Table of Contents

Main Report

1. Description of Proposed System

Proposed system: Car Inventory Management System

Introduction

A local car dealership is planning to integrate an electronic-based car record management system into their operations. Specifically, they want to use this record management system to efficiently manage their car inventory and help keep track of purchases and deals of cars that are made with their customers. The system that they require must be able to record all relevant information of the cars in their inventory, including plate number, car model, car brand, car type, colour, status, and price. The system must be able to add a new car record into the system, delete or edit an existing record in the system, search and print the details of a specific car record in the system, and print the details of all car records in the system.

Problem Statement

The car dealers currently use a paper-based system to manually manage car records. They have found their paper-based system to be inefficient and time-consuming as it can be a tedious and lengthy process just to manage car records. Additionally, the paper-based system also increases their expenses as it adds to their paper and stationery costs. The car dealers have also found that their current system is sometimes unsafe for keeping records as paper is easily susceptible to environment damage, such as water leakage or spilled drinks. Because of these problems, they decided that they need an electronic-based system to manage their inventory with minimal costs, time, and effort so that they can focus and improve on more important operations.

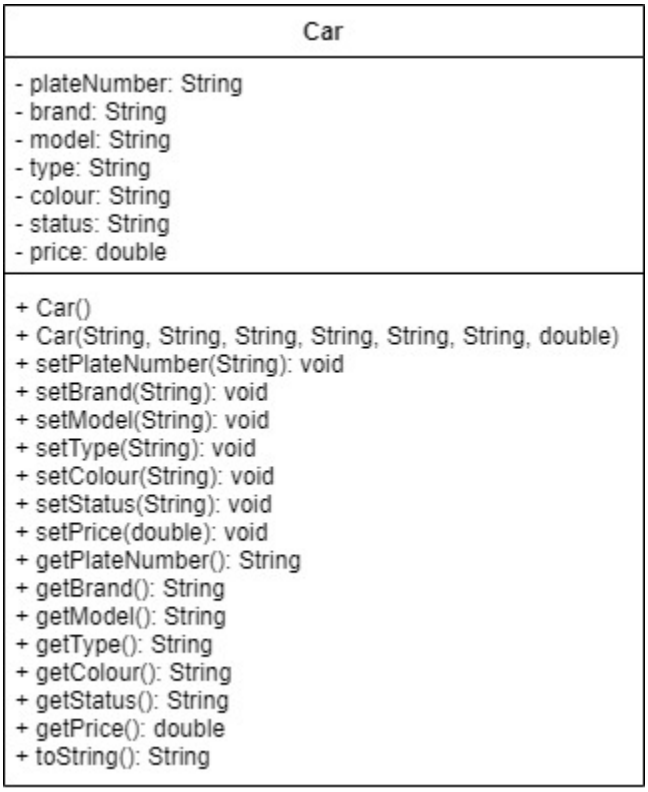
Limitations

The system has a few limitations that may negatively affect the users' tasks and workflow. Firstly, the system has limited space for storing records, as it can only store up to 100 car records at a time. The next limitation is the users will not be able to manually change the username and password for their account. If they want to, they will have to contact the developers to make the changes for them.

Conclusion

In conclusion, despite the minor limitations, an electronic-based car inventory management system would be highly beneficial for them as managing records will require less time and effort, making it more efficient. It will also reduce their inventory management expenses. Furthermore, the electronic-based system would also have error prevention features that would help detect and prevent certain human errors that may sometimes be overlooked when they use the manual paper-based system. Overall, a car inventory management system that is simple enough to quickly let users get used to and enable users to perform required task should be able to increase work efficiency in the establishment.

2. Class Diagram



3. Complete Program

- **Car.java**

```
public class Car {
    private String plateNumber;
    private String brand;
    private String model;
    private String type;
    private String colour;
    private String status;
    private double price;

    public Car() {
        plateNumber = "";
        brand = "";
        model = "";
        type = "";
        colour = "";
        status = "";
        price = 0.00;
        price = 0;
    }

    public Car(String pn, String br, String mo, String ty, String co, String st,
double pr) {
        plateNumber = pn;
        brand = br;
        model = mo;
        type = ty;
        colour = co;
        status = st;
        price = pr;
    }

    public void setPlateNumber(String pn) {
        plateNumber = pn;
    }

    public void setBrand(String br) {
        brand = br;
    }

    public void setModel(String mo) {
        model = mo;
```

```
    }

    public void setType(String ty) {
        type = ty;
    }

    public void setColour(String co) {
        colour = co;
    }

    public void setStatus(String st) {
        status = st;
    }

    public void setPrice(double pr) {
        price = pr;
    }

    public String getPlateNumber() {
        return plateNumber;
    }

    public String getBrand() {
        return brand;
    }

    public String getModel() {
        return model;
    }

    public String getType() {
        return type;
    }

    public String getColour() {
        return colour;
    }

    public String getStatus() {
        return status;
    }

    public double getPrice() {
        return price;
    }
```

```
    public String toString() {
        return "Plate number: " + plateNumber
                + "\nBrand: " + brand
                + "\nModel: " + model
                + "\nType: " + type
                + "\nColour: " + colour
                + "\nStatus: " + status
                + "\nPrice: " + price;
    }
}
```

