**ЗВІТ**

до лабораторної роботи №8

з предмету “Методи об’єктно орієнтованого програмування”

Виконав

Студент групи БІ-1

Франчук Іван

1. Постановка задачі

Реалізувати за допомогою REST web services

Відомості про об'єкти повинні зберігатися в базі даних.

Розробити клієнтську програму, що відправляє серверу запити на введення, редагування і отримання інформації про об'єкти.

Взаємодія між клієнтом і сервером має здійснюватися за технологією REST.

2. Вихідний код програми

Клас “Модель”:

import java.sql. \*;  
  
public class Model {  
 private final Connection con; // connection to db  
 private final Statement stmt; // operator  
  
 // constructor  
 public Model(String DBName, String ip, int port)  
 throws Exception {  
  
 String url = "jdbc:mysql://" + ip + ":" + port + "/" +  
 DBName + "?serverTimezone=Europe/Kiev&useSSL=FALSE";  
 con = DriverManager.*getConnection*(url, "admin", "Password\_1");  
 stmt = con.createStatement();  
 }  
  
 // models list  
 public void showModels() {  
 String sql = "SELECT ID, Name, ManufacturerID, ColorID, Year, EngineCapacity, Count FROM Model";  
 try {  
 ResultSet rs = stmt.executeQuery(sql);  
 System.*out*.println("Cars list:");  
 System.*out*.println("ID - Name - ManufacturerID - ColorID - Year - EngineCapacity - Count");  
 while (rs.next()) {  
 int id = rs.getInt("ID");  
 String name = rs.getString("Name");  
 int man\_id = rs.getInt("ManufacturerID");  
 int col\_id = rs.getInt("ColorID");  
 int year = rs.getInt("Year");  
 int eng\_cap = rs.getInt("EngineCapacity");  
 int count = rs.getInt("Count");  
 System.*out*.println(">>" + id + " - " + name + " - " + man\_id + " - " + col\_id + " - " + year +  
 " - " + eng\_cap + " - " + count);  
 }  
 rs.close();  
 } catch (SQLException e) {  
 System.*out*.println(  
 "ERROR while getting auto's list");  
 System.*out*.println(" >> " + e.getMessage());  
 }  
 }  
  
 // models by manufacturer ID  
 public void findModelsByManID(int manufacturer\_id) {  
 String sql = "SELECT ID, Name, ManufacturerID, ColorID, Year, EngineCapacity, Count FROM Model " +  
 "WHERE ManufacturerID = " + manufacturer\_id;  
 try {  
 ResultSet rs = stmt.executeQuery(sql);  
 System.*out*.println("Cars list:");  
 System.*out*.println("ID - Name - ManufacturerID - ColorID - Year - EngineCapacity - Count");  
 while (rs.next()) {  
 int id = rs.getInt("ID");  
 String name = rs.getString("Name");  
 int man\_id = rs.getInt("ManufacturerID");  
 int col\_id = rs.getInt("ColorID");  
 int year = rs.getInt("Year");  
 int eng\_cap = rs.getInt("EngineCapacity");  
 int count = rs.getInt("Count");  
 System.*out*.println(">>" + id + " - " + name + " - " + man\_id + " - " + col\_id + " - " + year +  
 " - " + eng\_cap + " - " + count);  
 }  
 rs.close();  
 } catch (SQLException e) {  
 System.*out*.println(  
 "ERROR while getting auto's list");  
 System.*out*.println(" >> " + e.getMessage());  
 }  
 }  
  
 // stop work  
 public void stop() throws SQLException {  
 con.close();  
 }  
  
 // add model  
 public boolean addModel(String name, int man\_id, int col\_id, int year, int eng\_cap, int count) {  
 String sql = "INSERT INTO Model (Name, ManufacturerID, ColorID, Year, EngineCapacity, Count) " +  
 "VALUES ('" + name + "', " + man\_id + ", " + col\_id + ", "  
 + year + ", " + eng\_cap + ", " + count + ")";  
 try {  
 stmt.executeUpdate(sql);  
 System.*out*.println("Model " + name + " added successfully");  
 return true;  
 } catch (SQLException e) {  
 System.*out*.println("ERROR! Model " + name + " not added!");  
 System.*out*.println(" >> " + e.getMessage());  
 return false;  
 }  
 }  
  
