Lab 1 JDBC

使用JDBC连接数据库，需要用到mysql-connector-j-8.0.33.jar依赖包也就是mysql JDBC的API（中间件middleware）

Datebase Preparation

- Once logged in with your username and password, the system will automatically create a database having the same name as your user name.
- Create a table by run the following statement

```
CREATE TABLE Orf_Motif (orf varchar(12),
acc_num varchar(12), num int(3), pos int(3), len
int(3), mmatch varchar(12));
```

- Upload data into the table by importing the file '*yeast_prosite.sql*'. (这里是事先准备好的数据内容，直接引入创建好的新数据库)

Task 1:

- Run the following query:

```
SELECT * FROM Orf_Motif WHERE num=7
```

  - Record the time MySql needs to run this query
- Create an index on attribute *num*

```
CREATE INDEX numIndex ON Orf_Motif (num);
```

  - Run the same query:

```
SELECT * FROM Orf_Motif WHERE num=7
```

  - Record the time to run this query

Task 2;

- Open the *JDBCIndex.java* in any editor. Replace the following three parameters with your own ones.

```
static final String DB_URL = "jdbc:mysql://10.7.1.36/[DATABASE_NAME]";
static final String USER = "[USERNAME]";
static final String PASS = "[PASSWORD]";
```

- 使用命令行测试连接，并且会执行JDBC中的sql语句

  Windows: java -cp mysql-connector-java-8.0.18.jar; JDBCIndex

  MAC OS: java -cp mysql-connector-java-8.0.18.jar:. JDBCIndex

```
package Lab1;/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
```

```
 * and open the template in the editor.
 */

/**
 *
 * @author weiwang
 */
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;

public class JDBCIndex{

    static final String JDBC_DRIVER = "com.mysql.cj.jdbc.Driver";
    //replace the following three parameter values with your own ones
    static final String DB_URL = "jdbc:mysql://10.7.1.127/junhaohuang2202";
    static final String USER = "JunhaoHuang2202";
    static final String PASS = "123";

    public static void main(String[] args) {
        Connection conn;
        Statement stmt;
        ResultSet rs;
        String sql;
        String first;
        long begin;
        long end;

        try {
            Class.forName(JDBC_DRIVER);

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

            stmt = conn.createStatement();
//TEST QUERY WITHOUT AN INDEX
            //drop the existing index
            sql = "Drop INDEX numIndex ON Orf_Motif";
            stmt.execute(sql);
            System.out.println("Existing index on num dropped.");

            //prepare query without an index
            System.out.println("Start Query without an index.");
            sql = "SELECT * FROM Orf_Motif WHERE num=7";
            begin = System.currentTimeMillis();
            rs = stmt.executeQuery(sql);
            end = System.currentTimeMillis();
            System.out.println("-----------------");
            System.out.println("the 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()) {
                first = rs.getString("orf");
                System.out.println(rs.getString("acc_num") + "\t" + first);
            }
            System.out.println("End query without an index.");
            System.out.println("Query without an index takes " + (end - begin) +
"milliseconds");
//END TEST QUERY WITHOUT AN INDEX

//TEST QUERY WITH AN INDEX
            //prepare query with an index
            //build an index on num for Orf_Motif
            System.out.println("Start creating an index on num.");
            sql = "CREATE INDEX numIndex ON Orf_Motif (num)";
            stmt.execute(sql);
            //index created
            System.out.println("Index on num created.");
            //start query
            sql = "SELECT * FROM Orf_Motif WHERE num=7";
            begin = System.currentTimeMillis();
            rs = stmt.executeQuery(sql);
            end = System.currentTimeMillis();
            System.out.println("-----------------");
            System.out.println("the 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()) {
                //从列名为orf的列中读取数据
                first = rs.getString("orf");
                System.out.println(rs.getString("acc_num") + "\t" + first);
            }
            System.out.println("End query with an index.");
            System.out.println("Query with an index takes " + (end - begin) +
"milliseconds");
//END TEST QUERY WITH AN INDEX

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

        } catch (ClassNotFoundException e) {

            System.out.println("Cannot find JDBC Driver!");
            e.printStackTrace();

        } catch (SQLException e) {

            e.printStackTrace();

        } catch (Exception e) {
```

```
            e.printStackTrace();

        }
    }

}
```

Lab 2 JTA (Java Transaction API)

familiarise with the very basic programming for distributed transactions.

