Documentation for the Tax Calculation Application

**Property Class**

Fields:

* **baseValueOfLand** (double): Represents the base value of the land.
* **isInCity** (char): Indicates if the property is located in a city ('Y' for yes, 'N' for no).
* **ageOfProperty** (int): Represents the age of the property in years.
* **propertyTax** (double): Stores the calculated property tax.
* **id** (int): Unique identifier for the property.
* **builduparea** (int): Represents the built-up area of the property.
* **tax** (float): Stores the calculated tax.

Constructors:

* **Property(int id, int builduparea, double baseValueOfLand, int ageOfProperty, char isInCity)**: Initializes a property with specified parameters. Calculates property tax based on provided factors.
* **Property()**: Default constructor.

Methods:

* **calculatePropertyTax()**: Calculates the property tax based on factors like built-up area, base value of land, age, and city location.

**Vehicle Class**

Fields:

* **RegNum** (int): Vehicle registration number.
* **brand** (String): Brand of the vehicle.
* **maxvelocity** (int): Maximum velocity of the vehicle in kilometers per hour.
* **noOfSeats** (int): Number of seats in the vehicle.
* **vehicleType** (String): Type of the vehicle (e.g., PETROL, DIESEL, CNG/LPG).
* **cost** (float): Purchase cost of the vehicle.
* **vehicletax** (float): Stores the calculated vehicle tax.

Constructors:

* **Vehicle(int RegNum, String brand, int maxvelocity, int noOfSeats, String vehicleType, float cost)**: Initializes a vehicle with specified parameters.
* **Vehicle()**: Default constructor.

Methods:

* **calculateVehicleTax()**: Calculates the vehicle tax based on the type of fuel and other factors.

**TaxEntry Class**

Fields:

* **particular** (String): Describes the tax entry.
* **quantity** (int): Quantity of the entry (e.g., number of properties or vehicles).
* **tax** (double): Amount of tax.

Constructors:

* **TaxEntry(String particular, int quantity, double tax)**: Initializes a tax entry with specified details.

Methods:

* **getParticular()**: Returns the particular of the tax entry.
* **getQuantity()**: Returns the quantity of the tax entry.
* **getTax()**: Returns the tax amount.

**PropertyOperations Class**

Fields:

* **propertyList** (List<Property>): A list to store property objects.

Constructors:

* **PropertyOperations()**: Default constructor.

Methods:

* **addPropertyDetails1()**: Adds property details by taking user input.
* **addPropertyDetails()**: Initializes sample property objects and adds them to the property list.
* **viewPropertyDetails()**: Displays details of all properties.
* **calculatePropertyTax(Scanner scanner)**: Prompts user to enter property ID and calculates property tax for that property.
* **propertytaxmenu(Scanner scanner)**: Displays a menu for property tax operations.
* **getPropertyTaxSummary()**: Generates a summary of property tax details.
* **calculateTotalPropertyTax()**: Calculates the total property tax for all properties.

**Vehicleoperations Class**

Fields:

* **vehiclelist** (List<Vehicle>): A list to store vehicle objects.

Constructors:

* **Vehicleoperations()**: Default constructor.

Methods:

* **addVehicleDetails()**: Adds vehicle details by taking user input.
* **addvehicle()**: Initializes sample vehicle objects and adds them to the vehicle list.
* **viewvehicle()**: Displays details of all vehicles.
* **calculateVehicleTaxByRegNum1()**: Calculates vehicle tax based on registration number.
* **vehicletaxmenu(Scanner scanner)**: Displays a menu for vehicle tax operations.
* **getVehicleTaxSummary()**: Generates a summary of vehicle tax details.
* **calculateTotalVehicleTax()**: Calculates the total vehicle tax for all vehicles.

**TaxMain Class**

Methods:

* **calculateTotalTax()**: Calculates and displays the total tax amount, including both property and vehicle taxes.

**Tax Class (Main Class)**

Methods:

* **login(String username, String password)**: Checks if the provided username and password match the expected values (admin/admin123).
* **main(String[] args)**: Entry point of the program. Allows the user to interact with the tax calculation system.

***each part of the code and its purpose:***

1. **Property Class**:
   * This class represents a property and contains information about it such as base value of land, whether it's in the city, age of the property, property tax, ID, built-up area, and tax.
   * The class has a method **calculatePropertyTax()** which calculates the property tax based on various factors.
2. **Vehicle Class**:
   * This class represents a vehicle and contains information about it like registration number, brand, maximum velocity, number of seats, vehicle type, cost, and vehicle tax.
   * The class has a method **calculateVehicleTax()** which calculates the vehicle tax based on the type of fuel.
3. **TaxEntry Class**:
   * This class represents an entry in the tax summary. It includes information about the particular (e.g., properties or vehicles), the quantity, and the tax amount.
4. **PropertyOperations Class**:
   * This class manages operations related to properties, such as adding property details, viewing property details, and calculating property tax.
   * It also includes a method **getPropertyTaxSummary()** which generates a summary of property tax details.
   * The **calculateTotalPropertyTax()** method calculates the total property tax for all properties.
5. **Vehicleoperations Class**:
   * This class manages operations related to vehicles, such as adding vehicle details, viewing vehicle details, and calculating vehicle tax.
   * It also includes a method **getVehicleTaxSummary()** which generates a summary of vehicle tax details.
   * The **calculateTotalVehicleTax()** method calculates the total vehicle tax for all vehicles.
6. **TaxMain Class**:
   * This class handles the overall tax calculation process. It includes a method **calculateTotalTax()** which calculates and displays the total tax amount, including both property and vehicle taxes.
7. **Tax Class (Main Class)**:
   * This is the main class of the program. It contains the **main** method, which is the entry point of the program. It allows the user to interact with the tax calculation system.
   * It also includes a method **login** for user authentication.
8. **Documentation**:
   * I provided documentation to explain the purpose and functionality of each class, along with descriptions of their fields and methods.

Overall, this program simulates a system for calculating property and vehicle taxes. It allows users to add property and vehicle details, view details, and calculate taxes. The **TaxMain** class serves as a coordinator, calculating the total tax amount and providing an entry point for the program. The various classes and methods are organized to provide a structured and modular design for tax calculation operations.