 // update model  
 public boolean updateModel(int id, String name, int man\_id, int col\_id, int year, int eng\_cap, int count) {  
 String sql = "UPDATE Model SET ID = " + id;  
 if (!name.equals("")) {  
 sql += ", Name = '" + name + "'";  
 }  
  
 if (man\_id != 0) {  
 sql += ", ManufacturerID = " + man\_id;  
 }  
  
 if (col\_id != 0) {  
 sql += ", ColorID = " + col\_id;  
 }  
  
 if (year > 0) {  
 sql += ", Year = " + year;  
 }  
  
 if (eng\_cap > 0) {  
 sql += ", EngineCapacity = " + eng\_cap;  
 }  
  
 if (count >= 0) {  
 sql += ", Count = " + count;  
 }  
 sql += " WHERE ID = " + id;  
  
 try {  
 stmt.executeUpdate(sql);  
 System.*out*.println("Model " + name + " added successfully");  
 return true;  
 } catch (SQLException e) {  
 System.*out*.println("ERROR! Model " + name + " not added!");  
 System.*out*.println(" >> " + e.getMessage());  
 return false;  
 }  
 }  
  
 // delete model  
 public boolean deleteModel(int id) {  
 String sql = "DELETE FROM Model WHERE ID =" + id;  
 try {  
 int c = stmt.executeUpdate(sql);  
 if (c > 0) {  
 System.*out*.println("Model with id " + id + " deleted successfully!");  
 return true;  
 } else {  
 System.*out*.println("Model with id " + id + " not found!");  
  
 return false;  
 }  
 } catch (SQLException e) {  
 System.*out*.println("ERROR while deleting model with id " + id);  
 System.*out*.println(" >> " + e.getMessage());  
 return false;  
 }  
 }  
  
 public static void main(String[] args) throws Exception {  
 }  
}

Клас “Виробник”:

import java.sql. \*;  
import java.util.Calendar;  
  
public class Manufacturer {  
 private final Connection con; // connection to db  
 private final Statement stmt; // operator  
  
 // constructor  
 public Manufacturer(String DBName, String ip, int port)  
 throws Exception {  
  
 String url = "jdbc:mysql://" + ip + ":" + port + "/" +  
 DBName + "?serverTimezone=Europe/Kiev&useSSL=FALSE&allowPublicKeyRetrieval=true";  
 con = DriverManager.*getConnection*(url, "admin", "Password\_1");  
 stmt = con.createStatement();  
 }  
  
 // manufacturers list  
 public void showManufacturers() {  
 String sql = "SELECT ID, Name, FoundationDate FROM Manufacturer";  
 try {  
 ResultSet rs = stmt.executeQuery(sql);  
 System.*out*.println("Manufacturers list:");  
 while (rs.next()) {  
 int id = rs.getInt("ID");  
 String name = rs.getString("Name");  
 Date foundation = rs.getDate("FoundationDate");  
 System.*out*.println(">>" + id + " - " + name + " - " + foundation.toString());  
 }  
 rs.close();  
 } catch (SQLException e) {  
 System.*out*.println(  
 "ERROR while getting Manufacturer`s list");  
 System.*out*.println(" >> " + e.getMessage());  
 }  
 }  
  
 public String getManufacturerFoundationDate(int id) {  
 String sql = "SELECT FoundationDate FROM Manufacturer WHERE ID = " + id;  
 try {  
 ResultSet rs = stmt.executeQuery(sql);  
 while (rs.next()) {  
 Date foundation = rs.getDate("FoundationDate");  
 return foundation.toString();  
 }  
 rs.close();  
 } catch (SQLException e) {  
 System.*out*.println(  
 "ERROR while getting Manufacturer foundation date");  
 System.*out*.println(" >> " + e.getMessage());  
 }  
  
