Learning Java 9 : Databases and Multithreading in Java

User Assignment-3

Problem Statement:

1. Create a json file called **customers.json**, with fields custId, custName, city and pin and populate with some data.
2. Write a hibernate program to read the json data from the json file and store it in the Postgre mysql database.
3. To access the postgre mysql, create a docker image with postgre mysql and run it to perform the database operations.
4. In the same program read the data which is inserted in the above from the postgre mysql and display on the console.

```java
package Product4.Hibernate;

import org.hibernate.Session;
import org.hibernate.SessionFactory;
import org.hibernate.Transaction;
import org.hibernate.cfg.Configuration;

import java.net.URL;
import java.util.List;

public class assign3 {
    private static SessionFactory factory;

    public static void main(String args[]) throws Exception {
        setUp();

        getCustomer();

        URL file_path =
Main.class.getClassLoader().getResource("customers.json");
        JSONProcessor jsonProcessor = new JSONProcessor(file_path.getPath());
        List<Customer> customer = jsonProcessor.parseFile();

        customer.forEach(Main::addCustomer);
    }

    private static void setUp() {
        factory = new Configuration()
                .addAnnotatedClass(Customer.class)
                .configure()
                .buildSessionFactory();
    }

    private static Integer addCustomer(Customer customer) {
        Session session = factory.openSession();
        Transaction tx = session.beginTransaction();
        Integer customerId = (Integer) session.save(customer);
        tx.commit();

        return customerId;
    }

    private static List<Customer> getCustomer() {
        Session session = factory.openSession();
        Transaction tx = session.beginTransaction();

        List<Customer> customer = session.createQuery("FROM Customer").list();
        System.out.println();

        return null;
    }
}
```

```java
import javax.persistence.*;

@Entity
@Table(name = "colibri.customer")
public class Car {
    @Column(name = "cusId")
    private String cusID;
    @Column(name = "cusName")
    private String cusName;
    @Column(name = "city")
    private String city;
    @Column(name = "pin")
    private int pin;
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "id")
    private int id;

    public Customer(String cusID, String cusName, String city,int pin) {
        this.cusID = cusID;
        this.cusName = cusName;
        this.city = city;
        this.pin = pin;
    }

    public Customer()
    {

    public int getPin() {
        return pin;
    }

    public String getCity() {
        return city;
    }

    public String getcusName() {
        return cusName;
    }

    public String getcusID() {
        return cusID;
    }

    public String toCSVString() {
        return cusID + ", " + cusName + ", " + city + ", " + pin;
    }

    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }

    public void setcusID(String cusID) {
        this.cusID = cusID;
    }

    public void setcusName(String cusName) {
        this.cusName = cusName;
    }

    public void setcity(String city) {
        this.city = city;
    }

    public void setpin(int pin) {
        this.pin = pin;
    }
}
```

```java
package Product4.Hibernate;



import org.json.simple.JSONArray;
import org.json.simple.JSONObject;
import org.json.simple.parser.JSONParser;
import org.json.simple.parser.ParseException;

import java.io.FileReader;
import java.io.IOException;
import java.util.List;
import java.util.stream.Collectors;

public class JSONProcessor {
    private final String targetFilePath;

    JSONProcessor(String targetFilePath) {
        this.targetFilePath = targetFilePath;
    }

    public List<Customer> parseFile() throws IOException, ParseException {
        JSONParser parser = new JSONParser();
        JSONObject json = (JSONObject) parser.parse(new
FileReader(targetFilePath));
        JSONArray customers = (JSONArray) json.get("customers");

        List<JSONObject> customerList = (List<JSONObject>)
customers.stream().collect(Collectors.toList());

        return customerList.stream()
            .map(x -> new Customer((String) x.get("cusId"), (String)
x.get("cusName"), (String) x.get("city"), (Double) x.get("pin")))
            .collect(Collectors.toList());
    }
}
```

```
{
    "customers": [
      {
        "custId": "1542",
        "custName": "jeethendra",
        "city": "Hassan",
        "pin": 573220
      },
      {
        "custId": "1652",
        "custName": "Shiva Kumar",
        "city": "gulbarga",
        "pin": 573201
      },
      {
        "custId": "1865",
        "custName": "jay",
        "city": "Hassan",
        "pin": 573201
      }
    ]
  }
```