Two Phase Commit Protocal (2PC)

- The site at which the transaction originated is the *coordinator*. Other sites at which sub transactions are executed are *subordinates*.
- When a user decides to commit a transaction, the commit command is sent to the *coordinator* for the transaction.
- This initiates **2PC**
- **Phase 1**: Coordinator sends a **prepare** message to each subordinate; When subordinate receives a prepare message, it decides to abort or commit its sub transaction; Subordinate forces writes an **abort** or **ready** log record and sends a **no** or **yes** message to coordinator accordingly.
- **Phase 2**: If coordinator receives a **yes** message from all subordinates, force-writes a commit log record and sends **commit** message to all subordinates. Else force-writes an **abort** log record and sends an abort message; Subordinates forcewrite **abort** or **commit** log record based on the message they receive.
- In some implementations, after the two phases, subordinates send acknowledgement message to coordinator; After coordinator receives **ack** messages from all subordinates it writes an end log for the transaction.

Task Preparation:

- JDBC is needed.
- Create ‘distributed’ databases on MySQL
    - two relations/tables in your own database, or
    - two relations/tables in different databases
- Two relations/tables in your own database
    - *Not a true* distributed database
    - This can be done by yourself
- Two relations/tables in different databases
    - Distributed database
    - Need two people work in collaboration (share each others’ database relations)
- Basic relation/table design
    - A table with **two** columns such as user account (integer) and balance (double)
    - Add more columns if needed
    - Add some sample data for testing

- Modify the code
  - Fund transfer from one account at one database to another at the same or different database.
  - code should include your own database information

命令行：

Windows:

- javac -cp mysql-connector-j-8.0.31.jar; FundTransferJTA.java
- java -cp mysql-connector-j-8.0.31.jar; FundTransferJTA

MacOS:

- javac -cp mysql-connector-j-8.0.31.jar: FundTransferJTA.java
- java -cp mysql-connector-j-8.0.31.jar:. FundTransferJTA

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 *
 * CREATE TABLE t_account1(
    id INT(11),
    balance DOUBLE
)
 *
 */

/**
 *
 * @author weiwang
 */
import com.mysql.cj.jdbc.MysqlXADataSource;
import com.mysql.cj.jdbc.MysqlXid;
import java.sql.Connection;
import java.sql.SQLException;
import java.sql.Statement;

import javax.sql.XAConnection;
import javax.sql.XADataSource;
import javax.transaction.xa.XAException;
import javax.transaction.xa.XAResource;

public class FundTransferJTA {

    public static void main(String[] args) {

        // 连接数据库
        // 这里可以尝试连接两个不同的数据库
        XADataSource xads1 =
getMysqlXADS("jdbc:mysql://10.7.1.127/DATABASE_NAME", 3306, "JunhaoHuang2202",
                "123");
        XADataSource xads2 =
getMysqlXADS("jdbc:mysql://10.7.1.127/DATABASE_NAME", 3306, "JunhaoHuang2202",
```

```
                "123");

        // 创建两个XAConnection的对象，分别来管理和两个数据库的连接
        // 都设置成null代表手动来设置
        /*  XAConnection 是 XA 版本的连接。
            Connection 是标准的 JDBC 连接。
            Statement 用于执行 SQL 语句。
            XAResource 是 XA 资源，用于管理事务。
            MysqlXid 是 MySQL 的全局事务标识符。*/
        XAConnection xaconn1 = null, xaconn2 = null;
        Connection conn1 = null, conn2 = null;
        Statement stmt1 = null, stmt2 = null;
        XAResource xares1 = null, xares2 = null;

        MysqlXid xid1 = new MysqlXid(new byte[] { 0x01 }, new byte[] { 0x57 },
0);
        MysqlXid xid2 = new MysqlXid(new byte[] { 0x01 }, new byte[] { 0x58 },
0);

        try {
            xaconn1 = xads1.getXAConnection();
            conn1 = xaconn1.getConnection();
            stmt1 = conn1.createStatement();
            xares1 = xaconn1.getXAResource();

            xaconn2 = xads2.getXAConnection();
            conn2 = xaconn2.getConnection();
            stmt2 = conn2.createStatement();
            xares2 = xaconn2.getXAResource();