- **Login.java**

```
import java.util.Scanner;

public class Login {
    public Login() {
        printWelcome();
        login();
    }

    public static void printWelcome() {
        int i, len;
        String name= "WELCOME";
        len = name.length();
        name = name.toUpperCase();
        System.out.println();
        for (i = 0; i < len; i++) {
            if (name.charAt(i) == 'A' || name.charAt(i) == 'B' || name.charAt(i)
== 'C' || name.charAt(i) == 'E' || name.charAt(i) == 'F' || name.charAt(i) == 'G'
 || name.charAt(i) == 'I' || name.charAt(i) == 'J' || name.charAt(i) == 'O' || na
me.charAt(i) == 'P' || name.charAt(i) == 'Q' || name.charAt(i) == 'R' || name.cha
rAt(i) == 'S' || name.charAt(i) == 'T' || name.charAt(i) == 'Z') {
                System.out.print("\t\t   *****");
            } else if (name.charAt(i) == 'H' || name.charAt(i) == 'K' || name.cha
rAt(i) == 'M' || name.charAt(i) == 'N' || name.charAt(i) == 'U' || name.charAt(i)
 == 'V' || name.charAt(i) == 'W' || name.charAt(i) == 'X' || name.charAt(i) == 'Y
') {
                System.out.print("\t\t   *   *");
            } else if (name.charAt(i) == 'D') {
                System.out.print("\t\t       *");
            } else if (name.charAt(i) == 'L') {
                System.out.print("\t\t   *    ");
```

```
            }
        }
        System.out.print("\n");
        for (i = 0; i < len; i++) {
            if (name.charAt(i) == 'A' || name.charAt(i) == 'B' || name.charAt(i)
== 'G' || name.charAt(i) == 'H' || name.charAt(i) == 'O' || name.charAt(i) == 'P'
 || name.charAt(i) == 'Q' || name.charAt(i) == 'R' || name.charAt(i) == 'U' || na
me.charAt(i) == 'V' || name.charAt(i) == 'W') {
                System.out.print("\t\t   *   *");
            } else if (name.charAt(i) == 'C' || name.charAt(i) == 'E' || name.cha
rAt(i) == 'F' || name.charAt(i) == 'L' || name.charAt(i) == 'S') {
                System.out.print("\t\t   *    ");
            }
            if (name.charAt(i) == 'M') {
                System.out.print("\t\t   ** **");
            } else if (name.charAt(i) == 'Y') {
                System.out.print("\t\t    * * ");
            } else if (name.charAt(i) == 'K') {
                System.out.print("\t\t   *  * ");
            } else if (name.charAt(i) == 'I' || name.charAt(i) == 'J' || name.cha
rAt(i) == 'T') {
                System.out.print("\t\t     *  ");
            } else if (name.charAt(i) == 'D') {
                System.out.print("\t\t      *");
            } else if (name.charAt(i) == 'N') {
                System.out.print("\t\t   **  *");
            } else if (name.charAt(i) == 'Z') {
                System.out.print("\t\t     * ");
            } else if (name.charAt(i) == 'X') {
                System.out.print("\t\t       ");
            }
        }
        System.out.print("\n");
        for (i = 0; i < len; i++) {
            if (name.charAt(i) == 'A' || name.charAt(i) == 'B' || name.charAt(i)
== 'G' || name.charAt(i) == 'H' || name.charAt(i) == 'N' || name.charAt(i) == 'O'
 || name.charAt(i) == 'P' || name.charAt(i) == 'Q' || name.charAt(i) == 'R' || na
me.charAt(i) == 'U' || name.charAt(i) == 'V' || name.charAt(i) == 'W') {
                System.out.print("\t\t   *   *");
            } else if (name.charAt(i) == 'C' || name.charAt(i) == 'E' || name.cha
rAt(i) == 'F' || name.charAt(i) == 'L' || name.charAt(i) == 'S') {
                System.out.print("\t\t   *    ");
            }
            if (name.charAt(i) == 'M') {
                System.out.print("\t\t   * * *");
```

```
            } else if (name.charAt(i) == 'Y') {
                System.out.print("\t\t     *  ");
            } else if (name.charAt(i) == 'K') {
                System.out.print("\t\t   * *  ");
            } else if (name.charAt(i) == 'I' || name.charAt(i) == 'J' || name.cha
rAt(i) == 'T') {
                System.out.print("\t\t     *  ");
            } else if (name.charAt(i) == 'D') {
                System.out.print("\t\t      *");
            } else if (name.charAt(i) == 'X') {
                System.out.print("\t\t    * * ");
            } else if (name.charAt(i) == 'Z') {
                System.out.print("\t\t       ");
            }
        }
        System.out.print("\n");
        for (i = 0; i < len; i++) {
            if (name.charAt(i) == 'A' || name.charAt(i) == 'B' || name.charAt(i)
== 'D' || name.charAt(i) == 'E' || name.charAt(i) == 'F' || name.charAt(i) == 'G'
 || name.charAt(i) == 'H' || name.charAt(i) == 'P' || name.charAt(i) == 'Q' || na
me.charAt(i) == 'R' || name.charAt(i) == 'S') {
                System.out.print("\t\t   *****");
            } else if (name.charAt(i) == 'C') {
                System.out.print("\t\t   *    ");
            }
            if (name.charAt(i) == 'M' || name.charAt(i) == 'O' || name.charAt(i)
== 'V' || name.charAt(i) == 'W' || name.charAt(i) == 'U') {
                System.out.print("\t\t   *   *");
            } else if (name.charAt(i) == 'Z') {
                System.out.print("\t\t     *  ");
            } else if (name.charAt(i) == 'K') {
                System.out.print("\t\t   **   ");
            } else if (name.charAt(i) == 'I' || name.charAt(i) == 'J' || name.cha
rAt(i) == 'T' || name.charAt(i) == 'Y') {
                System.out.print("\t\t     *  ");
            } else if (name.charAt(i) == 'L') {
                System.out.print("\t\t   *    ");
            } else if (name.charAt(i) == 'N') {
                System.out.print("\t\t   * * *");
            } else if (name.charAt(i) == 'X') {
                System.out.print("\t\t     *  ");
            }
        }
        System.out.print("\n");
        for (i = 0; i < len; i++) {
```

```
            if (name.charAt(i) == 'A' || name.charAt(i) == 'B' || name.charAt(i)
== 'D' || name.charAt(i) == 'H' || name.charAt(i) == 'M' || name.charAt(i) == 'N'
 || name.charAt(i) == 'O' || name.charAt(i) == 'U' || name.charAt(i) == 'V') {
                System.out.print("\t\t   *   *");
            } else if (name.charAt(i) == 'C' || name.charAt(i) == 'E' || name.cha
rAt(i) == 'F' || name.charAt(i) == 'L' || name.charAt(i) == 'P') {
                System.out.print("\t\t   *    ");
            }
            if (name.charAt(i) == 'G' || name.charAt(i) == 'Q' || name.charAt(i)
== 'S') {
                System.out.print("\t\t      *");
            } else if (name.charAt(i) == 'K') {
                System.out.print("\t\t   * *  ");
            } else if (name.charAt(i) == 'I' || name.charAt(i) == 'J' || name.cha
rAt(i) == 'T' || name.charAt(i) == 'Y') {
                System.out.print("\t\t     *  ");
            } else if (name.charAt(i) == 'R') {
                System.out.print("\t\t   * *  ");
            } else if (name.charAt(i) == 'W') {
                System.out.print("\t\t   * * *");
            } else if (name.charAt(i) == 'X') {
                System.out.print("\t\t    * * ");
            } else if (name.charAt(i) == 'Z') {
                System.out.print("\t\t       ");
            }
        }
        System.out.print("\n");
        for (i = 0; i < len; i++) {
            if (name.charAt(i) == 'A' || name.charAt(i) == 'B' || name.charAt(i)
== 'D' || name.charAt(i) == 'H' || name.charAt(i) == 'M' || name.charAt(i) == 'O'
 || name.charAt(i) == 'U' || name.charAt(i) == 'V') {
                System.out.print("\t\t   *   *");
            } else if (name.charAt(i) == 'C' || name.charAt(i) == 'E' || name.cha
rAt(i) == 'F' || name.charAt(i) == 'L' || name.charAt(i) == 'P') {
                System.out.print("\t\t   *    ");
            }
            if (name.charAt(i) == 'G' || name.charAt(i) == 'Q' || name.charAt(i)
== 'S') {
                System.out.print("\t\t      *");
            } else if (name.charAt(i) == 'K') {
                System.out.print("\t\t   *  * ");
            } else if (name.charAt(i) == 'I' || name.charAt(i) == 'T' || name.cha
rAt(i) == 'Y') {
                System.out.print("\t\t     *  ");
            } else if (name.charAt(i) == 'R') {
```

```
                System.out.print("\t\t   *  * ");
            } else if (name.charAt(i) == 'W') {
                System.out.print("\t\t   ** **");
            } else if (name.charAt(i) == 'J') {
                System.out.print("\t\t   * *  ");
            } else if (name.charAt(i) == 'N') {
                System.out.print("\t\t   *  **");
            } else if (name.charAt(i) == 'Z') {
                System.out.print("\t\t    *    ");
            } else if (name.charAt(i) == 'X') {
                System.out.print("\t\t        ");
            }
        }
        System.out.print("\n");
        for (i = 0; i < len; i++) {
            if (name.charAt(i) == 'B' || name.charAt(i) == 'C' || name.charAt(i)
== 'D' || name.charAt(i) == 'E' || name.charAt(i) == 'G' || name.charAt(i) == 'I'
 || name.charAt(i) == 'O' || name.charAt(i) == 'L' || name.charAt(i) == 'S' || na
me.charAt(i) == 'U' || name.charAt(i) == 'Z') {
                System.out.print("\t\t   *****");
            } else if (name.charAt(i) == 'A' || name.charAt(i) == 'H' || name.cha
rAt(i) == 'M' || name.charAt(i) == 'N') {
                System.out.print("\t\t   *   *");
            }
            if (name.charAt(i) == 'F' || name.charAt(i) == 'P') {
                System.out.print("\t\t   *    ");
            } else if (name.charAt(i) == 'J') {
                System.out.print("\t\t   ***  ");
            } else if (name.charAt(i) == 'K' || name.charAt(i) == 'R') {
                System.out.print("\t\t   *   *");
            } else if (name.charAt(i) == 'Q') {
                System.out.print("\t\t      *");
            } else if (name.charAt(i) == 'T' || name.charAt(i) == 'V' || name.cha
rAt(i) == 'Y') {
                System.out.print("\t\t    *  ");
            } else if (name.charAt(i) == 'W' || name.charAt(i) == 'X') {
                System.out.print("\t\t   *   *");
            }
        }
        System.out.println("\n");
    }

    public static void login() {
        String username = "HelloWorld";
        String password = "123";
```

```
        boolean isGranted = false;

        System.out.println("Please login to continue.");
        for (int i = 1; i <= 3; i++) {
            Scanner input1 = new Scanner(System.in);
            System.out.print(String.format("%50s", "=").replace(' ', '=') + "\n")
;
            System.out.print("Enter Username : ");
            String usernameInput = input1.nextLine();

            System.out.print("Enter Password : ");
            String passwordInput = input1.nextLine();
            System.out.print(String.format("%50s", "=").replace(' ', '=') + "\n")
;

            if (usernameInput.compareTo(username)==0 && passwordInput.compareTo(p
assword)==0)
                isGranted = true;

            if (usernameInput.compareTo(username) != 0)
                System.out.println("Invalid Username!");

            if (passwordInput.compareTo(password) != 0)
                System.out.println("Invalid Password!");

            System.out.println();

            if (!isGranted) {
                if (i == 3) {
                    System.out.println("* Access Denied! You have reached the max
imum number of attempts. *");
                    System.exit(0);
                }else{
                    System.out.print("* YOU ONLY HAVE " + (3 - i) + " ATTEMPT");
                    if ((3 - i) > 1)
                        System.out.println("S LEFT! *");
                    else
                        System.out.println(" LEFT! *");
                }
            } else {
                System.out.println("Access granted, you have successfully logged
in.");
                break;
            }
        }
```

```
    }
}
```

