package com.yemreak.mongodbjava;

import com.mongodb.client.\*;

import com.mongodb.client.model.Filters;

import com.mongodb.client.model.Updates;

import com.mongodb.client.result.DeleteResult;

import com.mongodb.client.result.UpdateResult;

import org.bson.Document;

import java.util.Iterator;

import java.util.Scanner;

/\*\*

\* Odev5 - Veritabanı Yönetim Sistemleri

\* @author Yunus Emre Ak - 13006150001

\*/

public class Main {

// Database bilgileri

private static final String USERNAME = "admin";

private static final String PASSWORD = "admin123";

private static final String CLUSTER\_NAME = "iu-ce-azmzc";

private static final String DATABASE\_NAME = "odev5";

private static final String COLLECTION\_NAME = "Teacher";

// Kullanıcı işlemleri ve arayüz değişkenleri

private static final Scanner scan = new Scanner(System.in);

private static MongoCollection<Document> collection;

public static void main(String[] args) {

connectDB();

if (checkDBConnection()) {

userInterface();

} else {

System.out.println("Cannot connect DB!");

}

}

/\*\*

\* MongoDB veritanabına bağlanma

\*/

private static void connectDB() {

// Bağlantı URI'si oluşturma

String mongoURI = "mongodb+srv://" + USERNAME + ":" + PASSWORD + "@" + CLUSTER\_NAME + ".mongodb.net/" + DATABASE\_NAME + "?retryWrites=true";

// URI üzerinden mongo istemcisini başlatma

// MongoDB değişkenleri

MongoClient mongoClient = MongoClients.create(mongoURI);

// odev4 veritabanına bağlanma

MongoDatabase database = mongoClient.getDatabase(DATABASE\_NAME);

// Koleksiyonu (tabloyu) veritabanından almak

collection = database.getCollection(COLLECTION\_NAME);

}

/\*\*

\* Veri tabanı bağlantısın kontrol etme

\* @return Hata yoksa true

\*/

private static boolean checkDBConnection() {

return collection != null;

}

/\*\*

\* Ana kullanıcı arayüzü

\*/

private static void userInterface() {

// Kullanıcı cevabı

int answer;

while (true) {

newRow();

System.out.println("Main Menu");

newRow();

System.out.println("1- List");

System.out.println("2- Add");

System.out.println("3- Delete");

System.out.println("4- Update");

System.out.println("0- Exit");

answer = getNumberUI(null);

switch (answer) {

case 0:

System.exit(1);

case 1:

listUI();

break;

case 2:

addUI();

break;

case 3:

deleteUI();

break;

case 4:

updateUI();

break;

default:

System.out.println("Hatalı girdi girdiniz!");

}

}

}

/\*\*

\* Veritabanı verilerini listeme arayüzü

\*/

private static void listUI() {

// Koleksiyondaki verilleri tutacak bir yineleyici objesini alma

FindIterable<Document> iterDoc = collection.find();

// Yineleyici alma

Iterator iterator = iterDoc.iterator();

if (!iterator.hasNext()) {

System.out.println("No data!");

return;

}

// Yineyici devam ettiği sürece devam etme

while(iterator.hasNext()) {

System.out.println(iterator.next());

}

}

/\*\*

\* Veritabanına veri ekleme arayüzü

\*/

private static void addUI() {

// Kullanıcı değişkenleri

int tid = getNumberUI("Id");

String name = getStringUI("Name");

String placeOfBirth = getStringUI("Birthplace");

// Verilen kaydı veritabanına ekleme

collection.insertOne(new Document("tid", tid)

.append("name", name)

.append("placeOfBirth", placeOfBirth)

);

}

/\*\*

\* Veritabından kayıt silme arayüzü

\*/

private static void deleteUI() {

// Kullanıcı değişkenleri

int tid = getNumberUI("Id");

// Id'si verilen kaydı silme ve sonucu yazdırma

DeleteResult result = collection.deleteOne(Filters.eq("tid", tid));

if (result.getDeletedCount() == 0) {

System.out.println("The id of data is not valid");

System.out.println("Delete failed!");

} else {

System.out.println("Deletion successfully!");

}

}

/\*\*

\* Veritabanında veri güncelleme arayüzü

\* collection.updateOne(Filters.eq("id", 1), Updates.set("likes", 150));

\*/

private static void updateUI() {

// Kullanıcı değişkenleri

int tid = getNumberUI("Id");

String name = getStringUI("Name");

String placeOfBirth = getStringUI("Birthplace");

// Veri tabanı kaydını güncelleme

UpdateResult result = collection.updateOne(Filters.eq("tid", tid), Updates.combine(

Updates.set("name", name),

Updates.set("placeOfBirth", placeOfBirth)

));

if (result.getMatchedCount() == 0) {

System.out.println("The id of data is not valid");

System.out.println("Update failed!");

} else if (result.getModifiedCount() == 0){

System.out.println("May some of value be wrong or not different");

System.out.println("No update was done!");

} else {

System.out.println("Update successfully!");

}

}

/\*\*

\* Kullanıcıdan sayı verisi isteme arayüzü

\* @param text Ekrana yazılacak metin

\* @return Kullanıcının girdiği sayı

\*/

private static int getNumberUI(String text) {

newRow();

if (text != null)

System.out.println(text);

System.out.print("-> ");

try {

return Integer.parseInt(scan.nextLine());

} catch (NumberFormatException e) {

System.out.println("Input type error! (Must be number)");

return getNumberUI(text);

}

}

/\*\*

\* Kullanıcıdan dizgi verisi isteme arayüzü

\* @param text Ekrana yazılacak metin

\* @return Kullanıcının girdiği dizgi

\*/

private static String getStringUI(String text) {

newRow();

if (text != null)

System.out.println(text);

System.out.print("-> ");

return scan.nextLine();

}

/\*\*

\* Çizgilerden oluştan bir satır ekrana çizer

\*/

private static void newRow() {

System.out.println("-----------------------------");

}

}