TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI

VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

──────── \* ───────

BÁO CÁO BÀI TẬP LỚN

MÔN HỌC: NHẬP MÔN JAVA

**XÂY DỰNG CHƯƠNG TRÌNH QUẢN LÝ CÁC PHƯƠNG TIỆN GIAO THÔNG**

Sinh viên thực hiện : **Đình Đức Lâm**

**Ngô Quốc Dũng**

**Nguyễn Phạm Đức Kiên**

**Nguyễn Tiến Đạt**

Lớp : **IT-LTU-K64**

Giáo viên hướng dẫn : TS. **TRẦN HẢI ANH**

***Hà Nội, tháng 12 năm 2020***

MỤC LỤC

[MỤC LỤC 2](#_Toc59829666)

[LỜI NÓI ĐẦU 3](#_Toc59829667)

[PHÂN CÔNG THÀNH VIÊN TRONG NHÓM 4](#_Toc59829668)

[CHƯƠNG 1. GIỚI THIỆU CHUNG 5](#_Toc59829669)

[1.1. Đặt vấn đề 5](#_Toc59829670)

[1.2. Tổng quan các công nghệ sử dụng 5](#_Toc59829671)

[1.3. Mục tiêu đặt ra 5](#_Toc59829672)

[CHƯƠNG 2. CÔNG NGHỆ SỬ DỤNG 6](#_Toc59829673)

[2.1. JDK 8 6](#_Toc59829674)

[2.2. Eclipse IDE for Java Developers 6](#_Toc59829675)

[2.3. XAMPP Control Panel 6](#_Toc59829676)

[2.4. Thư viện poi-4.1.2 6](#_Toc59829677)

[2.5. Git và GitHub 6](#_Toc59829678)

[CHƯƠNG 3. XÂY DỰNG CHƯƠNG TRÌNH 7](#_Toc59829679)

[3.1. Xây dựng chương trình 7](#_Toc59829680)

[3.1.1. Tổng quan về chương trình 7](#_Toc59829681)

[3.1.2. Giao diện đồ họa. 7](#_Toc59829682)

[3.1.3. Kết nối và cách trình bày cơ sở dữ liệu. 10](#_Toc59829683)

[3.1.4. Nguyên lý hoạt động của chương trình. 12](#_Toc59829684)

[3.2. Kết quả thử nghiệm 20](#_Toc59829685)

[3.3. Kết quả chương trình 20](#_Toc59829686)

[CHƯƠNG 4. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN 21](#_Toc59829687)

[TÀI LIỆU THAM KHẢO 22](#_Toc59829688)

[PHỤ LỤC 23](#_Toc59829689)

LỜI NÓI ĐẦU

Việc ứng dụng máy tính để quản lý thông tin đã phát triển mạnh ở các nước tiên tiến từ những thập niên 70. Còn ở Nước ta hiện nay, vấn đề áp dụng Tin học để xử lý thông tin trong công tác quản lý đã trở thành nhu cầu bức thiết, nhất là trong thời đại "mở cửa" với các nước khác trên thế giới.

Hiện nay, việc quản lý lượng phương tiện giao thông là một vấn đề hết sức cần thiết. Việc Tin học hóa hệ thống quản lý phương tiện giao thông giúp chúng ta có thể kiểm soát số lượng, chất lượng của những phương tiện khi tham gia giao thông.

Lượng phương tiện giao thông tại Việt Nam hiện nay tăng đột biến, đơn cử là ô tô. Dựa theo bảng số liệu về doanh số bán xe và tốc độ tăng trưởng trung bình của ngành ô tô Việt Nam từ 2007 đến tháng 11 năm 2019 có thể thấy doanh số bán xe ô tô tại Việt Nam tăng đều qua các năm và đặc biệt là những năm gần đây thì doanh số đều đạt gần 300000 xe 1 năm. Điều đó cho thấy rằng lượng người có nhu cầu sở hữu những chiếc xe riêng đang ngày càng nhiều vì vậy một chương trình quản lý các loại phương tiện là một công cụ cần thiết và kịp thời lúc này.

Qua quan sát tình hình giao thông trên cả nước, đặc biệt tại TP Hà Nội, nhóm chúng em đã cùng nhau quyết định phân tích và thiết kế cho phần mềm quản lý phương tiện giao thông. Lượng phương tiện giao thông tại các thành phố lớn đang tăng nhanh chóng do số lượng người từ nông thôn chuyển lên thành thị ngày càng nhiều do vậy hệ thống Tin học hoá về việc cập nhật, sửa đổi, thêm và xoá các loại phương tiện giao thông như ô tô,xe máy ngày càng cần thiết. Việc có một phần mềm có thể quản lý phương tiện giao thông giúp cho kiểm soát về số lượng, chất lượng cũng như phạm vi hoạt động của phương tiện dễ dàng, từ đó có thể đưa ra phương hướng giúp cải thiện cơ sở hạ tầng, giảm thải ùn tắc ô nhiễm nhanh chóng thuận tiện hơn. Đó cũng là lý do chính mà nhóm chúng em lựa chọn đề tài này

Để hoàn thành được bài tập lớn này, nhóm chúng em xin được gửi lời cảm ơn chân thành đến thầy giáo hướng dẫn đề tài Thầy Trần Hải Anh, Giảng viên Viện Công nghệ Thông tin và Truyền thông, Trường Đại học Bách Khoa Hà Nội - đã hết lòng giúp đỡ, hướng dẫn, chỉ dạy tận tình để nhóm em hoàn thành được đề tài này.

***Hà Nội, tháng 12 năm 2020***

**Nhóm 7.1**

***Đình Đức Lâm***

***Ngô Quốc Dũng***

***Nguyễn Phạm Đức Kiên***

***Nguyễn Tiến Đạt***

PHÂN CÔNG THÀNH VIÊN TRONG NHÓM

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **STT** | **Họ tên** | **MSSV** | **Công việc đóng góp** | **Mức độ hoàn thành** |
| 1 | Đình Đức Lâm | 20198236 | * Xây dựng chương trình. * Viết báo cáo phần chương 2, 3, 4, kết luận và hướng phát triển. * Thuyết trình. | Tốt |
| 2 | Ngô Quốc Dũng | 20198219 | * Làm slide. * Viết báo cáo phần lời nói đầu và chương 1. * Đóng góp ý tưởng. | Tốt |
| 3 | Nguyễn Phạm Đức Kiên | 20198234 | * Viết báo cáo phần phụ lục. * Đóng góp ý tưởng. | Tốt |
| 4 | Nguyễn Tiến Đạt | 20198215 | * Viết báo cáo phần phụ lục. * Đóng góp ý tưởng. | Tốt |

# GIỚI THIỆU CHUNG

## Đặt vấn đề

Như đã nêu ở phần mở đầu, với việc phát triển của công nghệ và ứng dụng của công nghệ vào trong các lĩnh vực trong đời sống xã hội, nhằm phục vụ cho việc quản lý tra cứu thông tin cũng như kiểm soát về số lượng và chất lượng của các loại phương tiện giao thông một cách thuận tiện, nhanh chóng để phục vụ cho các yêu cầu riêng khi cần thiết nên nhóm em đã chọn đề tài này để làm mini-project lần này.