 return "1970-01-01";  
 }  
  
 // stop work  
 public void stop() throws SQLException {  
 con.close();  
 }  
  
 // add manufacturer  
 public boolean addManufacturer(String name, Calendar foundation\_date) {  
 int month = foundation\_date.get(Calendar.*MONTH*);  
 String month\_q;  
 if (month < 10) {  
 month\_q = "0" + String.*valueOf*(month);  
 } else {  
 month\_q = String.*valueOf*(month);  
 }  
  
 int day = foundation\_date.get(Calendar.*DAY\_OF\_MONTH*);  
 String day\_q;  
 if (day < 10) {  
 day\_q = "0" + String.*valueOf*(day);  
 } else {  
 day\_q = String.*valueOf*(day);  
 }  
  
 String sql = "INSERT INTO Manufacturer (Name, FoundationDate) " +  
 "VALUES ('" + name + "', '" + foundation\_date.get(Calendar.*YEAR*) + '-' +  
 month\_q + '-' + day\_q + "')";  
 try {  
 stmt.executeUpdate(sql);  
 System.*out*.println("Manufacturer " + name + " added successfully");  
 return true;  
 } catch (SQLException e) {  
 System.*out*.println("ERROR! Manufacturer " + name + " not added!");  
 System.*out*.println(" >> " + e.getMessage());  
 return false;  
 }  
 }  
  
 public boolean updateManufacturer(int id, String name, String foundation\_date) {  
 String sql;  
 if (name.equals("")) {  
 sql = "UPDATE Manufacturer SET FoundationDate = '" + foundation\_date + "' WHERE ID = " + id;  
 } else {  
 sql = "UPDATE Manufacturer SET Name = '" + name + "', FoundationDate = '" +  
 foundation\_date + "' WHERE ID = " + id;  
 }  
 try {  
 stmt.executeUpdate(sql);  
 System.*out*.println("Manufacturer " + name + " updated successfully");  
 return true;  
 } catch (SQLException e) {  
 System.*out*.println("ERROR! Manufacturer " + name + " not updated!");  
 System.*out*.println(" >> " + e.getMessage());  
 return false;  
 }  
 }  
  
 // delete Manufacturer  
 public boolean deleteManufacturer(int id) {  
 String sql = "DELETE FROM Manufacturer WHERE ID =" + id;  
 try {  
 int c = stmt.executeUpdate(sql);  
 if (c > 0) {  
 System.*out*.println("Manufacturer with id " + id + " deleted successfully!");  
 return true;  
 } else {  
 System.*out*.println("Manufacturer with id " + id + " not found!");  
  
 return false;  
 }  
 } catch (SQLException e) {  
 System.*out*.println("ERROR while deleting Manufacturer with id " + id);  
 System.*out*.println(" >> " + e.getMessage());  
 return false;  
 }  
 }  
  
 public static void main(String[] args) throws Exception {  
 }  
}

Клас “Сервер”:

package lab;  
  
import lab.Model;  
  
import java.io.\*;  
import java.net.ServerSocket;  
import java.net.Socket;  
import java.util.Calendar;  
import java.util.GregorianCalendar;  
import java.util.Vector;  
  
public class Server {  
  
 private static Socket *clientSocket*;  
 private static ServerSocket *server*;  
 private static BufferedReader *in*;  
 private static BufferedWriter *out*;  
  
 public static void main(String[] args) throws Exception {  
 try {  
 try {  
 *server* = new ServerSocket(8080);  
  
 System.*out*.println("Server is running!");  
  
 while (true) {  
 *clientSocket* = *server*.accept();  
 try {  
 *in* = new BufferedReader(new InputStreamReader(*clientSocket*.getInputStream()));  
 *out* = new BufferedWriter(new OutputStreamWriter(*clientSocket*.getOutputStream()));  
  
