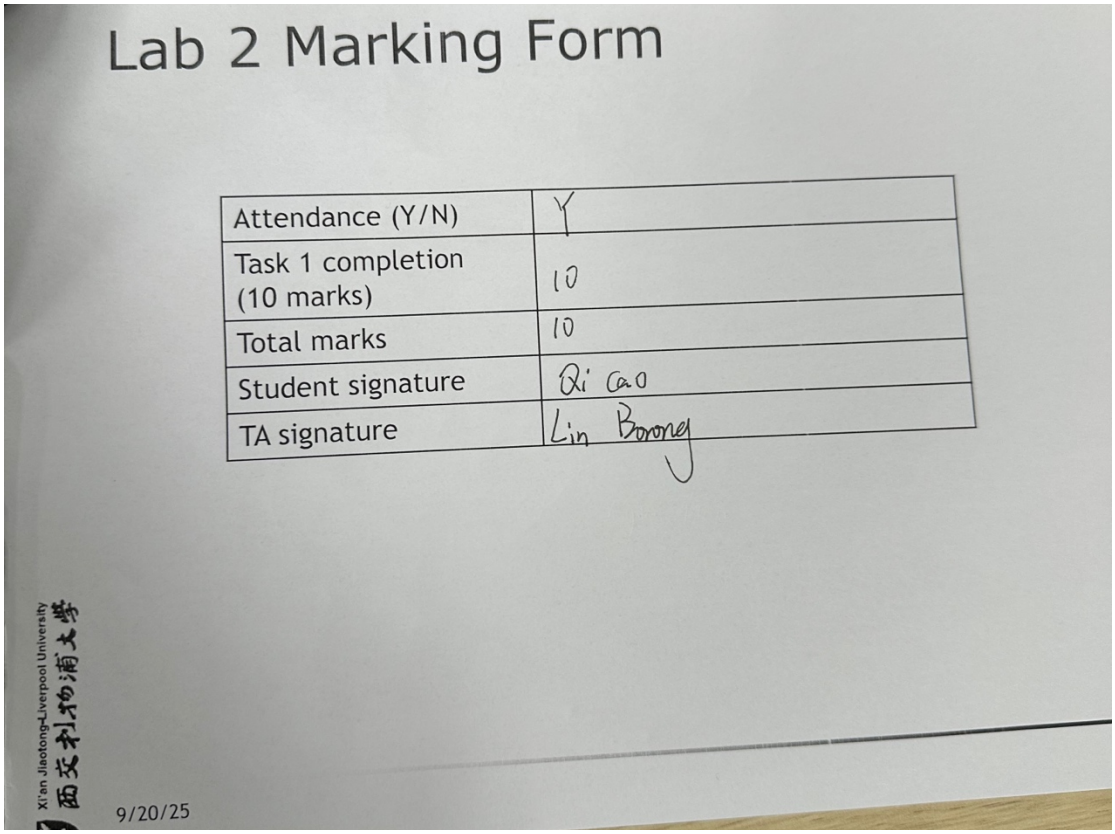
The source code is shown below.

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;

public class JDBCandIndex {

    //Database connection parameters
    //Replace following parameter values with one's own
    private static final String DB_URL =
"jdbc:mysql://127.0.0.1:3306/yeast_prosite";
    private static final String USER = "root";
    private static final String PASS = "02390239Caoqi";

    private static final String JDBC_DRIVER =
"com.mysql.cj.jdbc.Driver";
    private static Connection conn = null;
    private static Statement stmt = null;
    private static ResultSet rs = null;
    private static String sql = null;
    //Search key to create an index
    private static String searchKey = "num";
    //Search key value used in this example
    private static String searchKeyValue = "7";
    //Variables to computer query time
    private static long begin;
    private static long end;

    private void testJDBCQueryWithoutIndex() throws SQLException {
        //drop existing index
        try{
            sql = "Drop INDEX numIndex ON Orf_Motif";
            stmt.execute(sql);
            System.out.println("Existing index on num dropped.");
        }catch (SQLException e) {
            //Index doesn't exist
            System.out.println("No existing index to drop.");
        }

        //prepare query without an index
        System.out.println("Start Query without an index.");
```

```
        sql = "SELECT * FROM Orf_Motif WHERE " + searchKey + "=" +
searchKeyValue;
        begin = System.currentTimeMillis();
        rs = stmt.executeQuery(sql);
        System.out.println("-----------------");
        System.out.println("Result is:");
        System.out.println("-----------------");
        System.out.println(" first col" + "\t" + " second col");
        System.out.println("-----------------");

        //print actual data from the query
        while (rs.next()) {
            System.out.println(rs.getString("acc_num") + "\t" +
rs.getString("orf"));
        }
        end = System.currentTimeMillis();
        double time = end - begin;
        System.out.println("End query without an index.");
        System.out.printf("End query without an index. Time=%.3f
ms%n", time);
    }

    private void testJDBCQueryWithIndex() throws SQLException {
        try {
            sql = "CREATE INDEX numIndex ON Orf_Motif (num)";
            stmt.execute(sql);
            System.out.println("Index numIndex created on
Orf_Motif(num).");
        } catch (SQLException e) {
            System.out.println("Index numIndex already exists or cannot
be created: " + e.getMessage());
        }

        System.out.println("Start Query with an index.");

        sql = "SELECT * FROM Orf_Motif WHERE " + searchKey + "=" +
searchKeyValue;
        begin = System.currentTimeMillis();
        rs = stmt.executeQuery(sql);
        System.out.println("-----------------");
        System.out.println("Result is:");
        System.out.println("-----------------");
        System.out.println(" first col" + "\t" + " second col");
        System.out.println("-----------------");
```

```
        while (rs.next()) {
            System.out.println(rs.getString("acc_num") + "\t" +
rs.getString("orf"));
        }
        end = System.currentTimeMillis();
        double time = end - begin;
        System.out.printf("End query with an index. Time=%.3f ms%n",
time);
    }

    public static void main(String[] args) throws SQLException,
ClassNotFoundException {

        JDBCandIndex testInstance = new JDBCandIndex();

        Class.forName(JDBC_DRIVER);

        conn = DriverManager.getConnection(DB_URL, USER, PASS);
        if (!conn.isClosed()) {
            System.out.println("Succeeded connecting to Database.");
        }

        stmt = conn.createStatement();

        //TEST QUERY WITHOUT AN INDEX
        testInstance.testJDBCQueryWithoutIndex();
        //END TEST QUERY WITHOUT AN INDEX

        //TEST QUERY WITH AN INDEX
        testInstance.testJDBCQueryWithIndex();
        //END TEST QUERY WITH AN INDEX

        //close database connections.
        rs.close();
        stmt.close();
        conn.close();
    }

}
```

Lab 2 Java Transaction



Lab 2 Marking Form

Attendance (Y/N)	Y
Task 1 completion (10 marks)	10
Total marks	10
Student signature	Qi Cao
TA signature	Lin Boorong

Xi'an Jiaotong-Liverpool University 西交利物浦大学

9/20/25

Task 1.1

The test results are shown below. We can see that the code failed to prevent invalid transactions.