- **Main.java**

```
import java.util.Scanner;
import java.util.ArrayList;
import java.util.InputMismatchException;

public class Main {
    // Main class scope variables
    private static final int MAX_SIZE = 100; // for setting record limit

    public static void main(String[] args) {
        // Menu variables
        ArrayList<Car> carArrayList = new ArrayList<Car>();
        int menuInput = 0;
        char exitInput;
        boolean hasError = false, isEnded = false;
        Scanner scanner = new Scanner(System.in);

        // Prompt user to log in
        Login login = new Login();
        System.out.println("\nWelcome to Car Inventory Management System.");

        // Menu loop
        do {
            // Print menu header
            System.out.println();
            System.out.print(String.format("%44s", "=\n").replace(' ', '='));
            System.out.print(String.format("%24s", "MENU\n"));
            System.out.print(String.format("%44s", "=\n").replace(' ', '='));
            // Print menu options
            System.out.print(String.format("%12s", "") + "1. Add record\n");
            System.out.print(String.format("%12s", "") + "2. Delete record\n");
            System.out.print(String.format("%12s", "") + "3. Edit record\n");
            System.out.print(String.format("%12s", "") + "4. Search record\n");
            System.out.print(String.format("%12s", "") + "5. Display all records\
n");
            System.out.print(String.format("%12s", "") + "6. Exit\n\n");
```

```
            // Get user's selection input
            do {
                // Catch InputMismatchException (invalid data type input)
                try {
                    System.out.print("Selection: ");
                    menuInput = scanner.nextInt();
                    if (menuInput < 1 || menuInput > 6)
                        System.out.println("Invalid input, please choose from 1 -
 6.");
                    else
                        hasError = false;
                } catch (InputMismatchException e) {
                    hasError = true;
                    System.out.println("Invalid input, please choose from 1 - 6."
);
                    scanner.nextLine();
                }
            } while (menuInput < 1 || menuInput > 6 || hasError);
            System.out.println();

            // Process input, redirect user to selected function
            if (menuInput == 1) {
                createRecord(carArrayList);
            } else if (menuInput == 2) {
                deleteRecord(carArrayList);
            } else if (menuInput == 3) {
                editRecord(carArrayList);
            } else if (menuInput == 4) {
                searchRecord(carArrayList);
            } else if (menuInput == 5) {
                displayRecords(carArrayList);
            } else if (menuInput == 6) {
                // Exit confirmation
                do {
                    System.out.print("Are you sure? (Y/N): ");
                    exitInput = Character.toUpperCase(scanner.next().charAt(0));
                    if (exitInput != 'Y' && exitInput != 'N')
                        System.out.println("Invalid input, please enter 'Y' or 'N
'.");
                } while (exitInput != 'Y' && exitInput != 'N');
                // Exit if 'Y', return to menu if 'N'
                if (exitInput == 'Y')
                    isEnded = true;
                else if (exitInput == 'N')
                    isEnded = false;
```

```
            }
        } while (!isEnded);
    }

    public static void createRecord(ArrayList<Car> cars) {
        // Variable declarations
        String plateNumber, brand, model, type, colour, status;
        double price = 0;
        boolean isUnique = true, hasError = false;
        // Instantiate Scanner object
        Scanner scanner = new Scanner(System.in);

        // If Car array is full, cancel, else continue
        if (cars.size() >= MAX_SIZE) {
            System.out.println("* * * Maximum amount of records reached, unable t
o add new record. * * *");
        } else {
            // Prompt user for input
            System.out.println("Please enter details of the car.");
            System.out.print(String.format("%50s", "-\n").replace(' ', '-'));
            // Ensure plateNumber is unique
            do {
                plateNumber = getString("Plate Number", 12);
                // Cancel checking if there are no existing records yet
                if (cars.size() == 0)
                    isUnique = true;
                // Check if plateNumber already exists
                for (int i = 0; i < cars.size(); i++) {
                    if (plateNumber.equals(cars.get(i).getPlateNumber())) {
                        System.out.println("Invalid input, there is an existing c
ar record with this plate number.");
                        isUnique = false;
                        break;
                    }
                    // Set isUnique to true if input passes the checking
                    if (i == cars.size() - 1)
                        isUnique = true;
                }
            } while (!isUnique);

            brand = getString("Brand", 16);
            model = getString("Model", 16);
            type = getString("Type", 16);
            colour = getString("Colour", 16);
            status = getString("Status", 16);
```

```
            do {
                // Catch InputMismatchException (invalid data type input)
                try {
                    System.out.print("Price: ");
                    price = scanner.nextDouble();
                    if (price < 0)
                        System.out.println("Invalid input, Price cannot be negati
ve.");
                    else
                        hasError = false;
                } catch (InputMismatchException e) {
                    hasError = true;
                    System.out.println("Invalid input, please enter numbers only.
");
                    scanner.nextLine(); // next() discards the token(input)
                }
            } while (price < 0 || hasError);

            // Prompt user to confirm new record
            char confInput;
            System.out.println();
            do {
                System.out.print("Confirm new Car record? (Y/N): ");
                confInput = Character.toUpperCase(scanner.next().charAt(0));
                if (confInput != 'Y' && confInput != 'N')
                    System.out.println("Invalid input, please enter 'Y' or 'N'.")
;
            } while (confInput != 'Y' && confInput != 'N');

            // Add car record into array if Y, cancel if N
            if (confInput == 'Y') {
                Car carObj = new Car(plateNumber, brand, model, type, colour, sta
tus, price);
                cars.add(carObj);
                System.out.println("\n+ + + New Car record has been added. + + +"
);
                // Notify user if Car array is full
                if (cars.size() == MAX_SIZE)
                    System.out.println("\n* * * MAXIMUM AMOUNT OF RECORDS HAS BEE
N REACHED * * *");
            } else if (confInput == 'N') {
                System.out.println("\n* * * Process cancelled, returning to menu.
 * * *");
            }
```

```
        }
    }

    public static void deleteRecord(ArrayList<Car> cars) {
        // Variable declarations
        String plateNumber;
        char delInput;
        int delIndex = 0;
        boolean isFound = false;
        // Instantiate Scanner object
        Scanner scanner = new Scanner(System.in);

        // Check if car array is empty
        if (cars.size() == 0) {
            System.out.println("There are no existing car records in the system f
or you to delete.");
        } else {
            // Get plateNumber
            System.out.println("Please enter the Plate Number of the car you want
 to delete.");
            System.out.print(String.format("%70s", "-\n").replace(' ', '-'));
            plateNumber = getString("Plate Number", 12);
            // Check for car with matching plateNumber
            for (int i = 0; i < cars.size(); i++) {
                if (cars.get(i).getPlateNumber().compareToIgnoreCase(plateNumber)
 == 0) {
                    System.out.println("\nCar with Plate Number " + "\"" + plateN
umber + "\"" + " found." +
                            " Displaying its details: ");
                    System.out.print(String.format("%70s", "-\n").replace(' ', '-
'));
                    System.out.print(String.format("%-
30s", "Brand: " + cars.get(i).getBrand()));
                    System.out.print(String.format("%-
30s", "Model: " + cars.get(i).getModel()));
                    System.out.println();
                    System.out.print(String.format("%-
30s", "Type: " + cars.get(i).getType()));
                    System.out.print(String.format("%-
30s", "Colour: " + cars.get(i).getColour()));
                    System.out.println();
                    System.out.print(String.format("%-
30s", "Status: " + cars.get(i).getStatus()));
                    System.out.printf("Price: %.2f\n", cars.get(i).getPrice());
                    delIndex = i;
```

```
                    isFound = true;
                    break;
                }
            }

            if (isFound) {
                // Ask user if they want to delete the chosen car record
                do {
                    System.out.print("\nAre you sure you want to delete this car
record? (Y/N): ");
                    delInput = Character.toUpperCase(scanner.next().charAt(0));
                    if (delInput != 'Y' && delInput != 'N')
                        System.out.println("Invalid input, please enter 'Y' or 'N
'.");
                } while (delInput != 'Y' && delInput != 'N');
                // Delete if 'Y', cancel if 'N'
                if (delInput == 'Y') {
                    cars.remove(delIndex);
                    System.out.println("\n- - - The car record of " + plateNumber
 + " has been deleted. - - -");
                } else if (delInput == 'N') {
                    System.out.println("\n* * * Process cancelled, returning to m
enu. * * *");
                }
            } else {
                System.out.println("\nNo such car record found, returning to menu
.");
            }
        }
    }

    public static void editRecord(ArrayList<Car> cars) {
        // Variable declarations
        String plateNumber, brand, model, type, colour, status;
        double price = 0;
        int editChoice = 0, matchedIndex = 0;
        boolean isFound = false;
        // Instantiate Scanner object
        Scanner scanner = new Scanner(System.in);

        // Check if Car array is empty
        if (cars.size() == 0) {
            System.out.println("There are no existing car records in the system f
or you to edit.");
        } else {
```

```
            // Get plateNumber input
            System.out.println("Please enter the Plate Number of the car you want
 to edit.");
            System.out.print(String.format("%70s", "-\n").replace(' ', '-'));
            plateNumber = getString("Plate Number", 12);
            // Check for car with matching plateNumber
            for (int i = 0; i < cars.size(); i++) {
                if (cars.get(i).getPlateNumber().compareToIgnoreCase(plateNumber)
 == 0) {
                    matchedIndex = i;
                    isFound = true;
                }
            }
            // If matching plateNumber is found, allow user to edit the car recor
d
            if (!isFound) {
                System.out.println("\nNo such car record found, returning to menu
.");
            } else {
                // Display current details of the car record
                System.out.println("\nCar with Plate Number " + "\"" + plateNumbe
r + "\"" + " found." +
                                                " Displaying its details: ");
                System.out.print(String.format("%70s", "-\n").replace(' ', '-'));
                System.out.print(String.format("%-
30s", "Brand: " + cars.get(matchedIndex).getBrand()));
                System.out.print(String.format("%-
30s", "Model: " + cars.get(matchedIndex).getModel()));
                System.out.println();
                System.out.print(String.format("%-