 String word = *in*.readLine();  
 System.*out*.println("Server received msg: " + word);  
  
 if (word.equals("exit")) {  
 break;  
 }  
  
 *out*.write(*processClientMessage*(word) + "\n");  
 *out*.flush();  
 } finally {  
 *clientSocket*.close();  
 *in*.close();  
 *out*.close();  
 }  
 }  
 } finally {  
 System.*out*.println("Shutting down the server!");  
 *server*.close();  
 }  
 } catch (IOException e) {  
 System.*err*.println(e);  
 }  
 }  
  
 public static String processClientMessage(String mes) throws Exception {  
 Operation type;  
 int index = 0;  
 System.*out*.println("Operation: ");  
 if (mes.indexOf("Add") == 0) {  
 type = Operation.*Add*;  
 index += "Add".length();  
 System.*out*.println("Add");  
 } else if (mes.indexOf("Delete") == 0) {  
 type = Operation.*Delete*;  
 index += "Delete".length();  
 System.*out*.println("Delete");  
 } else if (mes.indexOf("Update") == 0) {  
 type = Operation.*Update*;  
 index += "Update".length();  
 System.*out*.println("Update");  
 } else if (mes.indexOf("Calculate") == 0) {  
 type = Operation.*Calculate*;  
 index += "Calculate".length();  
 System.*out*.println("Calculate");  
 } else if (mes.indexOf("Show") == 0) {  
 type = Operation.*Show*;  
 index += "Show".length();  
 System.*out*.println("Show");  
 } else {  
 type = Operation.*Unknown*;  
 System.*out*.println("Unknown");  
 }  
 if (type != Operation.*Calculate*) {  
 ++ index; // space after operation type in client msg  
 }  
 System.*out*.println("index = " + index);  
  
 System.*out*.println("Object is ");  
 Object obj = Object.*Unknown*;  
 ShowType show\_type = ShowType.*Unknown*;  
 switch (type) {  
 case *Add*:  
 case *Update*:  
 case *Delete*:  
 if (mes.indexOf("Model", index) == index) {  
 obj = Object.*Model*;  
 index += "Model".length();  
 System.*out*.println("Model");  
 } else if (mes.indexOf("Manufacturer", index) == index) {  
 obj = Object.*Manufacturer*;  
 index += "Manufacturer".length();  
 System.*out*.println("Manufacturer");  
 }  
 ++ index; // space after object type in client msg  
 break;  
 case *Calculate*:  
 break;  
 case *Show*:  
 System.*out*.println("Show type is ");  
 if (mes.indexOf("ManufacturerList", index) == index) {  
 show\_type = ShowType.*ManufacturerList*;  
 index += "ManufacturerList".length();  
 System.*out*.println("ManufacturerList");  
 } else if (mes.indexOf("ModelsWithManufacturer", index) == index) {  
 show\_type = ShowType.*ModelsWithManufacturer*;  
 index += "ModelsWithManufacturer".length();  
 System.*out*.println("ModelsWithManufacturer");  
 } else if (mes.indexOf("ModelsByManufacturer", index) == index) {  
 show\_type = ShowType.*ModelsByManufacturer*;  
 index += "ModelsByManufacturer".length();  
 ++index;  
 System.*out*.println("ModelsByManufacturer");  
 }  
 break;  
 case *Unknown*:  
 return "Wrong operation type provided. Try again.";  
 }  
  
 if (obj == Object.*Unknown* && show\_type == ShowType.*Unknown* && type != Operation.*Calculate*) {  
 return "Wrong client message. Try again";  
 }  
  
 if (type == Operation.*Add* || type == Operation.*Update* || type == Operation.*Delete*  
|| show\_type == ShowType.*ModelsByManufacturer*) {  
 return *makeQuery*(type, obj, show\_type, mes.substring(index));  
 } else {  
 return *makeQuery*(type, obj, show\_type, "");  
 }  
 }  
  