Bài toán yêu cầu xây dựng chương trình quản lý các phương tiện giao thông *(vehicle management)* bao gồm: ô tô *(car)*, xe máy *(bike)*, xe tải *(truck)*. Các phương tiện có 4 thuộc tính chung là:

* Hãng sản xuất *(Make).*
* Năm sản xuất *(Manufacturing Year).*
* Giá bán *(Price).*
* Màu xe *(Color).*

Và các thuộc tính riêng:

* Ô tô *(car)*: số chỗ ngồi *(number of seats)* và kiểu động cơ *(type of engine).*
* Xe máy *(bike)*: Công suất *(power).*
* Xe tải *(truck)*: Trọng tải *(load).*

Chương trình cung cấp các chức năng như:

* Thêm, sửa, xóa thông tin các loại xe *(add, update, delete).*
* Tìm kiếm các loại xe với các loại thông tin tìm kiếm khác nhau *(find)*.

## Tổng quan các công nghệ sử dụng

Chương trình sử dụng:

* JDK 8
* Eclipse IDE để viết code.
* MySQL để quản trị dữ liệu.
* XAMPP để tạo máy chủ local và sử dụng MySQL.
* Thư viện *poi-4.1.2*sử dụng để xuất file Excel.
* Thư viện *mysql-connector-java-8.0.22* sử dụng để kết nối tới cơ sở dữ liệu.
* *Git và GitHub* sử dụng để lưu trữ mã nguồn.

## Mục tiêu đặt ra

* Kết hợp giữa làm việc nhóm và kiến thức mỗi cá nhân làm ra một chương trình có thể đáp ứng tốt các yêu cầu bài toán đề ra.
* Xây dựng chương trình đáp ứng những nhu cầu cơ bản của người sử dụng với giao diện thân thiện, ưa nhìn, không quá cầu kì nhưng cũng không được quá sơ sài, đơn điệu và đặc biệt là dễ dàng sử dụng
* Ngoài những tính năng đặc biệt như yêu cầu của bài toán, có thể làm thêm những tính năng khác ví dụ như xuất ra file excel,…

# CÔNG NGHỆ SỬ DỤNG

## JDK 8

Java Development Kit 8 (JDK – Bộ công cụ cho người phát triển ứng dụng bằng ngôn ngữ lập trình Java) là một bộ bao gồm các công cụ cần thiết trong quá trình phát triển các ứng dụng Java + JRE. Các công cụ đó là: javac.exe, java.exe, applet viewer,… vai trò của từng công cụ như sau:

* Javac.exe – giúp biên dịch các file nguồn viết bằng ngôn ngữ Java (\*.java) thành mã máy (byte code)
* Java.exe – gọi tới JRE, tải nạp các file mã máy đã được biên dịch và gọi tới method main để thực thi các mã đó.
* Applet Viewer – để chạy các ứng dụng viết bằng applet.

## Eclipse IDE for Java Developers

Eclipse là một trình IDE dùng để lập trình Java (Eclipse cũng có thể dùng để lập trình các ngôn ngữ khác nhau C/C++, PHP… nhưng mạnh nhất là về Java).

## XAMPP Control Panel

Xampp là chương trình tạo máy chủ Web (Web Server) được tích hợp sẵn Apache, PHP, MySQL, FTP Server, Mail Server và các công cụ như phpMyAdmin. Không như Appserv, Xampp có chương trình quản lý khá tiện lợi, cho phép chủ động bật tắt hoặc khởi động lại các dịch vụ máy chủ bất kỳ lúc nào.

## Thư viện poi-4.1.2

Apache POI là một thư viện mã nguồn mở Java, được cung cấp bởi Apache. Thư viện này cung cấp các API (phương thức) làm việc với các tài liệu của Microsoft như Word, Excel, Power point, Visio,…

## Git và GitHub

* Là công cụ giúp quản lý source code tổ chức theo dạng dữ liệu phân tán.
* Giúp đồng bộ source code của team lên 1 server.
* Hỗ trợ các thao tác kiểm tra source code trong quá trình làm việc (diff, check modifications, show history, merge source, …).