            // 禁止自动提交
            conn1.setAutoCommit(false);
            conn2.setAutoCommit(false);

            // 开始事务并执行更新操作
            xares1.start(xid1, XAResource.TMNOFLAGS);
            // 注意这里需要更信息
            // update [test database name 1] set [atribute name] = [attribute
name] + 100 where [key name]=1
            stmt1.execute("update t_account1 set balance = balance + 100 where id
= 1");
            xares1.end(xid1, XAResource.TMSUCCESS);

            xares2.start(xid2, XAResource.TMNOFLAGS);
            // update [test database name 2] set [atribute name] = [attribute
name] + 100 where [key name]=1
            stmt2.execute("update t_account2 set balance = balance - 100 where id
= 1");
            xares2.end(xid2, XAResource.TMSUCCESS);

            // 准备和提交事务
            int ret1 = xares1.prepare(xid1);
            int ret2 = xares2.prepare(xid2);

            if (ret1 == XAResource.XA_OK && ret2 == XAResource.XA_OK) {
                xares1.commit(xid1, false);
```

```
                xares2.commit(xid2, false);
                System.out.println("OK!");
            } else {
                xares1.rollback(xid1);
                xares2.rollback(xid2);
            }
        } catch (SQLException e) {
            try {
                xares1.rollback(xid1);
                xares2.rollback(xid2);
            } catch (XAException xae) {
                xae.printStackTrace();
            }
        } catch (XAException e) {
            e.printStackTrace();
        } finally {
            try {
                stmt1.close();
                stmt2.close();
                conn1.close();
                conn2.close();
                xaconn1.close();
                xaconn2.close();
            } catch (SQLException e) {
                e.printStackTrace();
            }
        }
    }

    public static XADataSource getMysqlXADS(String url, int port, String user,
String pass) {
        MysqlXADataSource xads = new MysqlXADataSource();
        xads.setUrl(url);
        xads.setPort(port);
        xads.setUser(user);
        xads.setPassword(pass);

        return xads;
    }
}
```

Lab 3 XML

Task 1: XML data processing with SAX:

SAX (Simple API for XML)

- Based on a parser model, user provides event handlers for parsing events
    - e.g. start of element, end of element

SAX Introduction:

- **SAX** is an event-based parser for XML document processing.

- Applications using SAX receive event notifications about the XML document being processed (e.g. an element or attribute), at a time, in sequential order, starting at the top of the document, and ending with the closing of the ROOT element.

XML Processing with SAX

- Reads an XML document from top to bottom.
- Tokens are processed in the same order that they appear in the document.
- Reports to the application program the nature of tokens that the parser has encountered as they occur.
- The application program provides an "event" handler that must be registered with the parser.
- As the tokens are identified, callback methods in the handler are invoked.

When to use SAX:

- Process the XML document in a linear fashion top-down.
- The document is not deeply nested.
- Process a very large XML document whose DOM tree would consume too much memory.
- The problem to be solved involves only a part of the XML document.
- Data is available as soon as it is seen by the parser, so SAX works well for an XML document that arrives over a stream.

Limitation of SAX

- No random access to an XML document since it is processed in a forward-only manner.
- If one needs to keep track of data that the parser has seen or to change the order of items, s/he must write the code and store the data on her/his own.

Codes:

- Important Classes and Interfaces in SAX