```
Initial balances:
  1: Alice = 1000.0000
  2: Bob = 500.0000

--- Successful transfer (commit) ---
  Before: from=1000.0000, to=500.0000
  Transfer committed.
  1: Alice = 750.0000
  2: Bob = 750.0000

--- Transfer that should not be allowed ---
  Before: from=750.0000, to=750.0000
  Transfer committed.
  1: Alice = -1250.0000
  2: Bob = 2750.0000

Demo finished.

进程已结束，退出代码为 0
```

Task 1.2

The results of the new method for Task 1.1, which validates transactions using JDBC, are shown below.

```
Initial balances:
  1: Alice = 1000.0000
  2: Bob = 500.0000

--- Successful transfer (commit) ---
  Validation passed: Transfer of 250.00 from account 1 to account 2
  Before: from=1000.0000, to=500.0000
  Transfer committed.
  1: Alice = 750.0000
  2: Bob = 750.0000

--- Transfer that should not be allowed ---
  Validation failed: Insufficient balance. Available: 750.0000, Requested: 2000.00
  Transfer failed validation and was rolled back.
  1: Alice = 750.0000
  2: Bob = 750.0000

Demo finished.

进程已结束，退出代码为 0
```

The source code is shown below.

```
import java.math.BigDecimal;
import java.sql.*;

public class TransactionDemo {

    // TODO: adjust these for your environment
    private static final String DB_URL =
"jdbc:mysql://127.0.0.1:3306/transactiondb?useSSL=false&serverTimezon
e=UTC";
    private static final String DB_USER = "root";
    private static final String DB_PASSWORD = "02390239Caoqi";

    private static final String CREATE_TABLE_SQL =
            "CREATE TABLE IF NOT EXISTS acco" +
                    "unts ("
                    + "id INT PRIMARY KEY,"
                    + "name VARCHAR(100) NOT NULL,"
                    + "balance DECIMAL(19,4) NOT NULL"
                    + ") ENGINE=InnoDB";

    private static final String DELETE_DEMO_SQL = "DELETE FROM
accounts WHERE id IN (1,2)";
    private static final String INSERT_ACCOUNT_SQL = "INSERT INTO
accounts (id, name, balance) VALUES (?, ?, ?)";
    private static final String SELECT_ALL_SQL = "SELECT id, name,
balance FROM accounts ORDER BY id";
    private static final String SELECT_FOR_UPDATE_SQL = "SELECT
balance FROM accounts WHERE id = ? FOR UPDATE";
    private static final String UPDATE_BALANCE_SQL = "UPDATE accounts
SET balance = ? WHERE id = ?";

    public static void main(String[] args) {
        try {
            // Optional: explicit driver load (usually not necessary
with modern JDBC)
            try {
                Class.forName("com.mysql.cj.jdbc.Driver");
            } catch (ClassNotFoundException ignored) {}

            try (Connection conn = DriverManager.getConnection(DB_URL,
DB_USER, DB_PASSWORD)) {
                System.out.println("Connected to database: " + DB_URL);

                initSchema(conn);
```

```
            seedDemoData(conn);

            System.out.println("\nInitial balances:");
            printAllBalances(conn);

            System.out.println("\n--- Successful transfer (commit)
---");
            performTransferWithValidationAndRollback(conn, 1, 2,
new BigDecimal("250.00"));
            printAllBalances(conn);

            System.out.println("\n--- Transfer that should not be
allowed ---");
            performTransferWithValidationAndRollback(conn, 1, 2,
new BigDecimal("2000.00"));
            printAllBalances(conn);

            System.out.println("\nDemo finished.");
        }
    } catch (SQLException ex) {
        System.err.println("Database error: " + ex.getMessage());
        ex.printStackTrace();
    }
}


   private static boolean validateTransfer(Connection conn, int
fromId, int toId, BigDecimal amount) throws SQLException {
    if (amount.compareTo(BigDecimal.ZERO) <= 0) {
        System.out.println("  Validation failed: Amount must be
positive");
        return false;
    }

    BigDecimal fromBal = selectBalanceForUpdate(conn, fromId);
    if (fromBal.compareTo(amount) < 0) {
        System.out.println("  Validation failed: Insufficient
balance. Available: " + fromBal.toPlainString() + ", Requested: " +
amount.toPlainString());
        return false;
    }

    System.out.println("  Validation passed: Transfer of " +
amount.toPlainString() + " from account " + fromId + " to account " +
```

```
toId);
        return true;
    }


    private static void initSchema(Connection conn) throws
SQLException {
        try (Statement st = conn.createStatement()) {
            st.execute(CREATE_TABLE_SQL);
        }
    }

    private static void seedDemoData(Connection conn) throws
SQLException {
        try (Statement st = conn.createStatement()) {
            st.executeUpdate(DELETE_DEMO_SQL);
        }
        try (PreparedStatement ps =
conn.prepareStatement(INSERT_ACCOUNT_SQL)) {
            ps.setInt(1, 1);
            ps.setString(2, "Alice");
            ps.setBigDecimal(3, new BigDecimal("1000.00"));
            ps.executeUpdate();

            ps.setInt(1, 2);
            ps.setString(2, "Bob");
            ps.setBigDecimal(3, new BigDecimal("500.00"));
            ps.executeUpdate();
        }
    }

    private static void printAllBalances(Connection conn) throws
SQLException {
        try (Statement st = conn.createStatement();
             ResultSet rs = st.executeQuery(SELECT_ALL_SQL)) {
            while (rs.next()) {
                int id = rs.getInt("id");
                String name = rs.getString("name");
                BigDecimal bal = rs.getBigDecimal("balance");
                System.out.printf("  %d: %s = %s%n", id, name,
bal.toPlainString());
            }
        }
    }
```

```
    // Simple transfer: uses transaction and commits when all is ok
    private static void performTransfer(Connection conn, int fromId,
int toId, BigDecimal amount) {
        try {
            conn.setAutoCommit(false);

            BigDecimal fromBal = selectBalanceForUpdate(conn, fromId);
            BigDecimal toBal = selectBalanceForUpdate(conn, toId);

            if (fromBal == null || toBal == null) {
                throw new SQLException("Account not found.");
            }

            System.out.printf("  Before: from=%s, to=%s%n",
fromBal.toPlainString(), toBal.toPlainString());

            BigDecimal newFrom = fromBal.subtract(amount);
            BigDecimal newTo = toBal.add(amount);

            updateBalance(conn, fromId, newFrom);
            updateBalance(conn, toId, newTo);

            conn.commit();
            System.out.println("  Transfer committed.");
        } catch (SQLException ex) {
            safeRollback(conn);
            System.out.println("  Transfer rolled back: " +
ex.getMessage());
        } finally {
            safeResetAutoCommit(conn);
        }
    }

    private static BigDecimal selectBalanceForUpdate(Connection conn,
int id) throws SQLException {
        try (PreparedStatement ps =
conn.prepareStatement(SELECT_FOR_UPDATE_SQL)) {
            ps.setInt(1, id);
            try (ResultSet rs = ps.executeQuery()) {
                if (rs.next()) {
                    return rs.getBigDecimal("balance");
                } else {
                    return null;
```