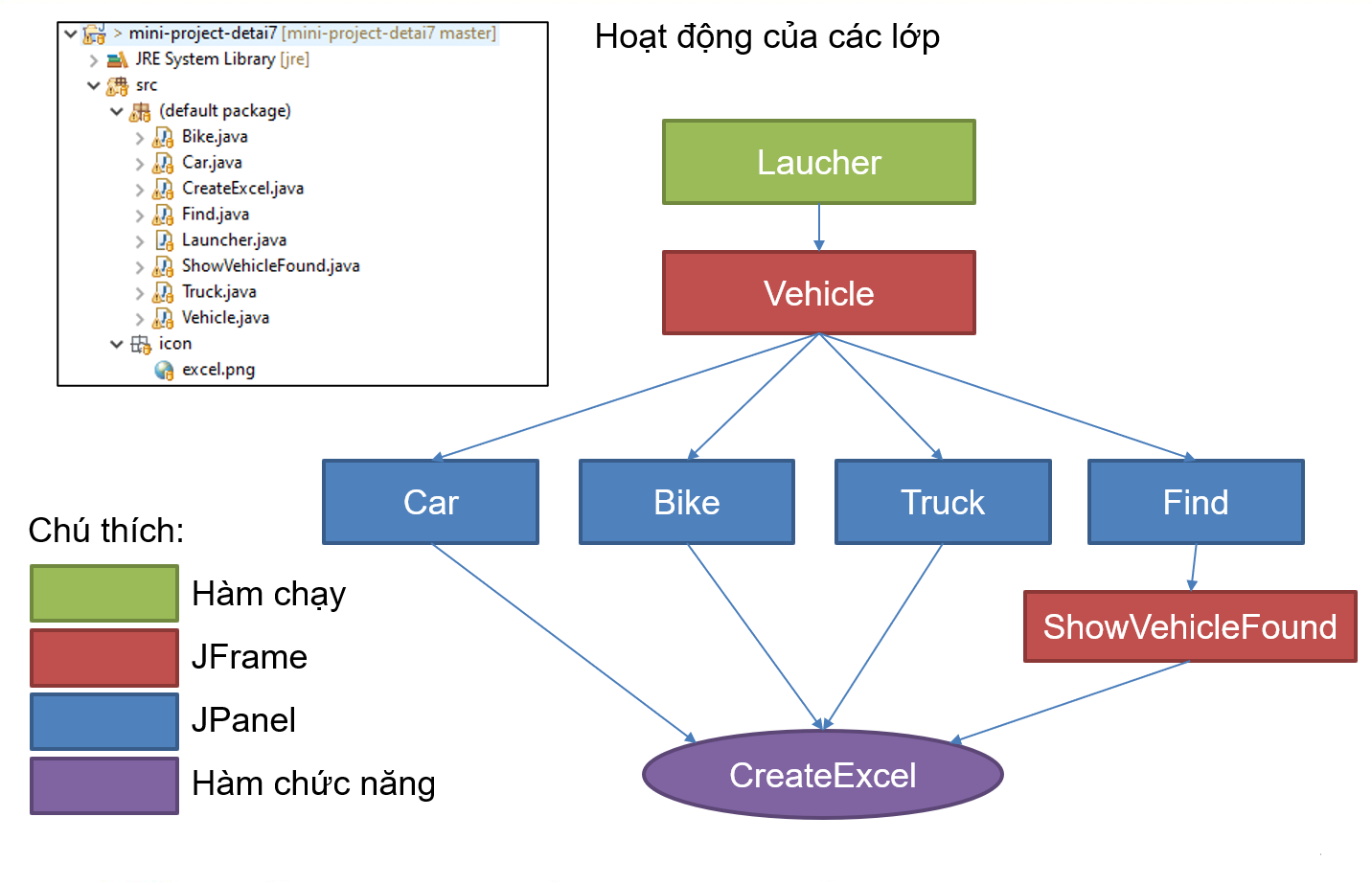
# XÂY DỰNG CHƯƠNG TRÌNH

## Xây dựng chương trình

### Tổng quan về chương trình

Chương trình gồm có 6 lớp:

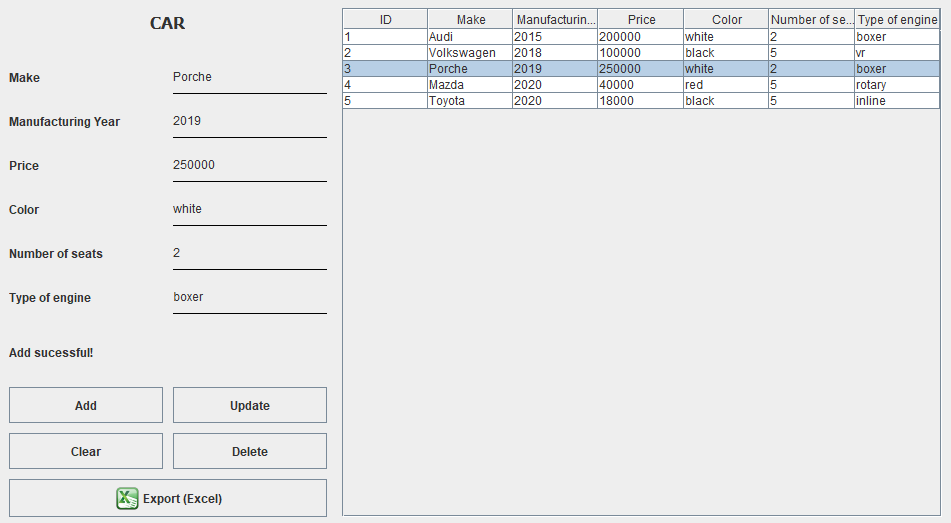
* Laucher
* Vehicle
* Car
* Bike
* Truck
* Find
* ShowVehicleFound
* CreateExcel



### Giao diện đồ họa.

#### Lớp Car, Bike và Truck.

Vì giao diện đồ họa của 3 lớp Car, Bike và Truck tương tự nhau nên chúng em xin phép chỉ trình bày về giao diện đồ họa của *lớp Car*.



*Hình 3.1.1.1*

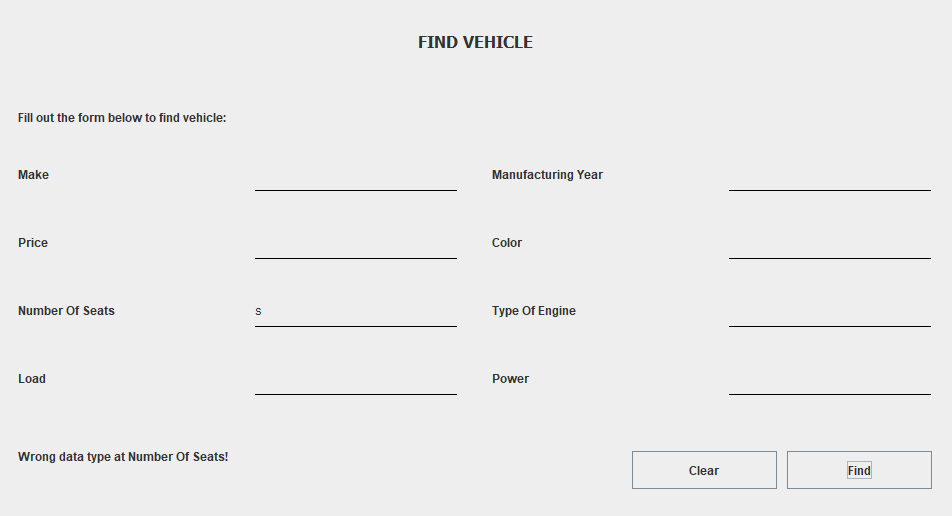
*Lớp Car* kế thừa từ *lớp JPanel*. JPanel chính sử BoxLayout (X-AXIS) chứa một *JPanel bên trái* vàmột *JTable bên phải (được bao bởi JScrollPane).*

JPanel bên trái sử dụng BorderLayout với:

* NORTH: JLabel(“CAR”).
* SOUTH: Một JPanel sử dụng BorderLayout chứa:
  + NORTH: Là JLabel *status* (Trong ảnh là dòng chữ *“Please fill all the font!”*).
  + SOUTH: Là JButton Export (Excel) *(export)*.
  + CENTER: Là JPanel sử dụng GridLayout(2, 2) chứa 4 JButton *add, update, clear, delete.*
* CENTER: Là JPanel sử dụng GridLayout(6, 2) bao gồm 6 JLabel và 6 JTextField *carMake, carManufacturingYear, carPrice, carColor, carNumberOfSeats, carTypeOfEngine.*

JScroll Pane (bên phải) chứa JTable gồm các cột "ID", "Make", "Manufacturing Year", "Price", "Color", "Number of seats", "Type of engine" (Object *carCol[]*).

