CODE FOR TAX CALCULATION APPLICATION

```
package tax_calculation_application;
import java.util.*;

class Property {
    private double baseValue;
    private boolean inCity;
    private int age;
    private double tax;
            private int area;


    public Property(int area, double baseValue, boolean inCity, int age) {
            this.area=area;
        this.baseValue = baseValue;
        this.inCity = inCity;
        this.age = age;

    }

    public double calculateTax() {

        if (inCity) {
            tax = (area * age * baseValue) + (0.5 * baseValue);

        } else {
            tax = area * age * baseValue;
        }
        return tax;
    }

    @Override
    public String toString() {
            char c='Y';
            if (inCity) {
            tax = (area * age * baseValue) + (0.5 * baseValue);
        } else {
            tax = area * age * baseValue;
            c='N';
        }

            return String.format("| %-16s | %-10.2f | %-10s | %-14s | $%-8.2f |\n", area,baseValue,c, age,tax);

    }
}

class Vehicles {
    private String regNumber;
```

```
private String brand;
private double purchaseCost;
private double velocity;
private int capacity;
private int type;
private double tax;

public Vehicles(String regNumber, String brand, double purchaseCost, double velocity, int capacity, int type) {
    this.regNumber = regNumber;
    this.brand = brand;
    this.purchaseCost = purchaseCost;
    this.velocity = velocity;
    this.capacity = capacity;
    this.type = type;

    if (isValidRegistrationNumber(regNumber)) {
        this.regNumber = regNumber;
    } else {
        throw new IllegalArgumentException("Invalid registration number format");
    }

    // Validate and set brand
    if (!brand.isEmpty()) {
        this.brand = brand;
    } else {
        throw new IllegalArgumentException("Brand cannot be empty");
    }

    // Validate and set purchase cost
    if (purchaseCost >= 50000 && purchaseCost <= 1000000) {
        this.purchaseCost = purchaseCost;
    } else {
        throw new IllegalArgumentException("Purchase cost must be between 50000 and 1000000");
    }

    // Validate and set maximum velocity
    if (velocity >= 120 && velocity <= 300) {
        this.velocity = velocity;
    } else {
        throw new IllegalArgumentException("Maximum velocity must be between 120 and 300 km/h");
    }

    // Validate and set capacity
    if (capacity >= 2 && capacity <= 50) {
        this.capacity = capacity;
    } else {
        throw new IllegalArgumentException("Capacity must be between 2 and 50");
    }

    // Validate and set type
```

```
        if (type >= 1 && type <= 3) {
            this.type = type;
        } else {
            throw new IllegalArgumentException("Type must be between 1 and 3");
        }

    }
    private boolean isValidRegistrationNumber(String regNumber) {
        // Check if it's a 4-digit number with optional leading zeros
        return regNumber.matches("0*[1-9][0-9]{0,3}");
    }

    public double calculateTax() {
        switch (type) {
            case 1:
                tax = velocity + capacity + 0.10 * purchaseCost;
                break;
            case 2:
                tax = velocity + capacity + 0.11 * purchaseCost;
                break;
            case 3:
                tax = velocity + capacity + 0.12 * purchaseCost;
                break;
            default:
                tax = 0.0;
        }
        return tax;
    }

    @Override
    public String toString() {
        return String.format("| %-11s | %-10s | %-15s | %-15s | %-15s | %-5s | %-7.2f|\n", regNumber, brand,
purchaseCost,velocity,capacity,type,tax);
    }
}

public class TaxCalculatorAp {
        static List<Property> properties = new ArrayList<>();
        static List<Vehicles> vehicles = new ArrayList<>();

        public static void main(String[] args) {

                System.out.println("**********WELCOME TO TAX CALCULATOR APPLICATION *****************
\n");
                System.out.println("My name is Yadav Dhiraj Rajendra Prasad and  I Developed this
application\n");

                System.out.println("************************USER INTERFACE*************************\n");
                user_interface();
```

```
}


private static void user_interface() {
	Scanner sc=new Scanner(System.in);

	while(true) {
	System.out.println("Choose an option:");
	System.out.println("1. Property Tax");
	System.out.println("2. Vehicle Tax");
	System.out.println("3. Total Tax");
	System.out.println("4. Close Application");

	int choice;

	try {
		choice=sc.nextInt();
	}
	catch (InputMismatchException exp) {
		System.out.println("Invalid input, please select a valid input :");
		sc.nextLine();
		continue;
	}

	switch(choice) {
	case 1:
		Property_Tax(sc,properties);
		break;
	case 2:
		Vehicles_Tax(sc,vehicles);
		break;
	case 3:
		double totalPropertyTax = properties.stream().mapToDouble(Property::calculateTax).sum();