 public static String makeQuery(Operation type, Object obj, ShowType show\_type, String query) throws Exception {  
 Model model = new Model("CarShowroom", "localhost", 3306);  
 Manufacturer manufacturer = new Manufacturer("CarShowroom", "localhost", 3306);  
 switch (obj) {  
 case *Model*:  
 switch (type) {  
 case *Add*:  
 {  
 // String name, int man\_id, int col\_id, int year, int eng\_cap, int count  
 int index = 0;  
 String name = "";  
 // man\_id, col\_id, year, eng\_cap, count;  
 Vector<Integer> vals = new Vector<>();  
  
 // read query  
 {  
 int ind = query.indexOf(" ");  
 if (ind != -1) {  
 name = query.substring(0, ind);  
 index += ind;  
 ++ index;  
 }  
 System.*out*.println("index = " + index);  
  
 try {  
 for (int i = 0; i < 4; ++i) {  
 ind = query.indexOf(" ", index);  
 if (ind != -1) {  
 vals.add(Integer.*parseInt*(query.substring(index, ind)));  
 index = ind + 1;  
 }  
  
 if (i == 3) {  
 // add last number (no space after last number)  
 vals.add(Integer.*parseInt*(query.substring(index)));  
 break;  
 }  
 }  
 } catch (Exception e) {  
 return "Error while processing query: " + e.getMessage();  
 }  
 }  
  
 boolean is\_ok = model.addModel(name, vals.elementAt(0), vals.elementAt(1),  
 vals.elementAt(2), vals.elementAt(3), vals.elementAt(4));  
  
 if (is\_ok) {  
 return "Model added successfully";  
 } else {  
 return "Error while adding model";  
 }  
 }  
 case *Update*:  
 {  
 // String name, int man\_id, int col\_id, int year, int eng\_cap, int count  
 int index = 0;  
 String name = "";  
 // man\_id, col\_id, year, eng\_cap, count;  
 Vector<Integer> vals = new Vector<>();  
  
 // read query  
 {  
 int ind = query.indexOf(" ");  
 if (ind != -1) {  
 vals.add(Integer.*parseInt*(query.substring(0, ind)));  
 index = (ind + 1);  
 }  
  
 ind = query.indexOf(" ", index);  
 if (ind != -1) {  
 name = query.substring(index, ind);  
 index = (ind + 1);  
 }  
  
 try {  
 for (int i = 0; i < 4; ++i) {  
 ind = query.indexOf(" ", index);  
 if (ind != -1) {  
 vals.add(Integer.*parseInt*(query.substring(index, ind)));  
 index = (ind + 1);  
 }  
  
 if (i == 3) {  
 // add last number (no space after last number)  
 vals.add(Integer.*parseInt*(query.substring(index)));  
 break;  
 }  
 }  
 } catch (Exception e) {  
 return "Error while processing query";  
 }  
 }  
  
 boolean is\_ok = model.updateModel(vals.elementAt(0), name, vals.elementAt(1), vals.elementAt(2),  
 vals.elementAt(3), vals.elementAt(4), vals.elementAt(5));  
  
 if (is\_ok) {  
 return "Model updated successfully";  
 } else {  
 return "Error while updating model";  
 }  
 }  
 case *Delete*:  
 {  
 int index = 0;  
 int id = 0;  
  
 // read query  
 {  
 int ind = query.indexOf(" ");  
 if (ind != -1) {  
 id = Integer.*parseInt*(query.substring(0, ind));  
 } else {  
 id = Integer.*parseInt*(query);  
 }  
 }  
  
 boolean is\_ok = model.deleteModel(id);  
  
 if (is\_ok) {  
 return "Model deleted succsessfully";  
 } else {  
 return "Error while deleting model";  
 }  
 }  
 default:  
 return "Wrong operation for Model object.";  
 }  
 case *Manufacturer*:  
 switch (type) {  
 case *Add*:  
 {  
 // String name, Calendar foundation\_date (year, month, day)  
 int index = 0;  
 String name = "";  
 Vector<Integer> vals = new Vector<>();  
  