30s", "Type: " + cars.get(matchedIndex).getType()));
                System.out.print(String.format("%-
30s", "Colour: " + cars.get(matchedIndex).getColour()));
                System.out.println();
                System.out.print(String.format("%-
30s", "Status: " + cars.get(matchedIndex).getStatus()));
                System.out.printf("Price: %.2f\n", cars.get(matchedIndex).getPric
e());

                // Loop for edit record
                System.out.println("\nPlease select the data that you want to edi
t.");
                System.out.print(String.format("%70s", "-\n").replace(' ', '-'));
                do {
                    // Submenu for edit record
```

```
                    System.out.print(String.format("%-20s", "1. Plate Number"));
                    System.out.println("5. Colour");
                    System.out.print(String.format("%-20s", "2. Brand"));
                    System.out.println("6. Status");
                    System.out.print(String.format("%-20s", "3. Model"));
                    System.out.println("7. Price");
                    System.out.print(String.format("%-20s", "4. Type"));
                    System.out.println("8. Back to main menu\n");
                    // Get input for user's choice
                    do {
                        try {
                            System.out.print("Selection: ");
                            editChoice = scanner.nextInt();
                            System.out.println();
                            if (editChoice < 1 || editChoice > 8)
                                System.out.println("Invalid input, please choose
from 1 - 8.");
                        } catch (InputMismatchException e) {
                            editChoice = 0; // setting editChoice to a value outs
ide 1 - 8 continues the loop
                            System.out.println("Invalid input, please enter numbe
rs only.");
                            scanner.nextLine();
                        }
                    } while (editChoice < 1 || editChoice > 8);

                    // Allow user to edit selected data
                    if (editChoice == 1) {
                        // Edit plate number
                        // Ensure plateNumber is unique
                        boolean isUnique = false;
                        do {
                            plateNumber = getString("Plate Number", 12);
                            for (int i = 0; i < cars.size(); i++) {
                                if (cars.get(i).getPlateNumber().equals(plateNumb
er)) {
                                    System.out.println("Invalid input, there is a
n existing car record with this plate number.");
                                    isUnique = false;
                                    break;
                                }
                                if (i == cars.size() - 1)
                                    isUnique = true;
                            }
                        } while (!isUnique);
```

```
                        // Set new plateNumber
                        cars.get(matchedIndex).setPlateNumber(plateNumber);
                        System.out.println("\n+ + + Plate Number successfully cha
nged. + + +\n");
                    } else if (editChoice == 2) {
                        // Edit brand
                        brand = getString("Brand", 16);
                        cars.get(matchedIndex).setBrand(brand);
                        System.out.println("\n+ + + Brand successfully changed. +
 + +\n");
                    } else if (editChoice == 3) {
                        // Edit model
                        model = getString("Model", 16);
                        cars.get(matchedIndex).setModel(model);
                        System.out.println("\n+ + + Model successfully changed. +
 + +\n");
                    } else if (editChoice == 4) {
                        // Edit type
                        type = getString("Type", 16);
                        cars.get(matchedIndex).setType(type);
                        System.out.println("\n+ + + Type successfully changed. +
+ +\n");
                    } else if (editChoice == 5) {
                        colour = getString("Colour", 16);
                        cars.get(matchedIndex).setColour(colour);
                        System.out.println("\n+ + + Colour successfully changed.
+ + +\n");
                    } else if (editChoice == 6) {
                        status = getString("Status", 16);
                        cars.get(matchedIndex).setStatus(status);
                        System.out.println("\n+ + + Status successfully changed.
+ + +\n");
                    } else if (editChoice == 7) {
                        // Edit price
                        do {
                            // Catch InputMismatchException (invalid data type in
put)
                            try {
                                System.out.print("Price: ");
                                price = scanner.nextDouble();
                                if (price < 0)
                                    System.out.println("Invalid input, Price cann
ot be negative.");
                            } catch (InputMismatchException e) {
```

```
                                price = -
69; // setting price to a negative integer continues the loop
                                System.out.println("Invalid input, please enter n
umbers only.");
                                scanner.nextLine();
                            }
                        } while (price < 0);
                        cars.get(matchedIndex).setPrice(price);
                        System.out.println("\n+ + + Price successfully changed. +
 + +\n");
                    } else if (editChoice == 8) {
                        // Back to main menu
                        System.out.println("Returning to main menu.");
                        break;
                    }
                } while(true);
            }
        }
    }

    public static void searchRecord(ArrayList<Car> cars) {
        // Variable declarations
        String plateNumber;
        char contInput;
        boolean isFound = false, isStopped = false;
        Scanner scanner = new Scanner(System.in);

        // Check if cars array is empty
        if (cars.size() == 0) {
            System.out.println("There are no existing car records in the system f
or you to search.");
        } else {
            do {
                // Get plateNumber input
                System.out.println("\nPlease enter the Plate Number of the car yo
u want to search.");
                System.out.print(String.format("%70s", "-\n").replace(' ', '-'));
                plateNumber = getString("Plate Number", 12);

                // Check for car with matching plateNumber in array
                for (int i = 0; i < cars.size(); i++) {
                    // Print the car details if found
                    if (cars.get(i).getPlateNumber().compareToIgnoreCase(plateNum
ber) == 0) {
```

```
                        System.out.println("\nCar with Plate Number " + "\"" + pl
ateNumber + "\"" + " found." +
                                " Displaying its details: ");
                        System.out.print(String.format("%70s", "-
\n").replace(' ', '-'));
                        System.out.print(String.format("%-
30s", "Brand: " + cars.get(i).getBrand()));
                        System.out.print(String.format("%-
30s", "Model: " + cars.get(i).getModel()));
                        System.out.println();
                        System.out.print(String.format("%-
30s", "Type: " + cars.get(i).getType()));
                        System.out.print(String.format("%-
30s", "Colour: " + cars.get(i).getColour()));
                        System.out.println();
                        System.out.print(String.format("%-
30s", "Status: " + cars.get(i).getStatus()));
                        System.out.printf("Price: %.2f\n", cars.get(i).getPrice()
);
                        isFound = true;
                        break;
                    } else {
                        isFound = false;
                    }
                }
                if (!isFound)
                    System.out.println("\nNo such car record found.");

                // Ask user if they want to continue searching
                do {
                    System.out.print("\nDo you want to continue searching? (Y/N):
 ");
                    contInput = Character.toUpperCase(scanner.next().charAt(0));
                    if (contInput == 'Y')
                        isStopped = false;
                    else if (contInput == 'N')
                        isStopped = true;
                    else
                        System.out.println("Invalid input, please enter 'Y' or 'N
'.\n");
                } while (contInput != 'Y' && contInput != 'N');
            } while (!isStopped);
        }
    }
```

```
    public static void displayRecords(ArrayList<Car> cars) {
        Scanner scanner = new Scanner(System.in);
        // If cars array is not empty, print records
        if (cars.size() != 0) {
            // Print table header
            System.out.println(String.format("%140s", "=").replace(' ', '='));
            System.out.print("No.    "); // numbering
            System.out.print(String.format("%-16s", "Plate Number"));
            System.out.print(String.format("%-20s", "Brand"));
            System.out.print(String.format("%-20s", "Model"));
            System.out.print(String.format("%-20s", "Type"));
            System.out.print(String.format("%-20s", "Colour"));
            System.out.print(String.format("%-20s", "Status"));
            System.out.print(String.format("%-16s", "Price"));
            System.out.println();
            System.out.println(String.format("%140s", "=").replace(' ', '='));
            // Print table contents
            for (int i = 0; i < cars.size(); i++) {
                System.out.print((i+1) + "      "); // numbering
                System.out.print(String.format("%-
16s", cars.get(i).getPlateNumber()));
                System.out.print(String.format("%-20s", cars.get(i).getBrand()));
                System.out.print(String.format("%-20s", cars.get(i).getModel()));
                System.out.print(String.format("%-20s", cars.get(i).getType()));
                System.out.print(String.format("%-
20s", cars.get(i).getColour()));
                System.out.print(String.format("%-
20s", cars.get(i).getStatus()));
                System.out.print(String.format("%-
16.2f", cars.get(i).getPrice()));
                System.out.println();
            }
        } else {
            // Display no records if array is empty
            System.out.println(String.format("%70s", "=").replace(' ', '='));
            System.out.format("%40s", "NO RECORDS\n");
            System.out.println(String.format("%70s", "=").replace(' ', '='));
        }
        System.out.print("\nPRESS ENTER TO RETURN TO MAIN MENU");
        scanner.nextLine();
    }

    // Helper method for getting String inputs with checking and validation
    public static String getString(String text, int limit) {
        String strInput;
```

```
        boolean hasChars;
        // Instantiate Scanner object
        Scanner scanner = new Scanner(System.in);
        do {
            hasChars = false;
            System.out.print(text + ": ");
            strInput = scanner.nextLine();
            // Check if the string has any characters excluding spaces
            for (int i = 0; i < strInput.length(); i++) {
                if (strInput.charAt(i) != ' ') {
                    hasChars = true;
                    break;
                }
            }
            // Check if the string is valid
            if (strInput.isEmpty() || !hasChars)
                System.out.println("Invalid input, " + text + " cannot be empty."
);
            else if (strInput.length() > limit)
                System.out.println(text + " cannot be longer than " + limit + " c
haracters, please try again.");
        } while (strInput.isEmpty() || !hasChars || strInput.length() > limit);

        return strInput.trim();
    }
}
```

