**Name: Tong Wu CWID: A20410395**

**Name: Jimmy Tran CWID: A20368250**

**Name: Matt Flavin CWID: A20348995**

Table of Contents

[PART 1: 2](#_Toc441827106)-28

[PART 2: 2](#_Toc441827106)8-29

**PART1:**

**Code:**

We have provided some csv files for you to view my input xml files.

Admin:

**package** Model;

**public** **class** Admin {

**private** String id;

**private** String fullname;

**private** String password;

**public** Admin() {

}

**public** String getId() {

**return** id;

}

**public** **void** setId(String id) {

**this**.id = id;

}

**public** String getFullname() {

**return** fullname;

}

**public** **void** setFullname(String fullname) {

**this**.fullname = fullname;

}

**public** String getPassword() {

**return** password;

}

**public** **void** setPassword(String password) {

**this**.password = password;

}

}

CarRental:

**package** Model;

**public** **class** CarRental {

**private** String id;

**private** String type;

**private** String rented;

**private** String returned;

**private** **double** cost;

**public** CarRental() {

}

**public** String getId() {

**return** id;

}

**public** **void** setId(String id) {

**this**.id = id;

}

**public** String getType() {

**return** type;

}

**public** **void** setType(String type) {

**this**.type = type;

}

**public** String getDateRented() {

**return** rented;

}

**public** **void** setDateRented(String rented) {

**this**.rented = rented;

}

**public** String getDateReturned() {

**return** returned;

}

**public** **void** setDateReturned(String returned) {

**this**.returned = returned;

}

**public** **double** getRentCostDaily() {

**return** cost;

}

**public** **void** setRentCostDaily(**double** cost) {

**this**.cost = cost;

}

}

Customer:

**package** Model;

**public** **class** Customer {

**private** String id;

**private** String email;

**private** String fullname;

**private** String password;

**private** String streetaddress;

**private** String city;

**private** String state;

**private** **double** balance;

**public** Customer() {

}

**public** String getId() {

**return** id;

}

**public** **void** setId(String id) {

**this**.id = id;

}

**public** String getEmail() {

**return** email;

}

**public** **void** setEmail(String email) {

**this**.email = email;

}

**public** String getFullname() {

**return** fullname;

}

**public** **void** setFullname(String fullname) {

**this**.fullname = fullname;

}

**public** String getPassword() {

**return** password;

}

**public** **void** setPassword(String password) {

**this**.password = password;

}

**public** String getStreetaddress() {

**return** streetaddress;

}

**public** **void** setStreetaddress(String streetaddress) {

**this**.streetaddress = streetaddress;

}

**public** String getCity() {

**return** city;

}

**public** **void** setCity(String city) {

**this**.city = city;

}

**public** String getState() {

**return** state;

}

**public** **void** setState(String state) {

**this**.state = state;

}

**public** **double** getBalance() {

**return** balance;

}

**public** **void** setBalance(**double** balance) {

**this**.balance = balance;

}

}

PartService:

**package** Model;

**public** **class** PartService {

**private** String id;

**private** String type;

**private** **double** amount;

**private** **double** cost;

**public** PartService() {

}

**public** String getId() {

**return** id;

}

**public** **void** setId(String id) {

**this**.id = id;

}

**public** String getType() {

**return** type;

}

**public** **void** setType(String type) {

**this**.type = type;

}

**public** **double** getAmount() {

**return** amount;

}

**public** **void** setAmount(**double** amount) {

**this**.amount = amount;

}

**public** **double** getCost() {

**return** cost;

}

**public** **void** setCost(**double** cost) {

**this**.cost = cost;

}

}

User:

**package** Model;

**public** **class** User {

**private** String id;

**private** String mobile;

**private** String email;

**private** String fullname;

**private** String password;

**public** User() {

}

**public** String getId() {

**return** id;

}

**public** **void** setId(String id) {

**this**.id = id;

}

**public** String getMobile() {

**return** mobile;

}

**public** **void** setMobile(String mobile) {

**this**.mobile = mobile;

}

**public** String getEmail() {

**return** email;

}

**public** **void** setEmail(String email) {

**this**.email = email;

}

**public** String getFullname() {

**return** fullname;

}

**public** **void** setFullname(String fullname) {

**this**.fullname = fullname;

}

**public** String getPassword() {

**return** password;

}

**public** **void** setPassword(String password) {

**this**.password = password;

}

}

ParseAdmin:

package Control;

import java.io.IOException;

import java.util.ArrayList;

import javax.xml.parsers.DocumentBuilder;

import javax.xml.parsers.DocumentBuilderFactory;

import javax.xml.parsers.ParserConfigurationException;

import org.w3c.dom.Document;

import org.w3c.dom.Element;

import org.w3c.dom.Node;

import org.w3c.dom.NodeList;

import org.xml.sax.SAXException;

import Model.Admin;

public class ParseAdmin {

public static ArrayList<Admin> getParseAdmin() {

ArrayList<Admin> list= new ArrayList<Admin>();

DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance();

DocumentBuilder builder;

try {

builder = factory.newDocumentBuilder();

Document doc;

doc = builder.parse("File//Admin.xml");

NodeList adminNodes = doc.getElementsByTagName("Admin");

for(int i = 0; i<adminNodes.getLength();i++){

Element adminElement = (Element) adminNodes.item(i);

Admin admin = new Admin();

NodeList childNodes =adminElement.getChildNodes();

for (int j = 0; j < childNodes.getLength(); j++) {

if(childNodes.item(j).getNodeType()==Node.ELEMENT\_NODE){

if("AdminID".equals(childNodes.item(j).getNodeName())){

admin.setId((childNodes.item(j).getFirstChild().getNodeValue()));

}else if("FullName".equals(childNodes.item(j).getNodeName())) {

admin.setFullname(childNodes.item(j).getFirstChild().getNodeValue());

}else if("Password".equals(childNodes.item(j).getNodeName())) {

admin.setPassword(childNodes.item(j).getFirstChild().getNodeValue());

}

}

}list.add(admin);

}

}catch (SAXException e) {

e.printStackTrace();

} catch (IOException e) {

e.printStackTrace();

} catch (ParserConfigurationException e) {

e.printStackTrace();

}

return list;

}

}

ParseCar:

package Control;

import java.io.IOException;

import java.util.ArrayList;

import javax.xml.parsers.DocumentBuilder;

import javax.xml.parsers.DocumentBuilderFactory;

import javax.xml.parsers.ParserConfigurationException;

import org.w3c.dom.Document;

import org.w3c.dom.Element;

import org.w3c.dom.Node;

import org.w3c.dom.NodeList;

import org.xml.sax.SAXException;

import Model.CarRental;

public class ParseCar {

public static ArrayList<CarRental> getParseCarRental() {

ArrayList<CarRental> list= new ArrayList<CarRental>();

DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance();

DocumentBuilder builder;

try {

builder = factory.newDocumentBuilder();

Document doc;

doc = builder.parse("File//CarRental.xml");

NodeList carNodes = doc.getElementsByTagName("Car");

for(int i = 0; i<carNodes.getLength();i++){

Element carElement = (Element) carNodes.item(i);

CarRental car = new CarRental();

NodeList childNodes =carElement.getChildNodes();

for (int j = 0; j < childNodes.getLength(); j++) {

if(childNodes.item(j).getNodeType()==Node.ELEMENT\_NODE){

if("carId".equals(childNodes.item(j).getNodeName())){

car.setId((childNodes.item(j).getFirstChild().getNodeValue()));

}else if("carType".equals(childNodes.item(j).getNodeName())) {

car.setType((childNodes.item(j).getFirstChild().getNodeValue()));

}else if("dateRented".equals(childNodes.item(j).getNodeName())) {

car.setDateRented((childNodes.item(j).getFirstChild().getNodeValue()));

}else if("dateReturned".equals(childNodes.item(j).getNodeName())) {

car.setDateReturned((childNodes.item(j).getFirstChild().getNodeValue()));

}else if("rentCostDaily".equals(childNodes.item(j).getNodeName())) {

car.setRentCostDaily(Double.parseDouble((childNodes.item(j).getFirstChild().getNodeValue())));

}

}

}

list.add(car);

}

} catch (SAXException e) {

e.printStackTrace();

} catch (IOException e) {

e.printStackTrace();

} catch (ParserConfigurationException e) {

e.printStackTrace();

}

return list;