		double totalVehicleTax = vehicles.stream().mapToDouble(Vehicles::calculateTax).sum();
		double totalTax = totalPropertyTax + totalVehicleTax;
		int a = properties.size();
		int b = vehicles.size();
		System.out.println("****************** TOTAL TAX CALCULATIONS ******************");
	   System.out.println("+------+------------------+-----------------+----------------------+");
	   System.out.println("|SR.NO |3    Tax Category   |     Quanity     |       Total Tax      |");
	   System.out.println("+------+------------------+-----------------+----------------------+");
	   System.out.printf("|  1   |     Property Tax   |       %-5d     |      $%-8.2f     |\n", a,totalPropertyTax);
	   System.out.printf("|  2   |     Vehicle Tax    |       %-5d     |      $%-8.2f     |\n", b,totalVehicleTax);
	   System.out.println("+-------------------------------------------+----------------------+");
	   System.out.printf("| Total Tax Payable         |       %-5d     |       $%-5.2f   |\n",(a+b), totalTax);
	   System.out.println("+-------------------------------------------+----------------------+");

	   break;
	case 4:
		System.out.println("Closing the application. Goodbye!");
```

```
                System.exit(0);
            default:
                    System.out.println("Invalid input, please select a valid input :");

    }
}

    }
    private static void Property_Tax(Scanner sc, List<Property> properties) {
            while(true) {
            System.out.println("Choose an option:");
            System.out.println("1. Add Property");
            System.out.println("2. Calculate Property Tax");
            System.out.println("3. Display All Property");
            System.out.println("4. Back To Main Menu");

            int ch;
            try {
                    ch=sc.nextInt();
            }
            catch (InputMismatchException exp){
                    System.out.println("Invalid input, please select a valid option");
                    sc.nextLine();
                    continue;
            }
            switch(ch) {
            case 1:
                    System.out.print("Enter the Built-up Area: ");
                    int area;
                    try {
                    area=sc.nextInt();
                }
                catch (InputMismatchException exp) {
                    System.out.println("Invalid input, please select a valid input :");
                    sc.nextLine();
                    continue;
                }

                System.out.print("Enter base value of land: ");
                double baseValue;
                try {
                    baseValue=sc.nextDouble();
                }
                catch (InputMismatchException exp) {
                    System.out.println("Invalid input, please select a valid input :");
                    sc.nextLine();
                    continue;
                }
                System.out.print("Is the property in the city? (Y/N): ");
                boolean inCity;
```

```
            try {
                inCity = sc.next().equalsIgnoreCase("Y");
            }
            catch (InputMismatchException exp) {
                System.out.println("Invalid input, please select a valid input :");
                sc.nextLine();
                continue;
            }
            System.out.print("Enter age of construction: ");
            int age;
            try {
                age = sc.nextInt();
            }
            catch (InputMismatchException exp) {
                System.out.println("Invalid input, please select a valid input :");
                sc.nextLine();
                continue;
            }
            properties.add(new Property(area,baseValue, inCity, age));
            System.out.println("Property added.");
            break;
        case 2:
                double totalPropertyTax = properties.stream().mapToDouble(Property::calculateTax).sum();
            System.out.println("Total Property Tax: " + totalPropertyTax);
            break;
        case 3:
                System.out.println("********************************* PROPERTY TAX CALCULATION
*********************************");


System.out.println("+--------------+-----------------+------------+------------+---------------+-------------+");
            System.out.printf("| %-12s | %-16s | %-10s | %-10s | %-14s | %-11s |\n","Property-No", "Built-up
Area", "Base Price", "In-City", "Age", "Tax");

            System.out.println("+--------------+-----------------+------------+------------+---------------+-------------+");


                if (properties.isEmpty()) {
               System.out.println("No Property Details Found.");
            } else {
               for (int i = 0; i < properties.size(); i++) {
                  System.out.printf("|  %d           " + properties.get(i),(i + 1));
               }
            }