4. Description of the Program

- **Login**

When the system starts, a big “WELCOME” message is shown. Then, the user is prompted to log into the system before they are granted access. This function operates with the assumption that the user is a registered user. The user must enter the correct username and password to gain access to the system. The user is granted 3 attempts to enter the correct username and password, and they will be denied access when they run out of attempts. If the user enters the correct username and password, they will be granted access and redirected to the main menu of the system.

Figure 1.0: Big “WELCOME” message displayed to user

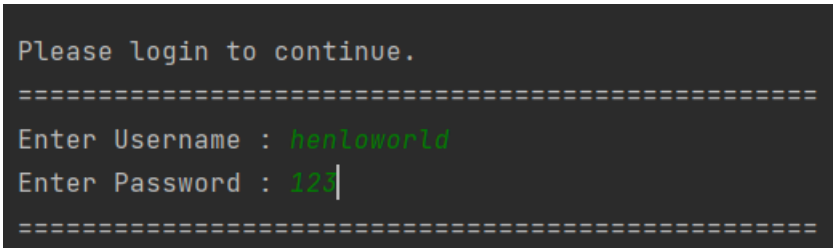


Figure 1.1: User is prompted to enter username and password to login

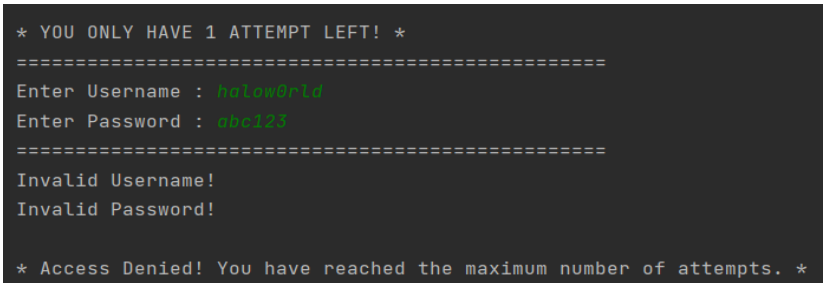


Figure 1.2: User is denied access and the system exits when attempts are depleted

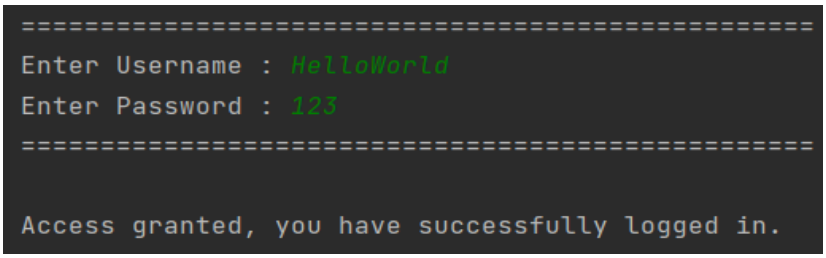


Figure 1.3: User is granted access when they enter the correct username and password

- **Main Menu**

The main menu consists of 6 options for the user to choose from. The 6 options include “Add Record, “Delete record”, “Edit record”, “Search record”, “Display all records”, and “Exit”. These are basically the main functions of the system.

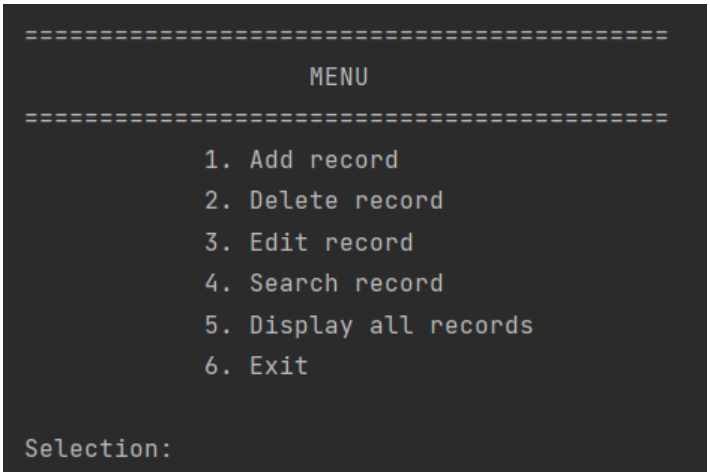


Figure 2.0: Main menu of the system

In the main menu, the user is also prompted for a “Selection” input, which represents the main function that the user wants to select. The system only accepts integer values from 1 to 6; if any other value is entered, the system will display an error message notifying the user that the input is invalid. The system will continuously prompt the user for a new input until a valid input is entered. When a valid input is received, the system will redirect the user to their selected function.