}

}

ParseCustomer:

package Control;

import java.io.IOException;

import java.util.ArrayList;

import javax.xml.parsers.DocumentBuilder;

import javax.xml.parsers.DocumentBuilderFactory;

import javax.xml.parsers.ParserConfigurationException;

import org.w3c.dom.Document;

import org.w3c.dom.Element;

import org.w3c.dom.Node;

import org.w3c.dom.NodeList;

import org.xml.sax.SAXException;

import Model.Customer;

public class ParseCustomer {

public static ArrayList<Customer> getParseCustomer() {

ArrayList<Customer> list= new ArrayList<Customer>();

DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance();

DocumentBuilder builder;

try {

builder = factory.newDocumentBuilder();

Document doc;

doc = builder.parse("File//Customer.xml");

NodeList cusNodes = doc.getElementsByTagName("Customer");

for(int i = 0; i<cusNodes.getLength();i++){

Element cusElement = (Element) cusNodes.item(i);

Customer customer = new Customer();

NodeList childNodes =cusElement.getChildNodes();

for (int j = 0; j < childNodes.getLength(); j++) {

if(childNodes.item(j).getNodeType()==Node.ELEMENT\_NODE){

if("UserID".equals(childNodes.item(j).getNodeName())){

customer.setId((childNodes.item(j).getFirstChild().getNodeValue()));

}else if("Email".equals(childNodes.item(j).getNodeName())) {

customer.setEmail(childNodes.item(j).getFirstChild().getNodeValue());

}else if("FullName".equals(childNodes.item(j).getNodeName())){

customer.setFullname(((childNodes.item(j).getFirstChild().getNodeValue())));

}else if("Password".equals(childNodes.item(j).getNodeName())) {

customer.setPassword(childNodes.item(j).getFirstChild().getNodeValue());

}else if("StreetAddress".equals(childNodes.item(j).getNodeName())) {

customer.setStreetaddress(childNodes.item(j).getFirstChild().getNodeValue());

}else if("City".equals(childNodes.item(j).getNodeName())) {

customer.setCity(childNodes.item(j).getFirstChild().getNodeValue());

}else if("State".equals(childNodes.item(j).getNodeName())) {

customer.setState(childNodes.item(j).getFirstChild().getNodeValue());

}else if("Balance".equals(childNodes.item(j).getNodeName())) {

customer.setBalance(Double.parseDouble((childNodes.item(j).getFirstChild().getNodeValue())));

}

}

}

list.add(customer);

}

} catch (SAXException e) {

e.printStackTrace();

} catch (IOException e) {

e.printStackTrace();

} catch (ParserConfigurationException e) {

e.printStackTrace();

}

return list;

}

}

ParsePart:

package Control;

import java.io.IOException;

import java.util.ArrayList;

import javax.xml.parsers.DocumentBuilder;

import javax.xml.parsers.DocumentBuilderFactory;

import javax.xml.parsers.ParserConfigurationException;

import org.w3c.dom.Document;

import org.w3c.dom.Element;

import org.w3c.dom.Node;

import org.w3c.dom.NodeList;

import org.xml.sax.SAXException;

import Model.PartService;

public class ParsePart {

public static ArrayList<PartService> getParseService() {

ArrayList<PartService> list= new ArrayList<PartService>();

DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance();

DocumentBuilder builder;

try {

builder = factory.newDocumentBuilder();

Document doc;

doc = builder.parse("File//PartsServices.xml");

NodeList partNodes = doc.getElementsByTagName("Part");

for(int i = 0; i<partNodes.getLength();i++){

Element partElement = (Element) partNodes.item(i);

PartService part = new PartService();

NodeList childNodes =partElement.getChildNodes();

for (int j = 0; j < childNodes.getLength(); j++) {

if(childNodes.item(j).getNodeType()==Node.ELEMENT\_NODE){

if("partId".equals(childNodes.item(j).getNodeName())){

part.setId((childNodes.item(j).getFirstChild().getNodeValue()));

}else if("partType".equals(childNodes.item(j).getNodeName())) {

part.setType((childNodes.item(j).getFirstChild().getNodeValue()));

}else if("partAmount".equals(childNodes.item(j).getNodeName())) {

part.setAmount(Double.parseDouble((childNodes.item(j).getFirstChild().getNodeValue())));

}else if("cost".equals(childNodes.item(j).getNodeName())) {

part.setCost(Double.parseDouble((childNodes.item(j).getFirstChild().getNodeValue())));