#### Lớp Find.

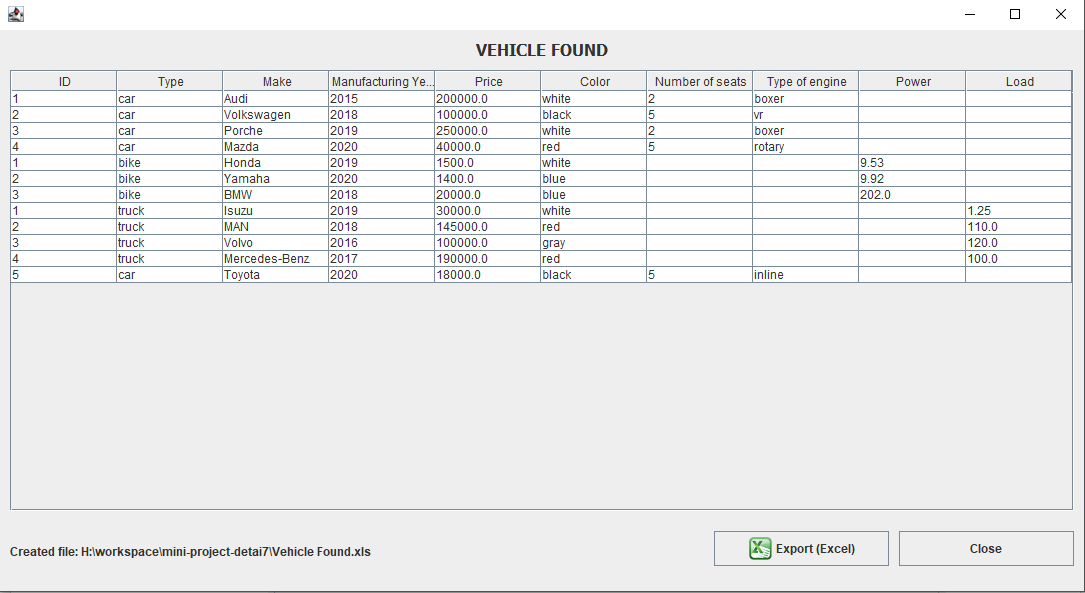


*Hình 3.1.1.2*

Như 3 lớp trên, *lớp Find* cũng được kế thừa từ *lớp JPanel.* JPanel chính sử dụng Border Layout với:

* NORTH: Là JLabel(“FIND VEHICLE”).
* SOUTH: Là 2 JButton Clear, Find *(clear, find)* và 1 JLabel *status (Là dòng chữ “Wrong data type at Number Of Seats” trong ảnh)*
* CENTER: Là một Border Layout :
  + NORTH: Là JLabel(“Fill out the form below to find vehicle”).
  + CENTER: Là một Jpanel sử dụng GridLayout(4, 4) gồm 8 JLabel và 8 JtextField.

#### Lớp ShowVehicleFound.



*Hình 3.1.1.3*

*Lớp ShowVehicleFound* được kế thừa từ *lớp JFrame* sử dụng BorderLayout là layout chính bao gồm :

* NORTH: Là JLabel (“VEHICLE FOUND”).
* SOUTH: Là 1 JPanel gồm 1 JLabel *status* và 2 JButton *export, close (như trong hình).*
* CENTER: Là 1 JTable được bao bởi JScrollPane gồm 10 cột ("ID", "Type", "Make", "Manufacturing Year", "Price", "Color", "Number of seats", "Type of engine", "Power", "Load") như trong hình *(tương tự JTable của lớp Car).*

#### Lớp Vehicle.

Ảnh có chứa văn bản

Mô tả được tạo tự động

*Hình 3.1.1.4*

*Lớp* *Vehicle* được kế thừa từ *lớp JFrame* và là lớp chính (chứa class main) của chương trình. Giao diện của *lớp Vehicle* sử dụng layout chính là BorderLayout bao gồm:

* NORTH: Là JLabel(“VEHICLE MANAGEMENT”).
* WEST: Là 1 JPanel bao gồm JLabel(“WELCOME!”) và 4 JButton Car, Bike, Truck, Find tương ứng với *btCar, btBike, btTruck, btFind.*
* CENTER: Là một Jpanel sử dụng CardLayout(15, 15) sử dụng để chuyển đổi giữa 4 JPanel *Car, Bike, Truck, Find (là 4 class kế thừa JPanel).*

### Kết nối và cách trình bày cơ sở dữ liệu.

#### Cơ sở dữ liệu và cách trình bày.

Project sử dụng hệ quản trị dữ liệu MySQL.

Cơ sở dữ liệu có 1 bảng *vehicle* bao gồm 12 cột:

Ảnh có chứa bàn

Mô tả được tạo tự động

*Hình 3.1.2.1*

* *id :* Là cột chính, lưu id cho mỗi dòng.
* *class\_id:* Lưu id của các xe trong từng loại xe (car, bike và truck).
* *switch:* Xác định sự tồn tại của hàng (1 là có, 0 là đã bị xóa).
* *type:* Lưu loại xe (car, bike hoặc truck).
* *make:* Lưu hãng sản xuất.
* *manufacturing\_year:* Lưu năm sản xuất.
* *price:* Lưu giá bán.
* *color:* Lưu màu xe.
* *number\_of\_seats:* Lưu số chỗ ngồi *(dành riêng cho Car).*
* *type\_of\_engine:* Lưu kiểu động cơ *(dành riêng cho Car).*
* *power:* Công suất *(dành riêng cho Bike).*
* *truck\_load:* Lưu trọng tải *(dành riêng cho Truck).*

#### Kết nối tới cơ sở dữ liệu.

Chương trình kết nối tới cơ sở dữ liệu qua hàm connect() như sau:

Connection con;

**private** **void** connect() {

// **TODO** Auto-generated method stub

String url = "jdbc:mysql://localhost:3306/vehicle";

String username = "root";

String password = "";

**try** {

con = DriverManager.*getConnection*(url, username, password);

} **catch** (Exception e) {

// **TODO**: handle exception

System.***out***.println("Connect to database fail");

e.printStackTrace();

}

}

### Nguyên lý hoạt động của chương trình.

#### Lớp Vehicle.

Lớp Vehicle có 4 JButton tương ứng với 4 lớp là Car, Bike, Truck và Find. Khi người dùng nhấn vào một trong 4 JButton thì chương trình sẽ thực hiện việc chuyển qua JPanel (JPanel ở trung tâm) tương ứng *(lớp Car, Bike, Truck và Find là 4 JPanel – 4 lớp kế thừa JPanel)* với cú pháp sau:

**public** **void** actionPerformed(ActionEvent arg0) {

// **TODO** Auto-generated method stub

btCar.setBackground(SystemColor.***activeCaption***);

btBike.setBackground(SystemColor.***activeCaption***);

btTruck.setBackground(SystemColor.***activeCaption***);

btFind.setBackground(SystemColor.***activeCaption***);