```
            }
        }
    }
}

    private static void updateBalance(Connection conn, int id,
BigDecimal newBalance) throws SQLException {
        try (PreparedStatement ps =
conn.prepareStatement(UPDATE_BALANCE_SQL)) {
            ps.setBigDecimal(1, newBalance);
            ps.setInt(2, id);
            int updated = ps.executeUpdate();
            if (updated != 1) {
                throw new SQLException("Expected to update 1 row for
id=" + id + " but updated " + updated);
            }
        }
    }

    private static void safeRollback(Connection conn) {
        if (conn == null) return;
        try {
            conn.rollback();
        } catch (SQLException e) {
            System.err.println("  Failed to rollback: " +
e.getMessage());
        }
    }

    private static void safeResetAutoCommit(Connection conn) {
        if (conn == null) return;
        try {
            conn.setAutoCommit(true);
        } catch (SQLException e) {
            System.err.println("  Failed to reset autoCommit: " +
e.getMessage());
        }
    }

    private static void
performTransferWithValidationAndRollback(Connection conn, int fromId,
int toId, BigDecimal amount){
        try {
            conn.setAutoCommit(false);
```

```
            if (!validateTransfer(conn, fromId, toId, amount)) {
                System.out.println("  Transfer failed validation and was
rolled back.");
                return;
            }

            BigDecimal fromBal = selectBalanceForUpdate(conn, fromId);
            BigDecimal toBal = selectBalanceForUpdate(conn, toId);

            if (fromBal == null || toBal == null) {
                throw new SQLException("Account not found.");
            }

            System.out.printf("  Before: from=%s, to=%s%n",
fromBal.toPlainString(), toBal.toPlainString());

            BigDecimal newFrom = fromBal.subtract(amount);
            BigDecimal newTo = toBal.add(amount);

            updateBalance(conn, fromId, newFrom);
            updateBalance(conn, toId, newTo);

            conn.commit();
            System.out.println("  Transfer committed.");
        } catch (SQLException ex) {
            safeRollback(conn);
            System.out.println("  Transfer rolled back: " +
ex.getMessage());
        } finally {
            safeResetAutoCommit(conn);
        }

    }
}
```

Task 1.3

The results from the implementation and testing of the distributed transaction application using the Two-Phase Commit (2PC) protocol are presented below. Initially, Alice's account had a balance of 0 and

Bob's had 50. The first screenshot shows Bob successfully transferring 10 to Alice. The second screenshot shows Bob attempting to transfer 100 to Alice, which failed.

```
--- Initial Balance---
  Alice: 0.0000
  Bob: 50.0000

Distributed transaction COMMITTED.

--- Final Result ---
  Alice: 10.0000
  Bob: 40.0000

进程已结束，退出代码为 0
```

```
--- Initial Balance---
  Alice: 0.0000
  Bob: 50.0000

Insufficient funds in accounts_Bob → ROLLBACK both branches
Distributed transaction ROLLED BACK.

--- Final Result ---
  Alice: 0.0000
  Bob: 50.0000
```

The source code is shown below.

```
import com.mysql.cj.jdbc.MysqlXADataSource;

import javax.sql.XAConnection;
import javax.transaction.xa.XAException;
import javax.transaction.xa.XAResource;
import javax.transaction.xa.Xid;
import java.math.BigDecimal;
import java.sql.*;

public class FundTransferJTA {
```

```
    private static final String DB_URL =
"jdbc:mysql://127.0.0.1:3306/transactiondb?useSSL=false&serverTimezon
e=UTC";
    private static final String DB_USER = "root";
    private static final String DB_PASSWORD = "02390239Caoqi";

    public static void main(String[] args) {

        double amount =100.0;

        MysqlXADataSource xaDs1 = new MysqlXADataSource();
        xaDs1.setUrl(DB_URL);
        xaDs1.setUser(DB_USER);
        xaDs1.setPassword(DB_PASSWORD);

        MysqlXADataSource xaDs2 = new MysqlXADataSource();
        xaDs2.setUrl(DB_URL);
        xaDs2.setUser(DB_USER);
        xaDs2.setPassword(DB_PASSWORD);

        byte[] globalId = new byte[]{0x01};
        byte[] branchIdAlice = new byte[]{0x57};
        byte[] branchIdBob   = new byte[]{0x58};

        Xid xid1 = new MyXid(100, globalId, branchIdAlice);
        Xid xid2 = new MyXid(100, globalId, branchIdBob);

        XAConnection xaCon1 = null;
        XAConnection xaCon2 = null;
        Connection conn1 = null;
        Connection conn2 = null;
        Statement stmt1 = null;
        Statement stmt2 = null;
        XAResource xares1 = null;
        XAResource xares2 = null;

        try {
            xaCon1 = xaDs1.getXAConnection();
            xaCon2 = xaDs2.getXAConnection();

            conn1 = xaCon1.getConnection();
            conn2 = xaCon2.getConnection();
```

```
            stmt1 = conn1.createStatement();
            stmt2 = conn2.createStatement();

            xares1 = xaCon1.getXAResource();
            xares2 = xaCon2.getXAResource();

            String table1 = "accounts_Alice";
            String table2 = "accounts_Bob";


            initSchema(conn1, table1);
            initSchema(conn2, table2);

            seedDemoData(conn1, table1, "Alice", 0.0);
            seedDemoData(conn2, table2, "Bob", 50.0);

            System.out.println("\n--- Initial Balance---");
            printBalance(conn1, table1, "Alice");
            printBalance(conn2, table2, "Bob");
            System.out.println("");


            // ================== Phase 1 ==================
            xares1.start(xid1, XAResource.TMNOFLAGS);
            stmt1.execute("UPDATE " + table1 + " SET balance = balance
+ " + amount + " WHERE id = 1");
            xares1.end(xid1, XAResource.TMSUCCESS);
            xares2.start(xid2, XAResource.TMNOFLAGS);


