TP11

Java DataBase Connection (JDBC)

Remarks:

- ✓ All validation errors are in form of Exceptions

TP11.1. Database schema and classes

Create tables:

1. **countries (id, country);**
   id integer, auto increment, and is a primary key. (Ex. 1)
   country character varying max length 50 and not NULL or EMPTY. (Ex. Cambodia)
2. **cities (id, city, countryid, ucity);**
   id integer, auto increment, and is a primary key. (Ex. 1)
   city character varying max length 50 and not NULL or EMPTY. (Ex. Phnom Penh)
   countryid integer, references to table *countries* field *id*. (Ex. 1 (Cambodia))
   ucity character varying max length 60 and unique.
   (Unique city id = *<city>* _*<countryid>.* Ex. Phnom Penh_1)
3. **hotels (id, hotel, countryid, cityid, stars, cost, info);**
   id integer, auto increment, and is a primary key. (Ex. 1)
   hotel character varying max length 100 and not NULL or EMPTY. (Ex. Raffles Hotel Le Royal)
   countryid integer, nullable, and references to table *countries* field *id*. (Ex. 1)
   cityid integer, nullable, and references to table *cities* field *id*. (Ex. 1)
   stars tiny integer, minimum 0, maximum 5. (Ex. 5)
   cost double, maximum digits 12 and max 2 numbers after period. (Ex. 270.99)
   info text, nullable or empty. (Ex. *Raffles Hotel Le Royal invites you to enjoy an unforgettable stay at its iconic luxury hotel just off Phnom Penh's main boulevard near foreign embassies.*)
4. **images (id, hotelid, imagepath);**
   id integer, auto increment, and is a primary key. (Ex. 1)
   hotelid integer, and references to table *hotels* field *id*. (Ex. 1)
   imagepath character varying max length 256, not NULL or EMPTY. (Ex. images/raffles.png)
5. **roles (id, role);**
   id integer, auto increment, and is a primary key. (Ex. 1)
   role character varying max length 20, unique, and not NULL or EMPTY. (Ex. Admin)
   (should have 2 records: 1 Admin, 2 Customer)
6. **users (id, username, pass, email, roleid, discount, avatar)**
   id integer, auto increment, and is a primary key. (Ex. 1)
   username character varying max length 30, minimum length 3, unique and not NULL. (Ex. bopha)
   pass character varying max length 80, minimum length 3. (Ex. ********)
   email character varying, nullable. (Ex. bopha@itc.edu.kh)
   roleid integer, not NULL, and references to table *roles* field *id*. (Ex. 2)
   discount tiny integer, minimum 0, maximum 100. (Ex. 50 = 50%)
   avatar character varying max length 256, and nullable. (Ex. avatar/bopha.png)

TP11.2. Model classes

Models are classes represents tables in Relational Database. For example, we create class **Country** with attributes: int id, and String country and other methods related to data validation and is mapped to table **countries** in Database.

```
public class Country {
    private int id;
    private String country;
    public Country(int id, String country) {
        this.id = id;
        this.country = country;
    }
    public int getId() { return id; }
    public void setId(int id) { this.id = id; }
    public String getCountry() { return country; }
    public void setCountry(String country) { this.country = country; }
}
```

Example 2: table **cities** have fields id, city and countryid. We should create a class called **City** containing fields:

- id as int
- city as String and
- country as object of class Country.
- ucity as String

```
public class City {
    private int id;
    private String city;
    private Country country;
    private String ucity;
    public City(int id, String city, Country country, String ucity) {
        this.id = id;
        this.city = city;
        this.country = country;
        this.ucity = ucity;
    }
    public int getId() { return id; }
    public void setId(int id) { this.id = id; }
    public String getCity() { return city; }
    public void setCity(String city) { this.city = city; }
    public Country getCountry() { return country; }
    public void setCountry(Country country) { this.country = country; }
    public String getUcity() { return ucity; }
    public void setUcity(String ucity) { this.ucity = ucity; }
}
```

Create all model classes for mapping to all tables in Database.

TP11.3. ORM classes

Object-relational Mapping (ORM) is programming technique for converting data between relational database datatypes and Object-oriented datatypes. ORM class should contain the following features:

1. List all rows as objects from a table (method body will be empty in this class)
2. Add new row to a table
3. Remove a row by id
4. Update a row by id
5. List rows by raw query

```
import java.sql.*;
import java.util.ArrayList;
public class ORM<T> {
    protected Connection connection; protected String tableName=null;
    public ORM(){
        try {
            connection = DriverManager
               .getConnection("jdbc:mysql://localhost:3306/i4?user=root&password=");
        } catch (SQLException e) { .printStackTrace(); }
    }
    public ArrayList<T> listAll(){return new ArrayList<>();}
    public T add(T t){return t;}
    public boolean delete(int id){return false;}
    public void update(T t){}
    public ArrayList<T> rawQueryList(String query){ return new ArrayList<>(); }
}
```

Create all sub-classes related to all tables including: CountryORM, CityORM, …

Example:

```
import java.sql.SQLException;
import java.sql.Statement;

public class CountryORM extends ORM<Country> {
    public CountryORM(){
        super();
        tableName = "countries";
    }
    @Override
    public Country add(Country t) {
        try(var stmt = connection.createStatement()){
            var sql = "INSERT INTO "+tableName
                +" VALUES(NULL,'"+t.getCountry()+"')";
            stmt.executeUpdate(sql, Statement.RETURN_GENERATED_KEYS);
            var rs = stmt.getGeneratedKeys();
            rs.next();
            t.setId(rs.getInt(1));
            return t;
```

```
        }catch(SQLException e){
            e.printStackTrace();
        }
        return null;
    }
    public static void main(String[] args) {
        CountryORM orm = new CountryORM();
        Country c = new Country(0, "France");
        orm.add(c);
        System.out.println("Id: "+c.getId()+"; Name: "+c.getCountry());
    }
}
```