```
javax.xml.parsers.SAXParser;
javax.xml.parsers.SAXParserFactory;
org.xml.sax.Attributes;
org.xml.sax.SAXException;
org.xml.sax.helpers.DefaultHandler;
...
```

Remove the package declaration if package is not used in your code

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package xml;

/**
 * Example adapted from tutorialspoint
 * https://www.tutorialspoint.com/java_xml/java_sax_parse_document.htm
 * @author weiwang
 */
```

```java
import java.io.File;
import javax.xml.parsers.SAXParser;
import javax.xml.parsers.SAXParserFactory;

import org.xml.sax.Attributes;
import org.xml.sax.SAXException;
import org.xml.sax.helpers.DefaultHandler;

public class SAXParserDemo {

   public static void main(String[] args) {

      try {
          // 注意这里的路径是项目路径
         File inputFile = new File("./xml/XMLDoc.xml");
         SAXParserFactory factory = SAXParserFactory.newInstance();
         SAXParser saxParser = factory.newSAXParser();
         UserHandler userhandler = new UserHandler();
         saxParser.parse(inputFile, userhandler);
      } catch (Exception e) {
         e.printStackTrace();
      }
   }
}

class UserHandler extends DefaultHandler {

   boolean bFirstName = false;
   boolean bLastName = false;
   boolean bNickName = false;
   boolean bMarks = false;

   @Override
   public void startElement(
      String uri, String localName, String qName, Attributes attributes)
      throws SAXException {

      if (qName.equalsIgnoreCase("student")) {
         String rollNo = attributes.getValue("rollno");
         System.out.println("Roll No : " + rollNo);
      } else if (qName.equalsIgnoreCase("firstname")) {
         bFirstName = true;
      } else if (qName.equalsIgnoreCase("lastname")) {
         bLastName = true;
      } else if (qName.equalsIgnoreCase("nickname")) {
         bNickName = true;
      }
      else if (qName.equalsIgnoreCase("marks")) {
         bMarks = true;
      }
   }

   @Override
   public void endElement(String uri,
      String localName, String qName) throws SAXException {
```

```
      if (qName.equalsIgnoreCase("student")) {
         System.out.println("End Element :" + qName);
      }
   }

   // 根据布尔变量的状态打印相应的内容，然后将布尔变量重置为 false。
   @Override
   public void characters(char ch[], int start, int length) throws SAXException {

      if (bFirstName) {
         System.out.println("First Name: " + new String(ch, start, length));
         bFirstName = false;
      } else if (bLastName) {
         System.out.println("Last Name: " + new String(ch, start, length));
         bLastName = false;
      } else if (bNickName) {
         System.out.println("Nick Name: " + new String(ch, start, length));
         bNickName = false;
      } else if (bMarks) {
         System.out.println("Marks: " + new String(ch, start, length));
         bMarks = false;
      }
   }
}
```

Task 2: XML data processing with DOM

- **DOM** (Document Object Model)
  - XML data is parsed into a tree representation
  - Variety of functions provided for traversing the DOM tree
  - e.g. Java DOM API provides Node class with methods

    getParentNode( ), getFirstChild( ), getNextSibling( )

    getAttribute( ), getData( ) (for text node)

    getElementsByTagName( ), ...
  - Also provides functions for updating DOM tree

DOM Introduction:

- Document Object Model (DOM) is an official recommendation of the World Wide Web Consortium (W3C).
- It defines an interface that enables programs to access and update the style, structure, and contents of XML documents.
- When one parses an XML document with a DOM parser, s/he gets a tree structure containing all elements of the document. DOM provides a variety of functions to examine the contents and structure of a document.

When to use DOM:

- Need to know a lot about the structure of a document.
- Need to move parts of an XML document around (e.g. might want to sort certain elements).

- Need to use the information in an XML document more than once.

Important Classes and Interfaces in SAX:

```
javax.xml.parsers.DocumentBuilderFactory;
javax.xml.parsers.DocumentBuilder;
org.w3c.dom.Document;
org.w3c.dom.NodeList;
org.w3c.dom.Node;
org.w3c.dom.Element;
…
```

Common Methods in DOM:

- Document.getDocumentElement() − Returns the root element of the document.
- Node.getFirstChild() − Returns the first child of a given Node.
- Node.getLastChild() − Returns the last child of a given Node.
- Node.getNextSibling() − Returns the next sibling of a given Node.
- Node.getPreviousSibling() − Returns the previous sibling of a given Node.
- Node.getAttribute(attrName) − For a given Node, it returns the attribute with the requested name.
- *For more refer to DOM Java API documentation*

Remove the package declaration if package is not used in your code

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package xml;

/**
 * Example adapted from tutorialspoint
 * https://www.tutorialspoint.com/java_xml/java_dom_parse_document.htm
 * @author weiwang
 */