            BigDecimal currentBalance = selectBalanceForUpdate(conn2,
table2, 1);
            BigDecimal transferAmount = BigDecimal.valueOf(amount);

            if (currentBalance == null ||
currentBalance.compareTo(transferAmount) < 0) {
                System.out.println("Insufficient funds in " + table2 +
" → ROLLBACK both branches");
                xares2.end(xid2, XAResource.TMFAIL);

                xares1.rollback(xid1);
                xares2.rollback(xid2);
```

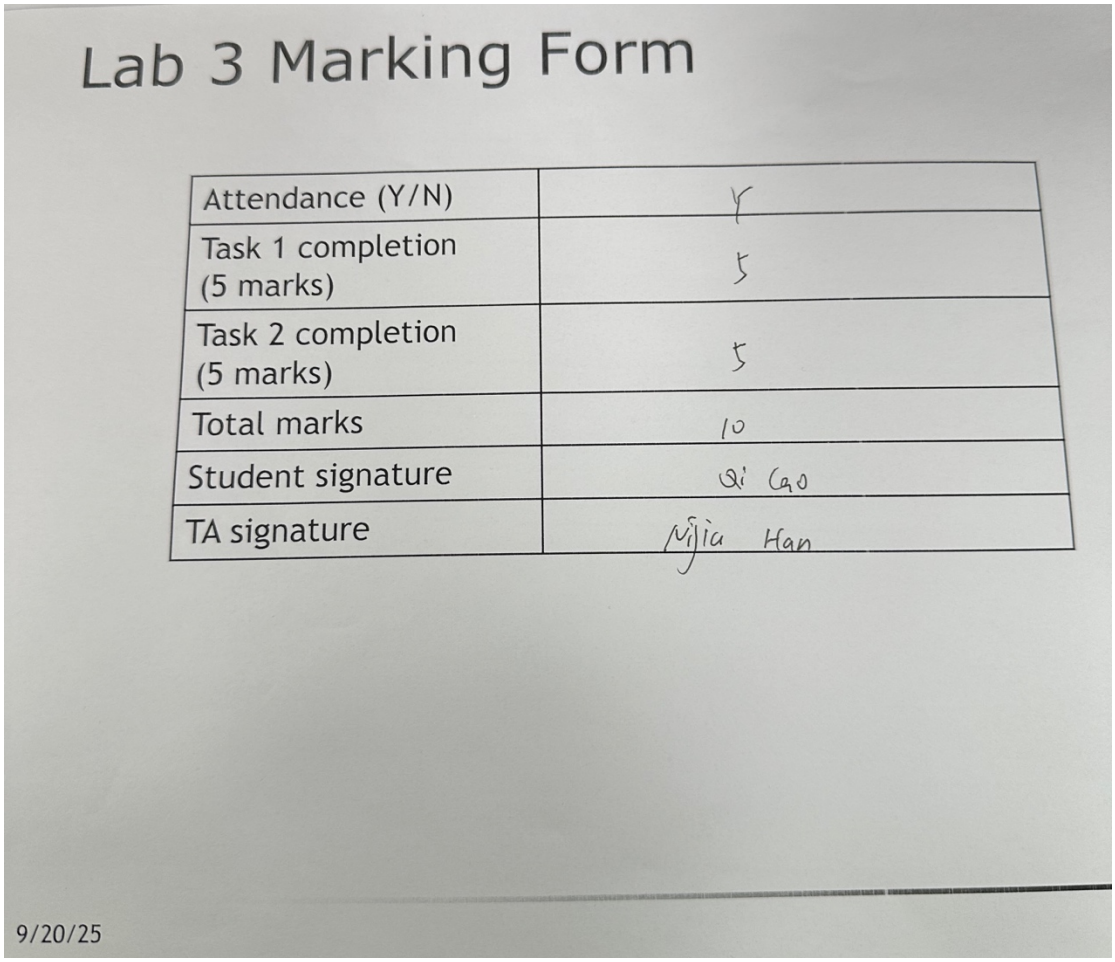
```
            System.out.println("Distributed transaction ROLLED
BACK.");
            System.out.println("\n--- Final Result ---");
            printBalance(conn1, table1, "Alice");
            printBalance(conn2, table2, "Bob");
            return;
        }

        stmt2.execute("UPDATE " + table2 + " SET balance = balance
- " + amount + " WHERE id = 1");
        xares2.end(xid2, XAResource.TMSUCCESS);

        int ret1 = xares1.prepare(xid1);
        int ret2 = xares2.prepare(xid2);

        // ================== Phase 2 ==================
        if (ret1 == XAResource.XA_OK && ret2 == XAResource.XA_OK) {
            xares1.commit(xid1, false);
            xares2.commit(xid2, false);
            System.out.println("Distributed transaction
COMMITTED.");
        } else {
            xares1.rollback(xid1);
            xares2.rollback(xid2);
            System.out.println("Prepare failed → ROLLBACK.");
        }

        System.out.println("\n--- Final Result ---");
        printBalance(conn1, table1, "Alice");
        printBalance(conn2, table2, "Bob");

    } catch (SQLException | XAException e) {
        System.out.println("Exception: " + e.getMessage());
    } finally {
        try {
            if (stmt1 != null) stmt1.close();
            if (stmt2 != null) stmt2.close();
            if (conn1 != null) conn1.close();
            if (conn2 != null) conn2.close();
            if (xaCon1 != null) xaCon1.close();
            if (xaCon2 != null) xaCon2.close();
        } catch (SQLException e) {
            e.printStackTrace();
        }
```

```
        }
    }

    private static void initSchema(Connection conn, String tablename)
throws SQLException {
        try (Statement st = conn.createStatement()) {
            String sql = "CREATE TABLE IF NOT EXISTS " + tablename + "
("
                         + "id INT PRIMARY KEY,"
                         + "name VARCHAR(100) NOT NULL,"
                         + "balance DECIMAL(19,4) NOT NULL"
                         + ") ENGINE=InnoDB";
            st.execute(sql);
        }
    }

    private static void seedDemoData(Connection conn, String
tableName, String ownerName, double initialBalance) throws
SQLException {
        String deleteSql = "DELETE FROM " + tableName;
        try (Statement st = conn.createStatement()) {
            st.execute(deleteSql);
        }

        String insertSql = "INSERT INTO " + tableName + " (id, name,
balance) VALUES (?, ?, ?)";

        try (PreparedStatement ps = conn.prepareStatement(insertSql))
{
            ps.setInt(1, 1);
            ps.setString(2, ownerName);
            ps.setBigDecimal(3, BigDecimal.valueOf(initialBalance));
            ps.executeUpdate();
        }
    }

    private static void printBalance(Connection conn, String
tableName, String name) throws SQLException {
        String sql = "SELECT balance FROM " + tableName + " WHERE name
= ?";
        try (PreparedStatement ps = conn.prepareStatement(sql)) {
            ps.setString(1, name);
            try (ResultSet rs = ps.executeQuery()) {
                if (rs.next()) {
```

```
                System.out.printf("  %s: %s%n", name,
rs.getBigDecimal("balance").toPlainString());
            } else {
                System.out.printf("  %s: Account does not exist%n",
name);
            }
        }
    }


    private static BigDecimal selectBalanceForUpdate(Connection conn,
String tableName, int id) throws SQLException {
        String sql = "SELECT balance FROM " + tableName + " WHERE id
= ? FOR UPDATE";

        try (PreparedStatement ps = conn.prepareStatement(sql)) {
            ps.setInt(1, id);
            try (ResultSet rs = ps.executeQuery()) {
                if (rs.next()) {
                    return rs.getBigDecimal("balance");
                } else {
                    return null;
                }
            }
        }
    }




    static class MyXid implements Xid {

        private int formatId;
        private byte[] globalTransactionId;
        private byte[] branchQualifier;

        public MyXid(int formatId, byte[] globalTransactionId, byte[]
branchQualifier) {
            this.formatId = formatId;
            this.globalTransactionId = globalTransactionId;
            this.branchQualifier = branchQualifier;
        }
```