**if** (arg0.getSource() == btCar) {

card.show(cardVehicle, "car");

btCar.setBackground(SystemColor.***activeCaptionBorder***);

}

**else** **if** (arg0.getSource() == btBike) {

card.show(cardVehicle, "bike");

btBike.setBackground(SystemColor.***activeCaptionBorder***);

}

**else** **if** (arg0.getSource() == btTruck) {

card.show(cardVehicle, "truck");

btTruck.setBackground(SystemColor.***activeCaptionBorder***);

}

**else** **if** (arg0.getSource() == btFind) {

card.show(cardVehicle, "find");

btFind.setBackground(SystemColor.***activeCaptionBorder***);

}

}

*Ghi chú:*

* *carVehicle:* Là 1 JPanel sử dụng CardLayout(15, 15) được thêm vào 4 JPanel *car, bike, truck, find*

car = **new** Car();

bike = **new** Bike();

truck = **new** Truck();

find = **new** Find();

cardVehicle = **new** JPanel();

card = **new** CardLayout(15, 15);

cardVehicle.setLayout(card);

cardVehicle.add("car", car);

cardVehicle.add("bike", bike);

cardVehicle.add("truck", truck);

cardVehicle.add("find", find);

* *btCar, btBike, btTruck, btFind:* Là 4 JButton với sự kiện hiện khi người dùng click vào JButton, chương trình bắt sự kiện và thực hiện các bước:
  + Bước 1: Đổi màu 4 JButton về màu của JButton Bike *(hình 3.1.1.4)*.
  + Bước 2: Lật tới JPanel trên cardVehicle tương ứng với JButton tạo ra sự kiện và đổi màu JButton đấy thành màu của JButton Car *(hình 3.1.1.4).*

btCar.setBackground(SystemColor.***activeCaption***);

btBike.setBackground(SystemColor.***activeCaption***);

btTruck.setBackground(SystemColor.***activeCaption***);

btFind.setBackground(SystemColor.***activeCaption***);

**if** (arg0.getSource() == btCar) {

card.show(cardVehicle, "car");

btCar.setBackground(SystemColor.***activeCaptionBorder***);

}

**else** **if** (arg0.getSource() == btBike) {

card.show(cardVehicle, "bike");

btBike.setBackground(SystemColor.***activeCaptionBorder***);

}

**else** **if** (arg0.getSource() == btTruck) {

card.show(cardVehicle, "truck");

btTruck.setBackground(SystemColor.***activeCaptionBorder***);

}

**else** **if** (arg0.getSource() == btFind) {

card.show(cardVehicle, "find");

btFind.setBackground(SystemColor.***activeCaptionBorder***);

}

#### Lớp Car, Bike và Truck.

Vì nguyên lý hoạt động của 3 lớp Car, Bike và Truck tương tự nhau nên chúng em xin phép chỉ trình bày về nguyên lý hoạt động của *lớp Car.*

Trong *lớp Car* có 6 JtextField dùng để lấy thông tin được nhập từ người dùng, 1 bảng dùng để hiện thông tin các xe ô tô trong cơ sở dữ liệu và 5 JButton với những tác dụng khác nhau *(thêm, sửa, xóa thông tin xe; xóa thông tin ở các JtextField; xuất file Excel từ bảng).*

Khi *lớp Car* được gọi đến, việc đầu tiên đó là kết nối và đọc dữ liệu từ cơ sở dữ liệu đang có sẵn bằng hàm *readdb(),* dữ liệu đọc được sẽ hiển thị lên bảng trong *lớp Car.*

**private** **void** readdb() **throws** SQLException {

Statement statement = con.createStatement();

ResultSet rs = statement.executeQuery("SELECT \* FROM vehicle.vehicle");

**while** (rs.next()) {

String type = rs.getString("type");

**if** ((type.compareTo("car") == 0)) {

carId = rs.getInt("class\_id");

**if** (rs.getBoolean("switch")) {

carRow[0] = carId;

String make = rs.getString("make");

carRow[1] = make;

**int** my = rs.getInt("manufacturing\_year");

carRow[2] = my;

**double** price = rs.getDouble("price");

carRow[3] = price;

String color = rs.getString("color");

carRow[4] = color;

**int** nos = rs.getInt("number\_of\_seats");

carRow[5] = nos;

String toe = rs.getString("type\_of\_engine");

carRow[6] = toe;

carModel.addRow(carRow);

}

}

}

statement.close();

rs.close();

}

Sau đó, người dùng có thể sử dụng chương trình bằng cách nhập thông tin vào các JTextField và gọi các chức năng bằng cách nhấn chuột vào các nút.

*Nút clear* có chức năng xóa tất cả các ký tự trong 6 JTextField.

**private** **void** clear() {

carMake.setText("");

carManufacturingYear.setText("");

carPrice.setText("");

carColor.setText("");

carNumberOfSeats.setText("");

carTypeOfEngine.setText("");

carMake.requestFocus();

}

*Nút add* có chức năng thêm 1 xe *(đã được điền đầy đủ thông tin trong 6 JtextField)* vào bảng. Khi người dùng ấn *nút add*, chương trình thực hiện các bước:

* Bước 1: Kiểm tra xem người dùng đã điền đủ thông tin chưa, nếu chưa thì thông báo yêu cầu nhập đủ thông tin.

String noti = "Please fill all the form!";

**if** (carMake.getText().equals("")) {

carMake.requestFocus();

status.setText(noti);

}**else** **if** (carManufacturingYear.getText().equals("")){

carManufacturingYear.requestFocus();

status.setText(noti);

} **else** **if** (carPrice.getText().equals("")) {

carPrice.requestFocus();

status.setText(noti);

} **else** **if** (carColor.getText().equals("")) {

carColor.requestFocus();

status.setText(noti);

} **else** **if** (carNumberOfSeats.getText().equals("")) {

carNumberOfSeats.requestFocus();

status.setText(noti);

} **else** **if** (carTypeOfEngine.getText().equals("")) {

carTypeOfEngine.requestFocus();

status.setText(noti);

}

* Bước 2: Kiểm tra xem người dùng đã nhập đúng kiểu dữ liệu chưa, nếu đúng thì thêm thông tin vào bảng và cơ sở dữ liệu, nếu sai thì thông báo và yêu cầu nhập lại.