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 DOMParserDemo {

   public static void main(String[] args) {

      try {
         File inputFile = new File("src/xml/XMLDoc.xml");
```

```
         DocumentBuilderFactory dbFactory = DocumentBuilderFactory.newInstance();
         DocumentBuilder dBuilder = dbFactory.newDocumentBuilder();
         Document doc = dBuilder.parse(inputFile);
         doc.getDocumentElement().normalize();
         System.out.println("Root element :" +
doc.getDocumentElement().getNodeName());
         NodeList nList = doc.getElementsByTagName("student");
         System.out.println("----------------------------");

         for (int temp = 0; temp < nList.getLength(); temp++) {
            Node nNode = nList.item(temp);
            System.out.println("\nCurrent Element:" + nNode.getNodeName());

            if (nNode.getNodeType() == Node.ELEMENT_NODE) {
               Element eElement = (Element) nNode;
               System.out.println("Student roll no: "
                  + eElement.getAttribute("rollno"));
               System.out.println("First Name: "
                  + eElement
                  .getElementsByTagName("firstname")
                  .item(0)
                  .getTextContent());
               System.out.println("Last Name: "
                  + eElement
                  .getElementsByTagName("lastname")
                  .item(0)
                  .getTextContent());
               System.out.println("Nick Name: "
                  + eElement
                  .getElementsByTagName("nickname")
                  .item(0)
                  .getTextContent());
               System.out.println("Marks: "
                  + eElement
                  .getElementsByTagName("marks")
                  .item(0)
                  .getTextContent());
            }
         }
      } catch (Exception e) {
         e.printStackTrace();
      }
   }
}
```

XML Data:

```
<?xml version="1.0" encoding="UTF-8"?>
<!--
To change this license header, choose License Headers in Project Properties.
To change this template file, choose Tools | Templates
and open the template in the editor.
-->
```

```
<class>
   <student rollno = "393">
      <firstname>dinkar</firstname>
      <lastname>kad</lastname>
      <nickname>dinkar</nickname>
      <marks>85</marks>
   </student>

   <student rollno = "493">
      <firstname>Vaneet</firstname>
      <lastname>Gupta</lastname>
      <nickname>vinni</nickname>
      <marks>95</marks>
   </student>

   <student rollno = "593">
      <firstname>jasvir</firstname>
      <lastname>singn</lastname>
      <nickname>jazz</nickname>
      <marks>90</marks>
   </student>
</class>
```

Lab 4 Knowledge Graph and Ontology

RDF knowledge; RDF queries using **SPARQL** language; and **Linked open data**

DBPedia:

- **DBpedia** is a community project that creates and provides public access to critical structured data for what's commonly referred to as the **Linked Open Data Cloud**.
- DBpedia provides a globally accessible **Knowledge Graph** derived from **Wikipedia** content. You can query this tremendous Knowledge Graph using the powerful **SPARQL** query language .
- Dbpedia can be accessed through its **SPARQL endpoint**.

RDF Data Query:

- Knowledge takes the form of a collection of **RDF** Triples, which structures data using a (**subject-predicate-object**) object model

  知识的形式是 **RDF** 三元组的集合,它使用 (**主体-谓词-客体**) 对象模型构建数据

- The most basic query example can take the form of a **SELECT** Query where the triple pattern in the Query Body comprises a subject, predicate, and object.

  最基本的查询示例可以采用 **SELECT** 查询的形式,其中查询体中的三重模式包括主语、谓语和宾语。

  - Query Solution projection size is limited to 10 records presented in an HTML Table in this lab.

    查询解决方案的投影大小限制为本实验室中 HTML 表格中呈现的 10 条记录。

- DBpedia can be easily queried both with and without a deep knowledge of the DBpedia ontology

  无论是否对 DBpedia 本体有深入了解,都可以轻松查询 DBpedia

Query Example:

SELECT *

WHERE

{

?s ?p ?o

}LIMIT 10

1. Search for “an athlete with a literal value of Cristiano Ronaldo“
    - **dereferencing** 的意思就是你go through the URI listed in the column of the key word you search in the query(in this example is Cristiano Raonaldo in the column called 'athlete'), and identify the entity literally labeled as the key word you searched.