 // read query  
 {  
 int ind = query.indexOf(" ");  
 if (ind != -1) {  
 name = query.substring(0, ind);  
 index += ind;  
 ++ index;  
 }  
 System.*out*.println("index = " + index);  
  
 try {  
 for (int i = 0; i < 3; ++i) {  
 ind = query.indexOf(" ", index);  
 if (ind != -1) {  
 vals.add(Integer.*parseInt*(query.substring(index, ind)));  
 index = ind + 1;  
 }  
  
 if (i == 2) {  
 // add last number (no space after last number)  
 vals.add(Integer.*parseInt*(query.substring(index)));  
 break;  
 }  
 }  
  
 System.*out*.println("Foundation date:");  
 for (var el : vals) {  
 System.*out*.println(el);  
 }  
  
 } catch (Exception e) {  
 return "Error while processing query: " + e.getMessage();  
 }  
 }  
  
// System.out.println("");  
 Calendar calendar = new GregorianCalendar(vals.elementAt(0), vals.elementAt(1) - 1,  
 vals.elementAt(2));  
 boolean is\_ok = manufacturer.addManufacturer(name, calendar);  
  
 if (is\_ok) {  
 return "Manufacturer added successfully";  
 } else {  
 return "Error while adding manufacturer";  
 }  
  
 }  
 case *Delete*:  
 {  
 int id = 0;  
  
 // read query  
 {  
 int ind = query.indexOf(" ");  
 if (ind != -1) {  
 id = Integer.*parseInt*(query.substring(0, ind));  
 } else {  
 id = Integer.*parseInt*(query);  
 }  
 }  
  
 boolean is\_ok = manufacturer.deleteManufacturer(id);  
  
 if (is\_ok) {  
 return "Manufacturer deleted successfully";  
 } else {  
 return "Error while deleting manufacturer";  
 }  
 }  
 default:  
 return "Wrong operation for Manufacturer object.";  
 }  
 case *Unknown*:  
 switch (type) {  
 case *Show*:  
 switch (show\_type) {  
 case *ManufacturerList*:  
 return manufacturer.showManufacturers();  
 case *ModelsByManufacturer*:  
 {  
 int man\_id = 0;  
  
 // read query  
 {  
 int ind = query.indexOf(" ");  
 if (ind != -1) {  
 man\_id = Integer.*parseInt*(query.substring(0, ind));  
 } else {  
 man\_id = Integer.*parseInt*(query);  
 }  
 }  
  
 return model.findModelsByManID(man\_id);  
 }  
 case *ModelsWithManufacturer*:  
 return model.showModelsByManufacturer();  
 default:  
 return "Wrong show type provided";  
 }  
 case *Calculate*:  
 return model.countModelsByManufacturer();  
 default:  
 return "Wrong operation for non-selected object.";  
 }  
 default:  
 return "Something went wrong";  
 }  
 }  
  
 public static enum Operation {  
 *Add*,  
 *Delete*,  
 *Update*,  
 *Calculate*,  
 *Show*,  
 *Unknown*  
};  
  
 public static enum Object {  
 *Model*,  
 *Manufacturer*,  
 *Unknown*  
};  
  
 public static enum ShowType {  
 *ManufacturerList*,  
 *ModelsWithManufacturer*,  
 *ModelsByManufacturer*,  
 *Unknown*  
}  
}