**try** {

carId++;

//add to database

PreparedStatement preparestatement = con.prepareStatement("UPDATE vehicle.vehicle SET make = ?, manufacturing\_year = ?, price = ?, color = ?, number\_of\_seats = ?, type\_of\_engine = ? WHERE class\_id = ? AND type = 'car'");

preparestatement.setString(1, carMake.getText());

preparestatement.setInt(2, Integer.*parseInt*(carManufacturingYear.getText()));

preparestatement.setDouble(3, Double.*parseDouble*(carPrice.getText()));

preparestatement.setString(4, carColor.getText());

preparestatement.setInt(5, Integer.*parseInt*(carNumberOfSeats.getText()));

preparestatement.setString(6, carTypeOfEngine.getText());

preparestatement.setInt(7, class\_id);

preparestatement.executeUpdate();

preparestatement.close();

//add to JTable

carRow[0] = carId;

carRow[1] = carMake.getText();

carRow[2] = carManufacturingYear.getText();

carRow[3] = carPrice.getText();

carRow[4] = carColor.getText();

carRow[5] = carNumberOfSeats.getText();

carRow[6] = carTypeOfEngine.getText();

carModel.addRow(carRow);

clear();

status.setText("Add sucessful!");

} **catch** (IllegalArgumentException e) {

carId--;

status.setText("Wrong data type!");

e.printStackTrace();

} **catch** (SQLException e) {

// **TODO** Auto-generated catch block

carId--;

status.setText("Add fail!");

e.printStackTrace();

}

*Nút update* có chức năng sửa thông tin của xe. Người dùng chọn thông tin của 1 xe ở bảng, sau đó thông tin của xe sẽ được điền vào 6 JTextField cho phép người dùng sửa, sau khi sửa xong người dùng ấn *nút update* để cập nhật lại thông tin. Chương trình sẽ thực hiện các bước sau khi *nút update* được ấn:

* Bước 1: Kiểm tra xem người dùng đã chọn hàng chứa thông tin xe chưa, nếu chưa thì thông báo

JOptionPane.*showMessageDialog*(**null**, "Please Select A Row First");

Nếu người dùng đã chọn thì làm bước 2.

* Bước 2: Kiểm tra xem người dùng đã điền đủ thông tin chưa, nếu chưa thì thông báo yêu cầu nhập đủ thông tin *(tương tự nút add).*
* Bước 3: Kiểm tra xem người dùng đã nhập đúng kiểu dữ liệu chưa, nếu đúng thì sửa thông tin ở cột được chọn của bảng và cơ sở dữ liệu, nếu sai thì thông báo và yêu cầu nhập lại.

**int** i = carTable.getSelectedRow();

**try** {

**int** class\_id = (**int**) carTable.getValueAt(i, 0);

PreparedStatement preparestatement = con.prepareStatement("UPDATE vehicle.vehicle SET make = ?, manufacturing\_year = ?, price = ?, color = ?, number\_of\_seats = ?, type\_of\_engine = ? WHERE class\_id = ? AND type = 'car'");

preparestatement.setString(1, carMake.getText());

preparestatement.setInt(2, Integer.*parseInt*(carManufacturingYear.getText()));

preparestatement.setDouble(3, Double.*parseDouble*(carPrice.getText()));

preparestatement.setString(4, carColor.getText());

preparestatement.setInt(5, Integer.*parseInt*(carNumberOfSeats.getText()));

preparestatement.setString(6, carTypeOfEngine.getText());

preparestatement.setInt(7, class\_id);

preparestatement.executeUpdate();

preparestatement.close();

carTable.setValueAt(carMake.getText(), i, 1);

carTable.setValueAt(carManufacturingYear.getText(), i, 2);

carTable.setValueAt(carPrice.getText(), i, 3);

carTable.setValueAt(carColor.getText(), i, 4);

carTable.setValueAt(carNumberOfSeats.getText(), i, 5);

carTable.setValueAt(carTypeOfEngine.getText(), i, 6);

clear();

carTable.removeRowSelectionInterval(i, i);

status.setText("Update sucessful!");

} **catch** (IllegalArgumentException e) {

status.setText("Wrong data type!");

e.printStackTrace();

} **catch** (Exception e) {

// **TODO**: handle exception

status.setText("Update fail!");

e.printStackTrace();

}

*Nút delete* sử dụng để xóa thông tin của 1 xe, hay là xóa 1 hàng trong bảng, đối với cơ sở dữ liệu thì đơn giản là chuyển biến switch của xe đó từ 1 thành 0.

**int** i = carTable.getSelectedRow();

**if** (i >= 0) {

**while** (i >= 0) {

**int** class\_id = (**int**) carTable.getValueAt(i, 0);

**try** {

PreparedStatement preparestatement = con.prepareStatement("UPDATE vehicle.vehicle SET switch = ? WHERE class\_id = ? AND type = 'car'");

preparestatement.setBoolean(1, **false**);

preparestatement.setInt(2, class\_id);

preparestatement.executeUpdate();

preparestatement.close();

carModel.removeRow(i);

i = carTable.getSelectedRow();

status.setText("Delete sucessful!");

} **catch** (SQLException e) {

// **TODO** Auto-generated catch block

status.setText("Delete fail!");

e.printStackTrace();

}

}

} **else**

JOptionPane.*showMessageDialog*(**null**, "Please Select A Row First");

Cuối cùng là *nút Export (Excel)* sử dụng để chuyển bảng thông tin thành file Excel bằng cách gọi đến lớp CreateExcel.

ResultSet rs = con.createStatement().executeQuery("SELECT class\_id, make, manufacturing\_year, price, color, number\_of\_seats, type\_of\_engine FROM vehicle.vehicle WHERE switch = 1 AND type = 'car'");

CreateExcel excel = **new** CreateExcel(rs, carCol, carGetRs, "Car");

status.setText("Created file: " + excel.getPath());

rs.close();

#### Lớp Find.

Trong *lớp Find* có 8 JtextField dùng để lấy thông tin được nhập từ người dùng dùng để tìm kiếm và 2 JButton là clear và find. *Nút clear* tương tự *nút clear* của *lớp Car.* Còn khi *nút find* được nhấn:

* Bước 1: Tạo String sql = “SELECT class\_id, type, make, manufacturing\_year, price, color, number\_of\_seats, type\_of\_engine, power, truck\_load FROM vehicle.vehicle WHERE switch = 1”
* Bước 2: Lấy giá trị của từng JTextField, kiểm tra xem đã đúng kiểu dữ liệu chưa, sau đó cộng thêm vào sql như ví dụ sau:

**if** (price.getText().equals("") == **false**) {

**try** {

**double** p = Double.*parseDouble*(price.getText());

sql += " AND price = " + price.getText();

} **catch** (Exception e) {

// **TODO**: handle exception

status.setText("Wrong data type at Price!");

k = **false**;

}

}

* Bước 3: Nếu người dùng đã nhập đúng kiểu dữ liệu thì thực hiện tìm kiếm, kết quả là 1 ResultSet rs được hiển thị qua *lớp ShowVehicleFound*.