}

}

}

list.add(part);

}

} catch (SAXException e) {

e.printStackTrace();

} catch (IOException e) {

e.printStackTrace();

} catch (ParserConfigurationException e) {

e.printStackTrace();

}

return list;

}

}

ParseUser:

package Control;

import java.io.IOException;

import java.util.ArrayList;

import javax.xml.parsers.DocumentBuilder;

import javax.xml.parsers.DocumentBuilderFactory;

import javax.xml.parsers.ParserConfigurationException;

import org.w3c.dom.Document;

import org.w3c.dom.Element;

import org.w3c.dom.Node;

import org.w3c.dom.NodeList;

import org.xml.sax.SAXException;

import Model.User;

public class ParseUser {

public static ArrayList<User> getParseUser() {

ArrayList<User> list= new ArrayList<User>();

DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance();

DocumentBuilder builder;

try {

builder = factory.newDocumentBuilder();

Document doc;

doc = builder.parse("File//User.xml");

NodeList userNodes = doc.getElementsByTagName("User");

for(int i = 0; i<userNodes.getLength();i++){

Element userElement = (Element) userNodes.item(i);

User user = new User();

NodeList childNodes =userElement.getChildNodes();

for (int j = 0; j < childNodes.getLength(); j++) {

if(childNodes.item(j).getNodeType()==Node.ELEMENT\_NODE){

if("UserID".equals(childNodes.item(j).getNodeName())){

user.setId((childNodes.item(j).getFirstChild().getNodeValue()));

}else if("mobileN".equals(childNodes.item(j).getNodeName())){

user.setMobile(((childNodes.item(j).getFirstChild().getNodeValue())));

}else if("Email".equals(childNodes.item(j).getNodeName())) {

user.setEmail(childNodes.item(j).getFirstChild().getNodeValue());

}else if("FullName".equals(childNodes.item(j).getNodeName())) {

user.setFullname(childNodes.item(j).getFirstChild().getNodeValue());

}else if("Password".equals(childNodes.item(j).getNodeName())) {

user.setPassword(childNodes.item(j).getFirstChild().getNodeValue());

}

}

}list.add(user);

}

}catch (SAXException e) {

e.printStackTrace();

} catch (IOException e) {

e.printStackTrace();

} catch (ParserConfigurationException e) {

e.printStackTrace();

}

return list;

}

}

Connector:

package Control;

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.SQLException;

import java.sql.Statement;

import Model.Admin;

import Model.CarRental;

import Model.Customer;

import Model.PartService;

import Model.User;

public class Connector {

static String sqlStr = "jdbc:mysql://localhost/CarService";

static String rootName = "root";

static String rootPwd = "Tongky100%";

public static void writeToMysql(User user) {

try {

Class.forName("com.mysql.jdbc.Driver");

} catch (ClassNotFoundException e) {

System.out.println("Unable to find driver class, load driver failed.");

e.printStackTrace();

}

Statement st = null;

Connection con =null;

try {

con = (Connection)DriverManager.getConnection(sqlStr,rootName,rootPwd);

String UserID= user.getId();

String Mobile = user.getMobile();

String Email = user.getEmail();

String FullName = user.getFullname();

String Password = user.getPassword();

String sql = "insert into user(UserID,mobileN,Email,FullName,Password) values(\""+UserID+"\",\""+Mobile+"\",\""+Email+"\",\""+FullName+"\",\""+Password+"\")";

System.out.println(sql);

st = con.createStatement();

st.executeUpdate(sql);

} catch (SQLException e) {

e.printStackTrace();

}finally{

try {

st.close();

con.close();

} catch (SQLException e) {

e.printStackTrace();

}

}

}

public static void writeToMysql(Customer customer) {

try {

Class.forName("com.mysql.jdbc.Driver");

} catch (ClassNotFoundException e) {

System.out.println("Unable to find driver class, load driver failed.");

e.printStackTrace();

}

Statement st = null;

Connection con =null;

try {

con = (Connection)DriverManager.getConnection(sqlStr,rootName,rootPwd);

String UserID= customer.getId();

String Email = customer.getEmail();

String FullName = customer.getFullname();

String Password = customer.getPassword();

String StreetAddress = customer.getStreetaddress();

String City = customer.getCity();

String State = customer.getState();

double Balance = customer.getBalance();