```
SELECT *
WHERE
{
?athlete rdfs:label "Cristiano Ronaldo"@en
}
```

2. Add in the *dbo:birthPlace* property; And execute the following query
    - We can narrow the place results to only include cities by scoping the query body to instances of the *dbo:City* class.

```
SELECT *
WHERE
{
?athlete rdfs:label "Cristiano Ronaldo"@en ;
    dbo:birthPlace ?place .
}
```

3. Execute the following query.

```
SELECT *
WHERE
{
?athlete rdfs:label "Cristiano Ronaldo"@en ;
    dbo:birthPlace ?place .
?place a dbo:City ;
    rdfs:label ?cityName .
}
```

4. We can also limit the language tag to English via a FILTER on the *?cityName* variable.

```
SELECT *
WHERE
{
?athlete rdfs:label "Cristiano Ronaldo"@en ;
    dbo:birthPlace ?place .
?place a dbo:City ;
    rdfs:label ?cityName .
FILTER ( LANG ( ?cityName ) = 'en' )
}
```

5. Let's double-confirm that Funchal is in Madeira, by using the *dbp:region* property and its value.

```
SELECT *
WHERE
{
?athlete rdfs:label "Cristiano Ronaldo"@en ;
    dbo:birthPlace ?place .
?place a dbo:City ;
    rdfs:label ?cityName ;
    dbp:region ?region .
FILTER ( LANG ( ?cityName ) = 'en' )
}
```

6. We can also replace ***** in the SELECT List with a **specific list of variables** to be projected in the Query Solution.

```
SELECT
    ?athlete
    ?place
    ?region
WHERE
{
?athlete rdfs:label "Cristiano Ronaldo"@en ;
    dbo:birthPlace ?place .
?place a dbo:City ;
    rdfs:label ?cityName ;
    dbp:region ?region .
FILTER ( LANG ( ?cityName ) = 'en' )
}
```

7. Run the follwing query

```
SELECT *
WHERE
{
?athlete rdfs:label "Cristiano Ronaldo"@en ;
    dbo:birthPlace ?place .
?place a yago:PhysicalEntity100001930 ;
    rdfs:label ?cityName .
FILTER ( LANG ( ?cityName ) = 'en' )
}
```

Lab 5 Java and MongoDB

use Java to connect to MongoDB for **CRUD** (Create, Retrieve, Update and Delete) operations.

Task 1 Connect to MongoDB:

MongoClientConnectionExample.java

```
package xjtlu.cpt201.mongoDB;

/**
 *
 * code from https://cloud.mongodb.com/v2/
 * and revised by Wei Wang, CPT, SAT, XJTLU
 * functionalities: 1. ping the default cloud database
 * 2. display all databases (sample databases loaded from the cloud)
 */
import com.mongodb.ConnectionString;
import com.mongodb.MongoClientSettings;
import com.mongodb.MongoException;
import com.mongodb.ServerApi;
import com.mongodb.ServerApiVersion;
import com.mongodb.client.MongoClient;
import com.mongodb.client.MongoClients;
import com.mongodb.client.MongoDatabase;
import org.bson.Document;
public class MongoClientConnectionExample {
    public static void main(String[] args) {
        String connectionString =
"mongodb+srv://Junhao_Huang:A482645hjh@cluster0.xbb5i.mongodb.net/?
retryWrites=true&w=majority&appName=Cluster0";
        ServerApi serverApi = ServerApi.builder()
                .version(ServerApiVersion.V1)
                .build();
        MongoClientSettings settings = MongoClientSettings.builder()
                .applyConnectionString(new ConnectionString(connectionString))
                .serverApi(serverApi)
                .build();
        // Create a new client and connect to the server
        try (MongoClient mongoClient = MongoClients.create(settings)) {
            try {
                // Send a ping to confirm a successful connection
                MongoDatabase database = mongoClient.getDatabase("admin");
                database.runCommand(new Document("ping", 1));
                System.out.println("Pinged your deployment. You successfully
connected to MongoDB!");

                //retrieve all databases and print their names
                mongoClient.listDatabaseNames().forEach(System.out::println);
            } catch (MongoException e) {
                e.printStackTrace();
            }
        }
    }
}
```

Task 2 CRUD operations with MongoDB

CRUDExample.java