System.out.println("+--------------+-----------------+------------+------------+---------------+-------------+");
            break;
        case 4:
                user_interface();
        default:
                System.out.println("Invalid input, please select a valid option");
```

```
        }

        }

}
        private static void Vehicles_Tax(Scanner sc , List<Vehicles> vehicles) {
                while (true) {
                System.out.println("Select an option:");
                System.out.println("1. Add Vehicle");
                System.out.println("2. Calculate Vehicle Taxes");
                System.out.println("3. Display Vehicle Details");
                System.out.println("4. Back to Main Menu");

                int c;
                try {
                        c=sc.nextInt();
                }
                catch (InputMismatchException exp){
                        System.out.println("Invalid input, please select a valid option");
                        sc.nextLine();
                        continue;
                }

                switch (c) {
                   case 1:
                        String regNumber;
                        try {
                                System.out.print("Enter Registration Number: ");
                        regNumber = sc.next();

                      }
                        catch (IllegalArgumentException|InputMismatchException exp) {
                        System.out.println("Invalid input: " + exp.getMessage());
                        sc.nextLine();
                        continue;
                      }
                        String brand;
                        try {
                        System.out.print("Enter Brand of Vehicle: ");
                        brand = sc.next();
                      }
                        catch (IllegalArgumentException|InputMismatchException exp) {
                        System.out.println("Invalid input: " + exp.getMessage());
                        sc.nextLine();
                        continue;
                      }
                        double purchaseCost;
                        try {
                                System.out.print("Enter Purchase Cost: ");
                        purchaseCost = sc.nextDouble();
```

```
    }
        catch (IllegalArgumentException exp) {
        System.out.println("Invalid input: " + exp.getMessage());
        sc.nextLine();
        continue;
    }
        double velocity;
        try {
                System.out.print("Enter Maximum Velocity (km/h): ");
        velocity = sc.nextDouble();
    }
        catch (IllegalArgumentException|InputMismatchException exp) {
        System.out.println("Invalid input: " + exp.getMessage());
        sc.nextLine();
        continue;
    }
        int capacity;
        try {
                System.out.print("Enter Capacity (Number of Seats): ");
        capacity = sc.nextInt();
    }
        catch (IllegalArgumentException|InputMismatchException exp) {
        System.out.println("Invalid input: " + exp.getMessage());
        sc.nextLine();
        continue;
    }
        int type;
        try {
                System.out.print("Select Type of Vehicle (1-Petrol, 2-Diesel, 3-CNG/LPG): ");
        type = sc.nextInt();

    } catch (IllegalArgumentException|InputMismatchException exp) {
        System.out.println("Invalid input: " + exp.getMessage());
        sc.nextLine();
        continue;
    }
        vehicles.add(new Vehicles(regNumber, brand, purchaseCost, velocity, capacity, type));
        System.out.println("Vehicle added successfully.");

    break;

case 2:
    double totalVehicleTax = vehicles.stream().mapToDouble(Vehicles::calculateTax).sum();
    System.out.println("Total Vehicle Tax: " + totalVehicleTax);
    break;
case 3:
    if (vehicles.isEmpty()) {
        System.out.println("No Vehicle Details Found.");
    } else {
```

```
                    System.out.println("********** VEHICLE TAX CALCULATION **********");

        System.out.println("+-------------+------------+----------------+----------------+----------------+-------+--------+");
                    System.out.printf("| %-11s | %-10s | %-15s | %-15s | %-15s | %-5s |
        %-7s|\n","Reg-Number", "Brand", "Purchase Cost", "Velocity(km/h)", "Capacity", "Type", "Tax");

        System.out.println("+----------+--------------+----------------+----------------+----------------+-------+--------+");
                    for (int i = 0; i < vehicles.size(); i++) {
                        System.out.println(vehicles.get(i));
                    }

        System.out.println("+----------+--------------+----------------+----------------+----------------+-------+--------+");
                    }
                    break;
                case 4:
                    user_interface();
                default:
                    System.out.println("Invalid choice. Please select a valid option.");
        }
    }

        }



}
```