String sql = "insert into customer(UserID,Email,FullName,Password,StreetAddress,City,State,Balance) values(\""+UserID+"\",\""+Email+"\",\""+FullName+"\",\""+Password+"\",\""+StreetAddress+"\",\""+City+"\",\""+State+"\",\""+Balance+"\")";

System.out.println(sql);

st = con.createStatement();

st.executeUpdate(sql);

} catch (SQLException e) {

e.printStackTrace();

}finally{

try {

st.close();

con.close();

} catch (SQLException e) {

e.printStackTrace();

}

}

}

public static void writeToMysql(CarRental car) {

try {

Class.forName("com.mysql.jdbc.Driver");

} catch (ClassNotFoundException e) {

System.out.println("Unable to find driver class, load driver failed.");

e.printStackTrace();

}

Statement st = null;

Connection con =null;

try {

con = (Connection)DriverManager.getConnection(sqlStr,rootName,rootPwd);

String CarID= car.getId();

String CarType = car.getType();

String DateRented = car.getDateRented();

String DateReturned = car.getDateReturned();

double DailyPrice = car.getRentCostDaily();

String sql = "insert into carrental(carId,carType,dateRented,dateReturned,rentCostDaily) values(\""+CarID+"\",\""+CarType+"\",\""+DateRented+"\",\""+DateReturned+"\",\""+DailyPrice+"\")";

System.out.println(sql);

st = con.createStatement();

st.executeUpdate(sql);

} catch (SQLException e) {

e.printStackTrace();

}finally{

try {

st.close();

con.close();

} catch (SQLException e) {

e.printStackTrace();

}

}

}

public static void writeToMysql(Admin admin) {

try {

Class.forName("com.mysql.jdbc.Driver");

} catch (ClassNotFoundException e) {

System.out.println("Unable to find driver class, load driver failed.");

e.printStackTrace();

}

Statement st = null;

Connection con =null;

try {

con = (Connection)DriverManager.getConnection(sqlStr,rootName,rootPwd);

String AdminID= admin.getId();

String FullName = admin.getFullname();

String Password = admin.getPassword();

String sql = "insert into admin(AdminID,FullName,Password) values(\""+AdminID+"\",\""+FullName+"\",\""+Password+"\")";

System.out.println(sql);

st = con.createStatement();

st.executeUpdate(sql);

} catch (SQLException e) {

e.printStackTrace();

}finally{

try {

st.close();

con.close();

} catch (SQLException e) {

e.printStackTrace();

}

}

}

public static void writeToMysql(PartService part) {

try {

Class.forName("com.mysql.jdbc.Driver");

} catch (ClassNotFoundException e) {

System.out.println("Unable to find driver class, load driver failed.");

e.printStackTrace();

}

Statement st = null;

Connection con =null;

try {

con = (Connection)DriverManager.getConnection(sqlStr,rootName,rootPwd);

String PartID = part.getId();

String PartType = part.getType();

double PartAmount = part.getAmount();

double PartCost = part.getCost();

String sql = "insert into partservices(PartId,PartType,PartAmount,Cost) values(\""+PartID+"\",\""+PartType+"\",\""+PartAmount+"\",\""+PartCost+"\")";

System.out.println(sql);

st = con.createStatement();

st.executeUpdate(sql);

} catch (SQLException e) {

e.printStackTrace();

}finally{

try {

st.close();

con.close();

} catch (SQLException e) {

e.printStackTrace();

}

}

}

}

Test:

package Control;

import java.util.ArrayList;

import Model.Admin;

import Model.CarRental;

import Model.Customer;

import Model.PartService;

import Model.User;

public class Test {

public static void main(String[] args) {

// TODO Auto-generated method stub

ArrayList<User> user= new ArrayList<User>();

ArrayList<Customer> customer=new ArrayList<Customer>();

ArrayList<Admin> admin=new ArrayList<Admin>();

ArrayList<CarRental> cr=new ArrayList<CarRental>();

ArrayList<PartService> ps=new ArrayList<PartService>();

user = ParseUser.getParseUser();

customer= ParseCustomer.getParseCustomer();

admin= ParseAdmin.getParseAdmin();

cr=ParseCar.getParseCarRental();

ps=ParsePart.getParseService();

for(int i=0;i<user.size();i++)