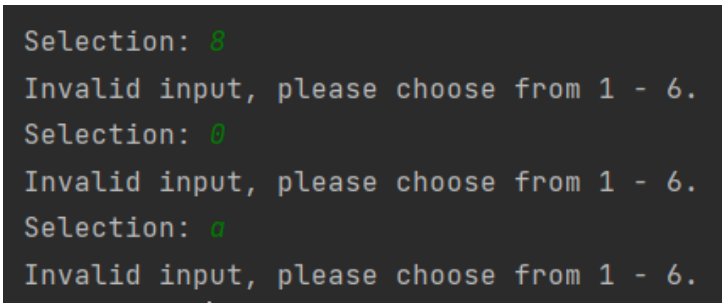


Figure 2.1: System rejects any invalid selection input, and repeatedly prompts user for new input

- **Add Record**

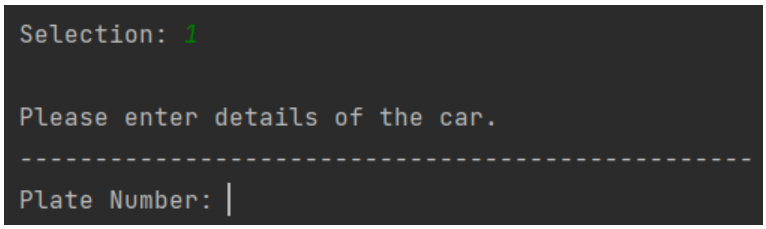


Figure 3.0: System redirects user to “Add record” function when user enters 1 in the menu

If the user enters “1” in the menu, the system redirects the user to the “Add record” function. The purpose of this function is to allow the user to add new car records into the inventory management system. In this function, the system prompts the user for details of the car that they want to add, including the plate number, car brand, car model, car type, colour, status, and price. The user must enter a unique plate number each time they want to add a new car record, otherwise the system will notify the user that there is an existing car record with the entered plate number. The user is also not allowed to enter empty values. Any time the user enters an invalid input, the system will notify them of the error and prompt them to re-enter the data again.

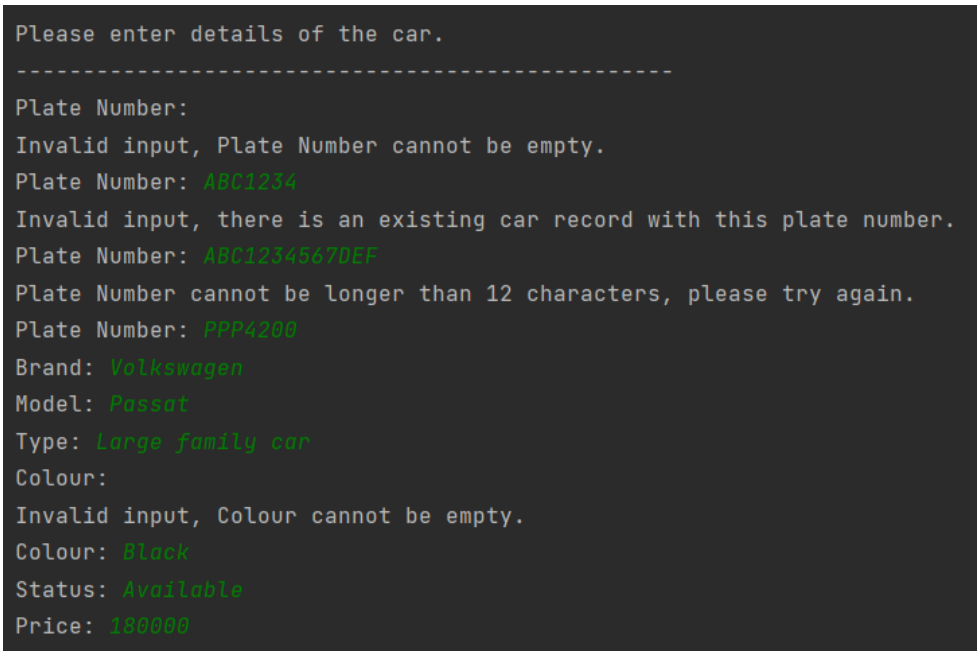


Figure 3.1: Process of entering details of new car record

Once the user has finished entering details of the car, the system asks the user to confirm whether they really want to create the new record. If the user enters 'Y', the system will save the new car record and notify the user that the record has been successfully created; if the user enters 'N', the system cancels the record creation process. Regardless of the user's choice, the system will redirect them to the main menu. If the user enters neither 'Y' nor 'N', the system will continuously prompt them for another input until a valid input is entered.

```
Confirm new Car record? (Y/N): Y

+ + + New Car record has been added. + + +
```

Figure 3.2: User enters 'Y'

```
Confirm new Car record? (Y/N): N

* * * Process cancelled, returning to menu. * * *
```

Figure 3.3: User enters 'N'

The process of adding record is the same each time the user performs this function. However, when the record limit of 100 is reached, the system the notify the user that the maximum number of records has been reached. If the user attempts to add another car record after the limit has been reached, the system will not allow the user to create any new record and will redirect the user back to the main menu.

```
Confirm new Car record? (Y/N): Y

+ + + New Car record has been added. + + +

* * * MAXIMUM AMOUNT OF RECORDS HAS BEEN REACHED * * *
```

Figure 3.4: System notifies the user that the maximum amount of records has been reached

```
Selection: 1

* * * Maximum amount of records reached, unable to add new record. * * *
```

Figure 3.5: System denies the user from making adding records once the limit is reached

- **Delete Record**

```
Selection: 2

Please enter the Plate Number of the car you want to delete.
------------------------------------------------------------------------
Plate Number:
```

Figure 4.0: System redirects user to “Delete record’ function when user enters 2 in the menu

If the user enters “2” in the menu, the system redirects the user to the “Delete record” function. The purpose of this function is to allow the user to delete existing car records in the system. If there are currently no existing car records in the system, this function is not usable. Otherwise, the user will be prompted to enter the plate number of the car record that they want to delete. If the user enters a plate number that does not exist in the system, the system will notify the user that there is no existing car record with the entered plate number, and it redirects them to the main menu.

```
Selection: 2

There are no existing car records in the system for you to delete.
```

Figure 4.1: User is unable to delete any records when there are no existing car records

```
Please enter the Plate Number of the car you want to delete.
------------------------------------------------------------------------
Plate Number: PPP4269

No such car record found, returning to menu.
```

Figure 4.2: User enters plate number that does not exist in the system, user is then redirected to the menu

If the user enters a valid plate number of an existing car record, the system will display the current details of that car record. Then, the user is asked to confirm whether they want to delete the car record or not. If the user enters ‘Y’, the system deletes the car record; if the user enters ‘N’, the system keeps the record as it is and redirects the user to the menu.

```
Please enter the Plate Number of the car you want to delete.
------------------------------------------------------------------------
Plate Number: PPP4200

Car with Plate Number "PPP4200" found. Displaying its details:
------------------------------------------------------------------------
Brand: Volkswagen             Model: Passat
Type: Large family car        Colour: Black
Status: Available             Price: 180000.00

Are you sure you want to delete this car record? (Y/N):
```

Figure 4.3: When a valid plate number is found, the system displays the car’s details and asks user for confirmation on deleting the car record

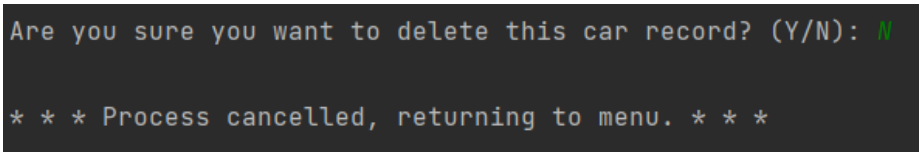


Figure 4.4: User enters 'N', car record is not deleted

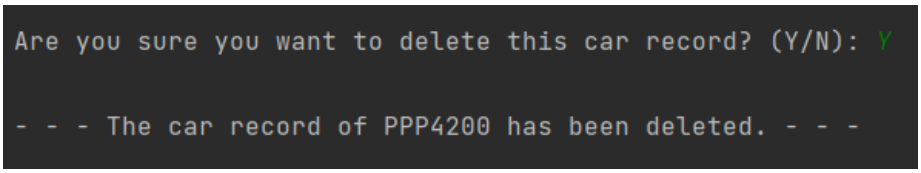


Figure 4.5: User enters 'Y', car record gets deleted

- **Edit Record**

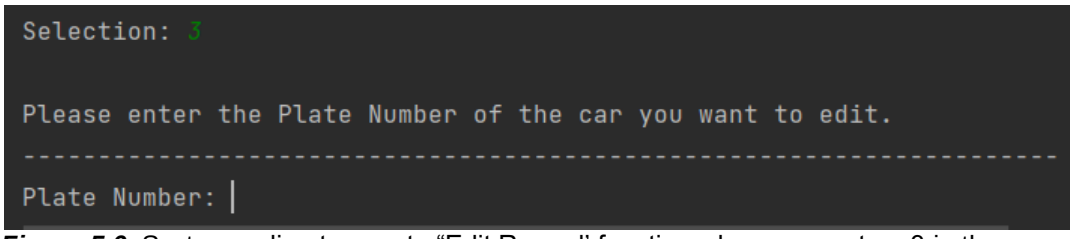


Figure 5.0: System redirects user to "Edit Record' function when user enters 3 in the menu

If the user enters "3" in the menu, the system redirects the user to the "Edit Record" function. The purpose of this function is to allow the user to edit the details of an existing car record in the system. If there are currently no existing records in the system, this function is not usable. Otherwise, the system will first prompt the user to enter the plate number of the car record that they want to edit. If the user enters a plate number that does not exist in the system, the system will notify the user that there is no existing car record with the entered plate number, and it redirects them to the main menu.