```
        @Override
        public int getFormatId() {
            return formatId;
        }

        @Override
        public byte[] getGlobalTransactionId() {
            return globalTransactionId;
        }

        @Override
        public byte[] getBranchQualifier() {
            return branchQualifier;
        }
    }


}
```

Lab 3 XML Processing



Lab 3 Marking Form

Attendance (Y/N)	Y
Task 1 completion (5 marks)	5
Task 2 completion (5 marks)	5
Total marks	10
Student signature	Qi Cao
TA signature	Mijia Han

9/20/25

Task 1

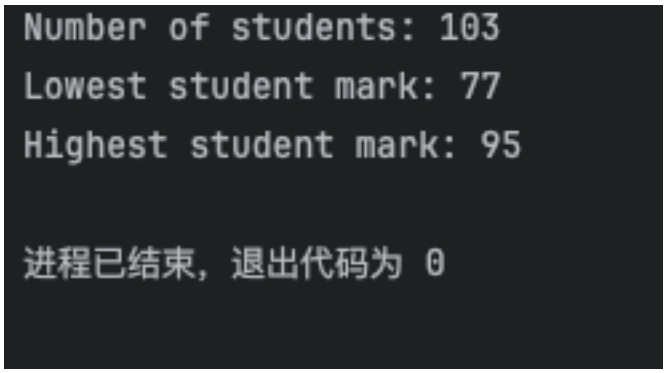
The results of testing XML data processing with SAX and DOM are shown below.

```
Roll No : 393
First Name: dinkar
Last Name: kad
Nick Name: dinkar
Marks: 85
End Element :student
Roll No : 493
First Name: Vaneet
Last Name: Gupta
Nick Name: vinni
Marks: 95
End Element :student
Roll No : 593
First Name: jasvir
Last Name: singn
Nick Name: jazz
Marks: 90
End Element :student

进程已结束，退出代码为 0
```

```
Root element :class
----------------------------

Current Element:student
Student roll no: 393
First Name: dinkar
Last Name: kad
Nick Name: dinkar
Marks: 85

Current Element:student
Student roll no: 493
First Name: Vaneet
Last Name: Gupta
Nick Name: vinni
Marks: 95

Current Element:student
Student roll no: 593
First Name: jasvir
Last Name: singn
Nick Name: jazz
Marks: 90

进程已结束，退出代码为 0
```

Task 2

I chose to use DOM to process a “larger” XML document. The results of the program are shown below.



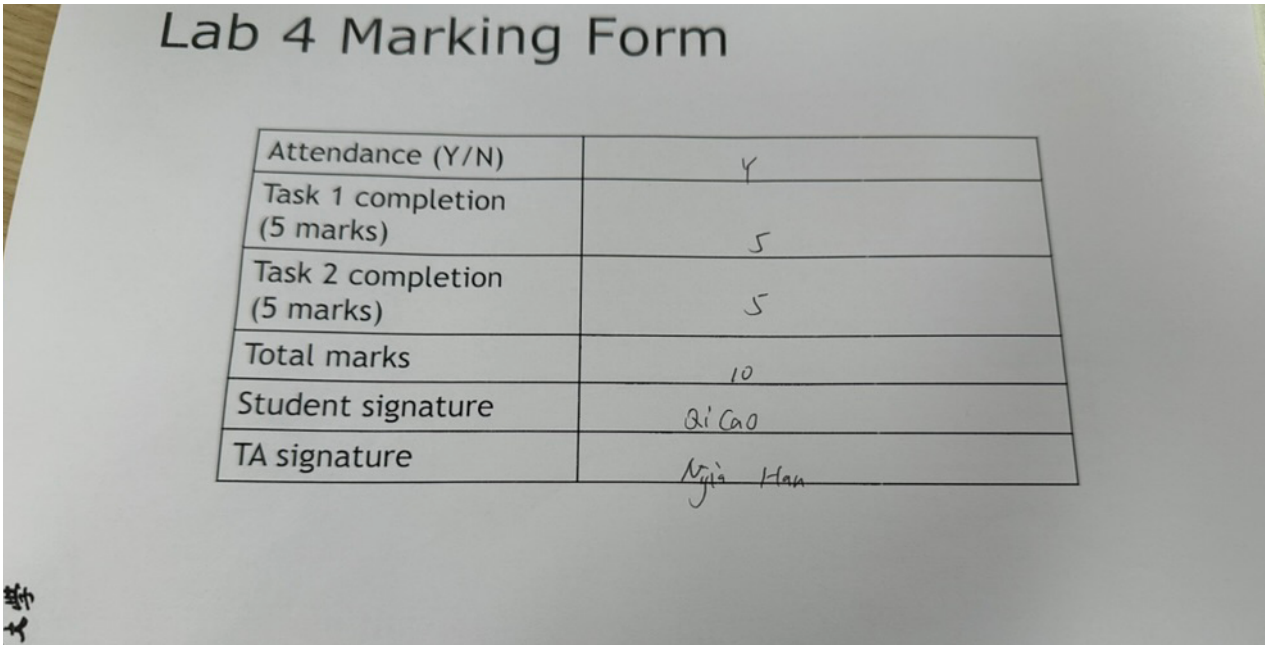
The source code is shown below.

```
import java.io.File;
import javax.xml.parsers.DocumentBuilderFactory;
import javax.xml.parsers.DocumentBuilder;
import org.w3c.dom.Document;
import org.w3c.dom.NodeList;
import org.w3c.dom.Node;
import org.w3c.dom.Element;

public class longVersion {
    public static void main(String[] args) {
        try {
            File inputFile = new File("src/xml/XMLDoc-
longVersion.xml");
            DocumentBuilderFactory dbFactory =
DocumentBuilderFactory.newInstance();
            DocumentBuilder dBuilder = dbFactory.newDocumentBuilder();
            Document doc = dBuilder.parse(inputFile);
            doc.getDocumentElement().normalize();

            int studentCount = countStudents(doc);
            int lowest = lowestMark(doc);
            int highest = highestMark(doc);

            System.out.println("Number of students: " + studentCount);
            System.out.println("Lowest student mark: " + lowest);
```

```
            System.out.println("Highest student mark: " + highest);