```
/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to
change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this
template
 */
package xjtlu.cpt201.mongoDB;

/**
 *
 * Code taken from https://www.baeldung.com/java-mongodb
 * and revised by Wei Wang, CPT, SAT, XJTLU
 */
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 CRUDExample {

    public static void main(String[] args) {
        String connectionString =
"mongodb+srv://Junhao_Huang:A482645hjh@cluster0.xbb5i.mongodb.net/?
retryWrites=true&w=majority&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("myMongoDb");

            boolean collectionExists =
mongoClient.getDatabase("myMongoDb").listCollectionNames()
                    .into(new ArrayList<>()).contains("customers");
            if (!collectionExists) {
                database.createCollection("customers");
            }
```

```
            // print all collections in myMongoDb database
            System.out.println("myMongoDb contains the following collections.");
            database.listCollectionNames().forEach(System.out::println);

            // create data
            MongoCollection<Document> collection =
database.getCollection("customers");
            Document document = new Document();
            document.put("name", "Leo Messi");
            document.put("company", "Barcelona FC");
            collection.insertOne(document);
            System.out.println("The record " + document.toString() + " is
inserted to collection " + collection.toString());

            // update data
            Document query = new Document();
            query.put("name", "Leo Messi");
            Document newDocument = new Document();
            newDocument.put("company", "Maimi FC");
            Document updateObject = new Document();
            updateObject.put("$set", newDocument);
            collection.updateOne(query, updateObject);
            System.out.println("The record has been updated.");

            // read data
            Document searchQuery = new Document();
            searchQuery.put("name", "Leo Messi");
            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.");

            // delete data
            Document deleteQuery = new Document();
            deleteQuery.put("name", "Leo Messi");
            collection.deleteOne(deleteQuery);
            System.out.println("The record has been deleted.");
        }
    }
}
```

Task 3 Task 3: Retrieve data from sample datasets

QueryCloudData.java

```
/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to
change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this
template
```

```
 */
package xjtlu.cpt201.mongoDB;

/**
 *
 * @author weiwang
 * functionalities: 1. connect to the cloud database "sample_airbnb"
 * 2. retrieve the collection "listingsAndReviews"
 * 3. retrieve and display the record with name "Private Room in Bushwick"
 */
import com.mongodb.ConnectionString;
import com.mongodb.MongoClientSettings;
import com.mongodb.MongoException;
import com.mongodb.ServerApi;
import com.mongodb.ServerApiVersion;
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;
import org.bson.Document;
import java.util.ArrayList;

public class QueryCloudData {

    public static void main(String[] args) {
        String connectionString =
"mongodb+srv://Junhao_Huang:A482645hjh@cluster0.xbb5i.mongodb.net/?
retryWrites=true&w=majority&appName=Cluster0";
        ServerApi serverApi = ServerApi.builder()
                .version(ServerApiVersion.V1)
                .build();
        MongoClientSettings settings = MongoClientSettings.builder()
                .applyConnectionString(new ConnectionString(connectionString))
                .serverApi(serverApi)
                .build();
        // Create a new client and connect to the server
        try ( MongoClient mongoClient = MongoClients.create(settings)) {
            try {
                // COMPLETE THE TASK BY WRITING YOUR OWN CODE HERE.
                MongoDatabase database =
mongoClient.getDatabase("sample_airbnb");

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

                Document searchQuery = new Document();
                searchQuery.put("name", "Private Room in Bushwick");
                FindIterable<Document> cursor = collection.find(searchQuery);
                System.out.println("Records with name 'Private Room in
Bushwick':");
                try (final MongoCursor<Document> cursorIterator =
cursor.cursor()) {
                    while (cursorIterator.hasNext()) {
                        System.out.println(cursorIterator.next().toJson());
```

```
                    }
                }
                System.out.println("The record has been retrieved.");


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

Sample Question and Answer

Briefly introduce the steps needed in rocessing XML document using DOM and Java:

1. Import DOM parser Packages
2. Create DocumentBuilder
3. Create Document object from XML file
4. Use various methods to address element
5. Extract XML element data (e.g. attribute node)