**if** (k) {

rs = st.executeQuery(sql);

**new** ShowVehicleFound(rs);

}

#### Lớp ShowVehicleFound.

*Lớp ShowVehicleFound* hiển thị dữ liệu từ ResultSet từ lớp Find đẩy vào như sau:

**private** Object[] col = {"ID", "Type", "Make", "Manufacturing Year", "Price", "Color", "Number of seats", "Type of engine", "Power", "Load" };

**private** Object[] getRs = {"class\_id", "type", "make", "manufacturing\_year", "price", "color", "number\_of\_seats", "type\_of\_engine", "power", "truck\_load"};

**private** Object[] row = **new** Object[10];

**private** DefaultTableModel model;

**while** (rs.next()) {

**for** (**int** i=0; i<col.length; i++) {

String s = rs.getString(getRs[i].toString());

row[i] = s;

}

model.addRow(row);

}

#### Lớp CreateExcel.

*Lớp CreateExcel* sử dụng để xuất file Excel từ dữ liệu từ ResultSet. Khi gọi đến *lớp CreateExcel* chương trình truyền vào các tham số ResultSet rs, Object []col, Object []getRs, String excelName

* ResultSet rs: Là dữ liệu sử dụng để tạo file Excel.
* Object []col: Là một mảng chứa tên các cột của bảng Excel.
* Object []getRs: Là mảng chứa tên của cột trong cơ sở dữ liệu tương ứng với tên cột của mảng []col, sử dụng để lấy dữ liệu ở ResultSet.
* String excelName: Là tên của file Excel và Sheet.

Đầu tiên là hàm tạo kiểu cho hàng đầu tiền (tên các cột)

**private** CellStyle createStyleForTitle(HSSFWorkbook workbook) {

Font font = workbook.createFont();

font.setBold(**true**);

CellStyle style = workbook.createCellStyle();

style.setFont(font);

**return** style;

}

Tạo bảng excel:

**int** rowNum = 0, colNum = col.length;

CellType setDataType[] = **new** CellType[colNum];

CellStyle style = createStyleForTitle(workbook);

row = sheet.createRow(rowNum);

//Write title row and set data type

**for** (**int** i=0; i<colNum; i++) {

cell = row.createCell(i, CellType.***STRING***);

cell.setCellValue(col[i].toString());

cell.setCellStyle(style);

String s = getRs[i].toString();

**if** (s.equals("class\_id") || s.equals("manufacturing\_year") || s.equals("price") || s.equals("number\_of\_seats") || s.equals("power") || s.equals("truck\_load")) setDataType[i] = CellType.***NUMERIC***;

**else** setDataType[i] = CellType.***STRING***;

}

// Write data to Excel

**while** (rs.next()) {

rowNum++;

row = sheet.createRow(rowNum);

**for** (**int** i=0; i<colNum; i++) {

cell = row.createCell(i, setDataType[i]);

cell.setCellValue(rs.getString(getRs[i].toString()));

}

## Kết quả thử nghiệm

Chương trình qua nhiều lần kiểm thử các chức năng đã được sửa lỗi và chạy ổn định, thực hiện đúng với yêu cầu đã đặt ra. Tuy nhiên vẫn còn một số hạn chế như chưa giải phóng hết các Connection, Statement, ResultSet sau khi kết thúc chương trình, chưa kết cấu cơ sở dữ liệu một cách khoa học.

## Kết quả chương trình

Đã được mô tả trong phần [*3.1.1. Giao diện đồ họa.*](#_Giao_diện_đồ)

# KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

* Ưu điểm:
  + Chương trình đã thực hiện được các yêu cầu đặt ra, ngoài ra còn thêm một số tính năng khác như xuất file Excel.
  + Bố cục đồ họa đẹp, đơn giản và dễ sử dụng.
* Nhược điểm:
  + Chưa giải phóng được bộ nhớ cho các Connection, Statement, ResultSet sau khi kết thúc chương trình.
  + Chưa có file .exe để dễ dàng cho người sử dụng.
  + Kết cấu cơ sở dữ liệu chưa hợp lý.
* Hướng phát triển cho đề tài, cho sản phẩm và khả năng ứng dụng:
  + Bổ sung thêm các tính năng khác như cửa sổ đăng nhập, tính năng undo, redo.
  + Bổ sung phím tắt cho các tính năng. (Ctrl + Z = Undo; Enter = Add;…).
  + Bố cục lại dữ liệu theo hướng đối tượng (4 thuộc tính chung).

TÀI LIỆU THAM KHẢO

1. Hướng dẫn kết nối Java với MySQL [*giasutinhoc.vn/huong-dan-thuc-hanh/huong-dan-java/huong-dan-ket-noi-java-voi-mysql-2/*](https://giasutinhoc.vn/huong-dan-thuc-hanh/huong-dan-java/huong-dan-ket-noi-java-voi-mysql-2/)
2. Đọc Ghi file excel trong Java sử dụng Apache POI [*o7planning.org/vi/11259/doc-ghi-file-excel-trong-java-su-dung-apache-poi*](https://o7planning.org/vi/11259/doc-ghi-file-excel-trong-java-su-dung-apache-poi)
3. Các câu hỏi và trả lời trên [stackoverflow.com/](https://stackoverflow.com/)

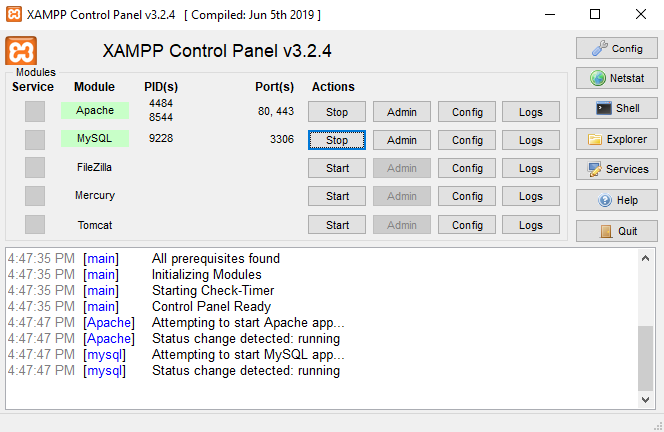
PHỤ LỤC

* + 1. Tải và cài đặt môi trường

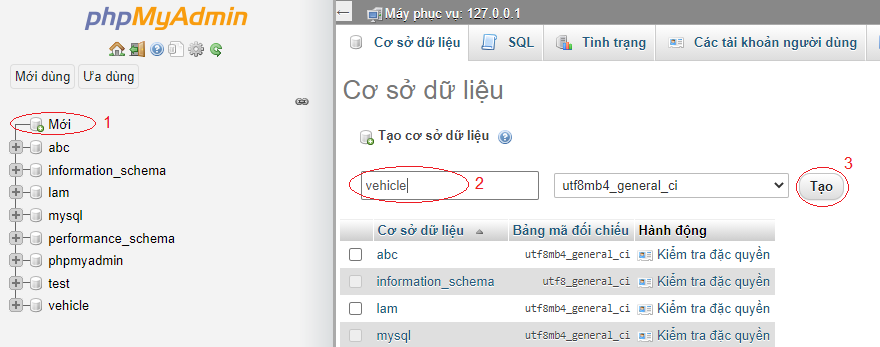
Bước 1: Tải các ứng dụng tạo môi trường.