{

if (user.get(i)!=null) {

Connector.writeToMysql(user.get(i));

}

}

for(int i=0;i<customer.size();i++) {

if(customer.get(i)!=null) {

Connector.writeToMysql(customer.get(i));

}

}

for(int i=0;i<admin.size();i++) {

if(admin.get(i)!=null) {

Connector.writeToMysql(admin.get(i));

}

}

for(int i=0;i<cr.size();i++) {

if(cr.get(i)!=null) {

Connector.writeToMysql(cr.get(i));

}

}

for(int i=0;i<ps.size();i++) {

if(ps.get(i)!=null) {

Connector.writeToMysql(ps.get(i));

}

}

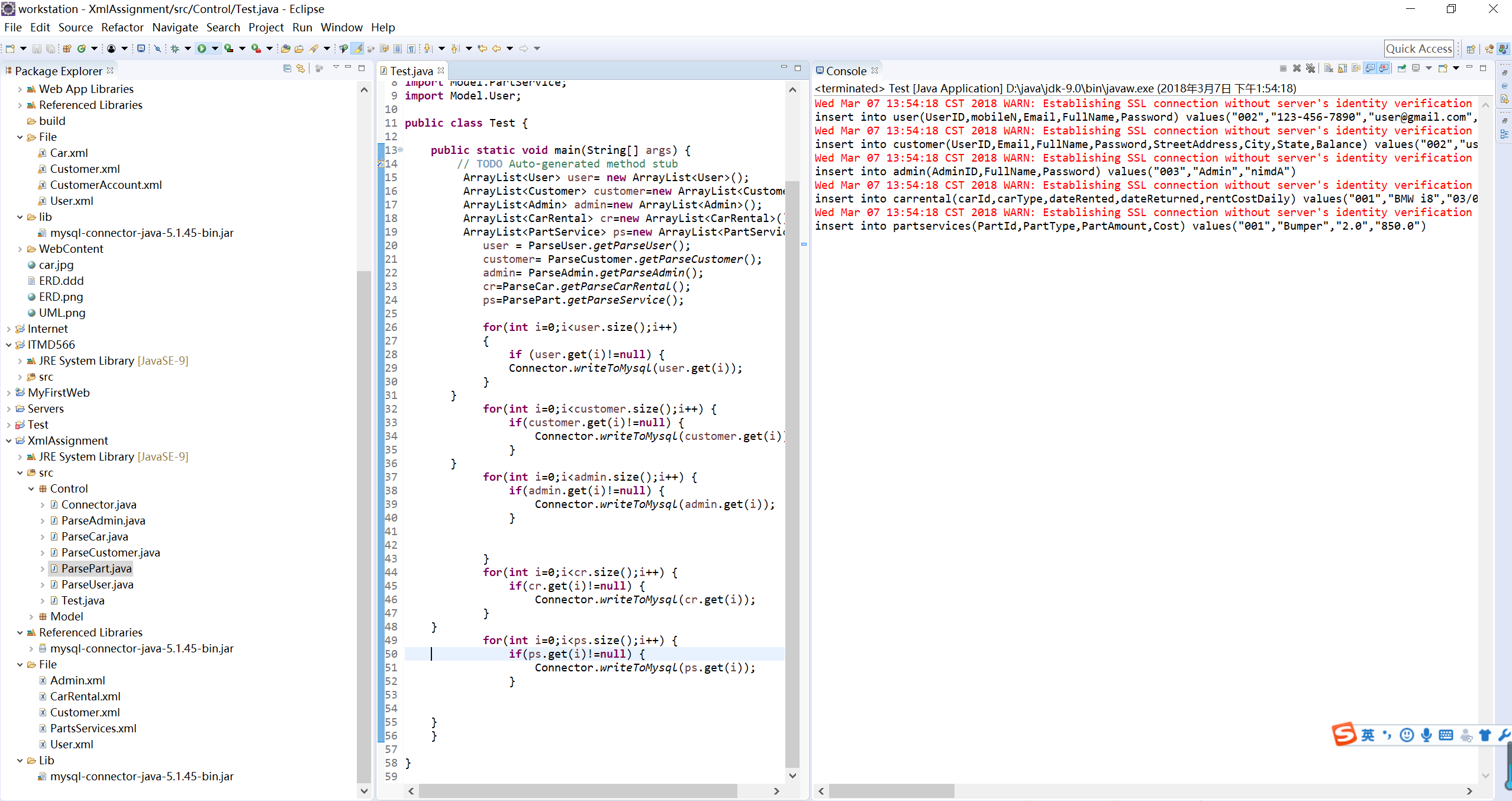
}

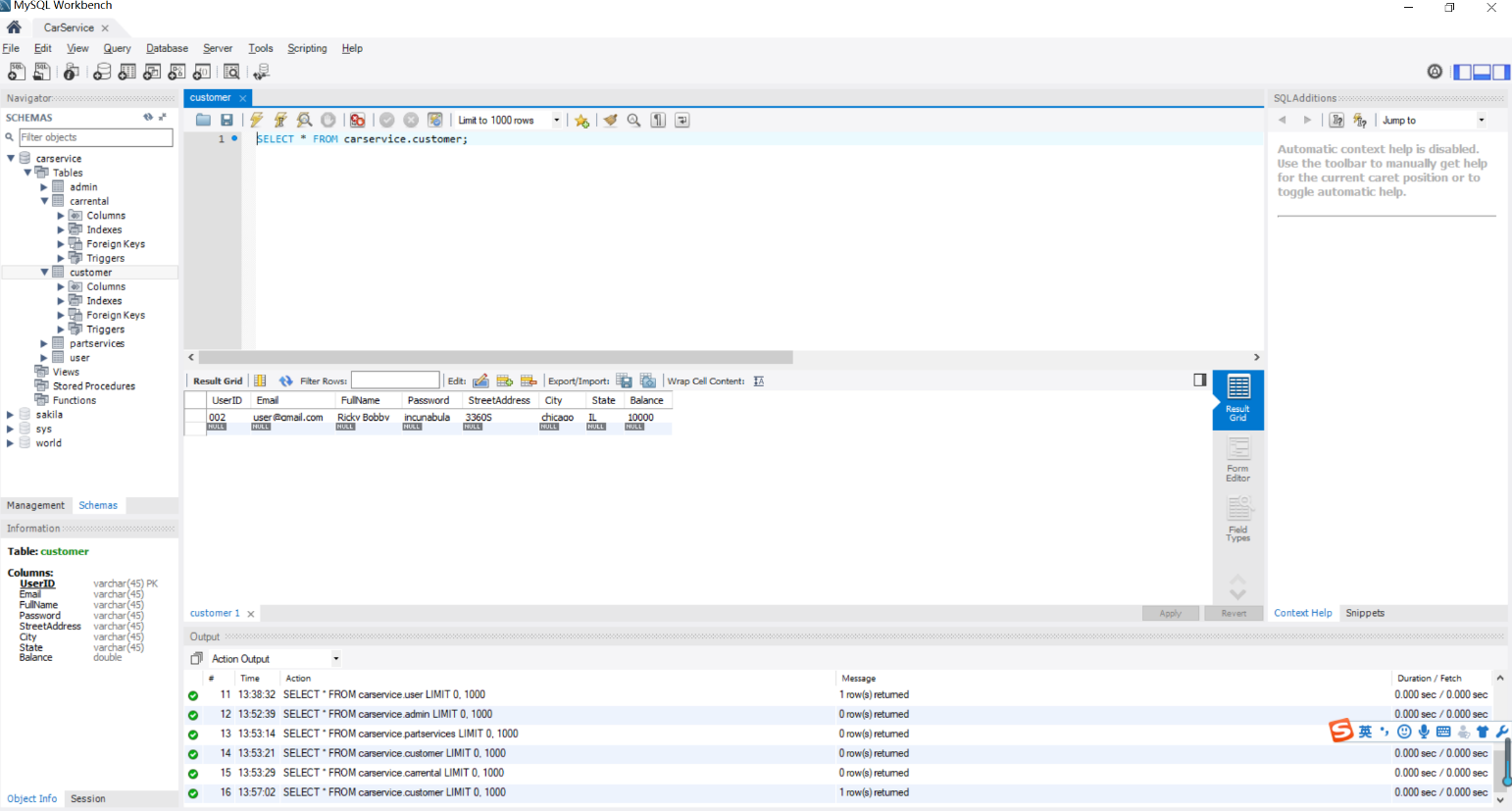
}

**PART1:**

**Screenshot:**

**Parse xml to Database；**

****

****