Project: Car Rental Management System

Introduction:

This project is a simple car rental system built using Java. It helps manage car rentals by keeping track of which cars are available, who rented them, and for how long. When a customer rents a car, the system marks that car as "unavailable," saves the customer's details (name, email, and address), and records the rental period. Everything is stored in an organized way so the system can check past rentals and available cars. It's a basic but functional program that teaches how real rental systems work behind the scenes.

Benefits:

1. Saves Time – No more paper lists! The computer tracks cars and customers fast.
2. No Mistakes – Automatically records rentals, so no mix-ups in bookings.
3. Easy to Use – Just enter customer details, pick a car, and done!
4. Safe Records – Customer info is stored digitally (better than paper receipts).
5. Makes More Money – Helps rent out cars faster and avoid empty cars sitting idle.

Drawbacks:

1. Costs Money – Need a computer, software, and maybe training for staff.
2. Can Crash – If the system stops working, rentals get stuck.
3. Learning Needed – Staff must learn how to use it (not everyone is tech-savvy).
4. Internet Needed – If Wi-Fi/power goes out, the system won't work.
5. Hack Risk – Bad people might steal customer data if security is weak.

Code:

```
static String[] carModels = {"Tesla Model 3", "Honda Civic"};
static double[] carPrices = {150, 70};     static boolean[]
carAvailable = {true, true};

   static int[] rentedCarIndex = new int[100];     static
String[] rentedCustomerName = new String[100];     static
String[] rentedCustomerEmail = new String[100];     static
String[] rentedCustomerAddress = new String[100];     static
int[] rentedDays = new int[100];     static int rentalCount =
0;


   public static void main(String[] args) {
Scanner sc = new Scanner(System.in);
       int choice = -1;


       while (choice != 0) {
          try {
              System.out.println("\n--- Simple Car Rental Menu ---");
              System.out.println("1. List Cars");
              System.out.println("2. Rent a Car");
              System.out.println("3. Return a Car");
              System.out.println("4. List Rentals");
              System.out.println("0. Exit");




              choice = sc.nextInt();
sc.nextLine(); // clear buffer


              switch (choice) {
```

```
                case 1:
listCars();
break;                case 2:
rentCar(sc);
break;                case 3:
returnCar(sc);
break;                case 4:
listRentals();
break;                case 0:
                    System.out.println("Goodbye!");
                    break;
default:
                    System.out.println("Invalid choice. Try again.");
                }
            } catch (InputMismatchException e) {
                System.out.println("Invalid input! Please enter a number.");
sc.nextLine(); // clear wrong input
            } catch (Exception e) {
                System.out.println("An unexpected error occurred: " + e.getMessage());
                sc.nextLine(); // clear buffer
            }
        }
    }

    public static void listCars() {
        System.out.println("\n--- Cars ---");
for (int i = 0; i < carModels.length; i++) {
            System.out.println((i + 1) + ". " + carModels[i] + " | $" + carPrices[i] + " per day |
Available: " + carAvailable[i]);
```

```
    }
  }

  public static void rentCar(Scanner sc) {
    try {
listCars();
      System.out.print("Enter Car Number to Rent: ");
int carNum = sc.nextInt();          sc.nextLine();
      if (carNum < 1 || carNum > carModels.length) {
System.out.println("Invalid Car Number.");
        return;
      }

      if (!carAvailable[carNum - 1]) {
        System.out.println("Car is not available.");
        return;
      }

      String name;
while (true) {
        System.out.print("Enter Your Name: ");
name = sc.nextLine().trim();            if
(!name.isEmpty()) break;
        System.out.println("Name cannot be empty. Please enter again.");
      }

      String email;
while (true) {
System.out.print("Enter
```

```
Your Email: ");
email =
sc.nextLine().trim();
if (email.contains("@"))
break;
            System.out.println("Invalid email. Please include '@' and try again.");
        }

        String address;
while (true) {
            System.out.print("Enter Your Address: ");
address = sc.nextLine().trim();              if
(!address.isEmpty()) break;
            System.out.println("Address cannot be empty. Please enter again.");
        }

        int days;
while (true) {
            System.out.print("Enter Number of Days: ");
            if (sc.hasNextInt()) {
days = sc.nextInt();
sc.nextLine();                  if
(days > 0) break;
            } else {
sc.nextLine();
            }
            System.out.println("Invalid number of days. Please enter a positive number.");
        }
```

```
            // Save rental            carAvailable[carNum -
1] = false;            rentedCarIndex[rentalCount] =
carNum - 1;
rentedCustomerName[rentalCount] = name;
rentedCustomerEmail[rentalCount] = email;
rentedCustomerAddress[rentalCount] = address;
rentedDays[rentalCount] = days;
rentalCount++;

            System.out.println("Car rented successfully!");

        } catch (InputMismatchException e) {
            System.out.println("Invalid input! Please enter correct values.");
            sc.nextLine();
        } catch (Exception e) {
            System.out.println("An error occurred during rental: " + e.getMessage());
            sc.nextLine();
        }
    }

    public static void returnCar(Scanner sc) {
        try {
            System.out.print("Enter Car Number to Return: ");
int carNum = sc.nextInt();            sc.nextLine();
            if (carNum < 1 || carNum > carModels.length) {
System.out.println("Invalid Car Number.");
                return;
            }
```

```
            if (carAvailable[carNum - 1]) {
                System.out.println("Car is already available.");
            } else {
                carAvailable[carNum - 1] = true;
                System.out.println("Car returned successfully!");
            }

        } catch (InputMismatchException e) {
            System.out.println("Invalid input! Please enter a valid number.");
            sc.nextLine();
        } catch (Exception e) {
            System.out.println("An error occurred while returning the car: " + e.getMessage());
            sc.nextLine();
        }
    }

    public static void listRentals() {
        try {
            System.out.println("\n--- Current Rentals ---");
if (rentalCount == 0) {
                System.out.println("No active rentals.");
                return;
            }

            for (int i = 0; i < rentalCount; i++) {                int
carIndex = rentedCarIndex[i];                double total =
carPrices[carIndex] * rentedDays[i];

                System.out.println("Car: " + carModels[carIndex] +
```

```
                " | Customer: " + rentedCustomerName[i] +
                " | Email: " + rentedCustomerEmail[i] +
                " | Address: " + rentedCustomerAddress[i] +
                " | Days: " + rentedDays[i] +
                " | Total: $" + total);
        }
    } catch (Exception e) {
        System.out.println("An error occurred while listing rentals: " + e.getMessage());
    }
} }
```