* Tải JDK phù hợp với hệ điều hành tại trang chủ của Oracle <https://www.oracle.com/java/technologies/javase/javase-jdk8-downloads.html>.
* Tải và cài đặt Eclipse IDE theo hướng dẫn sau: <https://o7planning.org/vi/10379/huong-dan-cai-dat-va-cau-hinh-eclipse>
* Tải và cài đặt XAMPP tại <https://www.apachefriends.org/index.html>

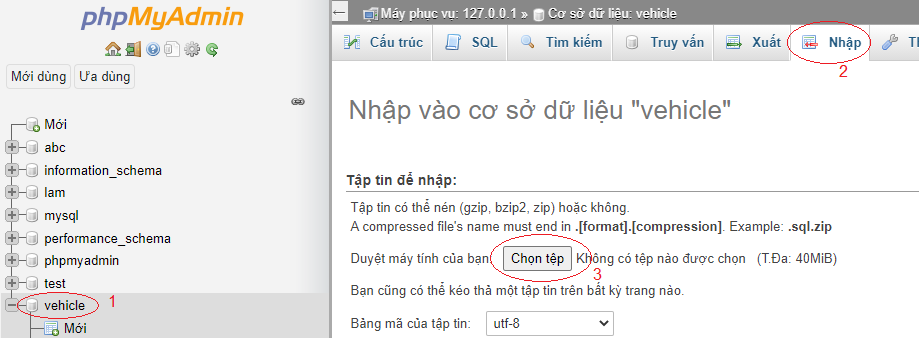
Bước 2: Bật XAMPP, sau đó nhấn vào nút Start tại Apache với MySQL. Nếu Apache và MySQL hiện màu xanh như trong hình là thành công



Bước 3: Bật ứng dụng duyệt web (Chrome, Cốc cốc,…), truy cập vào đường dẫn localhost/phpmyadmin

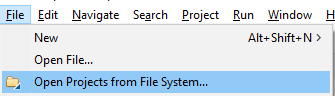
* Tại giao diện phpmyadmin, ta làm các bước sau:

Lưu ý: Đặt tên cơ sở dữ liệu là ***“vehicle”****.*

* Nhấn vào cơ sở dữ liệu vừa tạo, làm các bước như ảnh. Tại phần chọn tệp, ta chọn đường dẫn tới tệp ***“vehicle.sql”****,* sau đó kéo xuống và chọn *thực hiện.*

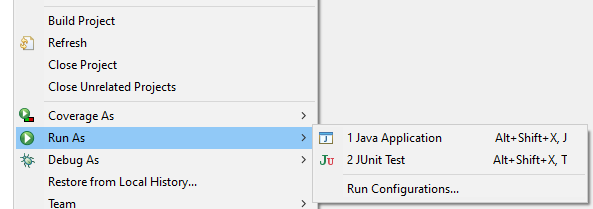
Bước 4: Giải nén file vừa tải về, bật Eclipse

* Chọn File, sau đó chọn Open Projects from File System

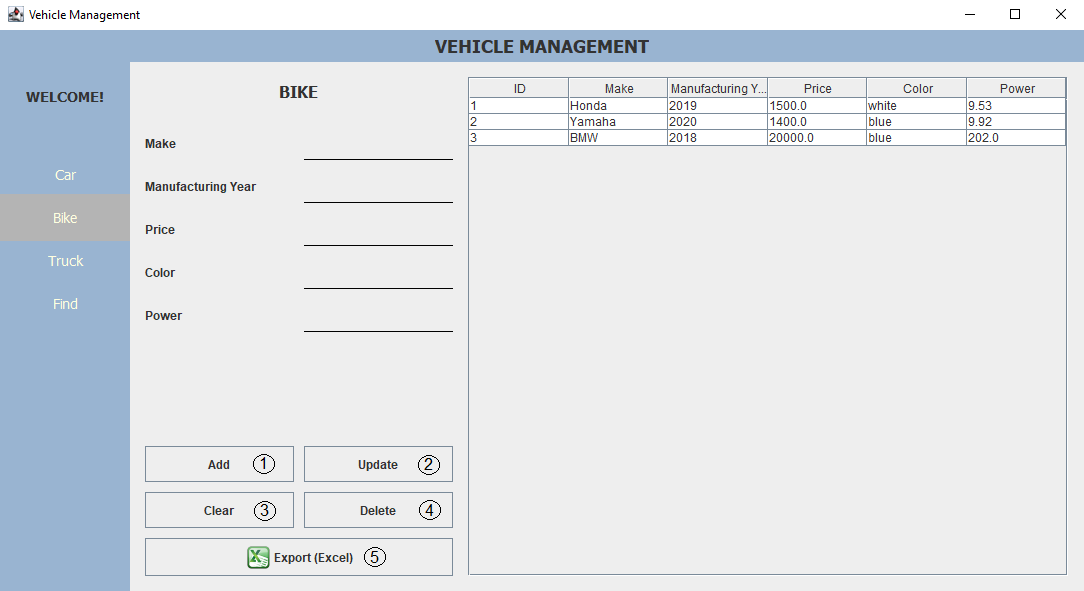


* Cửa sổ Import Projects from File System or Archive hiện ra, chọn Directory sau đó chọn đường dẫn đến file vừa giải nén rồi ấn Select Folder. Nhấn Finish để thêm project.

Bước 5: Tại thanh Project Explorer ta thấy xuất hiện project có tên *“miniproject-detai7-master”,* chuột phải vào project, chọn Run As sau đó chọn Java Application.



* + 1. Hướng dẫn sử dụng chương trình.



Để thêm, sửa, xóa, xuất file Excel thông tin của xe, bạn chọn ở thanh bên trái loại xe mà mình muốn tác động. Sau đó ấn nút tương ứng với hành động mình muốn làm:

* Nút Add (1): Thêm dữ liệu đã nhập vào bảng
* Nút Update (2): Chọn 1 hàng trong bảng, dữ liệu của hàng sẽ được điền vào các ô chữ để người dùng thay đổi. Sau khi thay đổi xong thì ấn nút *Update* để cập nhật lại vào bảng.
* Nút Clear (3): Dùng để xóa hết các chữ trong các ô chữ.
* Nút Delete (4): Dùng để xóa thông tin các xe trong bảng, yêu cầu chọn hàng trước.
* Nút Export (Excel) (5): Để xuất file Excel từ bảng.

Để tìm kiếm xe, người dùng điền thông tin vào các trường thông tin (có thể để trống) rồi ấn nút Find.