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

    }
    public static int countStudents(Document doc) {
        NodeList nList = doc.getElementsByTagName("student");
        return nList.getLength();
    }

    public static int lowestMark(Document doc) {
        NodeList nList = doc.getElementsByTagName("student");
        int lowest = Integer.MAX_VALUE;

        for (int i = 0; i < nList.getLength(); i++) {
            Node node = nList.item(i);
            if (node.getNodeType() == Node.ELEMENT_NODE) {
                Element student = (Element) node;
                int marks =
Integer.parseInt(student.getElementsByTagName("marks")
                        .item(0).getTextContent());
                if (marks < lowest) {
                    lowest = marks;
                }
            }
        }
        return lowest;
    }

    public static int highestMark(Document doc){
        NodeList nList = doc.getElementsByTagName("student");
        int highest = Integer.MIN_VALUE;

        for (int i = 0; i < nList.getLength(); i++) {
            Node node = nList.item(i);
            if (node.getNodeType() == Node.ELEMENT_NODE) {
                Element student = (Element) node;
                int marks =
Integer.parseInt(student.getElementsByTagName("marks")
                        .item(0).getTextContent());
                if (marks > highest) {
```

```
                highest = marks;
            }
        }
    }
    return highest;
}


}
```

Lab 4 Knowledge Graph and Ontology



Lab 4 Marking Form

Attendance (Y/N)	Y
Task 1 completion (5 marks)	5
Task 2 completion (5 marks)	5
Total marks	10
Student signature	Qi Cao
TA signature	Ngia Han

Task 1

The results of Linked Open Data query using SPARQL are shown below.

SPARQL | HTML5 table

athlete	place
http://dbpedia.org/resource/Cristiano_Ronaldo	http://dbpedia.org/resource/Portugal_national_football_team
http://dbpedia.org/resource/Cristiano_Ronaldo	http://dbpedia.org/resource/Funchal
http://dbpedia.org/resource/Cristiano_Ronaldo	http://dbpedia.org/resource/Madeira

SPARQL | HTML5 table

athlete

http://dbpedia.org/resource/Category:Cristiano_Ronaldo

http://dbpedia.org/resource/Cristiano_Ronaldo

SPARQL | HTML5 table

athlete	place	placeName
http://dbpedia.org/resource/Cristiano_Ronaldo	http://dbpedia.org/resource/Funchal	"Funchal"@en
http://dbpedia.org/resource/Cristiano_Ronaldo	http://dbpedia.org/resource/Funchal	"Funchal"@nl
http://dbpedia.org/resource/Cristiano_Ronaldo	http://dbpedia.org/resource/Funchal	"فونشال"@ar
http://dbpedia.org/resource/Cristiano_Ronaldo	http://dbpedia.org/resource/Funchal	"Funchal"@ca
http://dbpedia.org/resource/Cristiano_Ronaldo	http://dbpedia.org/resource/Funchal	"Funchal"@cs
http://dbpedia.org/resource/Cristiano_Ronaldo	http://dbpedia.org/resource/Funchal	"Φουνσάλ"@el
http://dbpedia.org/resource/Cristiano_Ronaldo	http://dbpedia.org/resource/Funchal	"Funchal"@eo
http://dbpedia.org/resource/Cristiano_Ronaldo	http://dbpedia.org/resource/Funchal	"Funchal"@de
http://dbpedia.org/resource/Cristiano_Ronaldo	http://dbpedia.org/resource/Funchal	"Funchal"@eu
http://dbpedia.org/resource/Cristiano_Ronaldo	http://dbpedia.org/resource/Funchal	"Funchal"@es
http://dbpedia.org/resource/Cristiano_Ronaldo	http://dbpedia.org/resource/Funchal	"Funchal"@in
http://dbpedia.org/resource/Cristiano_Ronaldo	http://dbpedia.org/resource/Funchal	"Funchal"@it
http://dbpedia.org/resource/Cristiano_Ronaldo	http://dbpedia.org/resource/Funchal	"フンシャル"@ja
http://dbpedia.org/resource/Cristiano_Ronaldo	http://dbpedia.org/resource/Funchal	"Funchal"@fr
http://dbpedia.org/resource/Cristiano_Ronaldo	http://dbpedia.org/resource/Funchal	"푼살"@ko
http://dbpedia.org/resource/Cristiano_Ronaldo	http://dbpedia.org/resource/Funchal	"Funchal"@pl

SPARQL | HTML5 table

athlete	place	placeName
http://dbpedia.org/resource/Cristiano_Ronaldo	http://dbpedia.org/resource/Funchal	"Funchal"@en

SPARQL | HTML5 table

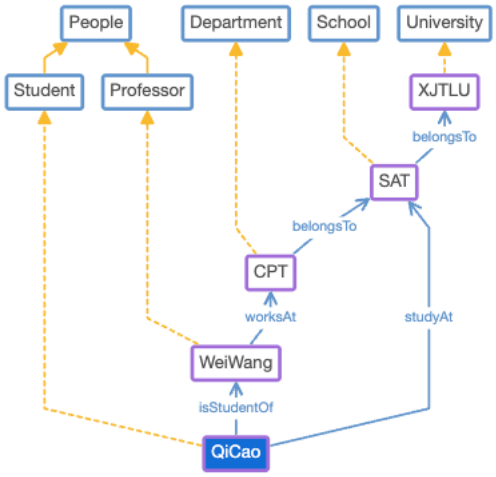
athlete	place	placeName	region
http://dbpedia.org/resource/Cristiano_Ronaldo	http://dbpedia.org/resource/Funchal	"Funchal"@en	"Madeira"@en

SPARQL | HTML5 table

athlete	place	region
http://dbpedia.org/resource/Cristiano_Ronaldo	http://dbpedia.org/resource/Funchal	"Madeira"@en

Task 2

The simple ontology for a university is shown below.



The source code is shown below.

```
<?xml version="1.0"?>
<rdf:RDF xmlns="urn:webprotege:ontology:7fe73c9f-943c-47b6-9be5-
c7f20820b456#"
    xml:base="urn:webprotege:ontology:7fe73c9f-943c-47b6-9be5-
c7f20820b456"
    xmlns:owl="http://www.w3.org/2002/07/owl#"
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:xml="http://www.w3.org/XML/1998/namespace"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
    xmlns:webprotege="http://webprotege.stanford.edu/">
   <owl:Ontology rdf:about="urn:webprotege:ontology:7fe73c9f-943c-
47b6-9be5-c7f20820b456"/>



    <!--

//////////////////////////////////////////////////////////////////////
//////////////////
    //
    // Object Properties
```

```
    //

////////////////////////////////////////////////////////////////////////
/////////////////
     -->




    <!-- http://webprotege.stanford.edu/R7L98OaeQeB87P5Ei5sYImO -->

    <owl:ObjectProperty
rdf:about="http://webprotege.stanford.edu/R7L98OaeQeB87P5Ei5sYImO">
        <rdfs:label>worksAt</rdfs:label>
    </owl:ObjectProperty>



    <!-- http://webprotege.stanford.edu/RSgMATzpr9SjSwBnmKpc5Q -->

    <owl:ObjectProperty
rdf:about="http://webprotege.stanford.edu/RSgMATzpr9SjSwBnmKpc5Q">
        <rdfs:label
rdf:datatype="http://www.w3.org/2001/XMLSchema#string">studyAt</rdfs:
label>
    </owl:ObjectProperty>