Клієнт:

package rest.client;  
  
import com.google.api.client.http.HttpRequest;  
import com.google.api.client.http.HttpRequestFactory;  
import com.google.api.client.http.HttpResponse;  
import com.google.api.client.http.HttpTransport;  
import com.google.api.client.http.apache.ApacheHttpTransport;  
import com.google.api.client.json.gson.GsonFactory;  
import com.google.gson.Gson;  
import com.google.gson.GsonBuilder;  
import com.lab.L8Server.ServerResponse;  
  
import java.io.\*;  
  
  
public class Client {  
 private static final Gson *GSON* = new GsonBuilder().create();  
 private static final HttpTransport *HTTP\_TRANSPORT* = new ApacheHttpTransport();  
 private static final GsonFactory *GSON\_FACTORY* = new GsonFactory();  
 private static final HttpRequestFactory *REQUEST\_FACTORY* = *HTTP\_TRANSPORT*.createRequestFactory();  
  
 public static String sendRequest(String command) throws IOException {  
 HttpRequest req = *REQUEST\_FACTORY*.buildRequest("GET", new Request(command), null);  
  
 HttpResponse httpResponse = req.execute();  
 ServerResponse response = *GSON*.fromJson(httpResponse.parseAsString(), ServerResponse.class);  
 return response.getResponse();  
 }  
  
 public static void main(String[] args) {  
 while (true) {  
 try {  
 BufferedReader reader = new BufferedReader(new InputStreamReader(System.*in*));  
  
 System.*out*.println("Input your query:");  
 String word = reader.readLine();  
  
 if (word.equals("exit")) {  
 break;  
 }  
  
 String response = *sendRequest*(word);  
 System.*out*.println(response);  
  
 } catch (Exception ex) {  
  
 }  
 }  
  
 }  
}

Запит

package rest.client;  
  
import com.google.api.client.http.GenericUrl;  
  
public class Request extends GenericUrl {  
 public Request(String request) {  
 super("http://localhost:8080/processRequest?request=" + request);  
 }  
}

3. Опис програми

У програмі реалізовано декуілька класів.

Перший – клас “Модель”. Об’єкт класу “Модель” може змінювати поля у базі даних у відповідній таблиці “Model”. У класі реалізовані операції додавання нового об’єкту, редагування існуючого, видалення існуючого об’єкту та пошук за заданим критерієм (виробником).

Другий клас – “Виробник”. Об’єкт класу “Виробник” може змінювати поля у базі даних у відповідній таблиці “Manufacturer”. У класі реалізовані операції додавання нового об’єкту, редагування існуючого та видалення існуючого об’єкту.

Третій клас – клас “Сервер”. Об’єкт класу сервер приймає повідомлення від клієнту та займається їх обробкою. Використовує класи “Модель” та “Виробник”.

Четвертий – головний клас для запуску застосунку і тестування. В ньому реалізована клієнтська частина. При тестуванні ведеться діалог з користувачем, задля можливості перевірки різних варіантів у ручному режимі.

4. Опис протоколу взаємодії клієнту із сервером.

Клієнтське повідомлення задається у строгому форматі. На початку пишеться необхідна операція, одна з множини – “*Add*, *Delete*, *Update*, *Calculate*, *Show”.* Далі структура повідомлення залежить від обраної команди:

* Якщо було обрано одну з команд “Додати, Видалили, Оновити”, то далі необхідно написати для якого об’єкта ця команда ( Model або Manufacturer). Далі, відповідно до команди, надати необхідні дані. Наприклад додавання нового виробника може виглядати так (Ім’я, дата заснування у формати рік місяць день): “Add Manufacturer Mazda 1920 1 30” – додати виробника Мазда, що був заснований 30го січня 1920 року.
* Якщо було обрано команду “Вирахувати” (Calculate), нічого не потрібно писати далі, адже є лише один запит, який необхідно вираховувати – це “Підрахунок кількості марок у виробника”.
* Якщо було обрано команду “Показати” ( Show), то далі необхідно обрати одну із опцій, що саме ми хочемо подивитися. Можливі варіанти: *ManufacturerList*, *ModelsWithManufacturer*, *ModelsByManufacturer –* список виробників, список співвідношення моделей із виробниками та пошук моделі за виробником відповідно. У разі обрання пошуку моделей за виробником ( *ModelsByManufacturer*) необхідно ще ввести ідентифікатор виробника. Наприклад, “Show ModelsByManufacturer 7”.