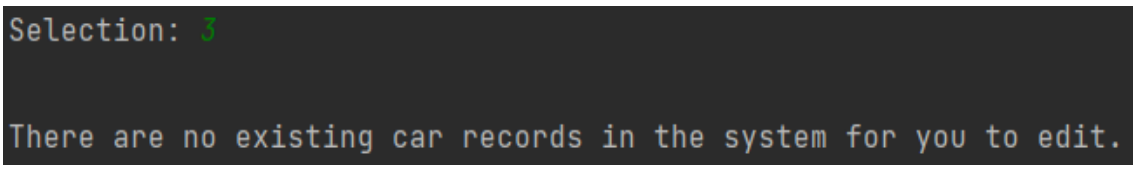


Figure 5.1: User is unable to edit any records when there are no existing car records

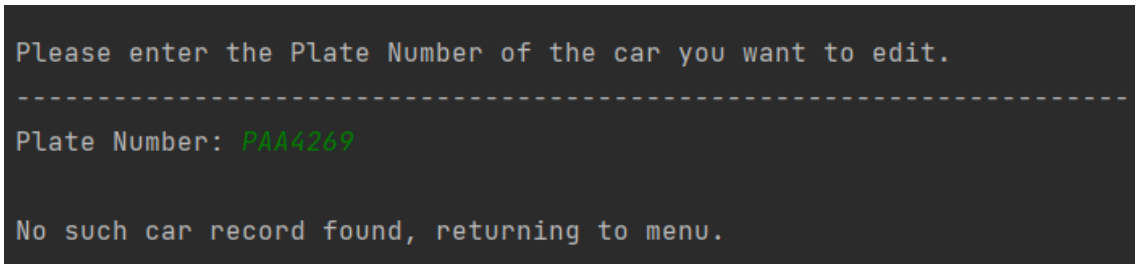


Figure 5.2: User enters plate number that does not exist in the system, and is redirected to the menu

If the user enters a valid plate number of an existing car record, the system will display the current details of that car record. Then, a submenu for “Edit Record” is displayed and the user is prompted to select the specific data which the user wants to edit. The user may enter numbers from 1 to 7 to select the data they want to edit, including plate number, car brand, car model, car type, colour, status, and price; or they can enter 8 to return to the menu. For example, if the user selects 6, the system prompts the user to enter a new status for the selected car. The system will repeatedly allow the user to edit data until they enter 8 to return to the menu.

```
Car with Plate Number "ABC1234" found. Displaying its details:
-----------------------------------------------------------------
Brand: Perodua                Model: Myvi
Type: Subcompact              Colour: White
Status: Available             Price: 30000.00
```

Figure 5.3: When a matching plate number is found, the system displays the car’s details

```
Please select the data that you want to edit.
-----------------------------------------------------------------
1. Plate Number     5. Colour
2. Brand            6. Status
3. Model            7. Price
4. Type             8. Back to main menu

Selection:
```

Figure 5.4: “Edit Record” submenu is displayed, and the system prompts the user to select a data to edit

```
Selection: 6

Status: Booked

+ + + Status successfully changed. + + +
```

Figure 5.5: User enters 6 to change the status, then the system notifies them that the change was successfull

```
Selection: 8


Returning to main menu.
```

Figure 5.6: User enters 8, and is redirected to the main menu

- **Search Record**

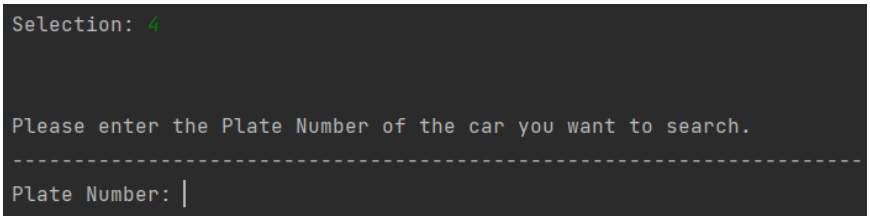


Figure 6.0: System redirects user to “Search record” function when user enters 4 in the menu

If the user enters “4” in the menu, the system redirects the user to the “Search record” function. The purpose of this function is to allow the user to search for an existing car record and check all its current details. If there are currently no existing records in the system, this function is not usable. Otherwise, the system will prompt the user to enter the plate number of the car record that they want to search. If the user enters a plate number that does not exist in the system, the system will notify the user that there is no existing car record with the entered plate number.

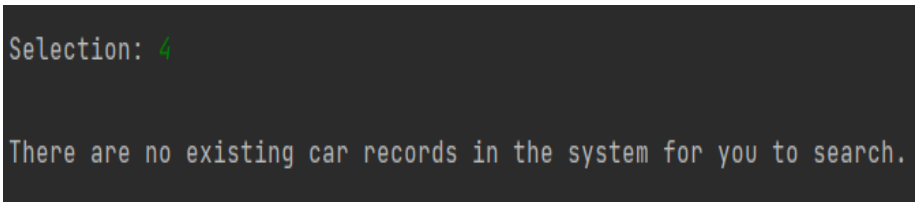


Figure 6.1: User is unable to search records when there are no existing car records

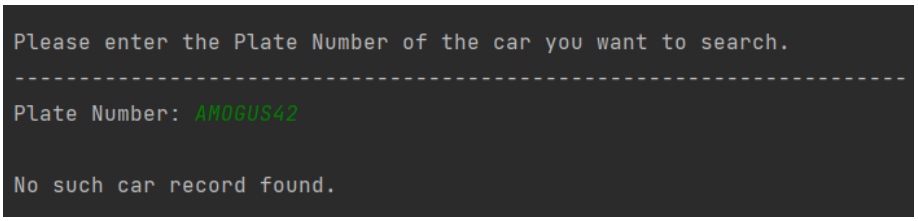


Figure 6.2: User enters a plate number that does not exist in the system

On the other hand, if a valid plate number of an existing car record is entered, the system will display all the details of that car record to the user. Regardless of whether a valid plate number is found in the system or not, the system will ask the user if they want to search again. If the user enters ‘Y’, the system will once again prompt the user to enter a plate number to search; if the user enters ‘N’, the system will end this function and redirect the user to the menu.

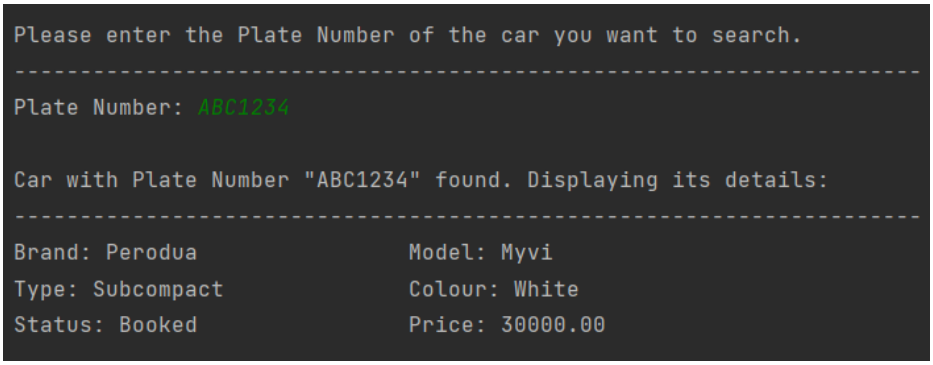


Figure 6.3: If a matching plate number is found, the system displays all its details