Working:

It allows users to:

- View available cars
- Rent a car
- Return a car
- List all rentals
- Arrays to track details of up to 100 rentals:
    - Which car was rented
    - Who rented it (name, email, address) ○ How many days they rented it for
    - A counter to track how many rentals have been made

Main () Method

The **starting point of the program**.

What it does:

- Creates a Scanner for user input.
- Displays a simple text-based menu repeatedly until the user chooses 0 to exit.
- Takes the user's choice and uses a switch statement to call the appropriate method.

Also has:

- **InputMismatchException** handling to catch invalid inputs (e.g., entering letters when numbers are expected).
- General **Exception** handling for unexpected errors.

Method Listcars ()

- Displays all the cars available for rentals.
- Shows the car number, name, price per day, and availability.

rentCar(Scanner sc) Method

Handles the rental process:

1. Calls listCars() to show available cars.
2. Asks the user to enter the car number they want to rent.
3. Validates:
    - If the car number is within range.
    - If the car is available.
4. Prompts for:
    - Customer name (non-empty) ○ Email (must contain @) ○ Address (non-empty) ○ Number of rental days (positive number)
5. Stores the rental details in the arrays.
6. Marks the car as **not available**.
7. Increments rentalCount.
8. Confirms successful rental.

returnCar(Scanner sc) Method

Handles car return:

1. Asks which car number to return.
2. Checks if it's within valid range.
3. If the car is already available:
    - Displays a message.
4. If not:
    - Marks the car as available. ○ Displays a confirmation message.

listRentals() Method

Displays a list of all current rentals:

1. If no rentals have been made (rentalCount == 0), show a message.
2. Otherwise, loop through the rental records:
    - Show car name, customer details, number of days, and total price (price per day × number of days).

Error Handling

Throughout the program:

- Uses **try-catch** blocks to gracefully handle:
    - Invalid number inputs (InputMismatchException) ○ Any other unexpected errors (Exception)

Summary: How It Works

It's a simple, menu-driven console app for a car rental service where:

- **Cars can be viewed, rented, and returned**
- **Customer details are collected for rentals**
- **Rental records are displayed with total costs**