    <!-- http://webprotege.stanford.edu/RU8r8U9EEhqRe8ko6huVKo -->

    <owl:ObjectProperty
rdf:about="http://webprotege.stanford.edu/RU8r8U9EEhqRe8ko6huVKo">
        <rdfs:label>belongsTo</rdfs:label>
    </owl:ObjectProperty>



    <!-- http://webprotege.stanford.edu/RkIjE0hQSHnJiUceZ1E9z6 -->

    <owl:ObjectProperty
rdf:about="http://webprotege.stanford.edu/RkIjE0hQSHnJiUceZ1E9z6">
        <rdfs:label>isStudentOf</rdfs:label>
    </owl:ObjectProperty>
```

```
    <!--

///////////////////////////////////////////////////////////////////////
//////////////////
    //
    // Classes
    //

///////////////////////////////////////////////////////////////////////
//////////////////
     -->




    <!-- http://webprotege.stanford.edu/R8fPvsQqN3tFXseY7WmFFd -->

    <owl:Class
rdf:about="http://webprotege.stanford.edu/R8fPvsQqN3tFXseY7WmFFd">
        <rdfs:subClassOf
rdf:resource="http://webprotege.stanford.edu/RCU9PJF663WeCGxXNt79590"
/>
        <rdfs:label>Professor</rdfs:label>
    </owl:Class>



    <!-- http://webprotege.stanford.edu/RBXglLLfJBYdymieDv4ETWT -->

    <owl:Class
rdf:about="http://webprotege.stanford.edu/RBXglLLfJBYdymieDv4ETWT">
        <rdfs:label>University</rdfs:label>
    </owl:Class>



    <!-- http://webprotege.stanford.edu/RCU9PJF663WeCGxXNt79590 -->

    <owl:Class
rdf:about="http://webprotege.stanford.edu/RCU9PJF663WeCGxXNt79590">
        <rdfs:label>People</rdfs:label>
```

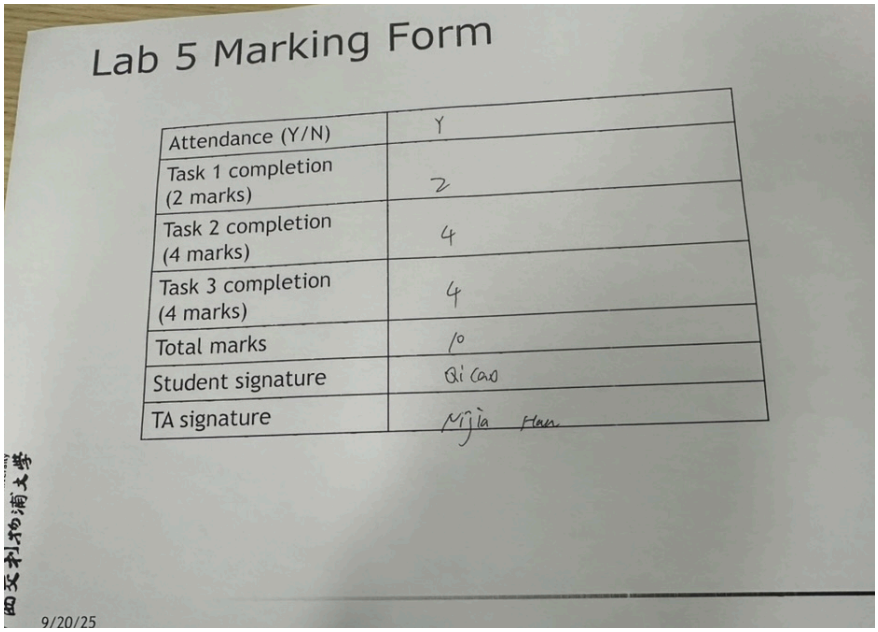
```
    </owl:Class>



    <!-- http://webprotege.stanford.edu/RIJzxMwL9Rf49lTXCvhBf2 -->

    <owl:Class
rdf:about="http://webprotege.stanford.edu/RIJzxMwL9Rf49lTXCvhBf2">
        <rdfs:subClassOf
rdf:resource="http://webprotege.stanford.edu/RCU9PJF663WeCGxXNt79590"
/>
        <rdfs:label>Student</rdfs:label>
    </owl:Class>



    <!-- http://webprotege.stanford.edu/RZ2RnQ62aG0gOFqo9uDPm1 -->

    <owl:Class
rdf:about="http://webprotege.stanford.edu/RZ2RnQ62aG0gOFqo9uDPm1">
        <rdfs:label>School</rdfs:label>
    </owl:Class>



    <!-- http://webprotege.stanford.edu/RbC1LwSjFc5sjBZ60Xl9gr -->

    <owl:Class
rdf:about="http://webprotege.stanford.edu/RbC1LwSjFc5sjBZ60Xl9gr">
        <rdfs:subClassOf
rdf:resource="http://www.w3.org/2002/07/owl#Thing"/>
        <rdfs:label>Department</rdfs:label>
    </owl:Class>



    <!--

////////////////////////////////////////////////////////////////////////
//////////////////
    //
    // Individuals
    //
```

```
//////////////////////////////////////////////////////////////////////
//////////////////
    -->




   <!-- http://webprotege.stanford.edu/R7NOqO4MBSrxz6qqFZYAG7L -->

   <owl:NamedIndividual
rdf:about="http://webprotege.stanford.edu/R7NOqO4MBSrxz6qqFZYAG7L">
      <rdf:type
rdf:resource="http://webprotege.stanford.edu/RIJzxMwL9Rf49lTXCvhBf2"/
>
      <webprotege:RSgMATzpr9SjSwBnmKpc5Q
rdf:resource="http://webprotege.stanford.edu/RBGDBaOofxhY5TDUJo07D12"
/>
      <webprotege:RkIjE0hQSHnJiUceZ1E9z6
rdf:resource="http://webprotege.stanford.edu/R9GZ2NFnKC0pZyEYu1Jr9q7"
/>
      <rdfs:label>QiCao</rdfs:label>
   </owl:NamedIndividual>



   <!-- http://webprotege.stanford.edu/R94oqFgnIme0JJFKpcJcwhj -->

   <owl:NamedIndividual
rdf:about="http://webprotege.stanford.edu/R94oqFgnIme0JJFKpcJcwhj">
      <rdfs:label>cq</rdfs:label>
   </owl:NamedIndividual>



   <!-- http://webprotege.stanford.edu/R9GZ2NFnKC0pZyEYu1Jr9q7 -->

   <owl:NamedIndividual
rdf:about="http://webprotege.stanford.edu/R9GZ2NFnKC0pZyEYu1Jr9q7">
      <rdf:type
rdf:resource="http://webprotege.stanford.edu/R8fPvsQqN3tFXseY7WmFFd"/
>
      <webprotege:R7L98OaeQeB87P5Ei5sYImO
rdf:resource="http://webprotege.stanford.edu/RCNCYFXV8FhwYtprWlCVqao"
/>
```

