S\*\*Showroom Management System in Java\*\*

Welcome to the Showroom Management System, a Java-based application that brings efficiency and organization to your car showroom operations. This project exemplifies key Object-Oriented Programming (OOP) principles, ensuring a robust and scalable solution.

### Key Features:

1. \*\*Encapsulation:\*\*

- Private properties in the `PurchaseForm` and `Car` classes.

- Public methods controlling access to encapsulated data.

2. \*\*Inheritance:\*\*

- The `UsedCar` class extends the `Car` class, promoting code reusability and hierarchy.

3. \*\*Polymorphism:\*\*

- Explore method overriding in the `UsedCar` class, specifically in the `sellCar` method.

4. \*\*Constructor:\*\*

- Constructors are utilized to initialize object properties during instantiation, ensuring proper setup.

5. \*\*Method Overriding:\*\*

- Witness method overriding in the `UsedCar` class, enhancing flexibility and adaptability.

6. \*\*Method Overloading:\*\*

- Multiple versions of the `generateNumberPlate` method accommodate various scenarios.

7. \*\*Object Instantiation:\*\*

- Instances of classes (`PurchaseForm`, `Car`, `UsedCar`) are created, representing real-world entities.

8. \*\*Composition:\*\*

- Lists of objects from other classes (`purchaseHistory` and `sellHistory`) in the `Main` class for a holistic approach.

9. \*\*Static Method:\*\*

- The `generateNumberPlate` method is designed as static for utility and ease of use.

10. \*\*Access Modifiers:\*\*

- Public, private, and protected access modifiers are thoughtfully applied, ensuring appropriate data access.

11. \*\*Instance Variables:\*\*

- Classes have well-defined instance variables representing distinct object states.

12. \*\*Conditional Statements:\*\*

- Experience intelligent decision-making with conditional statements (e.g., price adjustment in `sellCar` method).

13. \*\*Looping:\*\*

- Do-while and for loops are employed for streamlined execution and user interaction.

14. \*\*Input/Output:\*\*

- Utilizing `Scanner` for user input and `System.out.println` for output, the system ensures a user-friendly experience.

### How to Use:

1. \*\*Buy Cars:\*\*

- Input details like car name, model, manufacturing year, color, fuel type, and transmission type.

- The system generates a purchase form, followed by a buyer's information form.

- Finalize the purchase, and witness the automatic generation of a unique number plate.

2. \*\*Sell Cars:\*\*

- Provide necessary details to create a sell form.

- Fill in seller's information to complete the selling process.

- Track the selling history for comprehensive record-keeping.

3. \*\*Check History:\*\*

- Explore the purchase and sell history seamlessly, ensuring transparency and accountability.

Experience the power of OOP and elevate your showroom management with this Java-based system. Feel free to contribute, enhance, and tailor it to your specific needs. Happy coding!