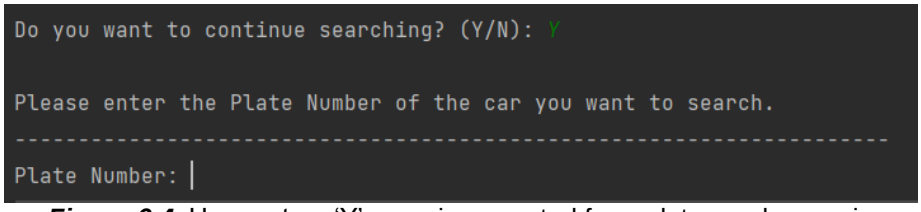


Figure 6.4: User enters 'Y', user is prompted for a plate number again

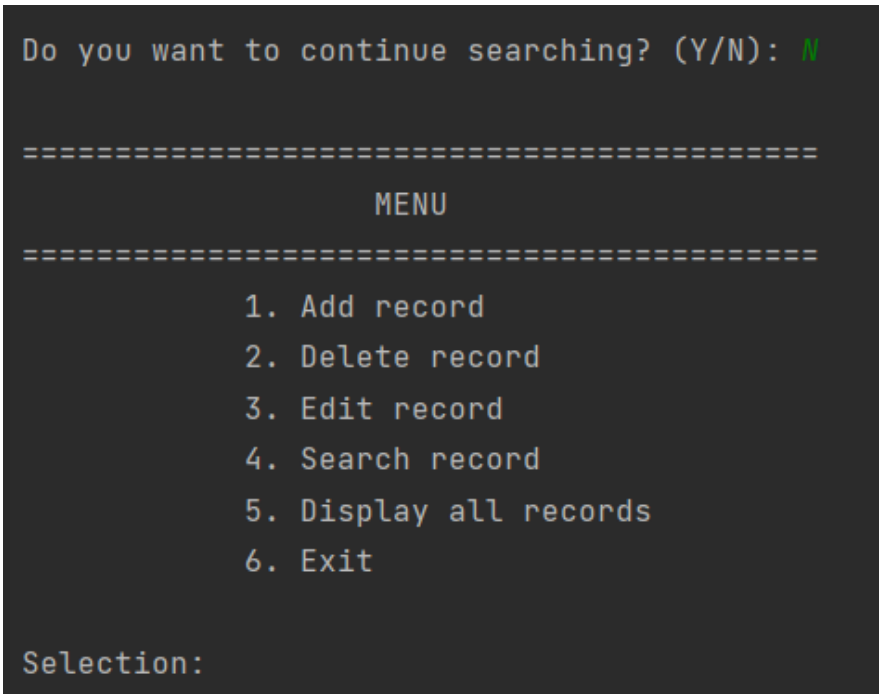


Figure 6.5: User enters 'N", user is redirected to the menu

- **Display All Records**

If the user enters “5” in the menu, the system redirects the user to the “Display all records” function. The purpose of this function is to allow the user to check the details of all existing car records in the system. If there are currently no existing car records in the system, the system would display “NO RECORDS” to the user. Conversely, if there are existing records in the system, the details of every car record will be displayed. The details of each car record are displayed row by row, under their respective columns of data, and the car records are listed from top to bottom based on which records were created earlier. After that, the user is prompted to press the “Enter” key to return to the menu.

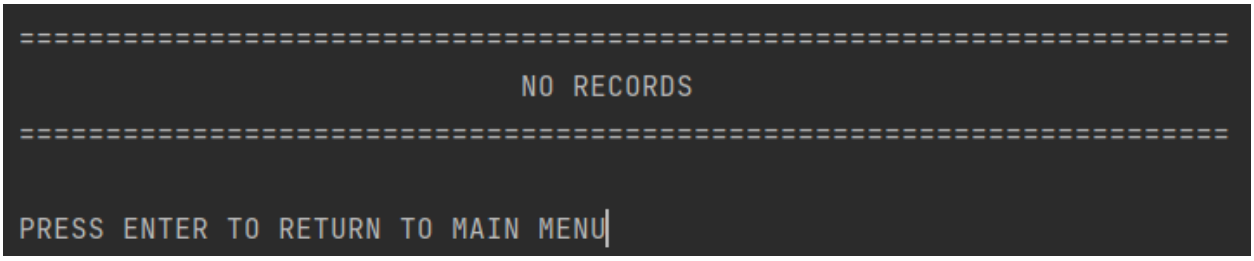


Figure 7.0: System displays “NO RECORDS” if there are no existing car records in the system

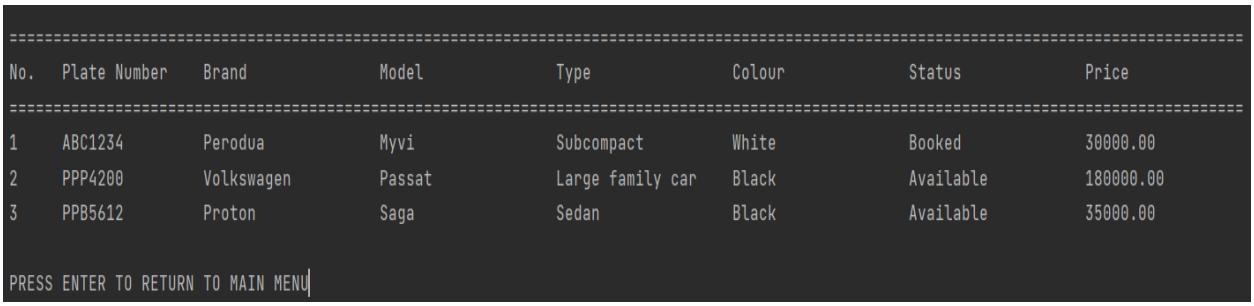


Figure 7.1: System displays a table containing all existing car records in the system

- **Exit**

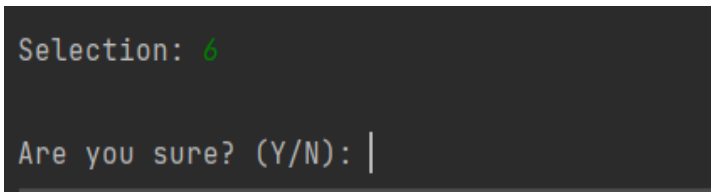


Figure 8.0: System rediects user to “Exit” function when user enters 6 in the menu

If the user enters “6” in the menu, the system redirects user to the “Exit” function. The purpose of this function is to allow the user to end and exit the system. Before exiting, the system asks the user to confirm whether they really want to exit the system or not. If the user enters ‘Y’, the system ends; if the user enters ‘N’, the system does not end, and it redirects the user to the main menu. If any other input besides ‘Y’ or ‘N’ is entered, the system will repeatedly prompt the user for input until a valid input is entered.

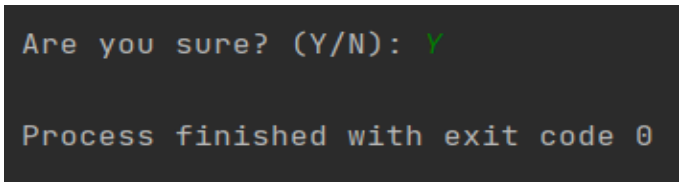


Figure 8.1: User enters ‘Y’, system ends

Figure 8.2: User enters ‘N’, system redirects user to main menu

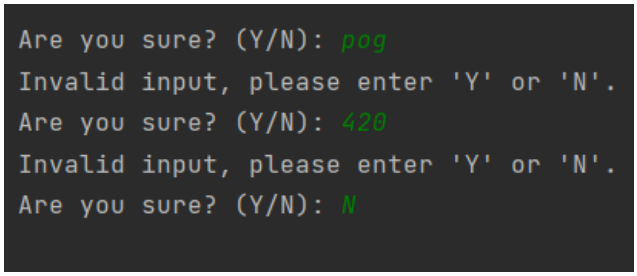


Figure 8.3: System rejects invalid inputs, and repeatedly prompts user for valid input

Bibliography List

Doug Lowe (2021) *How to Catch Exceptions in Java*. Dummies. Available from https://www.dummies.com/programming/java/how-to-catch-exceptions-in-java/ [accessed 11 June 2021].

Kukade, G. (2020) *How Do I Compare Strings in Java?*. Dzone. Available from https://dzone.com/articles/how-do-i-compare-strings-in-java [accessed 5 June 2021].

Oracle (2021) *ArrayList*. Oracle. Available from https://docs.oracle.com/javase/8/docs/api/java/util/ArrayList.html [accessed 11 June 2021].

Oracle (2020) *Formatter*. Oracle. Available from https://docs.oracle.com/javase/7/docs/api/java/util/Formatter.html [accessed 5 June 2021].

Oracle (2020) *Scanner*. Oracle. Available from https://docs.oracle.com/javase/7/docs/api/java/util/Scanner.html [accessed 11 June 2021].