```
        <rdfs:label>WeiWang</rdfs:label>
    </owl:NamedIndividual>



    <!-- http://webprotege.stanford.edu/RBGDBaOofxhY5TDUJo07D12 -->

    <owl:NamedIndividual
rdf:about="http://webprotege.stanford.edu/RBGDBaOofxhY5TDUJo07D12">
        <rdf:type
rdf:resource="http://webprotege.stanford.edu/RZ2RnQ62aG0gOFqo9uDPm1"/
>
        <webprotege:RU8r8U9EEhqRe8ko6huVKo
rdf:resource="http://webprotege.stanford.edu/RDRJLOGOUmnXqzgn7cTZWWs"
/>
        <rdfs:label>SAT</rdfs:label>
    </owl:NamedIndividual>



    <!-- http://webprotege.stanford.edu/RCNCYFXV8FhwYtprWlCVqao -->

    <owl:NamedIndividual
rdf:about="http://webprotege.stanford.edu/RCNCYFXV8FhwYtprWlCVqao">
        <rdf:type
rdf:resource="http://webprotege.stanford.edu/RbC1LwSjFc5sjBZ60Xl9gr"/
>
        <webprotege:RU8r8U9EEhqRe8ko6huVKo
rdf:resource="http://webprotege.stanford.edu/RBGDBaOofxhY5TDUJo07D12"
/>
        <rdfs:label>CPT</rdfs:label>
    </owl:NamedIndividual>



    <!-- http://webprotege.stanford.edu/RDRJLOGOUmnXqzgn7cTZWWs -->

    <owl:NamedIndividual
rdf:about="http://webprotege.stanford.edu/RDRJLOGOUmnXqzgn7cTZWWs">
        <rdf:type
rdf:resource="http://webprotege.stanford.edu/RBXglLLfJBYdymieDv4ETWT"
/>
        <rdfs:label
rdf:datatype="http://www.w3.org/2001/XMLSchema#string">XJTLU</rdfs:la
```

```
bel>
    </owl:NamedIndividual>
</rdf:RDF>



<!-- Generated by the OWL API (version 4.5.13)
https://github.com/owlcs/owlapi -->
```

Lab 5 Java and MongoDB



Lab 5 Marking Form

Attendance (Y/N)	Y
Task 1 completion (2 marks)	2
Task 2 completion (4 marks)	4
Task 3 completion (4 marks)	4
Total marks	10
Student signature	Qi Cao
TA signature	Mijia Han

9/20/25

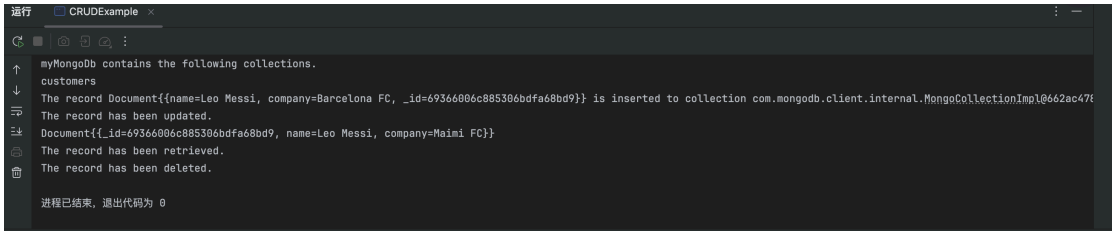
Task 1

The results of connecting to MongoDB by testing code are shown below.

```
Pinged your deployment. You successfully connected to MongoDB!
myMongoDb
sample_airbnb
admin
local

进程已结束，退出代码为 0
```

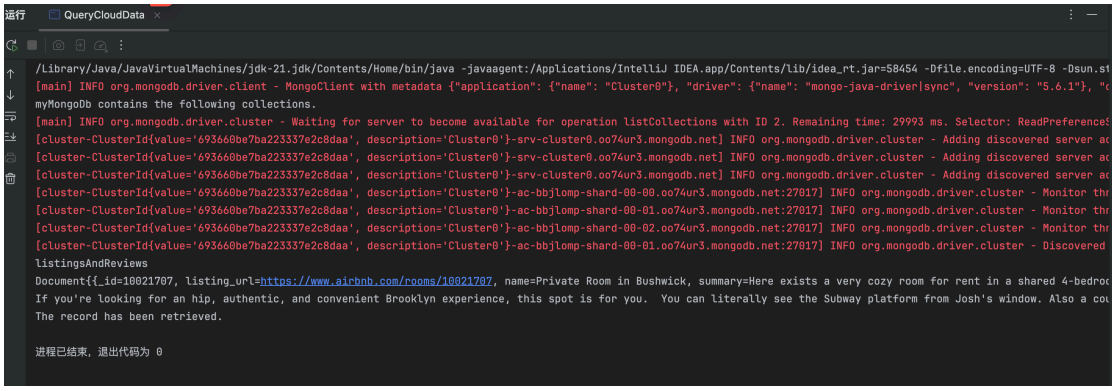
Task 2

The results of CRUD operations with MongoDB by testing code are shown below.



Task 3

The results of a new Java class called “QueryCloudData.java ” which retrieves data from a cloud dataset are shown below.



The codes of the new Java class called “QueryCloudData.java ” are shown below.

```
import com.mongodb.ConnectionString;
import com.mongodb.MongoClientSettings;
import com.mongodb.ServerApi;
import com.mongodb.ServerApiVersion;
import java.util.ArrayList;
import org.bson.Document;
import com.mongodb.client.FindIterable;
import com.mongodb.client.MongoClient;
import com.mongodb.client.MongoClients;
import com.mongodb.client.MongoCollection;
import com.mongodb.client.MongoCursor;
import com.mongodb.client.MongoDatabase;

public class QueryCloudData {
    public static void main(String[] args) {
        String connectionString =
```

```
"mongodb+srv://QiCao:20040915Cq@cluster0.oo74ur3.mongodb.net/?appName
=Cluster0";
        ServerApi serverApi = ServerApi.builder()
                .version(ServerApiVersion.V1)
                .build();
        MongoClientSettings settings = MongoClientSettings.builder()
                .applyConnectionString(new
ConnectionString(connectionString))
                .serverApi(serverApi)
                .build();
        try (MongoClient mongoClient = MongoClients.create(settings))
{

            MongoDatabase database =
mongoClient.getDatabase("sample_airbnb");


            // print all collections in customers database
            System.out.println("myMongoDb contains the following
collections.");

database.listCollectionNames().forEach(System.out::println);


            MongoCollection<Document> collection =
database.getCollection("listingsAndReviews");


            // query data
            Document searchQuery = new Document();
            searchQuery.put("name", "Private Room in Bushwick");
            FindIterable<Document> cursor =
collection.find(searchQuery);
            try (final MongoCursor<Document> cursorIterator =
cursor.cursor()) {
                while (cursorIterator.hasNext()) {
                    System.out.println(cursorIterator.next());
                }
            }
            System.out.println("The record has been retrieved.");

        }
    }
}
```